DISS. ETH N° 26214

Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Felix Mick Finn Berkenkamp

M.Sc. ETH, ETH Zurich

born on 10.12.1990

citizen of Germany

accepted on the recommendation of

Prof. Dr. Andreas Krause (ETH Zurich), examiner

Prof. Dr. Angela P. Schöllig (University of Toronto), co-examiner

Prof. Dr. Manfred Morari (University of Pennsylvania), co-examiner

2019

Abstract

Reinforcement learning has seen significant advances over the last decade in simulated or controlled environments. These successes have lead to interest in deploying learning algorithms in the real world, where they face significant prior uncertainties. While these algorithms are often able to find high-performance control strategies eventually, they typically do not provide any safety guarantees during the learning process. As a consequence, they cannot be deployed in safety-critical systems without posing a significant safety risk to both the learning system and its environment.

In this dissertation, we introduce safe exploration algorithms that provide rigorous safety guarantees during the learning process. In particular, our algorithms explicitly model uncertainty about their environment in order to make safe decisions. These kind of algorithms are conservative in the beginning, when uncertainties are large, but become more confident over time as they acquire more data and learn about their environment. Importantly, they remain safe at all times during the learning process.

We first consider direct policy optimization, where we optimize the parameters of a policy without explicitly learning a transition model of the environment. We extend existing Bayesian optimization algorithms to the setting with multiple safety constraints. Moreover, we show how to safely transfer knowledge between different tasks that are parameterized by 'contexts'. We evaluate the resulting algorithms in extensive experiments on a flying robotic vehicle and show that it can safely learn high-performance controllers.

Secondly, we consider model-based reinforcement learning. In this setting, algorithms learn an explicit model of the environment that includes uncertainties and actively collect data in order to improve it. We theoretically show that we can safely learn policies in this setting by restricting exploration to the set of states where a safe backup strategy is available. Moreover, we provide a safe exploration algorithm based on model predictive

control that safely collects data and restricts exploration to the region of the state space where it can always recover back to this safe region. Lastly, we provide regret bounds for an optimistic exploration scheme without safety constraints and discuss how it can be applied in the safety-constrained setting.

Zusammenfassung

Das Verstärkungslernen hat in den letzten Jahren erhebliche Fortschritte in simulierten oder kontrollierten Umgebungen gemacht. Diese Erfolge haben zu Interesse am Einsatz von Lernalgorithmen in der realen Welt geführt, wo sie mit erheblichen Unsicherheiten konfrontiert sind. Diese Algorithmen sind zwar oft in der Lage leistungsfähige Steuerungsstrategien zu finden, bieten aber in der Regel keine Sicherheitsgarantien während des Lernprozesses. Infolgedessen können sie nicht in sicherheitskritischen Systemen eingesetzt werden, ohne ein erhebliches Sicherheitsrisiko für das unterliegende System und seine Umgebung darzustellen.

In dieser Dissertation stellen wir sichere Erkundungsalgorithmen vor, die die Sicherheit während des Lernprozesses garantieren. Insbesondere modellieren unsere Algorithmen Unsicherheiten in ihrer Umgebung explizit, um sichere Entscheidungen zu treffen. Diese Art von Algorithmen sind zu Beginn konservativ, wenn die Unsicherheiten groß sind, werden aber mit der Zeit selbstbewusster wenn sie mehr Daten erfassen und somit mehr über ihre Umgebung erfahren. Wichtig ist, dass die Sicherheit des Lernprozesses zu jeder Zeit garantiert ist.

Wir betrachten zunächst die direkte Steuerungsoptimierung, bei der wir die Parameter eines Reglers optimieren, ohne ausdrücklich ein Modell des Systems zu lernen. Wir erweitern existierende Bayes'sche Optimierungsalgorithmen mit mehreren Sicherheitseinschränkungen. Darüber hinaus zeigen wir, dass Informationen zwischen verschiedenen Aufgaben, die von 'Kontexten' parametrisiert werden, sicher transferiert werden können. Wir werten die resultierenden Algorithmen in umfangreichen Experimenten auf einem fliegenden Roboter aus und zeigen, dass diese in der Lage sind leistungsfähige Regler sicher zu lernen.

Zweitens betrachten wir das modellbasierte Verstärkungslernen. Diese Algorithmen lernen ein explizites Modell der Umgebung inklusive Unsicherheiten und sammeln aktiv

Daten um das Modell zu verbessern und die Unsicherheiten zu reduzieren. Wir zeigen, dass wir in dieser Situation sicher Regler lernen können, indem wir die Erkundung auf Zuständen beschränken, in denen eine sichere Ausweichsstrategie verfügbar ist. Darüber hinaus entwickeln wir einen sicheren Erkundungsalgorithmus, der auf einer modellprädiktiven Steuerung basiert. Dieser Regler ist in der Lage Daten sicher zu sammeln, indem der die Erkundung auf Zustände beschränkt, von denen er sich jederzeit wieder in eine sichere Region zurückbewegen kann. Schlussendlich zeigen wir, dass ein optimistische Erkundungsstrategie ohne Sicherheitseinschränkungen garantiert zu nah-optimalen Reglern konvergiert und diskutieren wie diese Strategie mit einem sicherheitsbeschränkten Algorithmus kombiniert werden kann.

Acknowledgments

I would like to thank my two advisors, Andreas Krause and Angela Schöllig. Their respective expertise, advice, and willingness to explore new ideas have been essential for the success of this thesis. Thank you for all the invaluable discussions over the years. I would also like to thank Manfred Morari, who kindly agreed to serve on my Ph.D. committee.

I am extremely grateful to all my fantastic collaborators and co-authors. In particular Matteo Turchetta, who has been involved in large parts of the work presented in this dissertation. Within the group, I have also enjoyed the collaborations with Johannes Kirschner and Sebastian Curi, during which I have learned a lot from their different point of views. I am also thankful to have had the chance to collaborate with Alonso Marco, Manuel Wüthrich, Rikky Duivenvoorden, and Shromona Ghosh outside of the institute. Lastly, I want to thank Karen Bodie and Maximilian Brunner for the discussions about their cool Voliro robot.

I am also fortunate to have had the opportunity to work with numerous fantastic master students at ETH. In particular I want to thank Torsten Koller, Silvan Melchior, Robin Spiess, Nikolay Nikolov, Spencer M. Richards, Riccardo Moriconi, Nicolas Carion and Hany Abdelrahman, who went on to write publications beyond their respective projects. Your motivation and new ideas were part of what made this Ph.D. exciting. I also want to thank Jan Poland for the long and ongoing collaboration with ABB that enabled several of these projects.

Beyond academics, I would like to thank everyone at the Institute for Machine Learning, and at the Learning & Adaptive Systems group specifically, for all the great conversations and fun times both at and outside of work. I am also grateful to all the people in the robotics group at the University of Toronto, who were always happy to help and made my

short visits there extremely fun. I am also indebted to the administrative staff at ETH, especially Rita, for all their help over the years.

I am happy to acknowledge institutions that have supported the research in this dissertation. In particular, the Department of Computer Science at ETH Zurich. I was also supported by a fellowship from the Open Philanthropy Project, SNSF grant 200020_159557, ERC grant no. 815943, NSERC grant RGPIN-2014-04634, and a stipend from the Vector Institute.

I want to thank my girlfriend Özge for all her support, love, and making my life happier overall. I am also grateful to my family for their boundless support and encouragement.

Contents

1

Introduction

Over the last decade, reinforcement learning (Sutton and Barto, 1998) has become an increasingly popular paradigm to learn optimal control strategies directly by interacting with an *a priori unknown* dynamical system. For example, the resulting control strategies are able to control complex video games like Atari directly from image observations (Mnih et al., 2015) and can surpass the best human players at boardgames like Go (Silver, Huang, et al., 2016). While training in simulation can mostly be thought of as a form of model-based control, the data-driven nature of these methods means that they can also be applied directly in the physical world, notably in robotics (Kober and Peters, 2014).

Unlike in simulation, in the physical world actions have real consequences. As a result, any algorithm that is deployed on real-world systems needs to ensure the safety of itself and the environment that it interacts with. Safety in *known* environments has long been considered and formalized by the control and formal methods communities, where control strategies can be synthesized that comply with a given specification. Moreover, in stochastic systems one can define different stochastic measures of the risk of violating the safety constraints. However, all these methods rely on a *known* model of the system. In contrast, in reinforcement learning the environment is *a priori* unknown.

There are two main sources of unsafe behavior in reinforcement learning (Amodei et al., 2016). For the first one, we know how to quantify safety of our system, but we *do not know* the environment in advance. Without a known model of the environment, we cannot reason about the safety of control actions in advance and thus do not know *a priori* if a

control policy is safe to deploy. The second source is the consequence of not knowing or miss-specifying the performance objective for the reinforcement learning agent, so that the resulting system behaves in an undesirable, unfair, or unethical way. In these cases, we often *do not know* how to quantify safety and the unsafe behavior would even exist if the model was perfectly known. In this dissertation, we focus on the former, since it is more tractable to analyze. This means that, we must *safely explore* the environment to collect data without ever violating the safety constraints during exploration.

For example, consider an autonomous flying vehicle. While it is desirable for the algorithm steers the robot to improve over time (e.g., by adapting to changing environment conditions), any policy applied to the system has to guarantee the safety of the system. Safety here can, for example, be defined as not crashing into any obstacles. However, without a given model of the environment it is not possible to know which control actions are likely to cause the robot to crash. In this setting, safe exploration requires the system to fly conservative in the beginning, for example, by stabilizing the system in hover. Once the algorithm safely gathers data on this simple task and learns about the system, it can start to fly slowly so that it can always stop in time whenever it detects an obstacle. Only once it has learned about the environment, can the algorithm fly safely and with high performance.

In contrast to the safe exploration behavior described above, most existing learning algorithms rely on randomized exploration to gather relevant data for the learning process. This randomization ensures that, eventually, all actions have been tried and the algorithm can determine the optimal ones. At the same time, this random exploration is likely to violate the safety specification at some point during the exploration process. As a consequence, these algorithms *cannot* be applied to safety-critical, physical systems.

Safe exploration can be achieved in many ways by incorporating additional knowledge into the learning process (García and Fernández, 2015). For example, one may have access to human supervision or a known backup strategy at every state. However, these strategies often solve the problem by assumption and switch between learning and safety whenever possible. In this dissertation, we instead consider algorithms that directly quantify and use uncertainties during the learning process and only evaluate actions when they are known to be safe.

1.1 Contributions

In this dissertation, we investigate safe exploration schemes that explicitly reason about uncertainties during the learning process. These algorithms can guarantee safety by ensuring that they act safely with respect to their internal uncertainties. That is, they *know what they do not know* and only evaluate actions where they are confident about safety. The primary contributions of this dissertation are summarized below.

Chapter 3: We analyze a safe exploration scheme for direct policy search subject to *a priori* unknown constraints. The resulting Bayesian optimization algorithm can safely optimize controller parameters and transfer knowledge about tasks. We evaluate these algorithms in extensive experiments on a flying robotic vehicle.

Chapter 4: We show how to learn a reliable model of a dynamical system under certain statistical assumptions. Next, we show that, given a policy, this assumption can be used to analyze the stability of the closed-loop system. Furthermore, we construct confidence intervals on trajectories in order to verify finite-time properties of the system.

Chapter 5: We exploit the analysis tools from Chapter 4 to design safe exploration algorithms. First, we show that it is possible to safely learn a model of the system and optimize the control policy at the same time, while respecting stability constraints encoded by a Lyapunov function. Next, we explicitly design a safe exploration strategy based on model predictive control. Lastly, we analyze an exploration scheme without safety constraints and show that it provably converges and discuss how it can be combined with the safe learning algorithms derived before.

1.2 Publications Relevant to this Dissertation

This dissertation is to large parts based on the following publications and technical reports.

- Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause (2016). "Safe controller optimization for quadrotors with Gaussian processes". In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 493–496

- Felix Berkenkamp, Andreas Krause, and Angela P. Schoellig (2016). "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics". In: *arXiv:1602.04450 [cs.RO]*
- Felix Berkenkamp, Riccardo Moriconi, Angela P. Schoellig, and Andreas Krause (2016). "Safe learning of regions of attraction in nonlinear systems with Gaussian processes". In: *Proc. of the Conference on Decision and Control (CDC)*, pp. 4661–4666
- Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause (2017). "Safe model-based reinforcement learning with stability guarantees". In: *Proc. of Neural Information Processing Systems (NeurIPS)*, pp. 908–918
- Torsten Koller, Felix Berkenkamp, Matteo Turchetta, Joschka Boedecker, and Andreas Krause (June 27, 2019). "Learning-based model predictive control for safe exploration and reinforcement learning". In: arXiv: 1906.12189 [cs, eess]
- Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause (2019). "No-Regret Bayesian optimization with unknown hyperparameters". In: *Journal of Machine Learning Research (JMLR)* 20.50, pp. 1–24

While we point out the relevant publications at the beginning of each chapter, Chapter 1 and Chapter 2 are based on material from all these publications.

1.3 Collaborators

The main content of this dissertation was developed in collaboration with Andreas Krause and Angela P. Schoellig. Chapters 4 and 5 were developed together with Matteo Turchetta. Riccardo Moriconi contributed to the continuous-time analogue of the Lyapunov stability verification in Sections 4.2 and 5.1, while the results in Sections 4.3 and 5.2 were developed together with Torsten Koller and Joschka Boedecker. Section 5.3 was developed together with Sebastian Curi.

2

Background and Related Work

In this chapter, we provide an overview of the field and state relevant background information for the remainder of the dissertation.

2.1 Notation

We start by introducing an overview of the notation used. As any interdisciplinary work, we face the issue of different notations being used in the reinforcement learning and control communities. In this dissertation, we use the classical notation from control and dynamic programming (Bertsekas et al., 1995) to define states $\mathbf{x}$ and control actions $\mathbf{u}$, which as a rich history (Bennett, 1996), rather than the modern reinforcement learning notation with $\mathbf{s}$ and $\mathbf{a}$, respectively (Sutton and Barto, 1998). As a diplomatic middle ground, we use the more positive notion of rewards from the reinforcement learning literature, as opposed to costs, to define our control and learning objectives. Note that these choices are without loss of generality.

We generally use lower case variables like x and f to denote scalars or functions. Vectors are bold, $\mathbf{x}$, and matrices bold upper case, $\mathbf{A}$. Sets are generally denoted by calligraphic variables, e.g., $\mathcal{D}$.

A subscript $\mathbf{x}_i$ denotes the ith vector $\mathbf{x}$ and we use $\mathbf{x}_{1:4} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ do denote a range of vectors. We denote the ith element of the vector $\mathbf{x}$ as $[\mathbf{x}]_i$. For a function

$f(\mathbf{x}, \mathbf{u})$, we sometimes write a corresponding scalar function over an extended input space, $f(\mathbf{x}, \mathbf{u}, i) = [f(\mathbf{x}, \mathbf{u})]_i$, to index the output dimensions. We write $\mathbb{E}_{\boldsymbol{\omega}}[\cdot] = \mathbb{E}_{\boldsymbol{\omega} \sim p(\boldsymbol{\omega})}[\cdot]$ without specifying a distribution to denote the expected value when this is clear from context. Similarly, $\mathbb{E}_{\boldsymbol{\omega}_{1:2}}[\cdot]$ is the expectation over $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$. For discrete sets $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2\}$, $|\mathcal{A}| = 2$ denotes the size of the set, while $|\mathbf{A}|$ denotes the determinant of the matrix $\mathbf{A}$.

The following tables give an overview of the main symbols used throughout the thesis.

Table 2.1: Greek letters

Symbol	Meaning
β	Scaling factor for the Gaussian process confidence intervals, see Lemma 1
γ	The worst-case mutual information.
$\boldsymbol{\omega}$	The transition noise vector in (2.1) and (4.1). Also the observation noise in model-free RL.
$\boldsymbol{\theta}$	The parameters of the control policy π_θ.
π	The control policy π
σ	The noise is σ-sub Gaussian for the RKHS regression. For the GP, the noise is Gaussian with $\mathcal{N}(0, \sigma^2)$.
τ	A trajectory of a dynamic system
κ	The discretization constant for the grid in Section 4.2

Table 2.2: Sets

Symbol	Meaning
$\mathcal{D}$	Parameter space $\boldsymbol{\theta} \in \mathcal{D} \subseteq \mathbb{R}^d$
$\mathcal{H}_k$	Reproducing kernel Hilbert space associated with the kernel k
$\mathbb{N}$	The natural numbers
$\mathbb{R}$	The real numbers
$\mathcal{U}$	Feasible region of the linear input constraints
$\mathcal{U}_\kappa$	Discretization of the action-space
$\mathcal{V}(c)$	Level-set of the Lyapunov function s.t. $v(\mathbf{x}) \leq c$
$\mathcal{X}$	Feasible region of the linear state constraints
$\mathcal{S}_n$	Safe set of states (states that fulfill the stability constraint under the policy π_n.)
$\mathcal{X}_\kappa$	Discretization of the state-space

Table 2.3: Scalars and vectors

Symbol	Meaning
$\mathbf{a}$	Input to the statistical model in Section 2.6
$c(\cdot)$	Safety constraint in safe, model-based RL
d	Dimensions of the parameters $\theta \in \mathcal{D} \subseteq \mathbb{R}^d$
$d(\cdot, \cdot)$	Distance metric
$f(\cdot, \cdot)$	transition dynamics (2.1)
$f_\pi(\cdot)$	Closed-loop transition dynamics $f(\cdot, \pi(\mathbf{x}))$
g	Known model part in (4.1)
h	Unknown model part in (4.1)
i	Generic index
j	Generic index
$k(\cdot, \cdot)$	Kernel function for the Gaussian process Section 2.6.1 and RKHS functions Section 2.6.2
$l_n(\cdot)$	Lower confidence interval
m	Number of safety constraints in Chapter 3.
n	Iteration count.
p	Dimension of the state $\mathbf{x} \in \mathbb{R}^p$
q	Dimension of the control input $\mathbf{u} \in \mathbb{R}^q$
$r(\cdot, \cdot)$	Reward function
r_n	Regret at iteration n
R_n	The cumulative regret at iteration n, $R_n = \sum_{i=1}^{n} r_n$
t	Discrete time step in the state space model (2.1)
$\mathbf{u}$	Control input to the dynamic system (2.1)
$u_n(\cdot)$	Upper confidence interval
$v(\cdot)$	Lyapunov function, see Section 2.4.1.1
w_n	Width of the confidence interval. $w_n(\cdot) = u_n(\cdot) - l_n(\cdot)$
$\mathbf{x}$	State of the dynamic system (2.1)
$\mathbf{y}$	Noisy observations for the Gaussian process in Section 2.6.1
$\mathbf{z}$	Context variables in (2.36)

2.2 Dynamical Systems

The key component of both control theory and reinforcement learning are dynamical systems, which describe the behavior of a system over time. In particular, we consider a discrete-time Markovian system, which at every discrete time step t is completely described by a state $\mathbf{x}_t$, so that the next state $\mathbf{x}_{t+1}$ is independent of all past states given $\mathbf{x}_t \in \mathbb{R}^p$. We can control the system by applying control actions $\mathbf{u}_t \in \mathbb{R}^q$ at each time step t,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\omega}_t), \tag{2.1}$$

where the function f describes the evolution of the system and depends on the *i.i.d.* transition noise $\boldsymbol{\omega}_t$ with $\mathbb{E}[\boldsymbol{\omega}_t] = 0$. The distribution of the noise $\boldsymbol{\omega}_t$ encodes the probability of transitioning from one state to another. While we assume that the noise is *i.i.d.*, the mapping through the nonlinear function f can lead to *heteroscedastic* noise that depends on the state and control input. Instead of the stochastic system (2.1), some models consider a deterministic system with $\boldsymbol{\omega}_t = \mathbf{0}$ (a dirac distribution) for all $t \geq 0$, which we write as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t). \tag{2.2}$$

Throughout the dissertation we assume that the state $\mathbf{x}$ is observed directly without observation noise. This assumption is standard the Markov Decision Process framework that the reinforcement learning community typically considers, see Section 2.5. For notational consistency and convenience, we also keep this assumption in Chapter 3, even though that method is also applicable to partially-observed systems.

We generally assume that the state $\mathbf{x}_0$ is known in advance. This is a relatively weak assumption given that the state is observed directly. However, most results and definitions can be extended to the case where $\mathbf{x}_0$ is a random variable and its distribution is well-behaved (e.g., bounded moments or sub-Gaussian).

To control the system, we us a policy $\mathbf{u}_t = \pi(\mathbf{x}_t, t)$ that, given the state $\mathbf{x}_t$ and time step t, decides which control action $\mathbf{u}_t$ to apply to the system f. In general this policy can be time-dependent and depend on all past information up to time step t. We write $\pi(\mathbf{x}_t)$ when the policy is time-independent. We denote the resulting closed-loop system by $f_\pi(\mathbf{x}, \boldsymbol{\omega}) = f(\mathbf{x}, \pi(\mathbf{x}), \boldsymbol{\omega})$ and $f_\pi(\mathbf{x}) = f(\mathbf{x}, \pi(\mathbf{x}))$. Whenever we want to highlight that the policy is parameterized by parameters $\boldsymbol{\theta} \in \mathcal{D} \subseteq \mathbb{R}^d$, we write $\pi_{\boldsymbol{\theta}}$ for the resulting policy under the parameters $\boldsymbol{\theta}$. However, we still denote the closed-loop system under this policy

as $f_\pi = f_{\pi_{\boldsymbol{\theta}}}$ whenever it is clear from context. Note that not all policies are parameterized. For example, evaluating the policy might involve solving an optimization problem for every state $\mathbf{x}$, see Section 2.3.

2.3 Optimal Control

The goal of control is to make a dynamical system behave in a certain way or to solve a task. This goal is typically specified via a deterministic reward function $r\colon \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$, so that the reward signal $r(\mathbf{x}, \mathbf{u})$ encodes how desirable it is to apply the control input $\mathbf{u}$ in state $\mathbf{x}$ in order to solve the control task. Ultimately, we want to select a policy so that the closed-loop system f_π spends as much time as possible in high-reward states. We quantify this via the value $J_\pi(\mathbf{x})$ of a state $\mathbf{x}$ under the policy π, which is defined as the expected sum of all discounted future rewards under the closed-loop dynamics with policy π,

$$J_\pi(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\omega}_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t r\left(\mathbf{x}_t, \pi(\mathbf{x}_t)\right) \,\middle|\, \mathbf{x}_0 = \mathbf{x}\right], \tag{2.3}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi(\mathbf{x}_t), \boldsymbol{\omega}_t), \tag{2.4}$$

where $\gamma \in [0, 1)$ is a discount factor that ensures the infinite sum remains finite and well-defined. Setting $\gamma = 1$ is only allowed when the infinite sum in (2.3) is finite. For example, this happens when the system has absorbing (terminal) states with zero reward. Intuitively, the smaller we choose γ the more myopic the encoded value $J_\pi(\mathbf{x})$ is. Finding a policy π that maximizes the expected performance (2.3) subject to the system dynamics (2.4) is generally known as dynamic programming.

This setting can be modeled as a Markov Decision Process, see Figure 2.1. We are given a environment that internally represents a stochastic, dynamical system as in (2.1). We can apply control actions $\mathbf{u}_t$ and observe the resulting next state $\mathbf{x}_{t+1}$ together with the corresponding reward $r(\mathbf{x}_t, \mathbf{u}_t)$. A plethora of different methods exist to solve this problem under various assumptions and a full review is beyond the scope of this dissertation. Here, we provide a high-level introduction to approximate dynamic programming and review methods that are relevant for the remainder of the dissertation.

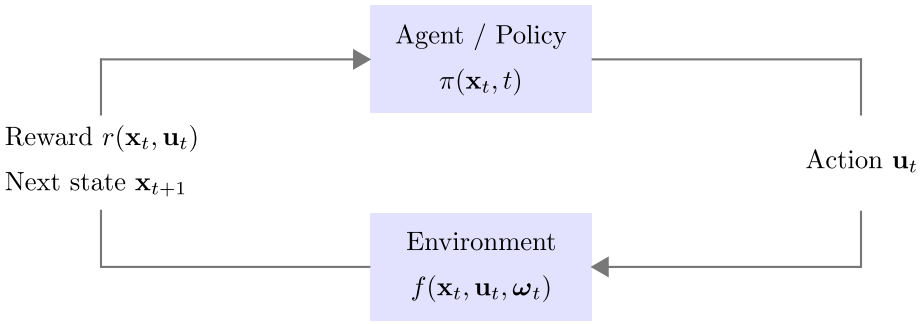


Figure 2.1: Illustration of a Markov Decision Process. At every time step t the agent (control policy) uses all past knowledge to decide on new actions $\mathbf{u}_t$. It then observes the effects of a noisy transition according to (2.1) in form of the next state $\mathbf{x}_{t+1}$ and the corresponding reward $r(\mathbf{x}_t, \mathbf{u}_t)$. We assume that the agent knows the reward function. The goal of the agent is to maximize the cumulative reward.

2.3.1 Approximate Dynamic Programming

Approximate dynamic programming summarizes a large class of methods that, for a given policy π, approximate the value function $J_\pi(\mathbf{x})$ for each state $\mathbf{x}$ with a parametric function approximator. This is known as *policy evaluation*. Subsequently this estimate can then be used for *policy improvement*, i.e., computing a policy with superior expected performance. In general, approximate dynamic programming is a large umbrella term that encompasses methods from reinforcement learning as well. Here we consider it only as a method to find an optimal policy given the dynamical system f and focus on continuous state-action spaces.

2.3.1.1 Policy Evaluation

For policy evaluation, we must approximate the true value function J_π under a given policy. It is easy to show that J_π satisfies the recursive definition

$$J_\pi(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\omega}_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t r\left(\mathbf{x}_t, \pi(\mathbf{x}_t)\right) \,\middle|\, \mathbf{x}_0 = \mathbf{x}\right], \tag{2.5a}$$

$$= r(\mathbf{x}, \pi(\mathbf{x})) + \mathbb{E}_{\boldsymbol{\omega}_{0:\infty}}\left[\sum_{t=1}^{\infty} \gamma^t r\left(\mathbf{x}_t, \pi(\mathbf{x}_t)\right) \,\middle|\, \mathbf{x}_0 = \mathbf{x}\right], \tag{2.5b}$$

$$= r(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}_{\boldsymbol{\omega}_0}\left[J_\pi(\mathbf{x}_1) \,\middle|\, \mathbf{x}_0 = \mathbf{x}\right] \tag{2.5c}$$

$$= r(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}_{\boldsymbol{\omega}}[J_\pi(f(\mathbf{x}, \pi(\mathbf{x}), \boldsymbol{\omega}))]. \tag{2.5d}$$

This recursive property of the value $J_\pi(\mathbf{x})$ of a state $\mathbf{x}$ under the policy π is usually taken as a starting point.

One of the simplest methods to approximate J_π is to minimize the *temporal difference error* between (2.5d) and $J_\pi(\mathbf{x})$, which is known as the TD(0) algorithm (Sutton, 1988; Dayan, 1992). To define a loss for the parametric approximation J_π^ψ, typically the squared distance is used,

$$\Delta J_\pi^\psi(\mathbf{x}) = \Big(\underbrace{r(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}_{\boldsymbol{\omega}}\left[J_\pi^\psi(f(\mathbf{x}, \pi(\mathbf{x}), \boldsymbol{\omega})\right]}_{\text{Training target (2.5d)}} - J_\pi^\psi(\mathbf{x})\Big)^2, \tag{2.6}$$

which measures the squared error from the Bellman equation. When this error is zero for all states $\mathbf{x} \in \mathbb{R}^p$, then we approximate the value function perfectly. However, $\Delta J_\pi^\psi(\mathbf{x}) = 0$ for some $\mathbf{x} \in \mathbb{R}^p$ generally does not imply that $J_\pi^\psi(\mathbf{x}) = J_\pi(\mathbf{x})$, since the definition recursively depends on the estimate J_π^ψ at other states. To minimize the error (2.6), we need to decide at which states $\mathbf{x}$ we want to approximate J_π. This is typically accomplished by specifying a distribution $p(\mathbf{x})$ (or weighting function) over states. Thus, we can obtain an estimate of the parameters ψ of our value function estimate J_π^ψ by minimizing the error,

$$\min_\psi \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\Delta J_\pi^\psi(\mathbf{x})\right]. \tag{2.7}$$

One choice for $p(\mathbf{x})$ that is particularly common in reinforcement learning is to manually specify a distribution over initial states $\mathbf{x}_0$ and then sample finite-length trajectories of states from the transition dynamics to obtain a distribution over $\mathbf{x}$ (Powell, 2009). The resulting distribution is

$$p(\mathbf{x}) \propto p(\mathbf{x}_0) \sum_{t=1}^{T} \mathbb{P}(\mathbf{x}_t = \mathbf{x} \,|\, \mathbf{x}_0, \pi), \tag{2.8}$$

which is proportional to the probability of the system being in state $\mathbf{x}$ at any of the T time steps. This is also known as the *state visitation probability* under the policy π. The intuition behind this choice is that we want to approximate the value function at states that are likely to be visited under the policy π when initial states are drawn from $p(\mathbf{x}_0)$.

2.3.1.2 Policy Improvement

Given an approximation J_π^ψ of J_π, we want to optimize a parametric policy $\pi_{\boldsymbol{\theta}}$ given the estimated value function; a process known as policy improvement. We write $J_{\boldsymbol{\theta}}$ for $J_{\pi_{\boldsymbol{\theta}}}$. A simple method to improve the policy is to maximize the expected reward under the dynamics and bootstraps after T time steps using the estimated value function,

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}_0 \sim p(\mathbf{x}_0)} \left[\mathbb{E}_{\boldsymbol{\omega}_{0:T}} \left[\gamma^T J_{\boldsymbol{\theta}}^\psi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \gamma^t r\left(\mathbf{x}_t, \pi(\mathbf{x}_t)\right) \right] \right], \tag{2.9}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi(\mathbf{x}_t), \boldsymbol{\omega}_t). \tag{2.10}$$

This optimization problem approximates the cost using the given system dynamics, but estimates the tail using the learned value function. One way to solve this problem is via gradient descent, where derivatives with respect to the expectations over the noise can be computed using the reparameterization trick (Kingma and Welling, 2013), see also (Lillicrap et al., 2015).

2.3.2 Model Predictive Control

An alternative method is to not parameterize the control policy by parameters θ, but instead optimize over control inputs $\mathbf{u}_t$ directly over a finite horizon T. To account for the truncated horizon, we can use a value function J_{π^*} under some policy π^* in order to approximate the cost beyond the T steps. Thus, the control policy is given by

$$\pi(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{u}_0} \min_{\mathbf{u}_{1:T}} \mathbb{E}_{\boldsymbol{\omega}_t} \left[\gamma^T J_{\pi^*}(\mathbf{x}_T) + \sum_{t=0}^{T-1} \gamma^t r\left(\mathbf{x}_t, \mathbf{u}_t\right) \,\middle|\, \mathbf{x}_0 = \mathbf{x} \right], \tag{2.11}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \boldsymbol{\omega}_t), \tag{2.12}$$

which plans an optimal sequence of control actions $\mathbf{u}_{0:T}$ for a given state $\mathbf{x}_0 = \mathbf{x}$ and then applies only the first control action of this sequence, $\mathbf{u}_0$. Solving this optimization problem at every time step t is known as model predictive control (Morari and H. Lee, 1999;

Francesco Borrelli et al., 2017). A notable advantage of this formulation is that constraints can be enforced explicitly as part of the optimization. Note that the bootstrapping with J_{π^*} can improve performance significantly in practice (U. Rosolia and F. Borrelli, 2018).

In practice, the nonlinear program (2.11) is often approximated to solve it efficiently. For example, approximating the dynamics to first order (linearizing) and the cost function to second order leads to a sequential quadratic programming (Boggs and Tolle, 1995) scheme. Without constraints, the quadratic programs can be solved in closed form using the iterative linear quadratic regulator (iLQR) (Tassa et al., 2012), while approximating the dynamics to second order leads to differential dynamic programming (DDP) (Jacobson and Mayne, 1970).

Rather than solving the optimization problem online, we can instead solve it offline and approximate the resulting control inputs, a process known as explicit model predictive control (Chen et al., 2018). From this perspective, the policy improvement step in Section 2.3.1.2 can be viewed as an explicit model predictive control scheme, which approximates the model predictive control policy $\pi(\mathbf{x})$ in (2.11) with a parametric policy $\pi_{\boldsymbol{\theta}}$ in expectation over $p(\mathbf{x}_0)$.

2.3.3 Episodic Control

One special case of the infinite-horizon optimal control problem is when the system is reset to the initial state $\mathbf{x}_0$ every T time steps. In this case, we can write the objective directly as a function of the policy parameters $\boldsymbol{\theta}$,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\omega}_{0:T-1}}\left[\sum_{t=0}^{T} \gamma^t r\left(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right) \,\middle|\, \mathbf{x}_0\right] \tag{2.13}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{\omega}_t). \tag{2.14}$$

This setting is known as *episodic* control, since the system operates in episodes that always start from the same state $\mathbf{x}_0$.

2.3.4 Sparse Rewards

In general, it is intractable to solve the optimization problems posed in this section. Thus, in practice, the model predictive control programs are solved with local approximations to both the dynamics and the cost. Similarly, the policy improvement step in Sections 2.3.1.2

and 2.3.3 is usually conducted via gradient descent, which leads to local improvements under the policy.

As a consequence of these local optimization strategies, optimal control and approximate dynamic programming become challenging when the rewards are sparse. This means that most states yield a reward of zero, except for a small set of states with positive reward. For example, an extreme case of sparsity is if we have $r(\mathbf{x}, \mathbf{u}) = 1$ only when $\mathbf{x} = \mathbf{0}$ and $r(\mathbf{x}, \mathbf{u}) = 0$ otherwise. Since the probability of the system visiting the state $\mathbf{x} = \mathbf{0}$ is zero, local changes to the policy do not improve the reward. The same holds true empirically if the policy is unlikely to visit states that incur reward.

As a result, policy optimization can only be expected to work if either the reward is sufficiently rich, or the state visitation probability $p(\mathbf{x})$ in (2.8) is non-zero for the states that have sparse rewards; that is, under our initial policy the system eventually visits states with high reward.

2.3.5 Certainty Equivalence

In linear, stabilizable systems with quadratic reward functions (even with $\gamma = 1$), it is well-known that the optimal policy is linear and can be computed in closed form (Kwakernaak and Sivan, 1972). More importantly, the certainty-equivalence principle holds, so that the optimal control policy for the stochastic linear system with additive, zero-mean noise with bounded second moment is the same as the optimal policy for the deterministic system. While this equivalence does not hold in general, this principle is often applied to nonlinear systems too; that is, the nonlinear stochastic optimal control problem is approximated with a nonlinear deterministic optimal control problem.

2.4 Safe Control

In Section 2.3, we consider optimizing the performance (cumulative reward) of a dynamical system. In practice, we also have to guarantee that the resulting system is safe. For example, we cannot only optimize a plane that carries passengers to fly as fast as possible, but also have to ensure that it is safe and respects the comfort constraints of passengers. In this section, we first discuss how to define safety in *deterministic* dynamical systems, and then

discuss extensions to stochastic systems. Lastly, we discuss these constraints in the context of optimal control from Section 2.3.

2.4.1 Definition

In general, the safety of a dynamical system f is a property of a sequence of states visited and actions applied under a control policy π. We denote these sequences (trajectories) of potentially infinite length with $\tau = \{(\mathbf{x}_0, \mathbf{u}_0), (\mathbf{x}_1, \mathbf{u}_1), \dots\}$. In general, safety can be defined for these trajectories with a specification (Woodcock et al., 2009). For example, this specification may analyze reachability or state and input constraints. It condenses all these constraints into a single function $c\colon \mathcal{T} \to \mathbb{R}$ that maps a trajectory of state-action pairs $\tau \in \mathcal{T}$ to a scalar value, so that a given trajectory is safe if

$$c(\tau) \geq 0. \tag{2.15}$$

Many frameworks to specify the function c exist that exploit different assumptions, e.g., (Maler and Nickovic, 2004). In the following, we review the two most commonly-used constraints in control: Stability and constraints on states and actions.

2.4.1.1 Lyapunov Stability and Regions of Attraction

A key concept in control is that of stability of an equilibrium point. Without loss of generality, this equilibrium point is assumed at the origin, so that $f_\pi(\mathbf{0}) = \mathbf{0}$. Stability is a property of infinitely long trajectories. Specifically, the origin of a closed-loop system is stable if the state stays within some norm-ball around the origin for all time steps and it is asymptotically stable if it converges to the origin eventually. We provide formal definitions below.

Definition 1 (Bof et al. (2018))**.** The equilibrium point $\mathbf{x} = \mathbf{0}$ of a closed-loop system $\mathbf{x}_{t+1} = f_\pi(\mathbf{x}_t)$ is

- **stable** if, for each $\epsilon > 0$, there exists a $\delta > 0$ such that

$$\|\mathbf{x}_0\| < \delta \implies \|\mathbf{x}_t\| < \epsilon, \forall t \geq 0; \tag{2.16}$$

- **unstable** if it is not stable;

- **asymptotically stable** if it is stable and there exists a $\delta > 0$ such that for any $\mathbf{x}_0$ with $\|\mathbf{x}_0\| < \delta$ it holds that
$$\lim_{t\to\infty} \|\mathbf{x}_t\| = 0; \tag{2.17}$$
- **globally asymptotically stable** if the system is asymptotically stable for any $\delta > 0$.

Remark 1. In the following, we call a closed-loop system asymptotically stable if the origin of that system is asymptotically stable as in Definition 1.

Since Definition 1 reasons about trajectories, for a given trajectory τ we can define $c(\tau) = 1$ whenever the corresponding property in Definition 1 holds and $c(\tau) = -1$ otherwise. In general, proving asymptotic stability of a system is a difficult proposition, as it requires reasoning about all trajectories of a system within a norm-ball around the origin. Fortunately, Lyapunov (1992) (originally published in 1892) provides a method to analyze the stability of systems that are Lipschitz continuous. This method uses Lyapunov functions $v(\mathbf{x})$ and allows to verify the asymptotic stability condition in Definition 1 with a simple one-step condition (Khalil and Grizzle, 1996).

Theorem 1 (Bof et al., 2018)**.** *Let $\mathbf{x} = \mathbf{0}$ be an equilibrium point for the autonomous system $\mathbf{x}_{t+1} = f_\pi(\mathbf{x}_t)$ where $f_\pi \colon \mathcal{F} \to \mathbb{R}^p$ is locally Lipschitz in $\mathcal{F} \subset \mathbb{R}^p$ and $\mathbf{0} \in \mathcal{F}$. Let $v \colon \mathbb{R}^p \to \mathbb{R}$ be a continuous function such that*

$$v(\mathbf{0}) = \mathbf{0}, \tag{2.18}$$
$$v(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathcal{F} \setminus \{\mathbf{0}\}, \tag{2.19}$$
$$\|\mathbf{x}\| \to \infty \implies v(\mathbf{x}) \to \infty, \tag{2.20}$$
$$v(f_\pi(\mathbf{x})) - v(\mathbf{x}) < 0,\ \forall \mathbf{x} \in \mathcal{F} \setminus \{\mathbf{0}\}, \tag{2.21}$$

then $\mathbf{x} = \mathbf{0}$ is globally asymptotically stable.

The key idea behind using Lyapunov functions to show stability of the system (2.2) is similar to that of gradient descent on strictly quasiconvex functions: if one can show that, given a policy π, applying the dynamics f_π on the state $\mathbf{x}$ maps it to strictly smaller values on the Lyapunov function ('going downhill') as in (2.21), then the state eventually converges to the equilibrium point at the origin (minimum, according to (2.18) and (2.19)). In contrast to reasoning about trajectories of the system in Definition 1, Theorem 1 only requires us to check the decrease condition throughout the state space.

In general, it is not easy to find suitable Lyapunov functions that satisfy (2.21). However, for physical models the energy of the system (e.g., kinetic and potential for mechanical systems) is a often a good candidate Lyapunov function. Moreover, it has recently been shown that it is possible to compute suitable Lyapunov functions (Li and Grüne, 2016; Giesl and Hafstein, 2015). In our experiments in Section 5.1, we exploit the fact that the negative value function $-J_\pi$ is a Lyapunov function if the rewards are strictly negative throughout the state space and zero only at the origin.

As a direct consequence of the proofs for Theorem 1, we can also define a local version of global asymptotic stability. In particular, each level set of the Lyapunov function that fulfills all the requirements of Theorem 1 forms a region of attraction, so that if the state $\mathbf{x}_0$ lies within this level set eventually $\mathbf{x}_t$ converges to the origin.

Corollary 1. *Under the assumptions of Theorem 1, if* $v(f_\pi(\mathbf{x})) < v(\mathbf{x})$ *for all* $\mathbf{x}$ *within the level set* $\mathcal{V}(c) = \{\mathbf{x} \in \mathcal{X} \setminus \{\mathbf{0}\} \,|\, v(\mathbf{x}) \leq c\}$, $c > 0$, *then* $\mathcal{V}(c)$ *is a region of attraction, so that* $\mathbf{x}_0 \in \mathcal{V}(c)$ *implies* $\mathbf{x}_t \in \mathcal{V}(c)$ *for all* $t > 0$ *and* $\lim_{t\to\infty} \mathbf{x}_t = \mathbf{0}$.

Note that Corollary 1 implicitly assumes that the Lyapunov function is radially increasing, so that level-sets of the Lyapunov function are connected. We assume that this holds true for the Lyapunov functions that we consider in this dissertation.

2.4.1.2 Constraint Satisfaction

Next to stability, we often want to enforce constraints on the controlled system. For example, we may want an autonomous car to stay within the lanes in order to avoid crashes. This kind of safety can be encoded as a constraint on the states $\mathbf{x}$ that the system is allowed to visit. In particular, we can write these requirements as inequality constraints of the form

$$c_x(\mathbf{x}_t) \geq 0, \, \forall t \geq 0, \tag{2.22}$$

$$c_u(\mathbf{u}_t) \geq 0, \, \forall t \geq 0. \tag{2.23}$$

As a result, we can write the corresponding constraint function $c(\cdot)$ for a trajectory in (2.15) as

$$c(\tau) = \min_{(\mathbf{x}_t, \mathbf{u}_t) \in \tau} \min(c_x(\mathbf{x}_t), c_u(\mathbf{u}_t)), \tag{2.24}$$

which is the minimum value of the two constraints in (2.22) and (2.23) along the trajectory τ. If we encode the feasible region of these constraints functions in the sets

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^p \,|\, c_x(\mathbf{x}) \geq 0\}, \tag{2.25}$$

$$\mathcal{U} = \{\mathbf{u} \in \mathbb{R}^q \,|\, c_u(\mathbf{u}) \geq 0\}, \tag{2.26}$$

we can also write the safety constraints as

$$\mathbf{x}_t \in \mathcal{X}, \quad \mathbf{u}_t \in \mathcal{U}, \ \forall t \geq 0, \tag{2.27}$$

For parametric policies $\pi_{\boldsymbol{\theta}}$, the input constraints can typically be enforced by structuring the policy in an appropriate way so that the range of the function is restricted to $\mathcal{U}$, that is $\pi_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathcal{U}$ for all $\mathbf{x} \in \mathbb{R}^p$. Enforcing the state constraints is equivalent to the requirement that the set $\mathcal{X}$ must be *positive invariant*.

Definition 2. A set $\mathcal{X}$ is *positive invariant* under the dynamics f_π if, for every $\mathbf{x}_0 \in \mathcal{X}$, we have $\mathbf{x}_t \in \mathcal{X}$ for all $t > 0$.

Notably the region of attraction in Corollary 1 is positive invariant, so that $\mathcal{V}(c) \subseteq \mathcal{X}$ implies that, if $\mathbf{x}_0 \in \mathcal{V}(c)$, we have $\mathbf{x}_t \in \mathcal{X}$ for all time steps $t > 0$ and we satisfy the state constraints. However, asymptotic stability is not a necessary condition to comply with the state constraints, nor vice-versa.

State constraints can be verified with control barrier functions (Ames et al., 2019), analogously to the Lyapunov functions in Section 2.4.1.1. While we do not review these functions, they can be thought of as enforcing the Lyapunov decrease condition only on the boundary of the set $\mathcal{X}$ in order to render $\mathcal{X}$ positive invariant, without requiring convergence of the state to the origin.

2.4.2 Stochastic Safety

So far, we have considered safety in deterministic systems. For the stochastic system (2.1) it is generally impossible to make any deterministic statements about safety. In particular, for a given starting state $\mathbf{x}_0$, the resulting trajectories are random variables. As a result, $c(\tau)$ is a random variable too. Thus, in order to reason about safety, we have to decide on a measure of uncertainty for this random variable.

The option that is closest to the optimal control framework is to consider expected safety. This notion is conceptually similar to certainty-equivalence control Section 2.3.5. However, while expected performance is a reasonable objective, expected safety is generally less desirable. In particular, we may have $\mathbb{E}[c(\tau)] \geq 0$ and still have the system violate the safety constraints frequently. Instead, risk-sensitivity (Luce and Raiffa, 1958) not only penalized the expected value but also includes higher moments. A stronger notion is the value at risk (Duffie and Pan, 1997), which considers safety with high probability. Lastly, one can also consider the conditional value at risk (Rockafellar and Uryasev, 2002), which considers the expected safety over the tail of the distribution.

Next to measures of uncertainty, formal methods and robust control (Woodcock et al., 2009; Zhou and Doyle, 1998) typically consider a worst-case criterion. For example, for the stochastic system stability can be required to hold with probability one (Afanas'ev et al., 1996).

2.4.3 Safety-constrained Markov Decision Process

Ultimately our goal is to find optimal policies as in Section 2.3 that also adhere to the safety constraints. To this end, we can choose any of the stochastic or worst-case safety definitions discussed in the previous section.

The risk-based notions of stochastic safety introduced in the previous section have all been explored in the context of finite Markov Decision Processes as well. Altman (1999) considers a constraint on the expected performance/safety, while Howard and Matheson (1972), Marcus et al. (1997), and Ruszczyński (2010) consider risk-sensitivity. Lastly, Bäuerle and Ott (2011) consider the value at risk for the performance. All of these typically assume that the safety function c decomposes additively over time steps, similar to the reward function for performance.

In continuous domains, safety constraints can be enforced by carefully designing controllers that are asymptotically stable (Khalil and Grizzle, 1996). The most direct way to incorporate constraints is in model predictive control. There, stability can be enforced by appropriately selecting the value function for bootstrapping or via terminal set constraints (Rawlings and Mayne, 2009). For stochastic systems, Schwarm and Nikolaou (1999) consider satisfying constraints with high-probability, which leads to so-called chance-constraints.

Another way to design safe controllers is by explicitly computing reachable subsets of the state space (Ding et al., 2011). These sets are characterized by a special kind of value function that measures worst-case disturbances from an adversary and also provide a corresponding safe policy. No closed-form solutions for these reachable sets exist, but they can be approximated using a discrete grid (Mitchell et al., 2005) or a function approximator (Akametalu et al., 2018).

2.5 Reinforcement Learning

In the optimal control section, Section 2.3, we have assumed that both the dynamics model f and the reward function r are known. The key different between reinforcement learning and optimal control is that the transition function f is *not known a priori* (Sutton and Barto, 1998). Despite this, we still aim to solve the optimal control problem in Section 2.3.

Remark 2. In the general reinforcement learning setting, the reward function r is often assumed to be unknown and stochastic as well. In this dissertation, we consider a less general setting and assume that the reward function is a known, deterministic function.

One strategy to find an optimal policy is to apply a two-stage method that first learns about the environment and then uses this knowledge to find the optimal policy as in Section 2.3. This is the typical combination of system identification and model-based control that is predominant in control theory. However, this procedure incurs low rewards during the system identification phase. Moreover, it has to learn about the environment globally, not only at the states that are relevant for solving the optimal control problem.

The goal of reinforcement learning instead is to maximize the rewards also during the learning process. Thus we have to *simultaneously* learn about the environment and optimize our actions to achieve high cumulative reward.

2.5.1 Policy Improvement Without a System Model

The key insight of many reinforcement learning algorithms is that we can solve the optimal control problems in Section 2.3 without a known dynamics model. Instead, we can use sampled trajectories from the system directly to optimize the policy.

In episodic reinforcement learning problems, see Section 2.3.3, we can estimate the gradients of the cost (2.14) directly. If the policy is stochastic, the REINFORCE algorithm (R. J.

Williams, 1992) estimates gradients by weighting the empirical sum of rewards observed for samples trajectories. Note that, in principle, one can also use gradient-free algorithms, e.g., genetic algorithms (Davidor, 1991), to optimize the policy directly.

We can also use the samples for policy evaluation in Section 2.3.1.1, by replacing the expectation with a finite sum over samples. Sutton, McAllester, et al. (1999) show that we can replace the empirical returns of REINFORCE by those of a learned value function. This method forms the basis of modern policy search algorithms (Peters and Schaal, 2006; Kober and Peters, 2014; Schulman et al., 2017). Similar results can be obtained for deterministic policies (Silver, Lever, et al., 2014), which leads to algorithms like the one proposed by Lillicrap et al. (2015). Their method optimizes (2.9) for $T = 0$ by taking gradients with respect to the learned value function directly. Notably, these methods can also be applied in the *off-policy* case, when the behavior policy that we collect samples from is not the same as the one that we optimize (Munos et al., 2016).

Similar 'model-free' results exist for model predictive control, where the optimal control policy can be computed by perturbing the selected actions by noise and updating the control actions based on the weighted trajectories (G. Williams et al., 2017). This approach is similar to path-integral reinforcement learning for parameterized policies (Theodorou et al., 2010).

These approaches are similar to the optimal control methods in Section 2.3, but replace the system model with a sample-based representation to, for example, estimate gradients or value functions. As a consequence, these algorithms inherit the properties of the original control problem in Section 2.3.1, including the difficulty of applying local optimization methods with sparse rewards, see Section 2.3.4.

2.5.2 Model-based Reinforcement Learning

Alternatively, we can use the data directly to learn a model and use the learned model to control the system using the methods in Section 2.3. This is often referred to as *model-based* reinforcement learning. The model here only refers to the dynamics f, since 'model-free' algorithms that do not learn f directly may also learn a model of, for example, the value function. This is similar to the typical system identification and control process (Atkeson and Santamaria, 1997), except that the model and policy are updated simultaneously online.

2.5.3 Aleatoric versus Epistemic Uncertainty

None of the methods presented in this section directly consider *uncertainty* during the learning process. This can cause these methods to be data-inefficient or even to converge to local optima. For example, a learned model or value function might be inaccurate due to lack of data, so that the policy update can decrease performance. In this section, we characterize uncertainty in reinforcement learning and discuss methods that use it for exploration in Section 2.7.

The uncertainty in the reinforcement learning problem can be split into two parts (Der Kiureghian and Ditlevsen, 2009). On the one hand, we face *aleatoric* uncertainty that is introduced by the transition noise $\boldsymbol{\omega}_t$ in (2.1). If we were to repeat the same experiment twice, starting from the same initial state $\mathbf{x}_0$, we would obtain two different trajectories. This uncertainty is inherent to the environment and is thus *irreducible*. This uncertainty is also present in the optimal control problem in Section 2.3.

The uncertainty arising from not knowing the dynamics f *a priori* is called *epistemic* or structural uncertainty. This uncertainty incorporates any knowledge about the environment that we *could know*, e.g., the transition function f, but that we do not know *a priori*. Epistemic uncertainty can be reduced by exploration and gathering additional data about the environment.

The difference between these two sources of uncertainty in reinforcement learning has been highlighted by Gal (2016) and exploiting this structure can lead to large performance gains in practice. In the next section, Section 2.6, we discuss how to model epistemic uncertainty and discuss uncertainty-based exploration schemes in Section 2.7.

2.6 Modelling Epistemic Uncertainty

The aleatoric uncertainty in the reinforcement learning problem is described by the process noise $\boldsymbol{\omega}$ in (2.1). In this section, we introduce models for the *epistemic* uncertainty of a generic function $f' \colon \mathcal{A} \to \mathbb{R}$. For now, this function f' is a placeholder and we discuss where it can be used in the context of reinforcement learning in Section 2.7.

In this setting, at iteration n we are given a finite dataset of function inputs $\mathcal{A}_n$ with corresponding, noisy observations of the function f' given by $\mathbf{y}_n = \mathbf{y}_{\mathcal{A}_n} = \{f'(\mathbf{a}_i) + \omega_i\}_{\mathbf{a}_i \in \mathcal{A}_n}$. Here ω_i is *i.i.d.* observations noise that follows a known distribution.

In general, any model that uses these observations to output a set of values or a distribution over values of f', rather than a point-estimate, can be used to model uncertainty. However, in for this model to be *reliable* it must describe the true function sufficiently well. We call such a model well-calibrated. In particular, we use the following type of error bounds for the remainder of the thesis.

Definition 3 (well-calibrated model)**.** Let $\mu_n(\mathbf{a})$ and $\sigma_n(\mathbf{a})$ denote outputs of our statistical model after observing data at inputs in $\mathcal{A}_n$ after iteration n. This model is well-calibrated with respect to f' over a set $\mathcal{A}$ if there exists a sequence $\beta_n \in \mathbb{R}_{>0}$ such that, with probability at least $(1-\delta)$, it holds jointly for all $n \geq 0$ and all $\mathbf{a} \in \mathcal{A}$ that $|f'(\mathbf{a}) - \mu_n(\mathbf{a})| \leq \beta_n \sigma_n(\mathbf{a})$.

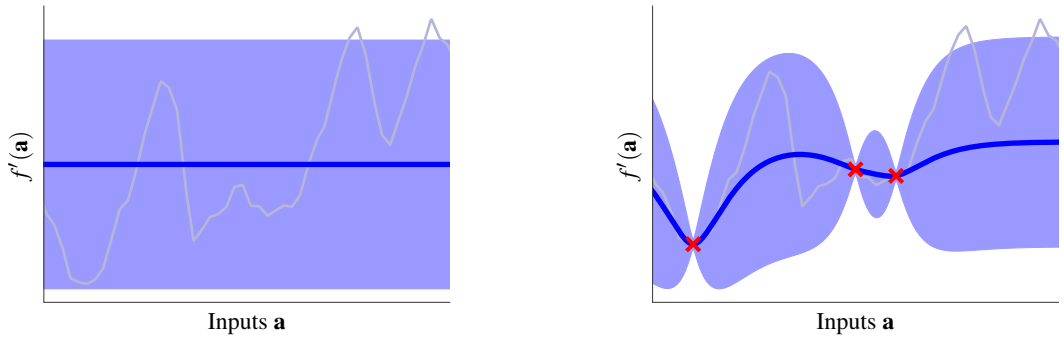
Definition 3 requires our model to be contained in a set of confidence intervals for all iterations and data points. If we have a model that provides reliable point-estimates (reliable marginal distributions) with high probability, we can generalize these to finite sets $\mathcal{A}$ by applying the union bound or to continuous domains by employing continuity arguments and additional assumptions on f'. While the requirement that this should hold for all $n > 0$ may seem like a difficult requirement, this can generally be achieved by applying a union bound with a particular choice of probability budget at every step n, see (Srinivas et al., 2012, Lemma 5.1).

Definition 3 places an assumption on the model. In practice, we often prefer assumptions on the function f' that imply Definition 3 for a given model. Moreover, Definition 3 does not provide any insights over whether the confidence intervals shrink as we gather more data. In the following, we discuss two special cases.

2.6.1 Gaussian Processes

A Bayesian belief over a function f' naturally models epistemic uncertainty given data with the posterior distribution. In general, computing the posterior distribution through Bayes' rule is not possible in closed form and approximate inference has to be used (Blei et al., 2017).

One particular, nonparametric model that has a tractable, closed-form posterior distribution are Gaussian processes (Carl Edward Rasmussen and C. K. Williams, 2006). The goal of Gaussian process inference is to infer a posterior distribution over the nonlinear map $f'(\mathbf{a}) : \mathcal{A} \to \mathbb{R}$ from an input vector $\mathbf{a} \in \mathcal{A}$ with $\mathcal{A} \subseteq \mathbb{R}^d$ to the function value $f'(\mathbf{a})$.



(a) Gaussian process prior.

(b) Gaussian process posterior.

Figure 2.2: Gaussian process model illustration for a Matérn kernel. Samples from a GP typically have a regular rate of change (gray line). By assumption, the Gaussian process prior mean (blue line) is zero in Figure 2.2a. Moreover, since the Matérn kernel depends only on the distance between two parameters, the prior marginal confidence intervals for each input $\mathbf{a}$ are also constant (blue shaded). After observing three measurements (red crosses) in Figure 2.2b, the Gaussian process confidence intervals are reduced close to the measurements.

This is accomplished by assuming that the function values $f'(\mathbf{a})$, associated with different values of $\mathbf{a}$, are random variables and that any finite number of these random variables have a *joint* normal distribution (Carl Edward Rasmussen and C. K. Williams, 2006).

A Gaussian process distribution is parameterized by a prior mean function and a covariance function or kernel $k(\mathbf{a}, \mathbf{a}')$, which defines the covariance of any two function values $f(\mathbf{a})$ and $f(\mathbf{a}')$ for $\mathbf{a}, \mathbf{a}' \in \mathcal{A}$. In this work, the mean is assumed to be zero without loss of generality. The choice of kernel function is problem-dependent and encodes assumptions about the unknown function. A review of potential kernels can be found in (Carl Edward Rasmussen and C. K. Williams, 2006).

We can condition a Gaussian process on the observations $\mathbf{y}_n$ at input locations $\mathcal{A}_n$. The Gaussian process model assumes that observations are noisy measurements of the true function value with Gaussian noise, $\omega \sim \mathcal{N}(0, \sigma^2)$. The posterior distribution is again a Gaussian process with mean μ_n, covariance k_n, and variance σ_n, where

$$\mu_n(\mathbf{a}) = \mathbf{k}_n(\mathbf{a})(\mathbf{K}_n + \mathbf{I}\sigma^2)^{-1}\mathbf{y}_n, \tag{2.28}$$

$$k_n(\mathbf{a}, \mathbf{a}') = k(\mathbf{a}, \mathbf{a}') - \mathbf{k}_n(\mathbf{a})(\mathbf{K}_n + \mathbf{I}\sigma^2)^{-1}\mathbf{k}_n^{\mathrm{T}}(\mathbf{a}'), \tag{2.29}$$

$$\sigma_n^2(\mathbf{a}) = k_n(\mathbf{a}, \mathbf{a}). \tag{2.30}$$

The covariance matrix $\mathbf{K}_n \in \mathbb{R}^{|\mathcal{A}_n| \times |\mathcal{A}_n|}$ has entries $[\mathbf{K}_n]_{(i,j)} = k(\mathbf{a}_i, \mathbf{a}_j)$ with $\mathbf{a}_i, \mathbf{a}_j \in \mathcal{A}_n$ and the vector $\mathbf{k}_n(\mathbf{a}) = \left[k(\mathbf{a}, \mathbf{a}_1), \ldots, k(\mathbf{a}, \mathbf{a}_{|\mathcal{A}_n|})\right]$ contains the covariances between the input $\mathbf{a}$ and the observed data points in $\mathcal{A}_n$. The identity matrix is denoted by $\mathbf{I}$. An illustration of the Gaussian process prior and posterior marginal distributions can be seen in Figure 2.2.

Given the Gaussian process assumptions, we obtain point-wise confidence estimates from the marginal Normal distribution specified by μ_n and σ_n. For finite sets, the Gaussian process belief induces a *joint* normal distribution over function values that is correlated through (2.29). We can use this to fulfill Definition 3 for continuous sets by using a union bound and exploiting that samples from a Gaussian process are Lipschitz continuous with high probability (Srinivas et al., 2012, Theorem 2).

2.6.1.1 Information Capacity

One important property of normal distributions is that the confidence intervals contract after we observe measurement data. How much data we require for this to happen generally depends on the variance of the observation noise, σ^2, and the size of the function class; i.e., the assumptions that we encode through the kernel. In the following, we use results by Srinivas et al. (2012) and use the mutual information to construct such a capacity measure.

Formally, the mutual information between the Gaussian process prior on f' at locations $\overline{\mathcal{A}}$ and the corresponding noisy observations $\mathbf{y}_{\overline{\mathcal{A}}}$ is given by

$$\mathrm{I}(\mathbf{y}_{\overline{\mathcal{A}}}; f) = 0.5 \log |\mathbf{I} + \sigma^{-2} \mathbf{K}_{\overline{\mathcal{A}}}|, \tag{2.31}$$

where $\mathbf{K}_{\overline{\mathcal{A}}}$ is the kernel matrix $[k(\mathbf{a}, \mathbf{a}')]_{\mathbf{a},\mathbf{a}' \in \overline{\mathcal{A}}}$ and $|\cdot|$ is the determinant. Intriguingly, for Gaussian process models this quantity only depends on the inputs in $\overline{\mathcal{A}}$ and not the corresponding measurements $\mathbf{y}_{\overline{\mathcal{A}}}$. Intuitively, the mutual information measures how informative the collected samples $\mathbf{y}_{\mathcal{A}}$ are about the function f. If the function values are independent of each other under the Gaussian process prior, they provide large amounts of new information. However, if measurements are taken close to each other as measured by the kernel, they are correlated under the Gaussian process prior and provide less information.

The mutual information in (2.31) depends on the locations $\mathcal{A}_n$ at which we obtain measurements. While it can be computed in closed-form, it can also be bounded by the largest mutual information that any algorithm could obtain from n noisy observations,

$$\gamma_n = \max_{\mathcal{A} \subset D, |\mathcal{A}| \leq n} \mathrm{I}(\mathbf{y}_{\mathcal{A}}; J). \tag{2.32}$$

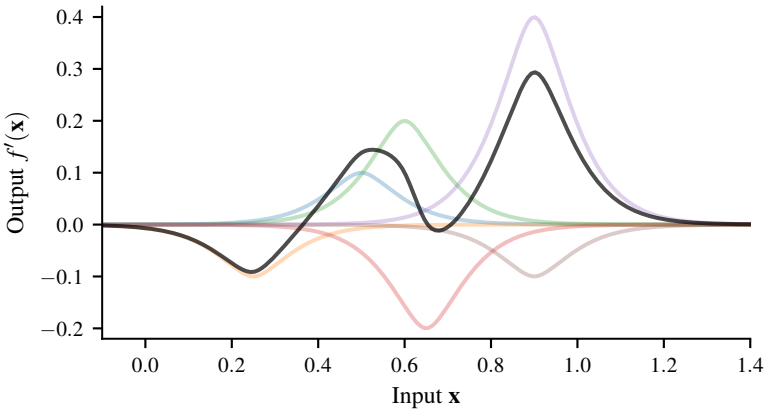


Figure 2.3: Example of an RKHS function. The individual weighted kernel evaluations at representer points (colored lines) add up to form a nonlinear function (black line).

We refer to γ_n as the *information capacity*, since it can be interpreted as a measure of complexity of the function class associated with a Gaussian process prior. It was shown by Srinivas et al. (2012) that γ_n has a sublinear dependence on n for many commonly used kernels such as the Gaussian kernel. This sublinear dependence is generally exploited by exploration algorithms in order to show convergence, see Section 2.7

2.6.2 Functions in a Reproducing Kernel Hilbert Space

Instead of the Bayesian Gaussian process framework, we can also consider frequentist confidence intervals. Unlike the Bayesian framework, which inherently models a belief over a random function, frequentists assume that there is an *a priori* fixed underlying function f' of which we observe noisy measurements.

The natural frequentist counterpart to Gaussian processes are functions inside the Reproducing Kernel Hilbert Space (RKHS) spanned by the same kernel $k(\mathbf{a}, \mathbf{a}')$ as used by the Gaussian process in Section 2.6.1. An RKHS $\mathcal{H}_k$ contains well-behaved functions of the form $f(\mathbf{a}) = \sum_{i \geq 0} \alpha_i \, k(\mathbf{a}, \mathbf{a}_i)$, for given representer points $\mathbf{a}_i \in \mathbb{R}^d$ and weights $\alpha_i \in \mathbb{R}$ that decay sufficiently quickly, see Figure 2.3 for an example. For example, the Gaussian process mean function (2.28) lies in this RKHS. The kernel function $k(\cdot, \cdot)$ determines the roughness and size of the function space and the induced RKHS norm $\|f'\|_k^2 = \langle f, \, f \rangle_k = \sum_{i,j \geq 0} \alpha_i \alpha_j k(\mathbf{a}_i, \mathbf{a}_j)$ measures the complexity of a function $f' \in \mathcal{H}_k$ with respect to the kernel. In particular, the function f' is Lipschitz continuous with

respect to the kernel metric

$$d(\mathbf{a}, \mathbf{a}') = \sqrt{k(\mathbf{a}, \mathbf{a}) + k(\mathbf{a}', \mathbf{a}') - 2k(\mathbf{a}, \mathbf{a}')}, \tag{2.33}$$

so that $|f'(\mathbf{a}) - f'(\mathbf{a}')| \leq \|f'\|_k d(\mathbf{a}, \mathbf{a}')$, see the proof of Proposition 4.30 by Christmann and Steinwart (2008).

2.6.2.1 Confidence Intervals

We can construct an estimate together with reliable confidence intervals if the measurements are corrupted by σ-sub-Gaussian noise. This is a class of noise where the tail probability decays exponentially fast, such as in Gaussian random variables or any distribution with bounded support. Specifically, we have the following definition.

Definition 4 (Vershynin (2010))**.** A random variable X is σ-sub-Gaussian if $\mathbb{P}\{|X| > s\} \leq \exp(1 - s^2/\sigma^2)$ for all $s \geq 0$.

While the Gaussian process framework makes different assumptions about the function and the noise, Gaussian processes and RKHS functions are closely related (Kanagawa et al., 2018) and it is possible to use the Gaussian process posterior marginal distributions to infer reliable confidence intervals on f'.

Lemma 1 (Abbasi-Yadkori (2012) and Chowdhury and A. Gopalan (2017))**.** *Assume that f has bounded RKHS norm $\|f'\|_k \leq B$ and that measurements are corrupted by σ-sub-Gaussian noise. If $\beta_n^{1/2} = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_n; f) + 1 + \ln(1/\delta)}$, then for all $\mathbf{a} \in \mathcal{A}$ and $n \geq 0$ it holds jointly with probability at least $1 - \delta$ that $|\, f'(\mathbf{a}) - \mu_n(\mathbf{a}) \,| \leq \beta_n^{1/2} \sigma_n(\mathbf{a})$.*

Lemma 1 implies that, with high probability, the true function f' is contained in the confidence intervals induced by the posterior Gaussian process distribution that uses the kernel k from Lemma 1 as a covariance function, scaled by an appropriate factor β_n. In contrast to Section 2.6.1, Lemma 1 does not make probabilistic assumptions on f'. In fact, f' could be chosen adversarially, as long as it has bounded norm in the RKHS.

Since the frequentist confidence intervals depend on the mutual information and the marginal confidence intervals of the Gaussian process model, they inherit the same contraction properties up to the factor β_n. However, note that the confidence intervals in Lemma 1 hold jointly through the continuous domain $\mathcal{A}$. This is not generally possible for Gaussian process models without employing additional continuity arguments, since

Gaussian process distributions are by definitions only defined via a multivariate Normal distribution over *finite* sets. This stems from the difference between a Bayesian belief and the frequentist perspective, where the function is unknown but fixed *a priori*.

2.6.3 Other Models

Beyond the Bayesian Gaussian processes and the frequentist confidence intervals for RKHS regression, many other methods exist to build confidence intervals. For example, Liu et al. (2019) and Guo et al. (2017) investigate providing calibrated confidence intervals for neural networks. Bayesian inference for more complicated models than Gaussian processes is often intractable and approximate inference methods have to be used to obtain uncertainty estimates (Gal, 2016). Another effective method is to train ensembles of point-wise function estimators in order to estimate uncertainty (Johnson, 2001; Lakshminarayanan et al., 2017).

Analyzing these methods theoretically is challenging. However, in practice one may have sufficient trust in these methods to yield reliable confidence intervals regardless. That is, in practice one might have sufficient trust in a particular model to fulfill Definition 3 by assumption.

2.7 Uncertainty-based Exploration

Given a statistical model of the uncertainty, we face a tradeoff between *exploration*, where we collect data to learn more about the environment and improve our statistical model, and *exploitation*, where we use the existing statistical model to make better decisions. In this section, we review several methods from the literature used to solve it in the context of reinforcement learning.

2.7.1 Bandits and Bayesian Optimization

The exploration-exploitation tradeoff has been studied extensively is in the bandit literature (Lattimore and Szepesvári, 2018). These are one-state Markov Decision Processes where all control actions lead back to the same state. Generally, bandit algorithms aim to solve optimization problems of the form $\max_{\boldsymbol{\theta} \in \mathcal{D}} J(\boldsymbol{\theta})$ efficiently by querying a sequence of inputs (arms/actions) $\boldsymbol{\theta}_n$ and observing the corresponding noise-perturbed realizations of $J(\boldsymbol{\theta})$.

For example, in a reinforcement learning setting bandit algorithms can be used to optimize the episodic cost (2.14) directly, where after each evaluation the system is reset back to its initial state and the bandit can evaluate a new set of parameters $\boldsymbol{\theta}$. In this case, $\boldsymbol{\theta}$ are the actions (arms) selected by the bandit algorithm, while $J(\boldsymbol{\theta})$ is the unknown objective function.

Regret In order to judge the quality of a bandit algorithm we must define what it means to solve the exploration - exploitation problem efficiently. We aim to construct a sequence of input evaluations $\boldsymbol{\theta}_n$ that eventually maximizes the function value $J(\boldsymbol{\theta}_n)$. One natural way to prove this convergence is to show that an algorithm has sublinear regret. The instantaneous regret at iteration n is defined as $r_n = \max_{\boldsymbol{\theta}\in\mathcal{D}} J(\boldsymbol{\theta}) - J(\boldsymbol{\theta}_n) \geq 0$, which is the loss incurred by evaluating the function at $\boldsymbol{\theta}_n$ instead of at the *a priori unknown* optimal inputs. The cumulative regret is defined as $R_N = \sum_{n=1}^{N} r_n$, the sum of instantaneous regrets incurred over N steps. If we can show that the cumulative regret is sublinear for a given algorithm, that is, $\lim_{n\to\infty} R_n \,/\, n = 0$, then eventually the algorithm evaluates the function at inputs that lead to close-to-optimal function values most of the time. We say that such an algorithm has *no-regret*. Intuitively, if the average regret approaches zero then, on average, the instantaneous regret must approach zero too, since r_n is strictly greater or equal than zero. This implies that there exists an iteration $n > 0$ such that $J(\boldsymbol{\theta}_n)$ is arbitrarily close to $J(\boldsymbol{\theta}^*)$ and the algorithm converges. Thus, we aim to design optimizations algorithm that have sublinear cumulative regret.

2.7.1.1 Bayesian Optimization

Several Bandit algorithms with various model assumptions and definitions of regret exist. We refer to (Lattimore and Szepesvári, 2018) for a thorough review. One class of Bandit algorithms that has been successfully applied to reinforcement learning and robotics is Bayesian optimization (Mockus, 2012). Bayesian optimization methods treat $J(\boldsymbol{\theta})$ as an unknown function and make regularity assumptions about it. These regularity properties are used to actively learn a model of the objective function.

Bayesian optimization methods often model the unknown function as a Gaussian process, see Section 2.6.1, and assume that any randomness in the evaluation of $J(\boldsymbol{\theta})$ can be modeled as noise. Gaussian process-based methods use the posterior mean and variance predictions in (2.28) and (2.30) to compute the next sample location (Mockus, 2012; Jones, 2001).

The resulting algorithms are practical and, if their assumptions are satisfied, provably find the global optimum of the objective function (Bull, 2011; Srinivas et al., 2012). Moreover, they tend to be data-efficient and only require few function evaluations.

Example applications of Bayesian optimization in robotics include gait optimization of legged robots (Calandra, N. Gopalan, et al., 2014; Lizotte et al., 2007) and the optimization of the controller parameters of a snake-like robot (Tesch et al., 2011). Marco et al. (2017) optimize the weighting matrices of an LQR controller for an inverted pendulum by exploiting additional information from a simulator. Several different Bayesian optimization methods are compared by Calandra, Seyfarth, et al. (2014) for the case of bipedal locomotion.

In the following, we discuss the main methods that are generally used to construct no-regret algorithms in bandits. These also form the basis of exploration in reinforcement learning. We specifically focus on Bayesian optimization.

Optimisim in the Face of Uncertainty One way to solve the exploration-exploitation dilemma is to be optimistic with respect to the uncertainty encoded in the statistical model (Lai and Robbins, 1985). For example, the GP-UCB algorithm by Srinivas et al. (2012) uses confidence intervals on the function J, e.g., from Lemma 1, in order to select as next input $\boldsymbol{\theta}_n$ that optimistically and plausibly can achieve the largest plausible performance value according to the model,

$$\boldsymbol{\theta}_{n+1} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \mu_n(\boldsymbol{\theta}) + \beta_n^{1/2} \sigma_n(\boldsymbol{\theta}). \tag{2.34}$$

Intuitively, (2.34) selects new evaluation points at locations where the upper bound of the confidence interval of the Gaussian process estimate is maximal. Repeatedly evaluating the function J at inputs $\boldsymbol{\theta}_{n+1}$ given by (2.34) improves the mean estimate of the underlying function and decreases the uncertainty at candidate locations for the maximum, so that the global maximum is provably found eventually (Srinivas et al., 2012). While (2.34) is also an optimization problem, it only depends on the Gaussian process model of J and solving it therefore does not require any expensive evaluations of J.

Srinivas et al. (2012) show that the GP-UCB algorithm has cumulative regret $R_n = \mathcal{O}\left(\sqrt{n\beta_n\gamma_n}\right)$ for all $n \geq 1$ with the same $(1-\delta)$ probability as the confidence intervals, e.g., in Lemma 1, hold. Since γ_n in (3.6) is sublinear for many commonly-used kernels, see Section 2.6.2, the cumulative regret R_n has a sublinear dependence on n so that $R_n/n \to 0$ and therefore GP-UCB converges to function evaluations close to $J(\boldsymbol{\theta}^*)$.

Thompson Sampling An alternative to optimism is to instead sample from the posterior distribution over plausible candidates and select the next parameters $\boldsymbol{\theta}_n$ that achieve maximal performance on this sample (D. J. Russo et al., 2018). This strategy was shown to have sublinear regret in the Bayesian optimization setting by Chowdhury and A. Gopalan (2017), specifically $R_n = \tilde{\mathcal{O}}\left(\sqrt{dn\beta_n\gamma_n}\right)$. Sampling from the posterior is especially efficient in linear models. This is exploited by Mutny and Krause (2018), who approximate the Gaussian process posterior with a provably accurate linear model, and then sample from this linear model.

Information-based Criteria More recently, it was proposed to trade off regret for the information gained during learning by Daniel Russo and Van Roy (2014). This exploration scheme can be particularly effective if the observation noise, and thus the amount of information obtained, depends on the parameters θ that we evaluate (Kirschner and Krause, 2018). This setting with *heteroscedastic* noise explicitly exploits the different sources of uncertainty, see Section 2.5.3.

Contextual Bayesian Optimization The bandit framework can be extended to be closer to state-full Markov Decision Processes, by additionally considering external variables during the optimization process (Auer, 2002; Langford and Zhang, 2007). These additional variables $\mathbf{z} \in \mathcal{Z}$ are often called contexts. For example, the performance of a robot may depend on its battery level or the weather conditions, both of which cannot be influenced directly. Alternatively, contexts can also represent different tasks that the robot has to solve, which are specified externally by a user. The idea is to include the functional dependence on the context in the Gaussian process model, but to consider them fixed when selecting the next parameters to evaluate.

This setting is called contextual Bayesian optimization (Krause and Ong, 2011). Given a context $\mathbf{z} \in \mathcal{Z}$ that is fixed by the environment, we can model how the performance and constraint functions change with respect to different contexts by multiplying the kernel function $k_{\boldsymbol{\theta}}$ over the parameters, with another kernel $k_z \colon \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ over the contexts,

$$k((\boldsymbol{\theta}, \mathbf{z}), (\boldsymbol{\theta}', \mathbf{z}')) = k_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\theta}') \cdot k_z(\mathbf{z}, \mathbf{z}'). \tag{2.35}$$

This kernel structure implies that function values are correlated when both parameters and the contexts are similar. For example, we would expect selecting the same parameters $\boldsymbol{\theta}$ for

a control algorithm to lead to similar performance values if the context (e.g., the battery level) is similar.

Since contexts are not part of the optimization criterion, a modified version of (2.34) has to be used. It was shown by Krause and Ong (2011) that an algorithm that evaluates the GP-UCB criterion given a fixed context $\mathbf{z}_n$,

$$\boldsymbol{\theta}_n = \underset{\boldsymbol{\theta} \in \mathcal{A}}{\operatorname{argmax}} \; \mu_{n-1}(\boldsymbol{\theta}, \mathbf{z}_n) + \beta_n^{1/2} \sigma_{n-1}(\boldsymbol{\theta}, \mathbf{z}_n), \tag{2.36}$$

enjoys similar convergence guarantees as GP-UCB so that, after seeing a particular context sufficiently often, the criterion (2.36) queries parameters that are close-to-optimal.

2.7.2 Reinforcement Learning

In the full reinforcement learning setting from Section 2.5, exploration algorithms have largely been inspired by the bandit algorithms from Section 2.7.1. The main conceptual difference is that uncertainty can be modeled at different levels. In the following, we review the three main uncertainties that are modeled and used by algorithms in order to improve exploration.

2.7.2.1 Reward Uncertainty

One option is to directly model uncertainty about the reward function $r(\mathbf{x}, \mathbf{u})$. This uncertainty can be incorporated into reinforcement learning by providing an exploration bonus (Sutton, 1990), which is conceptually similar to the optimism in the face of uncertainty principle in Bandits, see Section 2.7.1.1. This concept has also been called curiosity (Schmidhuber, 1991) and is related to intrinsic motivation (Chentanez et al., 2005). For example, (Tang et al., 2017) map the continuous domain to a discrete grid and provide a reward bonus during reinforcement learning when states with low counts are visited. Alternatively, Pathak et al. (2017) propose to use a dynamics-based prediction error as a bonus reward signal, which can be viewed as a deterministic estimate of uncertainty.

2.7.2.2 Value Uncertainty

Rather than using uncertain rewards, it is often more natural to consider uncertainty about the value of a state. In discrete state-action spaces, one of the most-well known algorithms

is Q-learning (Watkins and Dayan, 1992), a method that estimates values of state-action pairs and describes a policy directly through the corresponding Q-function. In this setting, convergence can be shown for a variant of Q-learning that initializes the estimates of the Q-value optimistically (Even-Dar and Mansour, 2002). Tighter bounds can be obtained by acting optimistically with respect to an estimate of the Q-function directly (Zanette and Brunskill, 2019).

As an alternative to optimism, Osband, Van Roy, and Wen (2014) provide regret bounds for Thompson sampling in the discrete setting with linear function approximation. At each step, their method samples a value function from a linear, statistical model and acts greedily with respect to it. Osband, Blundell, et al. (2016) and Osband, Van Roy, Daniel Russo, et al. (2017) extend this method heuristically to nonlinear function approximators by learning and sampling from an ensemble of value functions, see Section 2.6.3. As an alternative to the ensemble method, Deisenroth, C. Rasmussen, et al. (2009) and Engel et al. (2005) use a Gaussian process to approximate the value function.

Beyond epistemic uncertainty, Nikolov et al. (2019) additionally consider the aleatoric uncertainty of the value. They use an information-inspired exploration scheme that is similar to the one in Section 2.7.1.1, which encourages to visit states where the structural uncertainty is large, but the variance of the return is low.

2.7.2.3 Model Uncertainty

Lastly, one can consider uncertainty about the transition function f in (2.1). This is again best understood in discrete Markov Decision Processes. The E^3 algorithm by Kearns and S. Singh (2002) provides convergence guarantees for an exploration strategy that performs exploration only when it encounters a state that it has not visited before. Brafman and Tennenholtz (2003) extend this result and exploit an optimistic initialization of the value function in order to drive exploration, while Jaksch et al. (2010) provide regret bounds for an algorithm that acts with respect to an optimistic Markov Decision Process. Regret bounds also exist for Thompson sampling. These methods maintain a distribution over Markov Decision Processes and act optimally with respect to a sampled model (Osband, Dan Russo, et al., 2013; A. Gopalan and Mannor, 2015).

In continuous state spaces, exploration is best understood for linear models. Dean, Mania, et al. (2018) show that even random exploration can achieve sublinear regret. Regret bounds for optimistic algorithms (Abbasi-Yadkori and Szepesvári, 2011; Faradonbeh et al.,

2017; Ibrahimi et al., 2012) and Thompson sampling (Osband and Van Roy, 2016) have also been investigated.

In nonlinear systems, uncertainty-based exploration strategies have mostly been used heuristically. One of the first practical methods is PILCO by Deisenroth and Carl E. Rasmussen (2011), which considers optimizing parameterized policies over finite-horizon trajectories. It uses a Gaussian process dynamics model together with approximate uncertainty propagation scheme based on moment-matching (Girard et al., 2002) in order to construct a Gaussian approximation of the posterior distribution over trajectories. It then selects policy parameters that optimize the expected performance over finite horizons. Gal et al. (2016) propose to use a sampling-based representation over the posterior instead, which they then approximate with a Gaussian to compute gradients. Chatzilygeroudis et al. (2017) instead propose to use trajectory samples together with a gradient-free optimization. While optimizing for expected performance is not an exploration strategy in its own right, they often select specific reward functions in order to drive exploration.

Instead of optimizing over parametric policies, Cao et al. (2017) and Kamthe and Deisenroth (2018) use model predictive control to optimize the expected performance at every time step. (Chua et al., 2018) propose to replace the Gaussian process model by normal distributions that are parameterized by an ensemble of neural networks. This is extended by Malik et al. (2019), who show that better results can be obtained by calibrating the neural networks to yield reliable uncertainty estimates.

To use the uncertainty information for more efficient exploration, Bechtle et al. (2019) propose to use an optimal control method that encourages visiting uncertain states. As an alternative, (Moldovan, Levine, et al., 2015; Xie et al., 2016) propose to act optimistically with respect to the dynamics model.

In terms of theory for continuous domains, Osband and Van Roy (2014) provided regret bounds for both Optimistic exploration and Thompson sampling in parametric models. Recently, these results were extended to nonparametric models by Chowdhury and A. Gopalan (2019).

2.8 Safe Reinforcement Learning

Physical systems that operate in the real world typically have to adhere to rigorous safety constraints before they can be deployed. For a known model of the environment, we discuss

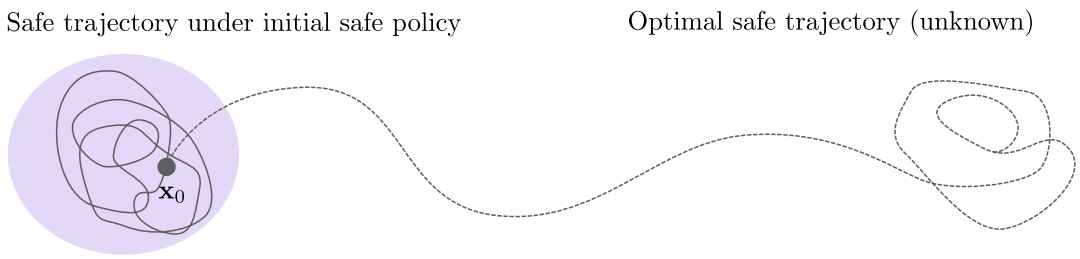


Figure 2.4: Illustration of the safe reinforcement learning process. While we have to assume access to sufficient prior knowledge in terms of an initially safe policy, this policy will generally be highly suboptimal. For example, in the example the policy is safe to use within the blue shaded region and the trajectory induced by the safe policy stays close to $\mathbf{x}_0$. However, the optimal safe trajectory in terms of cost is the dashed line. To achieve this trajectory, the system must safely learn about the system until it gathers sufficient information to determine the safety of the high-performance trajectory.

in Section 2.4 how we can define and quantify safety and reviewed methods that adhere to these safety constraints. For reinforcement learning to be applicable to real-world systems, we must provide the same kind of safety guarantees *during the learning process*. However, in reinforcement learning we do *not know the model* in advance and have to safely learn about it online, see Section 2.5. This means that we *do not know which control actions can be safely applied to the system*. As a result, guaranteeing safety in reinforcement learning is significantly more challenging than in control with a known system model. In particular, if we cannot make any assumptions about the system, the very first control input that we apply to the system may violate the safety constraints.

Thus, safe reinforcement learning requires us to have some prior information about the system. The minimal assumption that we require for safe learning is that of a safe starting point. This means that there must be sufficient initial knowledge about the system to control it without violating the safety constraints. For example, this may come in the form of an initial policy that is known to be safe or prior, local knowledge about the model of the system that allows us to safely control it. Note that this initial policy has no other requirements than safety. In particular, we do not expect it to be safe for any initial state in $\mathcal{X}$. Moreover, this policy can have arbitrarily poor performance as long as it is safe starting from $\mathbf{x}_0$. For example, the control objective may be to drive a car as quickly as possible subject to the safety constraint on staying on the track. A safe initial control policy for this system could be to stand still or drive very slowly in the center of the lane.

It is important to note that assuming the existence of a safe starting point is not the same as solving the safe reinforcement learning problem by assumption. In particular the initial policy is typically only certified for safety within a small region of the state space such as the blue shaded region in Figure 2.4. As a result, the data covered by the trajectories induced by the safe policy only cover a small part of the space (grey line). In particular, the optimal safe trajectory (grey dashed line) for the true system is typically *not known to be safe a priori*. As a consequence, safe reinforcement learning faces a challenging exploration problem where it has to safely gather data until it learns about the safety of the optimal trajectory. Notably, safely learning about the safety of the optimal trajectory typically requires learning about other trajectories that are known to be suboptimal. For example, to safely learn about the safety of the optimal trajectory in Figure 2.4, we may first have to learn how to return the system to the blue shaded region. This is not required for exploration in reinforcement learning, but it is required for exploration in safe reinforcement learning. As a consequence, applying safe variants of existing exploration schemes from Section 2.7 do not enjoy the same kind of performance guarantees as in the unsafe setting. Thus safe exploration for reinforcement learning is generally *significantly more challenging* than unsafe reinforcement learning.

In the following we provide an overview of safe reinforcement learning methods. Alternative reviews with a different focus can be found in (Pecka and Svoboda, 2014; García and Fernández, 2015).

2.8.1 Safe Model-free Reinforcement Learning

In this section, we review two model-free methods for safe reinforcement learning.

2.8.1.1 High-confidence Policy Improvement

A key requirement of safe reinforcement learning is to evaluate whether a new policy obtained from, e.g., an update to the policy parameters, is safe to apply to the system. The requirement for guaranteed safety during the learning process means that we must evaluate the safety of the new policy without trying it out on the system.

This turns out to be tractable when we care about the expected safety in a constrained Markov Decision Process, see Section 2.4.3. Achiam et al. (2017) and Chow et al. (2019) provide error bounds on the expected performance and safety of the new policy as a

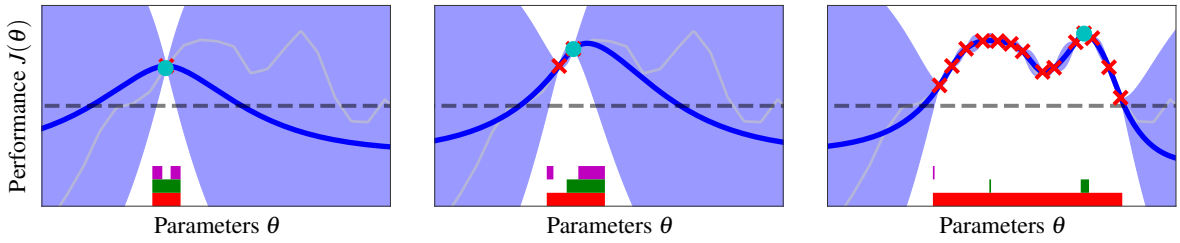
function of the average Kullback-Leibler divergence of the old and new policy. This is then used to guarantee that the updated policy is safe. Moreover, Thomas et al. (2015b) and Thomas et al. (2015a) show how to build confidence intervals on these estimates in the finite-sample case, which can build confidence about the safety of a new policy based on a finite number of episodic trajectories with the old policy. These confidence intervals can be further improved when the original policy is diverse, which leads to tighter error bounds (Cohen et al., 2018).

2.8.1.2 Safe Bayesian Optimization

The concept of constraints has also been incorporated into Bayesian optimization, see Section 2.7.1. Gelbart et al. (2014) introduce an algorithm to optimize an unknown function subject to an unknown constraint. However, this constraint is not considered to be safety-critical; that is, the algorithm is allowed to evaluate unsafe parameters. Schreiter et al. (2015) propose a method to find find a safe subset of the parameters without violating safety constraints, while Sui et al. (2015) present SafeOpt, a similar algorithm to safely optimize an objective function subject to a constraint on the minimum performance of any parameter that is evaluated.

Instead of optimizing the underlying performance function $J(\boldsymbol{\theta})$ globally, SafeOpt restricts itself to a safe set of parameters that achieve a certain minimum performance with high probability. This safe set is not known initially, but is estimated after each function evaluation. This can be seen in Figure 2.5a: The Gaussian process model over the true function provides uncertainty estimates of the performance $J(\boldsymbol{\theta})$ (blue shaded). Only parameters where the lower confidence interval is above the threshold (red set) are known to be safe and can thus be safely evaluated.

In this setting, the challenge is to find an appropriate evaluation strategy similar to (2.34), which at each iteration n not only aims to find the global maximum within the currently known safe set (exploitation), but also aims to increase the set of controllers that are known to be safe (exploration). To this end, SafeOpt keeps track of a set of plausible maximizers (green set) and parameters on the boundary of the safe set (purple set) that could potentially expand the current safe set. It trades off between selecting parameters from these two sets by choosing the parameters with with maximum Gaussian process posterior variance. While this procedure does not provide regret bounds, it automatically and safely trades off between exploration and exploitation. As a result, the method



(a) Initial, safe parameters. (b) Safe exploration. (c) Safe maximum found.

Figure 2.5: Illustration of the SafeOpt algorithm. Starting from safe, initial parameters in Figure 2.5a, it only evaluates safe parameters when $J(\boldsymbol{\theta})$ is known to be above the safety threshold (dashed line) with high probability (confidence intervals shaded in blue). It trades off between exploration on the boundary of the safe set (purple) and exploitation of the current plausible maximizers (green set) by selecting the most uncertain parameters. This way it safely explores in Figure 2.5b and eventually finds a good estimate of the safely reachable optimum (cyan circle) in Figure 2.5c.

eventually finds the safely reachable optimum of the unknown function in Figure 2.5c.

2.8.2 Safe Model-based Reinforcement Learning

In safe model-based reinforcement learning we learn a model of the system f in (2.1). However, since we do *not know the dynamics* in advance, this model has large errors at the beginning of the learning process. Thus we must guarantee safety during exploration despite these model errors.

2.8.2.1 Unknown Constraints

A simpler setting than the full safe reinforcement learning problem is to consider a *known* model of the environment, but to be uncertain about the safety constraints. For example, a robot might know its model, but be uncertain about the location of obstacles in the environment. This setting is strictly less general than the full safe reinforcement learning problem, since the uncertainty about the constraints can also be modeled by a dynamic system over an extended state-space, where the additional, static states encode the location of the constraints/obstacles.

This setting has been especially studied in discrete state-action spaces. Formally, a state is defined as safe whenever the constraint $c_x \geq 0$ is fulfilled, but we do not know the function c_x in advance and can only obtain noisy measurements of $c_x(\mathbf{x})$ by visiting the state $\mathbf{x}$. This problem was first studied by Moldovan and Abbeel (2012), who propose an exploration scheme that at each step solves for a policy that optimizes the amount of information gained about c_x subject to the constraint that the agent does not ever visits an unsafe state with high probability. Notably, the probability of failure is computed over an infinite horizon, so that the resulting exploration policy is safe in the *long term*. While the method guarantees safety, it does not provide exploration guarantees. Safe exploration guarantees were first provided by Turchetta et al. (2016) for an algorithm that operates in an environment without transition noise and aims to maximally reduce uncertainty within the current safe set (safe system identification). Without observation noise, Bıyık et al. (2019) provide stronger exploration guarantees. Lastly, Wachi et al. (2018) additionally consider exploration for a reinforcement learning objective by optimizing for performance. They encourage exploration of states where the difference in value between the safety-constrained Markov Decision Process and an optimistically safe Markov Decision Process is large.

In continuous state spaces, Sadigh and Kapoor (2016) propose a model predictive control scheme that does not violate the *a priori* unknown, but learned safety constraints with high probability. However, they do not actively learn about the constraints.

2.8.2.2 Robust Control

Guaranteeing safety with respect to model errors has traditionally be considered by the field of robust control in both discrete (Wiesemann et al., 2012) and continuous systems (Zhou and Doyle, 1998). Robust control methods characterize model uncertainty through an *a prior fixed* set of system models. They then provide safety and performance guarantees with respect to all models within this set. Computing robust controllers for linear systems is well-understood and the controllers can be computed by solving semi-definite programs (Zhou and Doyle, 1998). These methods can also be extended to model predictive control (Bemporad and Morari, 1999), where safety is defined through recursive feasibility of the optimization problem together with robust constraint satisfaction and stability. However, robust control methods tend to be conservative and achieve sub-optimal performance, since they do not update the model as more data becomes available.

2.8.2.3 Safe Learning Control

Robust control methods can be used directly to guarantee safety. For example, Aswani et al. (2013) and Wabersich and Zeilinger (2018) compute tubes around trajectories of a known nominal linear model, which accounts for all disturbances. While the former approach requires disturbances to be bounded in an *a priori* specified polytope, the latter relies on sampling to approximate model errors. A different approach by Jin and Lavaei (2018) provides guarantees for smooth policies when the nonlinearity is bounded. While these methods, provide safety guarantees and increase performance over time, they generally consider a fixed model error, so that the resulting controller necessarily stays conservative.

When the model is updated online, safety can be guaranteed by enforcing robustness online. Berkenkamp and A. P. Schoellig (2015) use a learned Gaussian process model to online compute robustly stabilizing linear controllers that get less conservative over time. For general systems, Vinogradska et al. (2017) uses the Gaussian process model together with quadrature to forward-integrate the uncertainty over the state space at every step. Since these schemes allow the quantification of errors, they provide reliable uncertainty estimates under the Gaussian process prior assumption. This enables the verification of stability and constraints for the learned system.

Beyond stability, safety can be guaranteed by computing a reachable set together with a safe policy. The safety constraints can be enforced by switching to this safe policy on the boundary of the safe set, so that the system remains safe (Fisac et al., 2018). This idea of switching between control policies is similar to that proposed by Alshiekh et al. (2018) for discrete spaces. This switching also forms the basis for early approaches. For example, Hans et al. (2008) assume access to a backup policy and Garcia and Fernandez (2012) define a heuristic to quantify the distance of unsafe states.

While these methods, explicitly compute reachable sets, terminal set constraints in model predictive control can be used to implicitly define the safety constraint. For example, robust constraint satisfaction and closed-loop stability for learning-based model predictive control with nominal linear models and state-dependent uncertainty is shown in Soloperto et al., 2018. Instead of updating the model of the system, (Ugo Rosolia and Francesco Borrelli, 2019) instead use samples to grow a safe set in a robust model predictive control setting over time. For learned, nonlinear systems, safety constraints can be satisfied over a finite horizon by predicting trajectories a finite number of steps and ensuring that the constraint is not violated over this time horizon. For tractability, Ostafew et al. (2016), Hewing et al.

(2017), and Polymenakos et al. (2019) use approximate uncertainty propagation schemes and enforce safety constraints with high probability. Dalal et al. (2018) take this to an extreme and only consider guaranteeing safety over one step based on a nominal model. These schemes do not provide recursive safety guarantees.

While these methods provide *safety guarantees* for learned models with specific structures, they do not consider *safe exploration*, where we actively and safely learn about the system in order to solve a task.

2.8.3 Safe Exploration

In continuous control, exploration guarantees for safe exploration are only understood in the linear setting with rewards that are quadratic in states and control actions. Dean, Tu, et al. (2018) and Lu et al. (2017) provide strong exploration, stability, and constraint satisfaction guarantees. They combine the error bounds from a reliable statistical model with the system level synthesis framework by Wang et al. (2019) to compute robust controllers. As in the setting without constraints in Section 2.7.2.3 they use Gaussian noise to drive the exploration and provide regret bounds for the control performance.

3

Safe Direct Policy Optimization

The results in this chapter have been previously published in (Berkenkamp, A. P. Schoellig, and Krause, 2016) and (Berkenkamp, Krause, et al., 2016).

The SafeOpt algorithm in Section 2.8.1.2 only considers the scalar case, where the safety constraint is encoded in terms of a lower bound on $J(\boldsymbol{\theta})$. However, in practice it is often desirable to encode safety constraints separately from the objective. In this chapter, we extend SafeOpt to multiple constraints and to contextual Bayesian optimization as in Section 2.7.1.1. Moreover, we evaluate the algorithm to solve several robotic control tasks.

3.1 Problem Statement

To solve a reinforcement learning task, we assume that we are given a control algorithm that is used to accomplish a certain task with a robot. In general, this algorithm is arbitrary and may contain several components including vision, state estimation, planning, and control laws. The algorithm is parameterized by parameters $\boldsymbol{\theta} \in \mathcal{D}$ in some specified, domain $\mathcal{D} \subseteq \mathbb{R}^d$.

The goal is to find the parameters within $\mathcal{D}$ that maximize a given, scalar performance measure, $J(\boldsymbol{\theta})$. For example, this performance measure may represent the negative tracking error of a robot (Berkenkamp, A. P. Schoellig, and Krause, 2016), the average walking speed of a bipedal robot (Calandra, N. Gopalan, et al., 2014), or any other quantity

that can be computed over a finite time horizon. We can only evaluate the performance measure for any parameter set $\boldsymbol{\theta}$ on finite-time trajectories from experiments on the real robot. The functional dependence of $J(\boldsymbol{\theta})$ on $\boldsymbol{\theta}$ is not known *a priori*. In the following, we write the performance measure as a function of the parameters $\boldsymbol{\theta}$, $J\colon \mathcal{D} \to \mathbb{R}$, even though measuring performance requires an experiment on the physical robot and typically depends on a trajectory of states, control inputs, and external signals. In our setting this function is the control performance from Section 2.3.

We assume that the underlying system is safety-critical; that is, there are constraints that the system must respect when evaluating parameters. Similarly to the performance measure, $J(\boldsymbol{\theta})$, these constraints can represent any quantity and may depend on states, inputs, or even environment variables. There are m safety constraints of the form $c_i(\boldsymbol{\theta}) \geq 0$, $c_i\colon \mathcal{D} \to \mathbb{R}$, $i = 1 \dots m$, which together define the safety conditions. This is without loss of generality, since any constraint function can be shifted by a constant in order to obtain this form. The functions c_i are unknown *a priori* but can be estimated through (typically noisy) experiments for a given parameter set $\boldsymbol{\theta}$. For example, in order to encode a state constraint on an obstacle for a robot, the safety function $c_i(\boldsymbol{\theta})$ can return the smallest distance to the obstacle along a trajectory of states when using algorithm parameters $\boldsymbol{\theta}$. Note that if the functions were known in advance, we could simply exclude unsafe parameters from the set $\mathcal{D}$.

The overall optimization problem can be written as

$$\max_{\boldsymbol{\theta} \in \mathcal{D}} J(\boldsymbol{\theta}) \quad \text{subject to} \quad c_i(\boldsymbol{\theta}) \geq 0 \, \forall \, i = 1, \dots, m. \tag{3.1}$$

The goal is to iteratively find the global maximum of this constrained optimization problem by, at each iteration n, selecting parameters $\boldsymbol{\theta}_n$ and evaluating (up to noise) the corresponding function values $J(\boldsymbol{\theta}_n)$ and $c_i(\boldsymbol{\theta}_n)$ until the optimal parameters are found. In particular, since the constraints define the safety of the underlying system, only parameters that are inside the feasible region of (3.1) are allowed to be evaluated; that is, only parameters that fulfill these safety requirements on the real system.

Since the functions $J(\boldsymbol{\theta})$ and c_i in (3.1) are unknown *a priori*, it is not generally possible to solve the corresponding optimization problem without violating the constraints. The first problem is that we do not know how to select a first, safe parameter to evaluate. In the following, we assume that an initial safe set of parameters $\mathcal{S}_0 \subseteq \mathcal{D}$ is known for which the constraints are fulfilled. These serve as a starting point for the exploration of the safe

region in (3.1). In robotics, safe initial parameters with poor performance can often be obtained from a simulation or domain knowledge.

Secondly, in order to safely explore the parameter space beyond $\mathcal{S}_0$, we must be able to infer whether parameters $\boldsymbol{\theta}$ that we have not evaluated yet are safe to use on the real system. To this end, we make regularity assumptions about the functions $J(\boldsymbol{\theta})$ and c_i in (3.1). In particular, in Section 3.2 we extend the assumption of bounded RKHS norm from Section 2.6.2 to be applicable to multiple functions. Using these continuity properties, we are able to generalize safety beyond the initial, safe parameters $\mathcal{S}_0$. Given the model assumptions, we require that the safety constraints hold with high probability over the entire sequence of experiments.

As a consequence of the safety requirements, it is not generally possible to find the global optimum of (3.1). Instead we aim to find the optimum in the part of the feasible region that is safely reachable from $\mathcal{S}_0$. We formalize this precisely in Section 3.3.

Lastly, whenever we evaluate parameters on the real system, we only obtain noisy estimates of both the performance function and the constraints, since both depend on noisy sensor data along trajectories. That is, for each parameter $\boldsymbol{\theta}$ the we evaluate, we obtain measurements $J(\boldsymbol{\theta}) + \omega_0$ and $c_i(\boldsymbol{\theta}) + \omega_i$, where ω_i, $i = 0, \dots, q$, is zero-mean, σ-sub-Gaussian noise. In general, the noise variables may be correlated, but we do not consider this case in our theoretical analysis in Section 3.3.2. We only want to evaluate parameters where all safety constraints are fulfilled, so that $c_i(\boldsymbol{\theta}_n) \geq 0$ for all $i \in \{1, \dots, q\}$ and $n \geq 1$.

3.2 Multi-output RKHS Functions

In Section 2.6, we considered a scalar RKHS function $f'(\mathbf{a})\colon \mathcal{A} \to \mathbb{R}$. In the following, we want to build confidence intervals that hold jointly for all functions $J(\theta)$ and c_i in Section 3.1. To this end, we consider learning a vector-valued function $f'(\boldsymbol{\theta}) = (J(\boldsymbol{\theta}), c_1(\boldsymbol{\theta}), \dots, c_m(\boldsymbol{\theta}))$. In the Gaussian process literature, vector-valued functions can be learned using matrix-valued kernel functions, where the off-diagonal entries in the matrix model correlation between different output dimensions (Álvarez et al., 2012). In particular, for the vector-

valued function f' we can use the kernel

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \begin{bmatrix} k_J(\boldsymbol{\theta}, \boldsymbol{\theta}) & k_{J,c_1}(\boldsymbol{\theta}, \boldsymbol{\theta}) & \dots & k_{J,c_m}(\boldsymbol{\theta}, \boldsymbol{\theta}) \\ k_{c_1,J}(\boldsymbol{\theta}, \boldsymbol{\theta}) & k_{c_1}(\boldsymbol{\theta}, \boldsymbol{\theta}) & \dots & k_{c_1,c_m}(\boldsymbol{\theta}, \boldsymbol{\theta}) \\ \vdots & \vdots & \vdots & \vdots \\ k_{c_m,J}(\boldsymbol{\theta}, \boldsymbol{\theta}) & k_{c_m,c_1}(\boldsymbol{\theta}, \boldsymbol{\theta}) & \dots & k_{c_m}(\boldsymbol{\theta}, \boldsymbol{\theta}) \end{bmatrix}. \tag{3.2}$$

As in Section 2.6.1, this kernel models the functions $J(\boldsymbol{\theta})$ and $c_i(\boldsymbol{\theta})$ with independent kernels k_J and k_{c_i} respectively. Additionally, it introduces covariance kernels k_{J,c_i} between all the function outputs, which model similarities between the different output dimensions of the function.

To apply the analysis of Lemma 1 to a vector-valued function f', we represent $f'(\boldsymbol{\theta})$ with an equivalent scalar-valued function that still accounts for the covariance. In particular, we consider the surrogate, scalar function

$$f'(\boldsymbol{\theta}, i) = [f'(\boldsymbol{\theta})]_i = \begin{cases} J(\boldsymbol{\theta}) & \text{if } i = 0 \\ c_i(\boldsymbol{\theta}) & \text{if } i \in \mathcal{I}_c, \end{cases} \tag{3.3}$$

which returns either the performance function or the individual safety constraints depending on the additional input $i \in \mathcal{I}$ with $\mathcal{I} = \{0, \dots, m\}$, where $\mathcal{I}_c = \{1, \dots, m\} \subset \mathcal{I}$ are the indices belonging to the constraints. The function $f'(\cdot, \cdot)$ is a single-output function as in Section 2.6.2 and can be modeled as a Gaussian process with scalar output over the extended parameter space $\mathcal{D} \times \mathcal{I}$. In particular, this function class has the same representation power as the matrix kernel in (3.2) if we define the kernel function for the scalar function as

$$k((\boldsymbol{\theta}, i), (\boldsymbol{\theta}', j)) = \begin{bmatrix} \delta_{i,0} & \delta_{i,1} & \dots & \delta_{i,m} \end{bmatrix} k(\boldsymbol{\theta}, \boldsymbol{\theta}) \begin{bmatrix} \delta_{j,0} \\ \delta_{j,1} \\ \vdots \\ \delta_{j,m} \end{bmatrix}, \tag{3.4}$$

where δ_{ij} is the Kronecker delta. Intuitively, when $i = j$ we obtain the kernel function that corresponds to the particular function value. However, when $i \neq j$ the kernel returns the covariance kernel between the corresponding function outputs.

Thus by extending the training data with the extra parameter i to index the function output, we can use the normal Gaussian process framework to predict function value and uncertainties of $f'(\boldsymbol{\theta}, i)$. However, in contrast to the scalar case in Lemma 1, at

every iteration n, we obtain $|\mathcal{I}| = m + 1$ measurements; one for each function. For ease of notation, we continue to write μ_n and σ_n, even though we have obtained $n \cdot (m + 1)$ measurements at locations $\mathcal{A}_n \times \mathcal{I}$ in the extended parameter space.

3.2.1 Confidence Intervals

In order to provide guarantees for safety, the confidence intervals in (3.9) must to hold for all iterations and functions. In the following, we assume that the surrogate function $f'(\boldsymbol{\theta}, i)$ has bounded norm in the RKHS corresponding to the kernel (3.4). Thus, the corresponding RKHS includes functions of the form $f'(\boldsymbol{\theta}, i) = \sum_j \alpha_j k((\boldsymbol{\theta}, i), (\boldsymbol{\theta}_j, i_j))$ with $\alpha_i \in \mathbb{R}$ and representer points $(\boldsymbol{\theta}_j, i_j) \in \mathcal{D} \times \mathcal{I}$. As in Section 2.6.2, the bounded norm property implies that the coefficients α_j decay sufficiently fast as j increases.

The following Lemma allows us to choose a scaling factor β_n for (3.9), so that the true function is contained in the confidence intervals with a desired probability for all iterations.

Lemma 2 (based on Chowdhury and A. Gopalan (2017)). *Assume that $f'(\boldsymbol{\theta}, i) = [f'(\boldsymbol{\theta})]_i$ has RKHS norm bounded by B and that measurements are corrupted by σ-sub-Gaussian noise. Let $\mathcal{A}_n = \mathcal{D}_n \times \mathcal{I}$ denote the measurements obtained up to iteration n. If $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f') + 1 + \ln(1/\delta)}$, then the following holds for all parameters $\boldsymbol{\theta} \in \mathcal{D}$, function indices $i \in \mathcal{I}$, and iterations $n \geq 0$ jointly with probability at least $1 - \delta$:*

$$\left| f'(\boldsymbol{\theta}, i) - \mu_n(\boldsymbol{\theta}, i) \right| \leq \beta_n \sigma_n(\boldsymbol{\theta}, i) \tag{3.5}$$

As before, the scaling factor β_n in Lemma 2 depends on the mutual information. For consistency, we define this quantity with respect to scalar observations,

$$\gamma_n = \max_{\mathcal{A} \subseteq \mathcal{D} \times \mathcal{I}, |\mathcal{D}| \leq n} \mathrm{I}(\mathbf{y}_{\mathcal{A}}; f'). \tag{3.6}$$

As a consequence, we have $\mathrm{I}(\mathbf{y}_{\mathcal{A}}; f') \leq \gamma_{|\mathcal{A}|}$, where $|\mathcal{A}|$ returns the number of observations in $\mathcal{A}$.

Importantly, using this surrogate function enables us to lift theoretical results of Sui et al., 2015 to the more general case with multiple constraints and provide theoretical guarantees for our algorithm in Section 3.3.2.

3.3 SafeOpt-MC (Multiple Constraints)

In this section, we use the confidence intervals from Lemma 2 and introduce the SafeOpt-MC algorithm for multiple constraints. The goal of the algorithm is to solve (3.1) by evaluating different parameters from the domain $\mathcal{D}$ without violating the safety constraints. To this end, any algorithm has to consider two important properties:

(i) Expanding the region of the optimization problem that is known to be feasible or safe as much as possible without violating the constraints,

(ii) Finding the optimal parameters within the current safe set.

For objective *i)*, we need quantify the size of the safe set. To do this in a tractable manner, we focus on finite sets $\mathcal{D}$ in the following. Moreover, while Lemma 2 already exploits the continuity assumptions in order to construct reliable confidence intervals, the generalization behavior is difficult to analyze. Instead, we use the continuity properties of the RKHS function in order to quantify the generalization behavior. In particular, we assume that $J(\boldsymbol{\theta})$ and $c_i(\boldsymbol{\theta})$ are L-Lipschitz continuous with respect to some metric $d(\cdot, \cdot)$. For example, the kernel metric (2.33) can be used.

Lastly, since we only observe noisy estimates of both the performance function and the constraints, we cannot expect to find the entire safe region encoded by the constraints within a finite number of evaluations. Instead, we follow Sui et al., 2015 and consider learning the safety constraint up to some accuracy ϵ. This assumption is equivalent to a minimum slack of ϵ on the constraints in (3.1).

Baseline As mentioned in Section 3.1, we assume that we have access to initial, safe parameters $\mathcal{S}_0 \subseteq \mathcal{D}$, for which we know that the safety constraints are satisfied *a priori*. Starting from these initial parameters, we ask what the best that any safe optimization algorithm could hope to achieve is. In particular, if we knew the safety constraint functions $c_i(\cdot)$ up to ϵ accuracy within some safe set of parameters S, we could exploit the continuity properties to expand the safe set to

$$R_\epsilon(S) \coloneqq S \cup \bigcap_{i \in \mathcal{I}_c} \left\{ \boldsymbol{\theta} \in \mathcal{D} \,|\, \exists\, \boldsymbol{\theta}' \in S : \, c_i(\boldsymbol{\theta}') - \epsilon - L d(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq 0 \right\}, \tag{3.7}$$

where $R_\epsilon(S)$ represents the number of parameters that can be classified as safe given that we know the safety constraints c_i up to ϵ-error inside S and exploiting the Lipschitz continuity

to generalize to new parameters outside of S. The baseline that we compare against is the limit of repeatedly applying this operator on $\mathcal{S}_0$; that is, with $R_\epsilon^n(S) = R_\epsilon(R_\epsilon^{n-1}(S))$ and $R_\epsilon^1(S) = R_\epsilon(S)$ the baseline is $\bar{R}_\epsilon(\mathcal{S}_0) \coloneqq \lim_{n\to\infty} R_\epsilon^n(\mathcal{S}_0)$. This set contains all the parameters in $\mathcal{D}$ that could be classified as safe starting from $\mathcal{S}_0$ if we knew the function up to ϵ error. This set does not include all the parameters that potentially fulfill the constraints in (3.1), but is the best we can do without violating the safety constraints. Hence the optimal value that we compare against is not the one in (3.1), but

$$J_\epsilon^* = \max_{\boldsymbol{\theta} \in \bar{R}_\epsilon(\mathcal{S}_0)} J(\boldsymbol{\theta}), \tag{3.8}$$

which is the maximum performance value over the set that we could hope to classify as safe starting from the initial safe set, $\mathcal{S}_0$.

3.3.1 The Algorithm

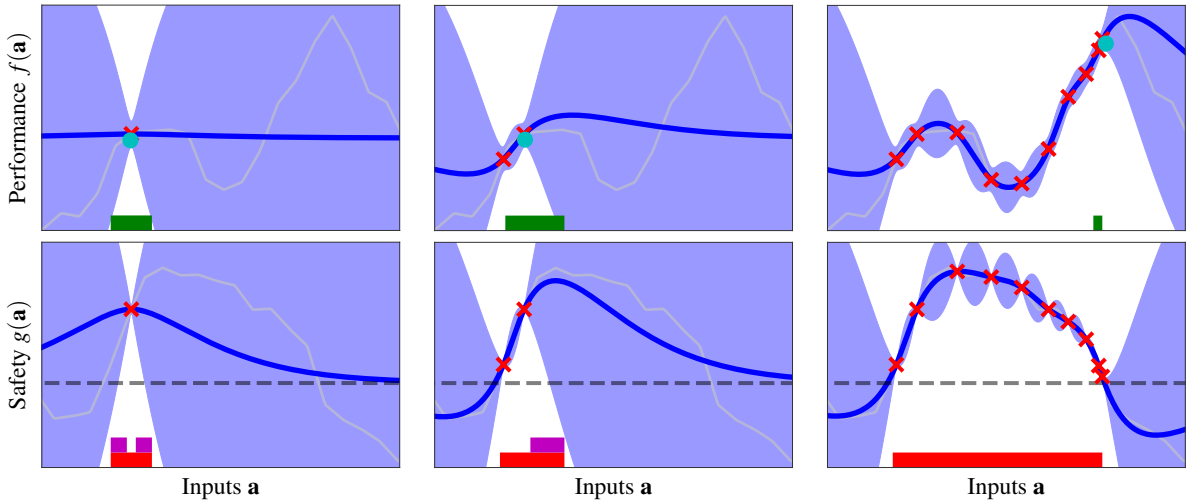
In this section, we present the SafeOpt-MC algorithm that guarantees convergence to the previously set baseline. The most critical aspect of the algorithm is safety. However, once safety is ensured, the second challenge is to find an evaluation criterion that enables trading off between exploration, trying to further expand the current estimate of the safe set, and exploitation, trying to improving the estimate of the best parameters within the current set.

To ensure safety, we construct confidence intervals that contain the true functions $J(\boldsymbol{\theta})$ and c_i with high probability. In particular, we use the posterior Gaussian process estimate given the data observed so far. The confidence intervals for the surrogate function in (3.3) are defined as

$$\mathcal{Q}_n(\boldsymbol{\theta}, i) \coloneqq \left[\mu_{n-1}(\boldsymbol{\theta}, i) \pm \beta_n^{1/2} \sigma_{n-1}(\boldsymbol{\theta}, i)\right], \tag{3.9}$$

where β_n is chose as in Lemma 2. This set contains all possible function values between the lower and upper confidence interval based on the Gaussian process posterior. This confidence interval contains the true function value with probability at least $(1 - \delta)$.

Rather than defining the lower and upper bounds based on (3.9), the following analysis requires that consecutive estimates of the lower and upper bounds are contained within each other. This assumption ensures that the safe set does not shrink from one iteration to the next, which we require to prove our results. We relax this assumption in Section 3.5. We define the contained set at iteration n as $\mathcal{C}_n(\boldsymbol{\theta}, i) = C_{n-1}(\boldsymbol{\theta}, i) \cap \mathcal{Q}_n(\boldsymbol{\theta}, i)$,



(a) Initial, safe parameters.

(b) Safe exploration.

(c) After 10 evaluations: safe maximum found.

Figure 3.1: Optimization with the SafeOpt-MC algorithm after 1, 2 and 10 parameter evaluations. Based on the mean estimate (blue) and the 2σ confidence interval (light blue), the algorithm selects evaluation points for which $c(\boldsymbol{\theta}) \geq 0$ (black dashed) from the safe set $\mathcal{S}_n$ (red), which are either potential maximizers $\mathcal{M}_n$ (green) or expanders $\mathcal{G}_n$ (magenta). It then learns about the function by drawing noisy samples from the unknown, underlying function (light gray). This way, we expand the safe region (red) as much as possible and, simultaneously, find the global optimum of the unknown function (3.16) (cyan circle).

where $C_0(\boldsymbol{\theta}, i)$ is $[0, \infty]$ for all $\boldsymbol{\theta} \in \mathcal{S}_0$ and $\mathbb{R}$ otherwise. This ensures that parameters in the initial safe set $\mathcal{S}_0$ remain safe according to the Gaussian process model after additional observations. The lower and upper bounds on this set are defined as $l_n^i(\boldsymbol{\theta}) \coloneqq \min \mathcal{C}_n(\boldsymbol{\theta}, i)$ and $u_n^i(\boldsymbol{\theta}) \coloneqq \max \mathcal{C}_n(\boldsymbol{\theta}, i)$, respectively. For notational clarity, we write $l_n^J(\boldsymbol{\theta}) \coloneqq l_n^0(\boldsymbol{\theta})$ and $u_n^J(\boldsymbol{\theta}) \coloneqq u_n^0(\boldsymbol{\theta})$ for the performance bounds.

Based on these confidence intervals for the function values and a current safe set S_{n-1}, we can enlarge the safe set using the Lipschitz continuity properties,

$$\mathcal{S}_n = \bigcap_{i \in \mathcal{I}_c} \bigcup_{\boldsymbol{\theta} \in S_{n-1}} \left\{ \boldsymbol{\theta}' \in \mathcal{D} \,|\, l_n^i(\boldsymbol{\theta}) - L d(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq 0 \right\}. \tag{3.10}$$

The set $\mathcal{S}_n$ contains all points in S_{n-1}, as well as all additional parameters that fulfill the safety constraints given the Gaussian process confidence intervals and the Lipschitz constant.

With the set of safe parameters defined, the last remaining challenge is to trade off between exploration and exploitation. One could, similar to Schreiter et al., 2015, simply select the most uncertain element over the entire set. However, this approach is not sample-efficient, since it involves learning about the entire function rather than restricting evaluations to the relevant parameters. To avoid this, we first define subsets of $\mathcal{S}_n$ that correspond to parameters that could either improve the estimate of the maximum or could expand the safe set. The set of potential maximizers is defined as

$$\mathcal{M}_n \coloneqq \left\{ \boldsymbol{\theta} \in \mathcal{S}_n \,|\, u_n^J(\boldsymbol{\theta}) \geq \max_{\boldsymbol{\theta}' \in \mathcal{S}_n} l_n^J(\boldsymbol{\theta}') \right\}, \tag{3.11}$$

which contains all parameters for which the upper bound of the current performance estimate is above the best lower bound. The parameters in $\mathcal{M}_n$ are candidates for the optimum, since they could obtain performance values above the current conservative estimate of the optimal performance.

Similarly, an optimistic set of parameters that could potentially enlarge the safe set is

$$\mathcal{G}_n \coloneqq \{ \boldsymbol{\theta} \in \mathcal{S}_n \,|\, e_n(\boldsymbol{\theta}) > 0 \}, \tag{3.12}$$

$$e_n(\boldsymbol{\theta}) \coloneqq \left| \left\{ \boldsymbol{\theta}' \in \mathcal{D} \setminus \mathcal{S}_n \,|\, \exists i \in \mathcal{I}_c \colon u_n^i(\boldsymbol{\theta}) - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq 0 \right\} \right|. \tag{3.13}$$

The function e_n enumerates the number of parameters that could additionally be classified as safe if a safety function obtained a measurement equal to its upper confidence bound. Thus, the set $\mathcal{G}_n$ is an optimistic set of parameters that could potentially expand the safe set.

We trade off between the two sets, $\mathcal{M}_n$ and $\mathcal{G}_n$, by selecting the most uncertain element across all performance and safety functions; that is, at each iteration n we select

$$\boldsymbol{\theta}_n = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{G}_n \cup \mathcal{M}_n} \max_{i \in \mathcal{I}} w_n(\boldsymbol{\theta}, i), \tag{3.14}$$

$$w_n(\boldsymbol{\theta}, i) = u_n^i(\boldsymbol{\theta}) - l_n^i(\boldsymbol{\theta}) \tag{3.15}$$

as the next parameter set to be evaluated on the real system. The implications of this selection criterion will become more apparent in the next section, but from a high-level view this criterion leads to a behavior that focuses almost exclusively on exploration initially, as the most uncertain points typically lie on the boundary of the safe set for many commonly used kernels. This changes once the constraint evaluations return results closer to the safety constraints. At this point, the algorithm keeps switching between selecting parameters that are potential maximizers, and parameters that could expand the safe set and lead to

Algorithm 1 SafeOpt-MC

Inputs: Domain $\mathcal{D}$,
Gaussian process prior $k((\boldsymbol{\theta}, i), (\boldsymbol{\theta}', j))$,
Lipschitz constant L,
Initial safe set $\mathcal{S}_0 \subseteq \mathcal{D}$

1: **for** $t = 1, \ldots$ **do**
2: $\quad \mathcal{S}_n \leftarrow \bigcap_{i \in \mathcal{I}_c} \bigcup_{\boldsymbol{\theta} \in S_{n-1}} \{\boldsymbol{\theta}' \in \mathcal{D} \,|\, l_n^i(\boldsymbol{\theta}) - L d(\boldsymbol{\theta}, \boldsymbol{\theta}') \geq 0\}$
3: $\quad \mathcal{M}_n \leftarrow \left\{\boldsymbol{\theta} \in \mathcal{S}_n \,|\, u_n^J(\boldsymbol{\theta}) \geq \max_{\boldsymbol{\theta}' \in \mathcal{S}_n} l_n^J(\boldsymbol{\theta}')\right\}$
4: $\quad \mathcal{G}_n \leftarrow \{\boldsymbol{\theta} \in \mathcal{S}_n \,|\, e_n(\boldsymbol{\theta}) \geq 0\}$
5: $\quad \boldsymbol{\theta}_t \leftarrow \operatorname{argmax}_{\boldsymbol{\theta} \in \mathcal{G}_n \cup \mathcal{M}_n} \max_{i \in \mathcal{I}} w_n(\boldsymbol{\theta}, i)$
6: $\quad$ Noisy measurements $J(\boldsymbol{\theta}_n),\ \hat{c}_i(\boldsymbol{\theta}_n)\, \forall i = 1, \ldots, m$
7: $\quad$ Update Gaussian process with new data

new areas in the parameter space with even higher function values. Pseudocode for the algorithm is found in Algorithm 1.

We show an example run of the algorithm in Figure 3.1. It starts from an initial safe parameter $\boldsymbol{\theta}_0 \in \mathcal{S}_0$ at which we obtain a measurement in Figure 3.1a. Based on this, the algorithms uses the continuity properties of the safety function and the Gaussian process in order to determine nearby parameters as safe (red set). This corresponds to the region where the high-probability confidence intervals of the Gaussian process model (blue shaded) are above the safety threshold (grey dashed line). At the next iteration in Figure 3.1b, the algorithm evaluates parameters that are close to the boundary of the safe set, in order to expand the set of safe parameters. Eventually the algorithm converges to the optimal parameters in Figure 3.1c, which obtain the largest performance value that is possible without violating the safety constraints. A local optimization approach, e.g. based on estimated gradients[1], would have gotten stuck in the local optimum at the initial parameter $\boldsymbol{\theta}_0$.

At any iteration, we can obtain an estimate for the current best parameters from

$$\hat{\boldsymbol{\theta}}_n = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{S}_n} l_n^J(\boldsymbol{\theta}), \tag{3.16}$$

which returns the best, safe lower-bound on the performance function $J(\boldsymbol{\theta})$.

[1] If gradient information is available, it can be incorporated in the Gaussian process model too (Solak et al., 2003)

3.3.2 Theoretical Results

In this section, we show that the same theoretical framework from the SafeOpt algorithm (Sui et al., 2015) can be extended to multiple constraints and the evaluation criterion (3.14). Here, we only provide the results and high-level ideas of the proofs. The mathematical details are provided in Appendix A.

Since the confidence intervals hold with probability $1-\delta$ and the safe set is not empty starting from $\mathcal{S}_0$, it is possible to prove that parameters within the safe set $\mathcal{S}_n$ are always safe with high probability. In order for the algorithm to compete with our baseline, we must additionally ensure that the algorithm learns the true function up to ϵ confidence in both the sets $\mathcal{M}_n$ and $\mathcal{G}_n$. The number of measurements required to achieve this depends on the information capacity γ_n, since it encodes how much information can be obtained about the true function from n measurements. We use the sublinearity of γ_n in order to bound the number of samples required to estimate the function up to ϵ accuracy. We have the following result:

Theorem 2. *Under the assumptions of Lemma 2, also assume that $\mathcal{S}_0 \neq \emptyset$ and $c_i(\boldsymbol{\theta}) \geq 0$ for all $\boldsymbol{\theta} \in \mathcal{S}_0$ and $i \in \mathcal{I}_c$. Choose β_n as in Lemma 2, define $\hat{\boldsymbol{\theta}}_n$ as in (3.16), and let $n^*(\epsilon, \delta)$ be the smallest positive integer satisfying*

$$\frac{n^*}{\beta_{n^*}^2 \gamma_{|\mathcal{I}|n^*}} \geq \frac{C_1(|\bar{R}_0(\mathcal{S}_0)| + 1)}{\epsilon^2}, \tag{3.17}$$

where $C_1 = 8/\log(1+\sigma^{-2})$. For any $\epsilon > 0$ and $\delta \in (0,1)$, when running Algorithm 1 the following inequalities jointly hold with probability at least $1-\delta$:

1. Safety: $\forall n \geq 1, \forall i \in \mathcal{I}_c \colon c_i(\boldsymbol{\theta}_n) \geq 0$
2. Optimality: $\forall n \geq n^*,\ J(\hat{\boldsymbol{\theta}}_n) \geq J_\epsilon^* - \epsilon$

Proof. Main idea: safety follows from Lemma 2, since accurate confidence intervals imply that we do not evaluate unsafe parameters. For the optimality, the main idea is that, since we evaluate the most uncertain element in $\mathcal{M}_n \cup \mathcal{G}_n$, the uncertainty about the maximum is bounded by $w_n(\boldsymbol{\theta}_n, i_n)$. The result follows from showing that, after a finite number of evaluations, either the safe set expands or the maximum uncertainty within $\mathcal{M}_n \cup \mathcal{G}_n$ shrinks to ϵ. See Appendix A for derivations and details. □ □

Theorem 2 states that, given the assumptions we made about the underlying function, Algorithm 1 explores the state space without violating the safety constraints and, after at most n^* samples, finds an estimate that is ϵ-close to the optimal value over the safely reachable region. The information capacity $\gamma_{|\mathcal{I}|n^*}$, grows at a faster rate of $|\mathcal{I}|n$ compared to n in SafeOpt, since we obtain $|\mathcal{I}|$ observations at the same parameters set $\boldsymbol{\theta}$, while the SafeOpt analysis assumes every measurement is optimized independently. However, $\gamma_{|\mathcal{I}|n}$ remains sublinear in n, see Appendix A.

3.4 Context

In this section, we show how the theoretical guarantees derived in the previous section transfer to contextual Bayesian optimization. In this setting, part of the variables that influence the performance, the contexts, are fixed by an external process that we do not necessarily control. In normal Bayesian optimization, it was shown by Krause and Ong, 2011 that an algorithm that optimizes the GP-UCB criterion in (2.36) for a fixed context converges to the global optimum after repeatedly seeing the corresponding context.

Intuitively, the fact that part of the variables that influence the performance, the contexts, are now fixed by an external process should not impact the algorithm on a fundamental level. However, safety is a critical issue in our experiments and, in general, one could always select an adversarial context for which we do not have sufficient knowledge to determine safe controller parameters. As a consequence, we make the additional assumption that only 'safe' contexts are visited; that is, we assume the following:

Assumption 1. For any $n \geq 1$, the context $\mathbf{z}_n \in \mathcal{Z}$ is selected such that $\mathcal{S}_n(\mathbf{z}_n) \neq \emptyset$.

Here, $\mathcal{S}_n(\mathbf{z}_n)$ denotes the safe set for the given context $\mathbf{z}_n$. Intuitively, Assumption 1 ensures that for every context there exists at least one parameter choice that is known to satisfy all safety constraints. This assumption includes the case where safe initial parameters for all contexts are known *a priori* and the case where the algorithm terminates and asks for help from a domain-expert whenever a context leads to an empty safe set.

In order to obtain results that hold jointly across all contexts in $\mathcal{Z}$, we adapt the information capacity (worst-case mutual information) γ_n to consider contexts,

$$\gamma_n = \max_{\mathcal{A} \subseteq \mathcal{D} \times \mathcal{Z} \times \mathcal{I}, |\mathcal{D}| \leq n} \mathrm{I}(\mathbf{y}_\mathcal{A}; f'), . \tag{3.18}$$

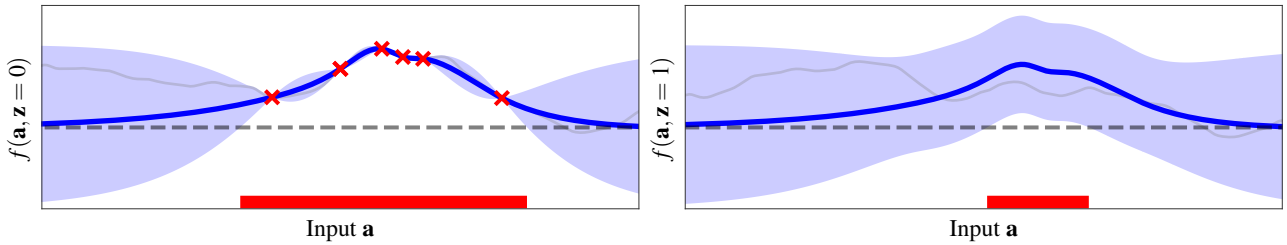


Figure 3.2: Example run of the context-dependent SafeOpt-MC algorithm. For the first six samples, the algorithm only sees the context $\mathbf{z} = 0$ (left function) and obtains measurements there (red crosses). However, by exploiting correlations between different contexts, the algorithm can transfer knowledge about the shape of the function and safe set over to a different context, $\mathbf{z} = 1$ (right function). This enables the algorithm to be significantly more data-efficient.

Unlike in (3.6), the mutual information is maximized across contexts in (3.18). As in the setting without contexts, we can use Lemma 2 to obtain confidence intervals that hold jointly across all contexts.

One challenge is that contexts are chosen in an arbitrary order. This is in stark contrast to the parameters $\boldsymbol{\theta}_n$, which are chosen according to (3.14) in order to be informative. This means that any tight finite sample bound on Algorithm 1 must necessarily depend on the order of contexts. The following theorem accounts for both of these challenges.

Theorem 3. *Under the assumptions of Theorem 2 and Assumption 1. Choose β_n as in Lemma 2, where γ_n is now the worst-case mutual information over contexts as in (3.18). For a given context order $(\mathbf{z}_1, \mathbf{z}_2, \dots)$ and any context $\mathbf{z} \in \mathcal{Z}$, let*

$$n(\mathbf{z}) = \sum_{n=1}^{n^*(\mathbf{z})} \mathbb{1}_{\mathbf{z}=\mathbf{z}_n} \tag{3.19}$$

be the number of times that we have observed the context $\mathbf{z}$ up to iteration n^* and $\mathbb{1}$ is the indicator function. Let $n^*(\mathbf{z})$ be the smallest positive integers such that

$$\frac{n(\mathbf{z})}{\beta_{n^*(\mathbf{z})}\ \gamma_{n(\mathbf{z})|\mathcal{I}|}(\mathbf{z})} \geq \frac{C_1(|\bar{R}_0(\mathcal{S}_0(\mathbf{z}))| + 1)}{\epsilon^2}, \tag{3.20}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$. We denote the information capacity for a fixed context $\mathbf{z}$ by $\gamma_n(\mathbf{z})$. That is, with $f'_{\mathbf{z}}(\boldsymbol{\theta}, i) = f'(\boldsymbol{\theta}, i, \mathbf{z})$ it is defined as $\gamma_n(\mathbf{z}) = \max_{\mathcal{A} \subseteq \mathcal{D} \times \mathcal{I}, |\mathcal{A}| \leq n} \mathrm{I}\left(\mathbf{y}_{\mathcal{A} \times \{\mathbf{z}\}}; f'_{\mathbf{z}}\right)$. For any $\epsilon > 0$ and $\delta \in (0, 1)$, let $f^*_\epsilon(\mathbf{z}) = \max_{\boldsymbol{\theta} \in \bar{R}_\epsilon(\mathcal{S}_0)} J(\boldsymbol{\theta}, \mathbf{z})$. Then, when running Algorithm 1 the following inequalities jointly hold with probability at least $1 - \delta$:

(i) $\forall n \geq 1, i \in \mathcal{I}_c \colon c_i(\boldsymbol{\theta}_n, \mathbf{z}_n) \geq 0$

(ii) $\forall \mathbf{z} \in \mathcal{Z}, n \geq n^*(\mathbf{z}) : J(\hat{\boldsymbol{\theta}}_n, \mathbf{z}) \geq J^*_\epsilon(\mathbf{z}) - \epsilon$

Proof. For a fixed context, $\mathbf{z}_n = \mathbf{z}\, \forall n$, we have $n^*(\mathbf{z}) = n(\mathbf{z})$ and the results follow directly as in Theorem 2. Otherwise, the only difference in the proofs is that β depends on the information capacity over contexts, making sure that the confidence intervals are valid across contexts. By visiting contexts in $\mathcal{Z} \setminus \{\mathbf{z}\}$, we obtain more measurements and β increases, which in turn increases the upper bound on the number of samples required at context $\mathbf{z}$. □

Theorem 3 states that the contextual variant of Algorithm 1 enjoys the same safety guarantees as the non-contextual version. Additionally, it shows that, after gaining enough information about a particular context, it can identify the optimal parameters. In practice, this upper bound is conservative, since it does not account for knowledge transfer across contexts. In practice, correlations between contexts significantly speed up the learning process. For example, in Figure 3.2 we show a contextual safe optimization problem with two contexts. Even though the algorithm has only been able to explore the parameter space at the first context ($\mathbf{z} = 0$, left function), the correlation between the functions means that information can be transferred to the as-of-yet unobserved context ($\mathbf{z} = 1$, right function). This knowledge transfer significantly improves data-efficiency and the number of evaluations required by the algorithm.

3.5 Practical Implementation

In this section, we discuss possible changes to Algorithm 1 that make the algorithm more practical, at the expense of loosing some of the theoretical guarantees. One challenge in applying Algorithm 1 in practice, is defining a suitable Lipschitz constant. In particular, specifying the wrong constant can lead to conservativeness or unsafe parameters being evaluated. Moreover, smoothness assumptions are already encoded by the kernel choice, which is more intuitive to specify than Lipschitz constants on their own. In practice, we use only the Gaussian process model to ensure safety (Berkenkamp, A. P. Schoellig, and Krause, 2016); that is, we define $l^i_n(\boldsymbol{\theta}) = \min \mathcal{Q}_n(\boldsymbol{\theta}, i)$ and $u^i_n(\boldsymbol{\theta}, i) = \max \mathcal{Q}_n(\boldsymbol{\theta}, i)$ in terms

of the confidence intervals of the Gaussian process directly. Thus, we can define the safe set without a Lipschitz constant as

$$\mathcal{S}_n = \mathcal{S}_0 \cup \left\{ \boldsymbol{\theta} \in \mathcal{D} \mid \forall i \in \mathcal{I}_c \colon l_n^i(\boldsymbol{\theta}) \geq 0 \right\}. \tag{3.21}$$

While it is difficult to prove the full exploration of the safely reachable set as in Theorem 2, the resulting algorithm remains safe:

Lemma 3. *With the assumptions of Lemma 2, $\mathcal{S}_0 \neq \emptyset$, and $c_i(\boldsymbol{\theta}) \geq 0$ for all $\boldsymbol{\theta} \in \mathcal{S}_0$ and $i \in \mathcal{I}_c$, when running Algorithm 1 with the safe set defined as in (3.21), the following holds with probability at least $1 - \delta$:*

$$\forall n \geq 1, \forall i \in \mathcal{I}_c \colon \; c_i(\boldsymbol{\theta}_n) \geq 0. \tag{3.22}$$

Proof. The confidence intervals hold with probability $1 - \delta$ following Lemma 2. Since $\mathcal{S}_n$ in (3.21) is defined as the set of parameters that fulfill the safety constraint and the safe set is never empty since $\mathcal{S}_0 \neq \emptyset$, the claim follows. □ □

Similarly, the set of expanders can be defined in terms of the Gaussian process directly, by adding optimistic measurements and counting the number of new parameters that are classified as safe, see (Berkenkamp, A. P. Schoellig, and Krause, 2016) for details. However, this potentially adds a large computational burden.

The parameter β_n, which determines the Gaussian process' confidence interval in Lemma 2, may be impractically conservative for experiments. Empirically, depending on the application, one may consider setting β_n to a constant value. This roughly corresponds to bounding the failure probability per iteration in the Bayesian setting, rather than over all iterations.

Learning all the different functions, $J(\boldsymbol{\theta})$ and c_i, up to the same accuracy ϵ may be restrictive if they are scaled differently. A possible solution is to either scale the observed data, or to scale the uncertainties in (3.14) by the prior variances of the kernels for that specific output. This enables (3.14) to make more homogeneous decisions across different scales.

3.6 Quadrotor Experiments

In this section, we demonstrate Algorithm 1 (with the changes discussed in Section 3.5) in experiments on a quadrotor vehicle, a Parrot AR.Drone 2.0.

A Python implementation of the SafeOpt-MC algorithm that builds on (GPy 2012), a Gaussian process library, is available at http://github.com/befelix/SafeOpt. Videos of the experiments can be found at

- Section 3.6.3: http://tiny.cc/icra16_video
- Section 3.6.4: https://youtu.be/rLmwYtoE3yg
- Section 3.6.5: https://youtu.be/4xC4OSiIDGk

3.6.1 Experimental Setup

During the experiments, measurements of all vehicle states are estimated from position and pose data provided by an overhead motion capture camera system. The quadrotor's dynamics can be described by six states: positions $\mathbf{x} = (x, y, z)$, velocities $\dot{\mathbf{x}} = (\dot{x}, \dot{y}, \dot{z})$, ZYX Euler angles (ϕ, θ, ψ), and body angular velocities $(\omega_x, \omega_y, \omega_z)$. The control inputs $\mathbf{u}$ are the desired roll and pitch angles θ_{des} and ϕ_{des}, the desired z-velocity $\dot{z}_{\text{des}}$, and the desired yaw angular velocity $\omega_{z,\text{des}}$, which in turn are inputs to an unknown, proprietary, on-board controller.

The position dynamics in the global coordinate frame are

$$\ddot{\mathbf{x}} = R_{\text{ZYX}}(\phi, \theta, \psi)\vec{f} - \vec{g}, \tag{3.23}$$

where R_{ZYX} is the rotation matrix from the body frame to the inertial frame, $\vec{f} = (0, 0, c)$ is the mass-normalized thrust, and $\vec{g} = (0, 0, g)$ is the gravitational force. The goal of the controller is to track a reference signal. We assume that z-position and the yaw angle are controlled by fixed control laws and focus on the position control in x- and y direction. We use two different control laws in the following experiments.

The simplest control law that can be used for this setting is a PD-controller, defined as

$$\phi_{\text{des}} = k_1(x_t - x_{\text{des}}) + k_2(\dot{x} - \dot{x}_{\text{des}}), \tag{3.24}$$

$$\theta_{\text{des}} = k_1(y_t - y_{\text{des}}) + k_2(\dot{y} - \dot{y}_{\text{des}}), \tag{3.25}$$

where $\boldsymbol{\theta} = (k_1, k_2)$ are the two tuning parameters. Intuitively, a larger parameter k_1 encourages tracking reference changes more aggressively, while k_2 is a damping factor that encourages moderate velocities.

A more sophisticated approach to control quadrotor vehicles is to use estimates of the angles and accelerations to solve for the thrust c. We use loop shaping on the horizontal position control loops so that they behave in the manner of a second-order systems with time constant τ and damping ratio ζ. Based on a given desired reference trajectory, commanded accelerations are computed as

$$\ddot{x}_c = \frac{1}{\tau^2}(x_{\text{des}} - x) + \frac{2\zeta}{\tau}(\dot{x}_{\text{des}} - \dot{x}), \tag{3.26}$$

$$\ddot{y}_c = \frac{1}{\tau^2}(y_{\text{des}} - y) + \frac{2\zeta}{\tau}(\dot{y}_{\text{des}} - \dot{y}). \tag{3.27}$$

From the commanded accelerations, we then obtain the control inputs for the desired roll and pitch angles by solving (3.23) for the angles. Here, the tuning parameters are $\boldsymbol{\theta} = (\tau, \zeta)$. For details regarding the controllers see (A. Schoellig et al., 2012; Lupashin et al., 2014).

The quadrotor was controlled using the ardrone_autonomy and vicon_bridge packages in ROS Hydro. Computations for the SafeOpt-MC algorithm in Algorithm 1 were conducted on a regular laptop and took significantly less than one second per iteration. As a result, experiments could be conducted continuously without interruptions or human interventions.

3.6.2 Kernel Selection

Algorithm 1 critically depends on the Gaussian process model for the performance and constraint functions. In this section, we review the kernel used in our experiments and the kind of models that they encode. A more thorough review of kernel properties can be found in (Carl Edward Rasmussen and C. K. Williams, 2006).

In our experiments, we use the Matèrn kernel with parameter $\nu = 3/2$ (Carl Edward Rasmussen and C. K. Williams, 2006),

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \kappa^2 \left(1 + \sqrt{3}\,\bar{d}(\boldsymbol{\theta}, \boldsymbol{\theta}')\right) \exp\left(-\sqrt{3}\,\bar{d}(\boldsymbol{\theta}, \boldsymbol{\theta}')\right), \tag{3.28}$$

$$\bar{d}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sqrt{(\boldsymbol{\theta} - \boldsymbol{\theta}')^{\mathrm{T}} \mathbf{M}^{-2} (\boldsymbol{\theta} - \boldsymbol{\theta}')}, \tag{3.29}$$

which encodes that mean functions are one-times differentiable. This is in contrast to the commonly used squared exponential kernels, which encode smooth (infinitely differentiable) functions. With the Matèrn kernel, the Gaussian process model is parameterized by three hyperparameters: measurement noise σ^2 in (2.28) and (2.30), the kernel's prior variance κ^2, and positive lengthscales $\mathbf{l} \in \mathcal{R}_+^{\mathcal{D}}$, which are the diagonal elements of the diagonal matrix $\mathbf{M}$, $\mathbf{M} = \mathrm{diag}(\mathbf{l})$. These hyperparameters have intuitive interpretations: the variance of the measurement noise κ^2 corresponds to the noise in the observations, which includes any randomness in the algorithm and initial conditions, and random disturbances. The prior variance κ^2 determines the expected magnitude of function values; that is, $|J(\boldsymbol{\theta})| \leq \kappa$ with probability 0.68 according to the Gaussian process prior. Lastly, the lengthscales $\mathbf{l}$ determine how quickly the covariance between neighboring values deteriorates with their distance. The smaller the lengthscales, the faster the function values can change from one parameter set to the next. In particular, the high-probability Lipschitz constant encoded by this kernel depends on the ratio between the prior variance and the lengthscales, $\kappa/\mathbf{l}$.

When using Gaussian processes to model dynamic systems, typically a maximum likelihood estimate of the hyperparameters is used based on data; see (Ostafew et al., 2016) for an example. For Bayesian optimization, the Gaussian process model is used to actively acquire data, rather than only using it for regression based on existing data. This dependence between the kernel hyperparameters and the acquired data is known to lead to poor results in Bayesian optimization when using a maximum likelihood estimate of the hyperparameters during data acquisition (Bull, 2011). In particular, the corresponding Gaussian process estimate is not guaranteed to contain the true function as in Lemma 2. In this work, we critically rely on these confidence bounds to guarantee safety. As a consequence, we do not adapt the hyperparameters as more data becomes available, but treat the kernel as a prior over functions in the true Bayesian sense; that is, the kernel hyperparameters encode our prior knowledge about the functions that we model and are fixed before experiments begin. While this requires intuition about the process, intuitively the less knowledge we encode in the prior, the more data and evaluations on the real system are required in order to determine the best parameters.

3.6.3 Linear Control

In this section, we use SafeOpt-MC to optimize the parameters of the linear control law in (3.24). The entire control algorithm consists of this control law together with the

on-board controller and state estimation.

The goal is to find controller parameters that maximize the performance during a 1-meter reference position change. For an experiment with parameters $\boldsymbol{\theta}_n$ at iteration t, the performance function is defined as

$$J(\boldsymbol{\theta}_n) = c(\boldsymbol{\theta}_n) - 0.95\, c(\boldsymbol{\theta}_0), \tag{3.30}$$

$$r(\boldsymbol{\theta}_n) = -\sum_{t=0}^{T} \mathbf{x}_t^{\mathrm{T}} \mathbf{Q} \mathbf{x}_t + \mathbf{u}_t^{\mathrm{T}} \mathbf{R} \mathbf{u}_t, \tag{3.31}$$

where, to compute the cost c, the states $\mathbf{x} = (x - 1, \dot{x}, \phi, \omega)$ and the input u are weighted by positive semi-definite matrices $\mathbf{Q}$ and $\mathbf{R}$. The subscript t indicates the state measurement at time step t in the trajectory and the time horizon is 5 s ($T = 350$). Performance is defined as the cost improvement relative to 95% of the initial controller cost. The safety constraint is defined only in terms of the performance; that is, the constraint is $c(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) \geq 0$, which encodes that we do not want to evaluate controller parameters that perform significantly worse than the initial parameters.

While the optimal controller parameters could be easily computed given an accurate model of the system, we do not have a model of the dynamics of the proprietary, on-board controller and the time delays in the system. Moreover, we want to optimize the performance for the real, nonlinear quadrotor system, which is difficult to model accurately. An inaccurate model of the system could be used to improve the prior Gaussian process model of the performance function, with the goal of achieving faster convergence. In this case, the uncertainty in the Gaussian process model of the performance function would account for inaccuracies in the system model.

We define the domain of the controller parameters as $[-0.6, 0.1]^2$, explicitly including positive controller parameters that certainly lead to crashes. In practice, one would exclude parameters that are known to be unsafe *a priori*. The initial controller parameters are $(-0.4, -0.4)$, which result in a controller with poor performance. Decreasing the controller gains further leads to unstable controllers.

The parameters for the experiments were set as follows: the length-scales were set to 0.05 for both parameters, which corresponds to the notion that a 0.05-0.1 change in the parameters leads to very different performance values. The prior standard deviation, κ, and the noise standard deviation, σ, are set to 5% and 10% of the performance of the initial controller, $J(\boldsymbol{\theta}_0)$, respectively. The noise standard deviation, σ, mostly models errors due to initial position offsets, since state measurements have low noise. The size of

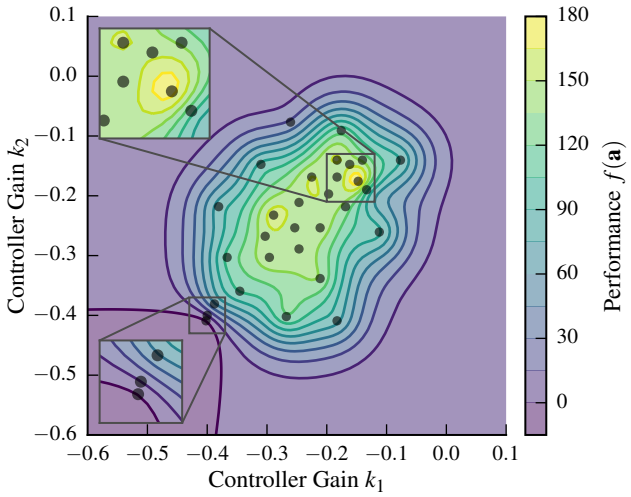


Figure 3.3: Gaussian process mean estimate of the performance function after 30 evaluations. The algorithm adaptively decides which parameters to evaluate based on safety and informativeness. In the bottom-left corner, there is the magnified section of the first three samples, which are close together to determine the location of the initial, safe region. The maximum, magnified in the top-left corner, also has more samples to determine the precise location of the maximum. Other areas are more coarsely sampled to expand the safe region.

these errors depends on the choice of the matrices $\mathbf{Q}$ and $\mathbf{R}$. By choosing σ as a function of the initial performance, we account for the $\mathbf{Q}$ and $\mathbf{R}$ dependency. Similarly, κ specifies the expected size of the performance function values. Initially, the best we can do is to set this quantity dependent on the initial performance and leave additional room for future, larger performance values. For the Gaussian process model, we choose $\beta_n = 2$ to define the confidence interval in (3.9).

The resulting, estimated performance function after running Algorithm 1 for 30 experiments is shown in Fig. 3.3. The unknown function has been reliably identified. Samples are spread out over the entire safe set, with more samples close to the maximum of the function and close to the initial controller parameters. No unsafe parameters below the safety threshold were evaluated on the real system.

Typically, the optimization behavior of Algorithm 1 can be roughly separated into three stages. Initially, the algorithm evaluates controller parameters close to the initial parameters in order for the Gaussian process to acquire information about the safe set (see

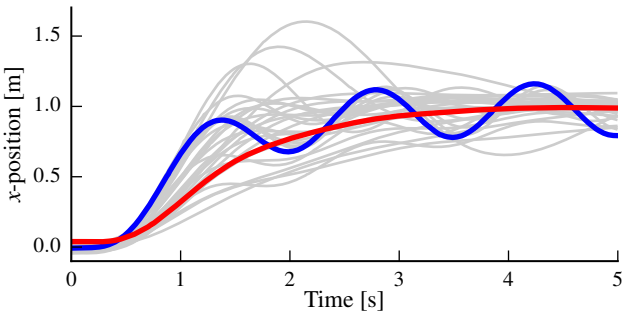


Figure 3.4: The quadrotor controller performance is evaluated during a 5 s evaluation interval, where a 1 m reference position change must be performed. The trajectories correspond to the optimization routine in Figure 3.3. The initial controller (blue) performs poorly but is stable. In contrast, the optimized controller (red) shows an optimized, smooth, and fast response. The trajectories of other controller parameters that were evaluated are shown in gray.

lower-left, zoomed-in section in Figure 3.3). Once a region of safe controller parameters is determined, the algorithm evaluates the performance function more coarsely in order to expand the safe set. Eventually, the controller parameters are refined by evaluating high-performance parameters that are potential maximizers in a finer grid (see upper-left, zoomed-in section in Figure 3.3). The trajectories of the initial, best and intermediate controllers can be seen in Figure 3.4.

3.6.4 Nonlinear Control

In the previous section, we showed how to optimize the performance of a linear control law subject to a simple constraint on performance. In this section, we optimize the nonlinear controller in (3.26) and (3.27) and show how more complex constraints can be used.

We use the same task as in the previous section, but this time the goal is to minimize the root-mean-square error (RMSE) over a time horizon of 5 s ($T = 350$ samples) during a 1-meter reference position change in x-direction. We define the performance function,

$$J(\boldsymbol{\theta}_n) = c(\boldsymbol{\theta}_n) - 0.75\, c(\boldsymbol{\theta}_0), \tag{3.32}$$

$$r(\boldsymbol{\theta}_n) = \frac{1}{\sqrt{T}} \left(\sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}_{\mathrm{des},t}\|_2^2 \right)^{1/2}, \tag{3.33}$$

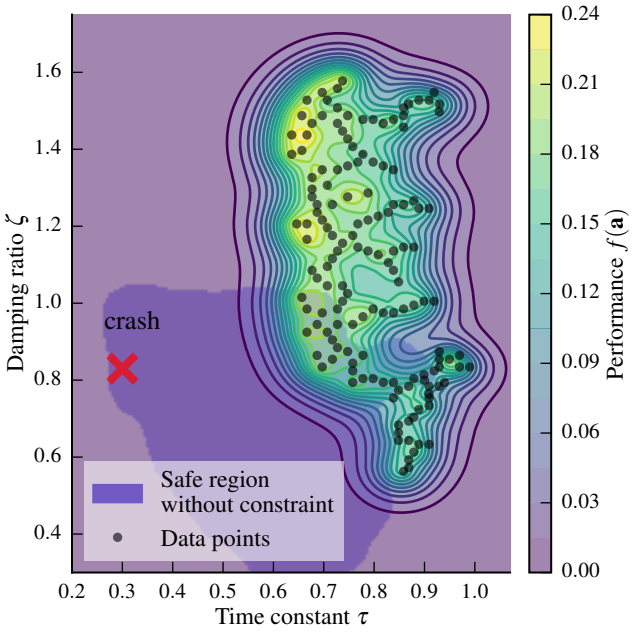


Figure 3.5: Mean estimate of the root-mean-square error when optimizing the parameters of the nonlinear control law for a step response, subject to safety constraints. The algorithm carefully evaluates only safe parameter combinations, until the safe region cannot be expanded further without violating constraints. Without the safety constraint, the algorithm explores a larger region of the parameter space (light blue) and eventually evaluates an unsafe parameter set.

as the performance relative to 75% of the performance of the initial parameters $\boldsymbol{\theta}_0 = (0.9, 0.8)$. We define the Gaussian process model of this performance function as follows: in this experiment, measurement noise is minimal, since the positions are measured accurately by the overhead camera system. However, to capture errors in the initial position, we define $\sigma = 0.05c(\boldsymbol{\theta}_0)$. We assume that we can improve the initial controller by roughly 20%, so we set $\kappa = 0.2c(\boldsymbol{\theta}_0)$. The lengthscales are set to 0.05 in order to encourage cautious exploration. These parameters turned out to be conservative for the real system. Notice that the cost is specified relative to $c(\boldsymbol{\theta}_0)$ instead of $J(\boldsymbol{\theta}_0)$ as in Section 3.6.3. Since $c(\boldsymbol{\theta}_0) > J(\boldsymbol{\theta}_0)$, these hyperparameters are more conservative, so that we require more evaluations on the real system. The reason for this more conservative choice is that the nonlinear controller is expected to have a less smooth performance function, unlike the one for linear control, which is expected to be roughly quadratic.

If, as in the previous section, one were to set the safety constraint to $c_1(\boldsymbol{\theta}) = J(\boldsymbol{\theta})$, the

algorithm would classify the blue shaded region in Figure 3.5 as safe. This region includes time constants as low as $\tau = 0.3$, which encourage highly aggressive maneuvers, as would be expected from a performance function that encourages changing position as fast as possible. However, these high gains amplify noise in the measurements, which can lead to crashes; that is, the performance-based constraint cannot properly encode safety. Notice that the blue shaded area does not correspond to full exploration, since the experiment was aborted after the first, serious crash. The reason for the unsafe exploration is that the RMSE performance function in (3.33) does not encode safety the same way as as weighting of state and input errors in Figure 3.3 does. Thus, in order to encode safety, we need to specify additional safety constraints.

One indication of unsafe behavior in quadrotors are high angular velocities when the quadrotor oscillates around the reference point. We define an additional safety constraint on the maximum angular velocity $\max_k |\omega_{x,k}| \leq 0.5\,\mathrm{rad/s}$ by setting $c_2(\boldsymbol{\theta}) = 0.5 - \max_k |\omega_{x,k}|$. The corresponding hyperparameters are selected as $\sigma_2 = 0.1$, $l = 0.2$, and $\kappa = 0.25$. The measurement noise can be chosen relatively small here, since it corresponds to a single measurement of angular velocity. Note that it is difficult to perform the step maneuver with an angular velocity lower than $0.4\,\mathrm{rad/s}$, so that typical values of c_2 are smaller than 0.1.

With this additional safety constraint, the algorithm explores the parameter space and stops before the safety constraints are violated, as can be seen in Figure 3.5. Rather than exploring smaller time constants τ (higher gains), the algorithm evaluates larger damping ratios, which allow slightly smaller values of τ and therefore higher performance without violating the safety constraints. The optimal parameters are to the top-left of the safe set, where small time constants encourage tracking the reference aggressively, while the increased damping ratio ensures a moderate angular velocity.

3.6.5 Circle Trajectory

In this section, we use the same nonlinear controller and cost function as in the previous section, but aim to optimize the RMSE with respect to a circle trajectory of radius 1 m at a speed of 1 m/s. The reference is defined as a point moving along the circle at the desired speed. Feasibility of such motions has been analyzed in A. P. Schoellig et al., 2011.

In order to ensure good tracking behavior, we define safety as a constraint on the maximum RMSE of 0.2 m. Additionally, we use the same constraint on the maximum angular

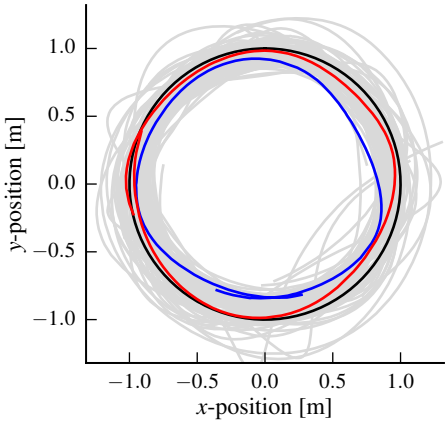


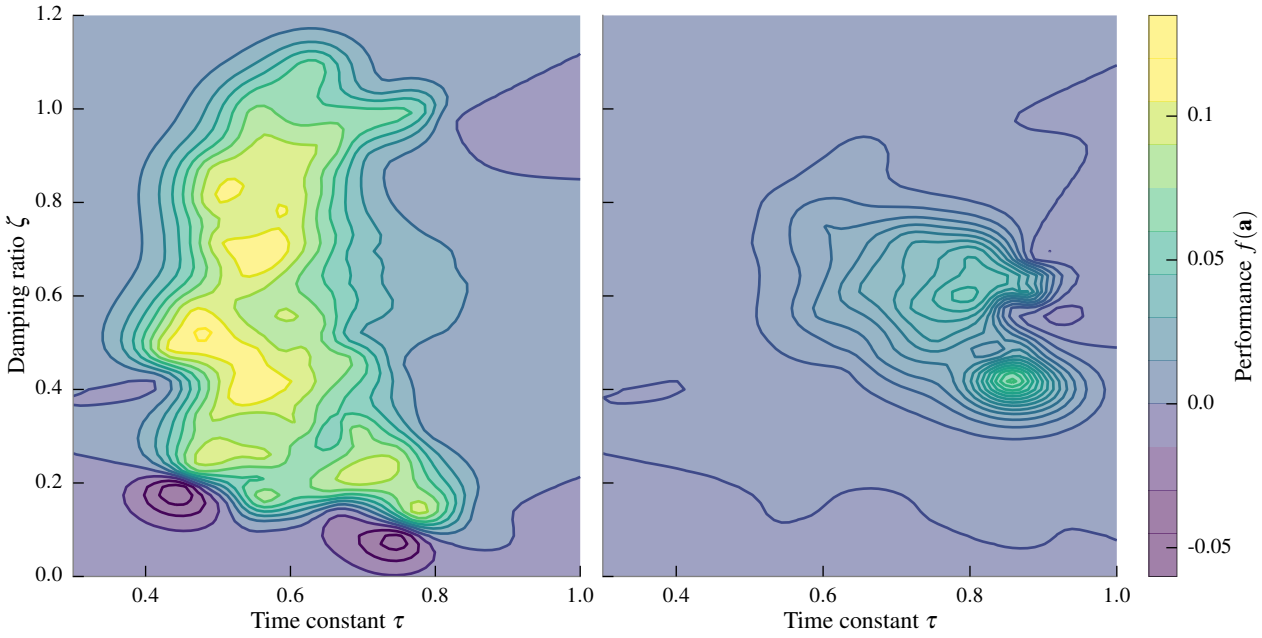
Figure 3.6: The trajectories (gray) resulting from iteratively optimizing the controller parameters for a unit circle reference trajectory at 1 m/s (black). The trajectory with the initial parameters (blue) has poor tracking performance, while the optimized parameters (red) perform significantly better. The flight is safe, i.e., only safe parameters are evaluated.

velocity around the x and y axis of 0.5 rad/s as before. The yaw-angle is set so that the quadrotor always points to the center of the circle, which ideally should lead to zero angular velocity. Deviations from this are an indication of unsafe behavior. We use the same kernel hyperparameters as in Section 3.6.4.

The trajectories that result from running the optimization algorithm are shown in Figure 3.6. The initial controller parameters lead to very poor performance. In particular, the initial time constant is too large, so that the quadrotor lags behind the reference. As a result, the quadrotor flies a circle of smaller radius. In contrast, the resulting optimized trajectory (in red) is the best that can be obtained given the safety constraints and controller structure above. The mean estimate of the performance function after the experiments can be seen in Section 3.6.5. The optimal parameters have smaller time constants, so that the position is tracked aggressively. Since the reference point moves of 1 m/s, these aggressive controller parameters do not lead to unsafe behavior. During the entire optimization, only safe parameters that keep the vehicle within the constraints on RMSE and angular velocity are evaluated.

3.6.6 Context-Dependent Optimization

In this section, we show how the knowledge about good controller parameters at low speeds can be used to speed up the safe learning at higher speeds.



(a) Speed: $\dot{x}_{\text{des}} = 1\,\frac{\text{m}}{\text{s}}$ (b) Speed: $\dot{x}_{\text{des}} = 1.8\,\frac{\text{m}}{\text{s}}$

Figure 3.7: The mean estimate of the performance function for the circle trajectory in Figure 3.6 for a speed of 1 m/s (left) and 1.8 m/s (right). Extending the kernel with a context for speed allows to transfer knowledge to different speeds and leads to speed-dependent optimal control parameters, speeding up the learning for higher speeds.

In our circle experiment, the quadrotor tracked a moving reference. As this reference moves with high velocities, the quadrotor gets pushed to its physical actuator limits and starts to lag behind the reference. This causes the circle that is flown by the quadrotor to have a smaller radius than the reference trajectory. In this section, the goal is to maximize the speed of the reference trajectory subject to the safety constraints of the previous experiment in Section 3.6.5. One way to achieve this goal, is to add the speed of the reference point to the performance function. However, this would lead to more experiments, as the algorithm will explore the safe parameter space for every velocity. Instead, here we define the trajectory speed as a context, which is set externally. In particular, we set

$$z_n = \underset{v\in\mathbb{R},\,\boldsymbol{\theta}\in\mathcal{D}}{\operatorname{argmax}}\, v \quad \text{subject to:} \quad c_i(\boldsymbol{\theta}, v) \geq 0,\ \forall i \in \mathcal{I}_c, \tag{3.34}$$

that is, we select the maximum velocity for which there are safe parameters known. While

here we select the context manually, in practice contexts can be used to model any external, measurable variables, such as the battery level, see Section 2.7.1.1.

In order to transfer knowledge about good controller parameters from the slow speed in Section 3.6.5 to higher speeds, we model how performance and constraints vary with desired speed by defining a kernel $k_z(\dot{x}_{\text{des}}, \dot{x}'_{\text{des}})$ over contexts. We use the same kernel structure as in (2.35) and hyperparameters $\kappa = 1$ and $l = 0.25$. Based on the data from Section 3.6.5, the extended model allows us to determine speeds for which safe controller parameters are known.

Starting from the data of the previous experiments in Section 3.6.5, we run SafeOpt-MC using the extended kernel with the additional speed context determined by (3.34). This allows us to find optimal parameters for increasingly higher speeds, which satisfy the constraints. We can safely increase the speed up to 1.8 m/s. We show the mean performance function estimates for two speeds in Figure 3.7. For lower speeds, the best controller parameters track the reference position more aggressively (low τ). For higher speeds, this behavior becomes unsafe as the quadrotor lags behind the reference point. Instead, the optimal parameters shift to higher time constants (lower gains). Additionally, as expected, high speeds lead to higher reference tracking errors. Increasing the reference velocity any further causes the performance constraint to be violated.

The trajectories that result from applying the optimal parameters for a speed of 1 m/s and the maximum safe speed of 1.8 m/s can be seen in Figure 3.8. For the relatively slow speed of 1 m/s the quadrotor can track the circle well using aggressive parameters. For the higher speed, the reference trajectory moves too fast for the quadrotor to track perfectly within the actuator limits, so that the best parameters just barely satisfy the safety constraint on the average deviation from the reference. Overall, this approach allows us to find context-dependent parameters, while remaining within the safety constraints.

3.7 Conclusion

In this chapter, we presented a generalization of the Safe Bayesian Optimization algorithm of Sui et al., 2015 that allows multiple, separate safety constraints to be specified and applied it to nonlinear control problems on a quadrotor vehicle. Moreover, we extended these results to contextual Bayesian optimization in order to enable safe transfer learning.

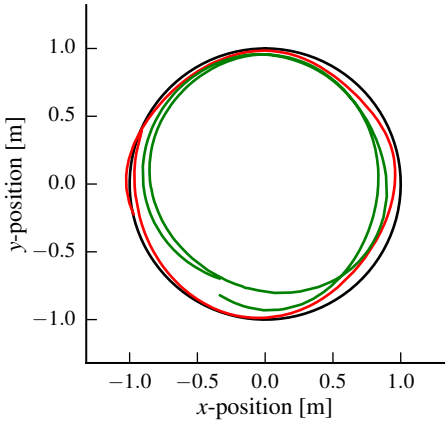


Figure 3.8: Trajectories with optimal parameters for speeds of 1 m/s (red) and 1.8 m/s (green) when tracking the black reference. At slower speeds there exist aggressive controller parameters that allow the quadrotor to track the reference almost perfectly. At higher speeds, actuator saturation limits the achievable performance. Due to the safe optimization framework the maximum speed can be found that does not deviate more from the reference trajectory than is allowed by the safety constraint. The corresponding performance functions can be seen in Figure 3.7.

Overall, the SafeOpt-MC algorithm enabled efficient and automatic optimization of parameters without violating the safety constraints, which would lead to system failures.

Other Related Work There are several publications which are not part of this dissertation, but that were written during the course of the PhD and are relevant to this chapter.

- In (Berkenkamp, A. P. Schoellig, and Krause, 2019), we investigated a variant of the GP-UCB algorithm that obtains regret bounds even when the prior model is miss-specified.
- In (Ghosh et al., 2018), we use Bayesian optimization to verify specifications in formal methods. In particular, we show how to exploit the problem structure to avoid modeling discontinuous functions with a Gaussian process.
- In (Marco et al., 2017), we investigate cost-efficient transfer learning between a cheap simulation and an expensive real-world experiment.

- In (Duivenvoorden et al., 2017), we investigate how to scale up safe Bayesian optimization heuristically.
- In (Abdelrahman et al., 2016), we apply unsafe Bayesian optimization to data from a real-world photovoltaic power plant.

4

Safety Analysis of Learned Dynamical Systems

Some of the results in this chapter have been previously published in (Berkenkamp, Moriconi, et al., 2016; Berkenkamp, Turchetta, et al., 2017; Koller, Berkenkamp, Turchetta, Boedecker, et al., 2019). Partial results of the last paper were shown in (Koller, Berkenkamp, Turchetta, and Krause, 2018).

In model-based reinforcement learning, we learn a model of the system dynamics in (2.1) directly and use it for planning, see Section 2.5. In this chapter, we first introduce assumptions that allow us to learn a reliable model of f in (2.1). Next, we analyze the safety of a the model both in terms of stability and finite-time constraint satisfaction. In contrast to existing methods in Section 2.8.2, we state our results with respect to specific assumptions about f, rather than broad assumptions about the statistical model.

4.1 Learning reliable models of dynamical systems

We consider a special case of the stochastic system in (2.1) with additive, *i.i.d.* transition noise $\boldsymbol{\omega}_t \in \mathbb{R}^p$,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}_t = \underbrace{h(\mathbf{x}_t, \mathbf{u}_t)}_{\text{Known model}} + \underbrace{g(\mathbf{x}_t, \mathbf{u}_t)}_{\text{Unknown error}} + \boldsymbol{\omega}_t, \tag{4.1}$$

so that $f\colon \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p$. This implies that the transition noise $\boldsymbol{\omega}_t$ does not depend on state and control input and is thus *homoscedastic*. We additionally split the model into a known part h and an *unknown* model error g. For example, the known part may come from an *a priori* known physical model, while g captures any model errors. To make inference tractable, we additionally assume that the noise $\boldsymbol{\omega}_t$ is σ-sub-Gaussian, see Definition 4.

Assumption 2. For all $t \geq 0$ the elements of the noise vector $\boldsymbol{\omega}_t$ are *i.i.d.* σ-sub-Gaussian.

4.1.1 Regularity Assumptions

Next to the noise properties, we require additional continuity assumptions for our theoretical results. In particular, in the following we make assumptions that ensure that f is Lipschitz continuous, which we require both to show Lyapunov stability and to over-approximate the uncertainty over multiple time steps. We also make stronger assumptions about continuous-differentiability, which we use to make the uncertainty propagation in Section 4.3 less conservative. However, Lipschitz continuity on its own would be sufficient for a more conservative version of the results.

Assumption 3. The known model h is continuously differentiable and L_h-Lipschitz continuous with respect to the 2-norm. Moreover, $\|h(\mathbf{x}_0) - \mathbf{x}_0\|_2 \leq B_h$.

Assumption 3 ensures the continuity properties of the known model. We introduce the bound on $\|h(\mathbf{x}_0) - \mathbf{x}_0\|_2$ here only for notational convenience later on, since a system that explodes to infinity is not particularly interesting for safe learning.

Assumption 4. The model error g has RKHS norm bounded by B_g with respect to a continuously differentiable kernel k with bounded derivative and $k(\mathbf{x}_0, \mathbf{x}_0) \leq k_{\max}$.

This assumption allows us to learn a calibrated model of the function g. Since RKHS functions are linear combinations of the kernel function evaluated at representer points, the continuity assumptions on the kernel directly transfer to continuity assumptions on the function g, so that we get the following result.

Corollary 2. *The model error g is continuously differentiable and L_g-Lipschitz continuous with respect to the 2-norm.*

Proof. For scalar functions, this is a direct consequence of Assumption 4 and (Christmann and Steinwart, 2008, Cor. 4.36). This directly generalizes to the vector case. □

In the following, we at times consider a closed-loop system under a policy π. To ensure that the resulting system is well-behaved, we assume that the policy is Lipschitz-continuous too.

Assumption 5. The control policies $\pi_{\boldsymbol{\theta}}$ lie in a set Π of functions that are L_π-Lipschitz continuous with respect to the 2-norm.

Corollary 3. *The closed-loop dynamics* (4.1) *under a policy* $\pi \in \Pi$ *are* L_f*-Lipschitz continuous with respect to the 2-norm.*

Proof.

$$\begin{aligned}
\|f(\mathbf{x}, \pi(\mathbf{x})) - f(\mathbf{x}', \pi(\mathbf{x}'))\|_2 &\leq (L_g + L_h)\|(\mathbf{x} - \mathbf{x}', \pi(\mathbf{x}) - \pi(\mathbf{x}'))\|_2 && (4.2)\\
&\leq (L_g + L_h)(\|(\mathbf{x} - \mathbf{x}'\|_2 + \|\pi(\mathbf{x}) - \pi(\mathbf{x}'))\|_2) && (4.3)\\
&\leq \underbrace{(L_g + L_h)(1 + L_\pi)}_{:=L_f}\|\mathbf{x} - \mathbf{x}'\|_2 && (4.4)
\end{aligned}$$

□

For simplicity, we assume that the policy class Π is designed to respect input constraints.

Assumption 6. For any policy $\pi \in \Pi$ we have $\pi(\mathbf{x}) \in \mathcal{U}$ for all $\mathbf{x} \in \mathbb{R}^p$.

While this may seem restrictive at first, this is common-practice in the reinforcement learning literature. For example, if Π is the class of policies parameterized by a certain neural network structure, a tanh nonlinearity can be used on the output of the neural network in order to ensure that suggested control actions are bounded within $[-1, 1]$. By scaling this output and adding a fixed offset, any interval input constraint can be enforced.

Lastly, whenever we have a performance objective we assume that the reward function is Lipschitz continuous.

Assumption 7. The reward $r(\mathbf{x}, \mathbf{u})$ is L_r-Lipschitz continuous with respect to the 2-norm.

This is not a restrictive assumption, since cost functions that are discontinuous are generally very difficult to optimize. In practice, most reward functions are continuously differentiable.

4.1.2 Bounding the Epistemic Uncertainty

Since the state $\mathbf{x}$ is observed directly, the assumptions in Section 4.1.1, especially Assumption 4, allow us to learn a reliable statistical model of g. In particular, for each transition from $(\mathbf{x}_t, \mathbf{u}_t)$ to $\mathbf{x}_{t+1}$, we add p observations, one for each output dimension, to $\mathcal{A}_n$ as in Lemma 2.

Corollary 4. *Under Assumptions 2 and 4 with β_n as in Lemma 2 and a Gaussian process model trained on observations $\mathbf{x}_{t+1} - h(\mathbf{x}_t, \mathbf{u}_t)$ based on an input $\mathbf{a} = (\mathbf{x}_t, \mathbf{u}_t)$, the following holds with probability $1 - \delta$ for all $n \geq 0$, $\mathbf{x} \in \mathbb{R}^p$, and $\mathbf{u} \in \mathbb{R}^q$:*

$$|f(\mathbf{x}, \mathbf{u}, i) - h(\mathbf{x}, \mathbf{u}, i) - \mu_n(\mathbf{x}, \mathbf{u}, i)| \leq \beta_n \sigma_n(\mathbf{x}, \mathbf{u}, i) \tag{4.5}$$

In the following, we write

$$\boldsymbol{\mu}_n(\mathbf{x}, \mathbf{u}) = (\mu_n(\mathbf{x}, \mathbf{u}, 1), \ldots, \mathbf{u}_n(\mathbf{x}, \mathbf{u}, p)), \tag{4.6}$$

$$\boldsymbol{\sigma}_n(\mathbf{x}, \mathbf{u}) = (\sigma_n(\mathbf{x}, \mathbf{u}, 1), \ldots, \sigma_n(\mathbf{x}, \mathbf{u}, p)) \tag{4.7}$$

to represent the individual elements as vectors. Corollary 4 allows us to build confidence intervals on the model error g based on the scaled Gaussian process posterior variance. A direct consequence of these point-wise error bounds is that we can also bound the norm of the error on the vector-output of f.

Corollary 5. *Under the assumption of Corollary 4, with probability $1 - \delta$ we have for all $n \geq 0$, $\mathbf{x} \in \mathbb{R}^p$, and $\mathbf{u} \in \mathbb{R}^q$ that*

$$\|f(\mathbf{x}, \mathbf{u}) - h(\mathbf{x}, \mathbf{u}) - \boldsymbol{\mu}_n(\mathbf{x}, \mathbf{u})\|_2 \leq \beta_n \|\sigma_n(\mathbf{x}, \mathbf{u})\|_2 \tag{4.8}$$

Proof.

$$\|f(\mathbf{x}, \mathbf{u}) - h(\mathbf{x}, \mathbf{u}) - \boldsymbol{\mu}_n(\mathbf{x}, \mathbf{u})\|_2 = \left(\sum_{i=1}^{p} |f(\mathbf{x}, \mathbf{u}, i) - h(\mathbf{x}, \mathbf{u}, i) - \mu_n(\mathbf{x}, \mathbf{u}, i)|^2 \right)^{1/2} \tag{4.9}$$

$$\leq \left(\sum_{i=1}^{p} |\beta_n \sigma_n(\mathbf{x}, \mathbf{u}, i)|^2 \right)^{1/2} = \beta_n \|\boldsymbol{\sigma}_n(\mathbf{x}, \mathbf{u})\|_2 \tag{4.10}$$

□

4.1.3 Bounding the Aleatoric Uncertainty

Corollary 4 allows us to obtain reliable confidence intervals on the model error g for all iterations n. We can also bound the aleatoric uncertainty by exploiting the sub-Gaussian property. In particular, we have the following result and provide a proof in Appendix B.1.

Lemma 4. *Let $\boldsymbol{\omega}_0, \boldsymbol{\omega}_1, \ldots$ be i.i.d. random vectors with $\boldsymbol{\omega}_t \in \mathbb{R}^p$ such that each entry of the vector is i.i.d. σ-sub-Gaussian. Then, with probability at least $(1-\delta)$,*

$$\|\boldsymbol{\omega}_t\|_2^2 \leq 2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{(t+1)^2 \pi^2}{3\delta} \tag{4.11}$$

holds jointly for all $t \geq 0$.

This means that, for all time steps t, the noise is bounded within the hyper-sphere defined through (4.11) with high probability. In particular, the joint confidence intervals only come at the cost of a $\mathcal{O}(\log t^2)$ increase in the confidence intervals over time. A simple union bound between Lemma 4 and Corollary 4 can be used to ensure that both the epistemic and the aleatoric bounds on the uncertainty hold jointly.

The aleatoric noise structure is not particular interesting and can be bounded according to Lemma 4. To avoid cumbersome notation, in the following we consider safety of the deterministic certainty-equivalent system (2.2) instead. However, we comment on how these results can be extended to stochastic systems when applicable.

4.2 Stability of Uncertain Systems

Based on the assumptions in Section 4.1, we can use Corollary 4 to learn a well-calibrated model of the dynamics f in (4.1). As a first step towards safe learning, we now ask the question whether a given policy $\pi \in \Pi$ leads to an asymptotically stable system. Note that the stochastic system in (4.1) is *not asymptotically stable* as defined in Definition 1, since the additive noise ensures that the state can never converge to the origin. Instead of considering a stochastic notion of stability (Khasminskii, 2012), we instead ask if the certainty-equivalent system (2.2) is asymptotically stable. That is, while we can learn confidence intervals from the stochastic system, we only analyze the deterministic system ($\boldsymbol{\omega}_t = \mathbf{0}$) here.

To compute the region of attraction for a fixed policy, we assume that we have access to a L_v-Lipschitz continuous Lyapunov function with connected level sets $\mathcal{V}(c)$ for all

$c > 0$. For example, this is fulfilled for continuously differentiable Lyapunov functions with $\partial V(\mathbf{x})/\partial \mathbf{x} = \mathbf{0}$ only at the origin.

In order to use Corollary 1 to estimate a region of attraction, we must verify that the decrease condition $v(f_\pi(\mathbf{x})) - v(\mathbf{x}) \leq 0$ holds over a continuous domain. However, the posterior uncertainty in the statistical model of the dynamics means that one step predictions about $v(f_\pi(\cdot))$ are uncertain too. We account for this by constructing high-probability confidence intervals on $v(f(\mathbf{x}, \mathbf{u}))$:

$$\mathcal{Q}_n(\mathbf{x}, \mathbf{u}) := \left[v(\mu_{n-1}(\mathbf{x}, \mathbf{u})) \pm L_v \beta_n \|\sigma_{n-1}(\mathbf{x}, \mathbf{u})\|_2\right]. \tag{4.12}$$

From Corollary 4 together with the Lipschitz property of v, we know that $v(f(\mathbf{x}, \mathbf{u}))$ is contained in $\mathcal{Q}_n(\mathbf{x}, \mathbf{u})$ with probability at least $(1 - \delta)$. Based on these confidence intervals, we define $u_n(\mathbf{x}, \mathbf{u}) := \max \mathcal{Q}_n(\mathbf{x}, \mathbf{u})$ and $l_n(\mathbf{x}, \mathbf{u}) := \min \mathcal{Q}_n(\mathbf{x}, \mathbf{u})$ as the upper and lower confidence interval on $v(f_\pi(\cdot))$.

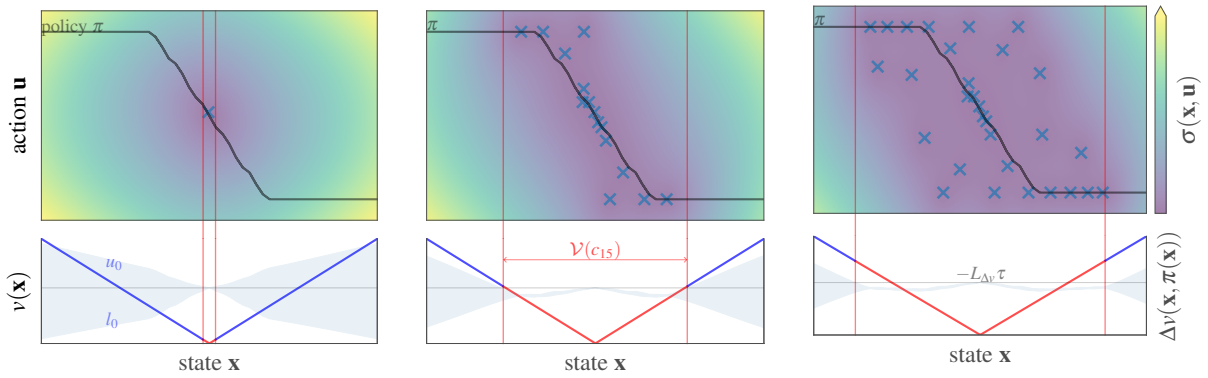
Given these high-probability confidence intervals, the system is stable according to Theorem 1 if $v(f(\mathbf{x}, \mathbf{u})) \leq u_n(\mathbf{x}) < v(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{V}(c)$. However, it is intractable to verify this condition directly on the continuous domain without additional, restrictive assumptions about the model. Instead, we consider a discretization of the state space $\mathcal{X}_\kappa \subset \mathcal{X}$ into cells, so that $\|\mathbf{x} - [\mathbf{x}]_\kappa\|_2 \leq \kappa$ holds for all $\mathbf{x} \in \mathcal{X}$. Here, $[\mathbf{x}]_\kappa$ denotes the point in $\mathcal{X}_\kappa$ with the smallest l_2 distance to $\mathbf{x}$. Given this discretization, we bound the decrease variation on the Lyapunov function for states in $\mathcal{X}_\kappa$ and use the Lipschitz continuity to generalize to the continuous state space $\mathcal{X}$.

Theorem 4. *Under Corollaries 3 and 4 with $L_{\Delta v} := L_v L_f (L_\pi + 1) + L_v$, let $\mathcal{X}_\kappa$ be a discretization of $\mathcal{X}$ such that $\|\mathbf{x} - [\mathbf{x}]_\kappa\|_2 \leq \kappa$ for all $\mathbf{x} \in \mathcal{X}$. If, for all $\mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa$ with $c > 0$, $\mathbf{u} = \pi(\mathbf{x})$, and for all $n \geq 0$ it holds that*

$$u_n(\mathbf{x}, \mathbf{u}) < v(\mathbf{x}) - L_{\Delta v} \kappa, \tag{4.13}$$

then $v(f(\mathbf{x}, \pi(\mathbf{x}))) < v(\mathbf{x})$ holds for all $\mathbf{x} \in \mathcal{V}(c)$ with probability at least $(1 - \delta)$ and $\mathcal{V}(c)$ is a positive invariant region of attraction for (2.2) under the policy π.

The proof is given in Appendix B.2. Theorem 4 states that, given confidence intervals on the statistical model of the dynamics, it is sufficient to check the stricter decrease condition in (4.13) on the discretized domain $\mathcal{X}_\kappa$ to guarantee the requirements for the region of attraction in the continuous domain in Corollary 1. The bound in (4.13) becomes tight



(a) $n = 1$. (b) $n = 15$. (c) $n = 30$.

Figure 4.1: Example for the Lyapunov stability verification using Theorem 4 on a simple one-dimensional system. Due to input constraints, the system becomes unstable for large states. Initially the dynamics are uncertain (top, background) in Figure 4.1a, which induces a large uncertainty about $v(f(\mathbf{x}, \pi(\mathbf{x})))$ under the given policy π (black line, top). As a result, initially we can only certify a small region of attraction as safe. As the GP model collect more observations (blue crosses), the uncertainty about the dynamics decreases and we can verify a large region of the state space as safe in Figure 4.1b. After 30 data points, we have almost identified the largest possible region of attraction in Figure 4.1c.

as the discretization constant κ and $|v(f(\cdot)) - u_n(\cdot)|$ go to zero. Thus, the discretization constant trades off computation costs for accuracy, while u_n approaches $v(f(\cdot))$ as we obtain more measurement data and the posterior model uncertainty about the dynamics, $\beta_n \|\boldsymbol{\sigma}_n\|_2$, decreases.

The confidence intervals on $v(f(\mathbf{x}, \pi(\mathbf{x}))$ are show in the bottom half of Figure 4.1 (blue shaded) for a given Lyapunov function (blue line). If we are uncertain about the transition dynamics (top, background) in Figure 4.1a, we can only certify a small region of attraction as safe (red lines, bottom plot). As we gather more data, the uncertainty about the dynamics decreases and, as a consequence, so does the uncertainty about $v(f(\mathbf{x}, \mathbf{u}))$. As a result, in Figures 4.1b and 4.1c we can certify larger subsets of the state space as safe.

Continuous-time These results can also be extended to continuous-time systems, see (Berkenkamp, Moriconi, et al., 2016). The main technical caveat is that, in order to build reliable confidence intervals, we have to assume that derivative observations are available. This is restrictive in general. We do not state these results here and refer to the paper

for details in order to avoid confusion between notation for continuous- and discrete-time systems.

4.2.1 State constraints

While we focus on stability constraints in the following, we can account for state constraints by limiting the maximum size of the region of attraction. In particular, we define

$$c_{\max} = \min_{\mathbf{x} \in \mathbb{R}^p \setminus \mathcal{X}} v(\mathbf{x}) \tag{4.14}$$

as the minimum value of the Lyapunov function outside the feasible region $\mathcal{X}$. If we make sure that the region of attraction is positive invariant and that it never includes states outside of $\mathcal{X}$, the policy cannot violate the state constraints.

Lemma 5. *Let $\mathcal{V}(c)$ with $0 < c < c_{\max}$ be positive invariant for the system (4.1) under a policy π. Then, for any state $\mathbf{x}_0 \in \mathcal{V}(c)$, we have*

$$\mathbf{x}_t \in \mathcal{X} \tag{4.15}$$

for all $t \geq 0$.

Proof. We have that $\mathcal{V}(c') \subseteq \mathcal{X}$ for any $c' < c_{\max}$ by definition of the level-set and $c_{\max}$. The result follows since $\mathcal{V}(c)$ is positive invariant. □

Lemma 5 ensures that as long as we limit the estimated region of attraction to states with $v(\mathbf{x}) < c_{\max}$, we satisfy the state constraints. Effectively, $\mathcal{V}(c_{\max})$ is the largest level-set of the Lyapunov function that is contained within the feasible region of the state space, $\mathcal{X}$.

4.3 Confidence Intervals for Finite-time Trajectories

Next to stability, we would like to verify that, under a given finite sequence of control inputs, the system adheres to safety constraints as in Section 2.4.1.2. To verify these conditions, we need to construct confidence intervals on the trajectories that might occur under a given policy. While Corollary 4 allows us to build confidence intervals for one-step predictions given the starting point $\mathbf{x}$, for multi-step predictions we must be able to propagate uncertainty through the uncertain model in a reliable way. While many

approximate uncertainty propagation methods have been used in the literature, we would like to exploit the properties of our model to construct them directly.

We start by reviewing basic properties of ellipsoids and then use these in order to over-approximate the uncertainty in our model.

4.3.1 Ellipsoids

We use ellipsoids to bound the uncertainty of our system when making multi-step ahead predictions. Due to appealing geometric properties, ellipsoids are widely used in the robust control community to compute *reachable sets* (Filippova, 2017; Asselborn et al., 2013). These sets intuitively provide an outer approximation on the next state of a system considering all possible realizations of uncertainties when applying a controller to the system at a given set-valued input. We briefly review some of these properties and refer to (Kurzhanskii and Vályi, 1997) for an exhaustive introduction to ellipsoids and to the derivations for the following properties.

We use the basic definition of an ellipsoid,

$$\mathcal{E}(\mathbf{p}, \mathbf{Q}) := \{\mathbf{x} \in \mathbb{R}^p | (\mathbf{x} - \mathbf{p})^{\mathrm{T}} \mathbf{Q}^{-1} (\mathbf{x} - \mathbf{p}) \leq 1\}, \tag{4.16}$$

with *center* $\mathbf{p} \in \mathbb{R}^p$ and a symmetric positive definite (s.p.d) *shape matrix* $\mathbf{Q} \in \mathbb{R}^{p \times p}$. Ellipsoids are invariant under *affine subspace transformations* such that, for $\mathbf{A} \in \mathbb{R}^{p \times r}, r \leq p$ with full column rank and $\mathbf{b} \in \mathbb{R}^r$, we have that

$$\mathbf{A} \cdot \mathcal{E}(\mathbf{p}, \mathbf{Q}) + \mathbf{b} = \mathcal{E}(\mathbf{p} + \mathbf{b}, \mathbf{A}\mathbf{Q}\mathbf{A}^{\mathrm{T}}). \tag{4.17}$$

The *Minkowski sum* $\mathcal{E}(\mathbf{p}_1, \mathbf{Q}_1) \oplus \mathcal{E}(\mathbf{p}_2, \mathbf{Q}_2)$, i.e. the pointwise sum between two arbitrary ellipsoids, is in general not an ellipsoid anymore, but we have

$$\mathcal{E}(\mathbf{p}_1, \mathbf{Q}_1) \oplus \mathcal{E}(\mathbf{p}_2, \mathbf{Q}_2) \subset \mathcal{E}_c(\tilde{\mathbf{p}}, \tilde{\mathbf{Q}}), \tag{4.18}$$

where $\tilde{\mathbf{p}} = \mathbf{p}_1 + \mathbf{p}_2$, $\tilde{\mathbf{Q}} = (1 + c^{-1})\mathbf{Q}_1 + (1 + c)\mathbf{Q}_2$ for all $c > 0$. Moreover, the minimizer of the trace of the resulting shape matrix is analytically given as $c = \sqrt{\mathrm{trace}(\mathbf{Q}_1)/\mathrm{trace}(\mathbf{Q}_2)}$. A particular problem that we encounter is finding the maximum distance ρ to the center of an ellipsoid $\mathcal{E}(\mathbf{0}, \mathbf{Q})$ under a special transformation, i.e.

$$\rho(\mathbf{Q}, \mathbf{S}) = \max_{\mathbf{x} \in \mathcal{E}(\mathbf{p}, \mathbf{Q})} \|\mathbf{S}(\mathbf{x} - \mathbf{p})\|_2 = \max_{\mathbf{s}^{\mathrm{T}} \mathbf{Q}^{-1} \mathbf{s} \leq 1} \mathbf{s}^{\mathrm{T}} \mathbf{S}^{\mathrm{T}} \mathbf{S} \mathbf{s}, \tag{4.19}$$

where $\mathbf{S} \in \mathbb{R}^{m \times n}$ has full column rank. This is a generalized eigenvalue problem of the pair $(\mathbf{Q}, \mathbf{S}^{\mathrm{T}}\mathbf{S})$ and the optimizer is given as the square-root of the largest generalized eigenvalue.

Lastly, we can over-approximate a hyper-rectangle. Let $\mathbf{a} \pm \mathbf{b} := [[\mathbf{a}]_1 \pm [\mathbf{b}]_1] \times [[\mathbf{a}]_p \pm [\mathbf{b}]_p]$ for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$ denote the hyper-rectangles centered at $\mathbf{a}$ with element-wise uncertainty in $\mathbf{b}$. We can over-approximate this rectangle with an ellipsoid

$$\mathbf{a} \pm \mathbf{b} \subset \mathcal{E}(\mathbf{a}, \sqrt{p} \cdot \mathrm{diag}(\mathbf{b})), \tag{4.20}$$

where $\mathrm{diag}(\mathbf{b})$ is is a square matrix thats zero everywhere, except on the diagonal which is given by the elements of $\mathbf{b}$.

4.3.2 Robust Multi-step Predictions

In order plan safe trajectories based on our statistical model, we need to reliably estimate the region of the state space that can be reached over multiple time steps under a sequence of control inputs. Based on Assumption 4 and our prior model $h(\mathbf{x}_t, \mathbf{u}_t)$, we directly obtain high-probability confidence intervals on $f(\mathbf{x}_t, \mathbf{u}_t)$ uniformly for all $t \in \mathbb{N}$ given a single control input $\mathbf{u}_t$ from Corollary 4. We extend this to over-approximate the system after a sequence of inputs $(\mathbf{u}_t, \mathbf{u}_{t+1}, .., \mathbf{u}_{T-1})$. The result is a sequence of set-valued confidence regions that contain the true trajectory of the system with high probability.

4.3.2.1 One-step Predictions with Uncertain Inputs

One key challenge in multi-step predictions is that, while the initial state $\mathbf{x}$ is known, beyond one step the input to the statistical model is uncertain. In this section, we derive a function that takes as an input an ellipsoidal subset of the state space and outputs a second ellipsoid that is an outer approximation to the next state if the current state is contained in the input ellipsoid.

In order to approximate the system, first we linearize our prior model $h(\mathbf{x}_t, \mathbf{u}_t)$ and use the affine transformation property (4.17) to compute the ellipsoidal next state of the linearized prior model. Next, we approximate the unknown model-error $g(\mathbf{x}_t, \mathbf{u}_t)$ using the confidence intervals of our statistical model. We propose a locally constant approximation of g in Section 4.3.2.1. We finally apply Lipschitz arguments to outer-bound the approximation errors. We sum up these individual approximations, which result in an ellipsoidal approximation of the next state of the system. This is illustrated in Figure 4.2. We formally

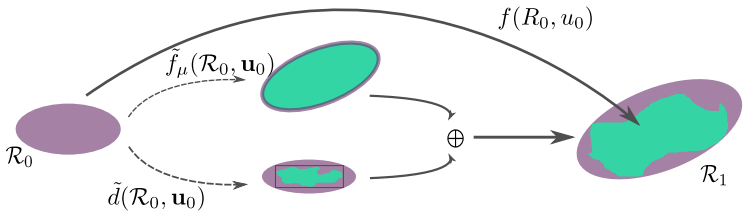


Figure 4.2: Decomposition of the over-approximated image of the system (4.1) under an ellipsoidal input R_0. The exact, unknown image of f (right, green area) is approximated by the linearized model $\tilde{f}_\mu$ (center, top) and the remainder term $\tilde{d}$, which accounts for the confidence interval and the linearization errors of the approximation (center, bottom). The resulting ellipsoid R_1 is given by the Minkowski sum of the two individual approximations.

Table 4.1: List of important variables, functions and constants.

Variable	Type	Definition
$\mathcal{R}$	Set	Ellipsoidal set of states $\mathcal{R} = \mathcal{E}(p, Q)$, with center p and shape matrix Q.
J_ϕ	Matrix	Jacobian matrix of a function ϕ with $J_\phi = [A_\phi, B_\phi]$, where A_ϕ, B_ϕ are the Jacobians w.r.t. the state and control inputs, respectively.
L_ϕ	Vector	Vector with Lipschitz constants for each output of a function ϕ.
$P_\phi^{\bar{\mathbf{a}}}$	Function	Taylor approximation of a function ϕ around linearization point $\bar{\mathbf{a}}$.
$\tilde{m}(\mathcal{R}, \pi_t)$	Function	Function taking ellipsoidal set of states $\mathcal{R}$ and affine feedback controller π_t. Outputs ellipsoidal over-approximation of next system state.

derive the necessary equations in the following paragraphs, which result in Lemma 6. We provide a list of important variables functions and constants in Table 4.1.

Locally constant model approximation We first consider the system f in (4.1) for a single input vector $\mathbf{a} = (\mathbf{x}, \mathbf{u})$ so that $f(\mathbf{a}) = h(\mathbf{a}) + g(\mathbf{a})$. We linearly approximate f around $\bar{\mathbf{a}} = (\bar{\mathbf{x}}, \bar{\mathbf{u}})$ via

$$f(\mathbf{a}) \approx h(\bar{\mathbf{a}}) + \mathbf{J}_h(\bar{\mathbf{a}})(\mathbf{a} - \bar{\mathbf{a}}) + g(\bar{\mathbf{a}}) = \tilde{f}(\mathbf{a}), \tag{4.21}$$

where $\mathbf{J}_h(\bar{\mathbf{a}}) = [\mathbf{A}_h, \mathbf{B}_h]$ is the Jacobian of h at $\bar{\mathbf{a}}$.

Next, we use the Lagrangian remainder theorem (Breiman and Cutler, 1993) on the linearization of h and apply a continuity argument on our locally constant approximation of g. This results in an upper-bound on the approximation error,

$$|[f(\mathbf{a})]_j - [\tilde{f}(\mathbf{a})]_j| \leq \frac{L_{\nabla h,j}}{2}\|\mathbf{a} - \bar{\mathbf{a}}\|_2^2 + L_g\|\mathbf{a} - \bar{\mathbf{a}}\|_2, \tag{4.22}$$

where $[f(\mathbf{a})]_j$ is the jth component of f with $1 \leq j \leq p$, $L_{\nabla h,j}$ is the Lipschitz constant of the gradient $\nabla[h]_j$, and L_g is the Lipschitz constant of g, which exists by Corollary 2.

The function $\tilde{f}$ depends on the unknown model error g. We approximate g with the statistical model from Corollary 4 with, $\boldsymbol{\mu}(\bar{\mathbf{a}}) \approx g(\bar{\mathbf{a}})$. From Assumption 4 we have

$$|[g(\bar{\mathbf{a}})]_j - [\boldsymbol{\mu}_n(\bar{\mathbf{a}})]_j| \leq \beta_n[\boldsymbol{\sigma}(\bar{\mathbf{a}})]_j,\ 1 \leq j \leq p, \tag{4.23}$$

with high probability. We combine (4.22) and (4.23) to obtain

$$|[f(\mathbf{a})]_j - [\tilde{f}_\mu(\mathbf{a})]_j| \leq \beta\sigma_n(\bar{\mathbf{a}}, j) + \frac{L_{\nabla h,j}}{2}\|\mathbf{a} - \bar{\mathbf{a}}\|_2^2 + L_g\|\mathbf{a} - \bar{\mathbf{a}}\|_2, \tag{4.24}$$

where $1 \leq j \leq p$ and $\tilde{f}_\mu(\mathbf{a}) = h(\bar{\mathbf{a}}) + \mathbf{J}_h(\bar{\mathbf{a}})(z - \bar{\mathbf{a}}) + \mu_n(\bar{\mathbf{a}})$. We can interpret (4.24) as the edges of the confidence hyper-rectangle

$$\tilde{m}(\mathbf{a}) = \tilde{f}_\mu(\mathbf{a}) \pm \left[\beta\boldsymbol{\sigma}_n(\bar{\mathbf{a}}) + \frac{1}{2}\mathbf{L}_{\nabla h}\|\mathbf{a} - \bar{\mathbf{a}}\|_2^2 + L_g\|\mathbf{a} - \bar{\mathbf{a}}\|_2\right], \tag{4.25}$$

where $\mathbf{L}_{\nabla h} = (L_{\nabla h,1}, .., L_{\nabla h,p})$.

We are now ready to compute a confidence region based on an ellipsoidal state $\mathcal{R} = \mathcal{E}(\mathbf{p}, \mathbf{Q}) \subset \mathbb{R}^p$ and a fixed input $\mathbf{u}$, by over-approximating the output of the system $f(\mathcal{R}, \mathbf{u}) = \{f(\mathbf{x}, \mathbf{u}) \,|\, \mathbf{x} \in \mathbb{R}^p\}$ for all inputs contained in an ellipsoid $\mathcal{R}$. Here, we choose $\mathbf{p}$ as the linearization center of the state and choose $\bar{\mathbf{u}} = \mathbf{u}$, i.e. $\bar{\mathbf{a}} = (\mathbf{p}, \mathbf{u})$. Since the function $\tilde{f}_\mu$ is affine, we can make use of (4.17) to compute

$$\tilde{f}_\mu(\mathcal{R}, \mathbf{u}) = \mathcal{E}(h(\bar{\mathbf{a}}) + \boldsymbol{\mu}_n(\bar{\mathbf{a}}), \mathbf{A}_h\mathbf{Q}\mathbf{A}_h^{\mathrm{T}}), \tag{4.26}$$

which results in an ellipsoid. This is visualized in Figure 4.2 by the upper ellipsoid in the center. To upper-bound the confidence hyper-rectangle on the right hand side of (4.25), we upper-bound the term $\|\mathbf{a} - \bar{\mathbf{a}}\|_2$ by

$$l(\mathcal{R}, \mathbf{u}) := \max_{\mathbf{x} \in \mathcal{R}} \|(\mathbf{x}, \mathbf{u}) - \bar{\mathbf{a}}\|_2, \tag{4.27}$$

which leads to

$$\tilde{d}(\mathcal{R}, u) := \beta_n\boldsymbol{\sigma}_n(\bar{\mathbf{a}}) + \mathbf{L}_{\nabla h}l^2(\mathcal{R}, \mathbf{u})/2 + L_g l(R, \mathbf{u}). \tag{4.28}$$

Due to our choice of $\bar{\mathbf{a}}$, we have that $\max_{\mathbf{x}\in\mathcal{R}} \|(\mathbf{x},\mathbf{u}) - \bar{\mathbf{a}}\|_2 = \|\mathbf{x}-\mathbf{p}\|_2$ and we can use (4.19) to get $l(\mathcal{R},\mathbf{u}) = \rho(\mathbf{Q},\mathbf{I}_p)$,, where r corresponds to the largest eigenvalue of $\mathbf{Q}^{-1}$ and is defined in (4.19). Using (4.27), we can now over-approximate the right side of (4.25) for inputs $\mathcal{R}$ by an ellipsoid

$$0 \pm \tilde{d}(\mathcal{R},\mathbf{u}) \subset \mathcal{E}(0, Q_{\tilde{d}}(\mathcal{R},\mathbf{u})), \tag{4.29}$$

where we obtain $Q_{\tilde{d}}(\mathcal{R},\mathbf{u})$ by over-approximating the hyper-rectangle $\tilde{d}(\mathcal{R},\mathbf{u})$ with the ellipsoid $\mathcal{E}(\mathbf{0}, Q_{\tilde{d}}(\mathcal{R},\mathbf{u}))$ as in (4.20). This is illustrated in Figure 4.2 by the lower ellipsoid in the center. Combining the previous results, we can compute the final over-approximation using (4.18),

$$\mathcal{R}_+ = \tilde{m}(\mathcal{R},\mathbf{u}) = \tilde{f}_\mu(R,\mathbf{u}) \oplus \mathcal{E}(0, Q_{\tilde{d}}(\mathcal{R},\mathbf{u})). \tag{4.30}$$

Since we carefully incorporated all approximation errors and extended the confidence intervals around our model predictions to set-valued inputs, we get the following generalization of the reliability Assumption 4 for single inputs.

Lemma 6. *Let $\delta \in (0,1]$ and choose β_n as in Corollary 4. Then, with probability greater than $1-\delta$, we have for any ellipsoid $\mathcal{R} = \mathcal{E}(\mathbf{p},\mathbf{Q}) \subset \mathbb{R}^p$ that*

$$\mathbf{x} \in \mathcal{R} \Rightarrow f(\mathbf{x},\mathbf{u}) \in \tilde{m}(\mathcal{R},\mathbf{u}), \tag{4.31}$$

uniformly for all $n \in \mathbb{N}$.

Proof. Define $m(\mathbf{x},\mathbf{u}) = h(\mathbf{x},\mathbf{u}) + \boldsymbol{\mu}_n(\mathbf{x},\mathbf{u}) \pm \beta_n \boldsymbol{\sigma}_n(\mathbf{x},\mathbf{u})$. From Corollary 4 we have that $f(\mathbf{x},\mathbf{u}) \in m(\mathbf{x},\mathbf{u})$. Due to the over-approximations, we have $m(\mathbf{x},\mathbf{u}) \subset \tilde{m}(\mathcal{R},\mathbf{u})$. □

Lemma 6 allows us to compute confidence ellipsoid around the next state of the system, given that the current state of the system is known to be contained in an ellipsoidal confidence region. Note that $\tilde{m}$ implicitly depends on the statistical model and thus the data observed up to time step n.

Stochastic system The one-step bound in Lemma 6 can easily be extended to the stochastic case by employing the bound on the transition noise Lemma 4 and adding the ellipsoid to the over-approximation in (4.30). Note that we need a union bound between Corollary 4 and Lemma 4 for the $1-\delta$ guarantee to hold jointly over all n and t.

4.3.2.2 Multi-step Predictions

We use the previous result to compute a sequence of ellipsoids that contain a trajectory of the system with high-probability, by iteratively applying the one-step ahead predictions (4.30).

Given an initial ellipsoid $\mathcal{R}_0 = \{\mathbf{x}\}$ and a sequence of control inputs $\mathbf{u}_t \in \mathbb{R}^q$, we iteratively compute confidence ellipsoids as

$$\mathcal{R}_{t+1} = \tilde{m}(\mathcal{R}_t, \mathbf{u}_t). \tag{4.32}$$

We can directly apply Lemma 6 to get the following result.

Corollary 6. *Under the assumptions of Lemma 6. Let $\mathbf{x}_0$ be a given state and $\mathbf{u}_0, \dots, \mathbf{u}_{T-1}$ be a sequence of control actions. Compute confidence intervals $\mathcal{R}_1, \dots, \mathcal{R}_T$ as in (4.32) and let $\mathbf{x}_1, \dots \mathbf{x}_T$ denote the true sequence of states of the deterministic system in (2.2). Then, with probability at least $1 - \delta$*

$$\mathbf{x}_t \in \mathcal{R}_t, \tag{4.33}$$

for all $0 < t \leq T$.

Proof. Direct consequence of Lemma 6 and Corollary 4. □

Corollary 6 guarantees that, with high probability, the system is always contained in the propagated ellipsoids (4.32).

4.3.2.3 Predictions under State-Feedback Control Laws

When applying multi-step ahead predictions under a sequence of feed-forward inputs $\mathbf{u}_t \in \mathcal{U}$, the individual sets of the corresponding reachability sequence can quickly grow large. This is because these *open loop* input sequences do not account for future control inputs that could correct deviations from the model predictions.

To account for future corrective control actions, we extend (4.30) to *affine state-feedback control laws* of the form

$$\mathbf{u}_{K,t}(\mathbf{x}_t) := \mathbf{K}_t(\mathbf{x}_t - \mathbf{p}_t) + \mathbf{k}_t, \tag{4.34}$$

where $\mathbf{K}_t \in \mathbb{R}^{q \times p}$ is a feedback matrix and $\mathbf{k}_t \in \mathbb{R}^q$ is the open-loop input. The parameter $\mathbf{p}_t$ is determined through the center of the current ellipsoid $\mathcal{R}_t = \mathcal{E}(\mathbf{p}_t, \mathbf{Q}_t)$. Given an

appropriate choice of $\mathbf{K}_t$, the control law actively contracts the ellipsoids towards their center. This technique is commonly used in *tube-based model predictive control*, to reduce the size of *tubes* around a nominal trajectory of the system that incorporate uncertainties and disturbances (Rawlings and Mayne, 2009). Similar to the derivations (4.21)–(4.30), we can compute the function $\tilde{m}$ for affine feedback controllers (4.34) and ellipsoids $\mathcal{R}_t = \mathcal{E}(\mathbf{p}_t, \mathbf{Q}_t)$. The resulting ellipsoid is

$$\tilde{m}(\mathcal{R}_t, \mathbf{u}_{K,t}) = \mathcal{E}(h(\bar{\mathbf{a}}_t) + \boldsymbol{\mu}_n(\bar{\mathbf{a}}_t), H_t \mathbf{Q}_t H_t^{\mathrm{T}}) \oplus \mathcal{E}(\mathbf{0}, Q_{\tilde{d}}(\mathcal{R}_t, \mathbf{u}_{K,t})), \tag{4.35}$$

where $\bar{\mathbf{a}}_t = [\mathbf{p}_t, \mathbf{k}_t]^{\mathrm{T}}$ and $\mathbf{H}_t = \mathbf{A}_h + \mathbf{B}_h \mathbf{K}_t$. The set $\mathcal{E}(\mathbf{0}, Q_{\tilde{d}}(\mathcal{R}_t, \mathbf{u}_{K,t}))$ is obtained similarly to (4.27) as the ellipsoidal over-approximation of

$$\mathbf{0} \pm \left[\beta_n \boldsymbol{\sigma}_n(\bar{\mathbf{a}}) + \mathbf{L}_{\nabla h} \frac{l^2(\mathcal{R}_t, S_t)}{2} + L_g l(\mathcal{R}_t, \mathbf{S}_t) \right], \tag{4.36}$$

with $\mathbf{S}_t = [\mathbf{I}_p, \mathbf{K}_t^{\mathrm{T}}]$ and $l(\mathcal{R}_t, \mathbf{S}_t) = \max_{x \in \mathcal{R}_t} \|\mathbf{S}_t((\mathbf{x}, \mathbf{u}_{\mathbf{K}_t}(x)) - \bar{\mathbf{z}}_t)\|_2$. The theoretical results of Lemma 6 and Corollary 6 directly apply to the case of the uncertainty propagation technique (4.35). For the remainder of this dissertation, we assume $\mathbf{K}_t$ is pre-specified, while $\mathbf{k}_t$ is assumed to be a decision variable. For the sake of generality, we drop the subscript K and the functional dependency on $\mathbf{x}$ in $\mathbf{u}_{K,t}(\mathbf{x})$ unless required and refer to (4.34) when writing $\mathbf{u}_t$.

4.3.2.4 Safety Constraints

The derived multi-step ahead prediction technique provides a sequence of ellipsoidal confidence regions around trajectories of the true system f through (4.32). We can use these to verify finite-time safety constraints in terms of reachability and constraint satisfaction.

A particular class of constraints that we consider are linear constraints on states and control inputs. In particular, we define the constraints from Section 2.4.1.2 as

$$-c_x(\mathbf{x}) = \mathbf{H}_x \mathbf{x} - \mathbf{h}_x \leq 0 \tag{4.37}$$

$$-c_u(\mathbf{u}) = \mathbf{H}_u \mathbf{u} - \mathbf{h}_u \leq 0 \tag{4.38}$$

where $\mathbf{H}_x \in \mathbb{R}^{m_x \times p}$, $\mathbf{h}_x \in \mathbb{R}^{m_x}$, $\mathbf{H}_u \in \mathbb{R}^{m_x \times q}$, and $\mathbf{h}_u \in \mathbb{R}^{m_u}$ are given and fixed matrices. The linear safety constraints must be satisfied for all time steps as in (2.24).

We can guarantee that the system trajectory predicted according to Corollary 6 satisfies the constraints in (4.37) and (4.38) by verifying that the computed confidence ellipsoids

are contained in the feasible region (2.25) and (2.26) that corresponds to the polytopic constraints. For the special case of linear constraints and ellipsoidal over-approximations, this is feasible and can be verified with

$$\mathcal{R}_{t+1} \subset \mathcal{X},\ \mathbf{u}_t(\mathcal{R}_t) \subset \mathcal{U},\ t = 0, .., T-1, \tag{4.39}$$

where $(\mathcal{R}_0, \dots, \mathcal{R}_T)$ is given through (4.32) and $\mathbf{u}_t(\mathcal{R}_t) := \{\mathbf{u}_{K,t}(\mathbf{x}) | \mathbf{x} \in \mathcal{R}_t\}$.

Since our constraints are polytopes, the feasible region $\mathcal{X}$ is defined as the intersection of m_x linear constraints, $\mathcal{X} = \bigcap_{i=1}^{m_x} \mathcal{X}_i$, where $\mathcal{X}_i = \{\mathbf{x} \in \mathbb{R}^p \,|\, [H_x]_{(i,\cdot)}\mathbf{x} - [h_x]_i \leq 0\}$ and $[\mathbf{H}_x]_{(i,\cdot)}$ is the ith row of $\mathbf{H}_x$. We can now formulate the state constraints through the condition $\mathcal{R}_t = \mathcal{E}(\mathbf{p}_t, \mathbf{Q}_t) \subset \mathcal{X}$ as m_x individual constraints $\mathcal{R}_t \subset \mathcal{X}_i$, $i = 1, .., m_x$, for which an analytical formulation exists (Hessem and Bosgra, 2002),

$$[\mathbf{H}_x]_{(i,\cdot)}\mathbf{p}_t + \sqrt{[\mathbf{H}_x]_{(i,\cdot)}\mathbf{Q}_t[\mathbf{H}_x]^T_{(i,\cdot)}} \leq [\mathbf{h}_x]_i, \tag{4.40}$$

$\forall i \in \{1, .., m_x\}$. Moreover, we can use the fact that $\mathbf{u}_t$ is affine in $\mathbf{x}_t$ to verify the input constraints. In particular, we use (4.17) to obtain $\mathbf{u}_t(\mathcal{R}_t) = \mathcal{E}(\mathbf{k}_t, \mathbf{K}_t\mathbf{Q}_t, \mathbf{K}_t^T)$, so that the corresponding control constraint $\mathbf{u}_t(\mathcal{R}_t) \subset \mathcal{U}$ can be equivalently written as

$$[\mathbf{H}_u]_{(i,\cdot)}\mathbf{k}_t + \sqrt{[\mathbf{H}_u]_{(i,\cdot)}\mathbf{K}_t\mathbf{Q}_t\mathbf{K}_t^{\mathrm{T}}[\mathbf{H}_u]^{\mathrm{T}}_{(i,\cdot)}} \leq [\mathbf{h}_u]_i, \tag{4.41}$$

$\forall i \in \{1, \dots, m_u\}$. This provides us with a closed-form expression of our safety constraints (4.39) that deterministically guarantees the safety of our system over an arbitrary finite horizon T, *given that* the system is contained in the sequence of ellipsoids $\mathcal{R}_t, t = 0, \dots, T$. Hence, these constraints are as reliable as our multi-step ahead prediction technique and, consequently, as reliable as our statistical model.

These results can also be used to verify any reachability condition that is framed as a polytopic constraint at a fixed time step t.

4.4 Conclusion

In this chapter, we stated formal assumptions that allow us to learn reliable models of a dynamical system in (4.1). We then showed how these reliable models can be used to analyze the stability of a fixed control policy. Lastly, we provided a reliable uncertainty propagation scheme, which can be used to verify properties of the system over a finite time horizon. We use all these properties in the following chapter to analyze safe reinforcement learning algorithms.

Other Related Work There are several publications which are not part of this dissertation, but that were written during the course of the PhD and are relevant to this chapter.

1. In (Melchior et al., 2019), we propose a variational inference scheme in order to learn Gaussian process models of partially observed systems.

2. In (Berkenkamp, Moriconi, et al., 2016), we investigate a continuous-time variant of the Lyapunov stability guarantees in Section 4.2.

5

Safe Exploration for Model-based Reinforcement Learning

Some of the results in this chapter have been previously published in (Berkenkamp, Moriconi, et al., 2016; Berkenkamp, Turchetta, et al., 2017; Koller, Berkenkamp, Turchetta, Boedecker, et al., 2019). Partial results of the last paper were shown in (Koller, Berkenkamp, Turchetta, and Krause, 2018).

In Chapter 4, we analyze a fixed control policy or a fixed sequence of control inputs for safety in terms of both stability and constraint satisfaction. However, the goal of reinforcement learning is not to analyze policies, but rather to synthesize them. That is, we want to actively learn about our dynamics model f in order to compute better policies.

In this section, we investigate how the analysis tools from Chapter 4 can be used for safe exploration in reinforcement learning. For safe exploration, we can only visit a particular state if it is contained within the region of attraction of an asymptotically stable policy. This means that for any state that we visit under our exploration policy, we must always have a backup policy that stabilizes the system starting form this state. Moreover, we require that state and input constraints (2.22) and (2.23) must hold throughout the exploration process. Note that this is a constraint both on the backup policy and on the exploration policy.

We first consider the problem of safely learning a backup policy in Section 5.1 under an

idealized exploration scheme. Next in Section 5.2, we propose a model predictive control scheme that provides safety guarantees during the exploration process. Lastly, we introduce an exploration scheme in Section 5.3 that has provably sublinear regret without safety constraints and extend it to the safety-constrained setting.

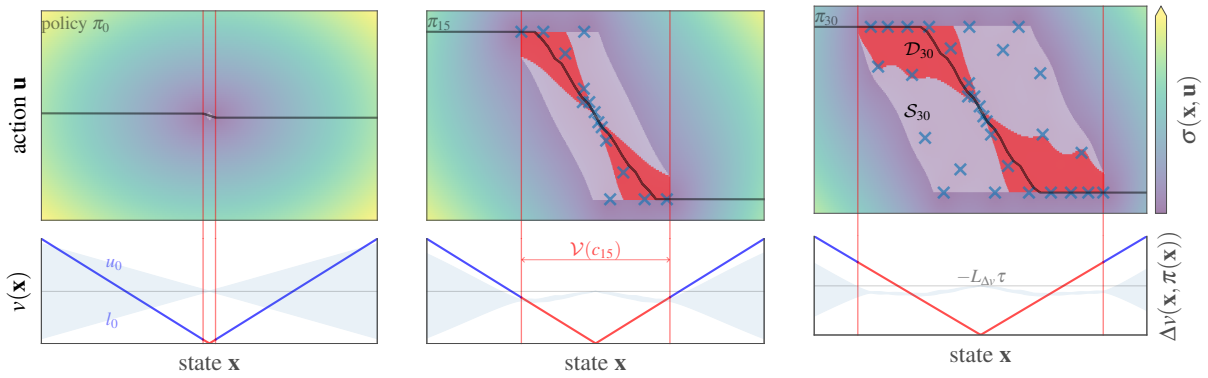
5.1 Exploration by Uncertainty Sampling

As a first step towards safe reinforcement learning, we consider the problem of learning a policy subject to safety constraints. In particular, we want to safely learn about the dynamics function f from measurements and adapt the policy for performance, without encountering system failures. Specifically, we define safety in terms of stability as in Section 2.4.1.1 together with the state and input constraints encoded through the sets $\mathcal{X}$ and $\mathcal{U}$. That means, we must learn a control policy $\pi\colon \mathbb{R}^p \to \mathbb{R}^q$ that, given the current state, determines the appropriate control action that drives the system to an equilibrium point as in Section 2.4.1.1, which we set as the origin without loss of generality (Khalil and Grizzle, 1996).

Due to the safety constraints, we can only learn about the dynamics $f(\mathbf{x}, \mathbf{u})$ when we can guarantee that doing so does not drive the system outside of the safe region of attraction under the current policy. Note that the region of attraction is *not known a priori*, but is implicitly defined through the system dynamics and the choice of policy, see Section 4.2. Thus, the policy not only defines performance as in typical reinforcement learning, but also determines safety and where we can obtain measurements.

Next to the safety constraints, we want this system to behave in a certain way, e.g., the car driving on the road. That means we want to achieve high performance eventually as in the general reinforcement learning setting in Section 2.5. We encode the performance requirements of how to drive the system to the origin through a negative reward (cost) function $r(\mathbf{x}, \mathbf{u}) \leq 0$ that is associated with states and actions and has $r(\mathbf{0}, \mathbf{0}) = 0$. The policy aims to maximize the cumulative, discounted reward for each starting state.

As described in Section 2.8, safe reinforcement learning is generally impossible without a safe starting point. Thus, we assume that we have an initial policy π_0 that renders the origin of the system in (4.1) asymptotically stable within some small set of states $\mathcal{S}_0^x$. For example, this policy may be designed using the prior model h in (4.1), since most models



(a) Initial safe set (in red). (b) Exploration, $n = 15$. (c) Final policy, $n = 30$.

Figure 5.1: Example application of Algorithm 2. Due to input constraints, the system becomes unstable for large states. We start from an initial, local policy π_0 that has a small, safe region of attraction (red lines) in Figure 5.1a. The algorithm selects safe, informative state-action pairs within $\mathcal{S}_n$ (top, white shaded), which can be evaluated without leaving the region of attraction $\mathcal{V}(c_n)$ (red lines) of the current policy π_n. As we gather more data (blue crosses), the uncertainty in the model decreases (top, background) and we use (5.2) to update the policy so that it lies within $\mathcal{F}_n$ (top, red shaded) and fulfills the Lyapunov decrease condition. The algorithm converges to the largest safe set in Figure 5.1c. It improves the policy without evaluating unsafe state-action pairs and thereby without system failure.

are locally accurate but deteriorate in quality as state magnitude increases. This policy is explicitly *not safe* to use throughout the state space $\mathcal{X} \setminus \mathcal{S}_0^x$.

Lastly, we assume that a Lipschitz-continuous Lyapunov function $v(\mathbf{x})$ is given as in Section 4.2.

5.1.1 Safe Policy Optimization

In Section 4.2, we have focused on estimating the region of attraction for a fixed policy. We now generalize this framework in order to compare several policies. Since safety in terms of stability is a property of states under a fixed policy, the chosen policy directly determines which states are safe. Specifically, to form a region of attraction all states in the discretization $\mathcal{X}_\tau$ within a level set of the Lyapunov function need to fulfill the decrease condition in Theorem 4, which depends on the selected policy. The set of all state-action

pairs that fulfill this decrease condition is given by

$$\mathcal{F}_n = \Big\{(\mathbf{x}, \mathbf{u}) \in \mathcal{X}_\tau \times \mathcal{U} \,|\, u_n(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}) < -L_{\Delta v}\tau\Big\}, \tag{5.1}$$

see Figure 5.1c (top, red shaded). In order to estimate the region of attraction based on this set, we need to commit to a policy. Specifically, we want to pick the policy that leads to the largest possible region of attraction according to Theorem 4. This requires that for each discrete state in $\mathcal{X}_\tau$ the corresponding state-action pair under the policy must be in the set $\mathcal{F}_n$. Thus, we optimize the policy according to

$$\pi_n, c_n = \underset{\substack{\pi \in \Pi_L, \\ c \in (0, c_{\max})}}{\operatorname{argmax}} c, \quad \text{such that for all } \mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\tau \colon (\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{F}_n, \tag{5.2}$$

where we additionally limit c_n by $c_{\max}$ from (4.14) in order to account for state constraints, see Lemma 5. The region of attraction that corresponds to the optimized policy π_n according to (5.2) is given by $\mathcal{V}(c_n)$, see Figure 5.1b. It is the largest level set of the Lyapunov function for which all state-action pairs $(\mathbf{x}, \pi_n(\mathbf{x}))$ that correspond to discrete states within $\mathcal{V}(c_n) \cap \mathcal{X}_\tau$ are contained in $\mathcal{F}_n$. This means that these state-action pairs fulfill the requirements of Theorem 4 and $\mathcal{V}(c_n)$ is a region of attraction of the true system under policy π_n. The following theorem is thus a direct consequence of Theorem 4 and (5.2).

Theorem 5. *Let $\mathcal{R}_{\pi_n}$ be the true region of attraction of (4.1) under the policy π_n. For any $\delta \in (0, 1)$, we have with probability at least $(1 - \delta)$ that $\mathcal{V}(c_n) \subseteq \mathcal{R}_{\pi_n}$ for all $n > 0$. Moreover, for any $\mathbf{x}_0 \in \mathcal{V}(c_n)$, we have for all $t \geq 0$ that*

$$\mathbf{x}_t \in \mathcal{X}, \qquad \pi(\mathbf{x}_t) \in \mathcal{U} \tag{5.3}$$

Thus, when we optimize the policy subject to the constraint in (5.2), the estimated region of attraction is always an inner approximation of the true region of attraction. However, solving the optimization problem in (5.2) is intractable in general. We approximate the policy update step in Section 5.1.3 to be tractable without violating the safety constraints.

5.1.2 Exploration Guarantees

So far, we have focused on learning a policy π_n that satisfies the safety guarantees. Next, we focus on actively and safely reducing the uncertainty about the transition model f in order to improve the policy.

In particular, it is natural to ask how one might obtain data points in order to improve the model of $g(\cdot)$ and thus efficiently increase the region of attraction. This question is difficult to answer in general, since it depends on the properties of the statistical model. In particular, for general statistical models it is often not clear whether the confidence intervals contract sufficiently quickly. In the following, we exploit the assumptions in Chapter 4 about the dynamics f and make additional assumptions about reachability within $\mathcal{V}(c_n)$ in order to provide exploration guarantees. These assumptions allow us to highlight fundamental requirements for safe data acquisition and that safe exploration is possible.

We want to exploit the stability analysis in Section 4.2 to verify the stability of a learned policy. However, for our exploration analysis we need to ensure that safe state-actions cannot become unsafe; that is, an initial set of safe set $\mathcal{S}_0$ remains safe (defined later). To this end, we intersect the confidence intervals returned by our statistical model: $\mathcal{C}_n(\mathbf{x}, \mathbf{u}) := \mathcal{C}_{n-1} \cap \mathcal{Q}_n(\mathbf{x}, \mathbf{u})$, where the set $\mathcal{C}$ is initialized to $\mathcal{C}_0(\mathbf{x}, \mathbf{u}) = (-\infty, v(\mathbf{x}) - L_{\Delta v}\tau)$ when $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}_0$ and $\mathcal{C}_0(\mathbf{x}, \mathbf{u}) = \mathbb{R}$ otherwise. Note that $v(f(\mathbf{x}, \mathbf{u}))$ is contained in $\mathcal{C}_n(\mathbf{x}, \mathbf{u})$ with the same $(1 - \delta)$ probability as in Corollary 4. We redefine the upper and lower bounds on $v(f(\cdot))$ as $u_n(\mathbf{x}, \mathbf{u}) := \max \mathcal{C}_n(\mathbf{x}, \mathbf{u})$ and $l_n(\mathbf{x}, \mathbf{u}) := \min \mathcal{C}_n(\mathbf{x}, \mathbf{u})$.

In order to quantify the exploration properties of our algorithm, we consider a discrete action space $\mathcal{U}_\tau \subset \mathcal{U}$. We define exploration as the number of state-action pairs in $\mathcal{X}_\tau \times \mathcal{U}_\tau$ that we can safely learn about without leaving the true region of attraction. Note that, despite this discretization, the policy takes values on the continuous domain. Moreover, instead of using the confidence intervals directly as in (5.2), we consider an algorithm that uses the Lipschitz constants to slowly expand the safe set, similar to the model-free setting in Chapter 3. We use this in our analysis to quantify the ability to generalize beyond the current safe set. In practice, nearby states are sufficiently correlated under the model to enable generalization using (5.1).

Suppose we are given a set $\mathcal{S}_0$ of state-action pairs about which we can learn safely. Specifically, this means that we have a policy such that, for any state-action pair $(\mathbf{x}, \mathbf{u})$ in $\mathcal{S}_0$, if we apply action $\mathbf{u}$ in state $\mathbf{x}$ and then apply actions according to the policy, the state converges to the origin. Such a set can be constructed using the initial safe policy π_0 as $\mathcal{S}_0 = \{(\mathbf{x}, \pi_0(\mathbf{x})) \,|\, \mathbf{x} \in \mathcal{S}_0^x\}$. Starting from this set, we want to update the policy to expand the region of attraction according to Theorem 4. To this end, we use the confidence intervals on $v(f(\cdot))$ for states inside $\mathcal{S}_0$ to determine state-action pairs that

fulfill the decrease condition. We thus redefine $\mathcal{F}_n$ for the exploration analysis to

$$\mathcal{F}_n = \bigcup_{(\mathbf{x},\mathbf{u})\in\mathcal{S}_{n-1}} \left\{\mathbf{a}' \in \mathcal{X}_\tau \times \mathcal{U}_\tau \,|\, u_n(\mathbf{x},\mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|\mathbf{a}' - (\mathbf{x},\mathbf{u})\|_2 < -L_{\Delta v}\tau\right\}. \tag{5.4}$$

This formulation is equivalent to (5.1), except that it uses the Lipschitz constant to generalize safety. Given $\mathcal{F}_n$, we can again find a region of attraction $\mathcal{V}(c_n)$ by committing to a policy according to (5.2). In order to expand this region of attraction effectively, we need to decrease the posterior model uncertainty about the dynamics of the Gaussian process by collecting measurements. However, to ensure safety in terms of stability, we are not only restricted to states within $\mathcal{V}(c_n)$, but also need to ensure that the state after taking an action is safe; that is, the dynamics map the state back into the region of attraction $\mathcal{V}(c_n)$. We again use the Lipschitz constant in order to determine this set,

$$\mathcal{S}_n = \bigcup_{\mathbf{a}\in\mathcal{S}_{n-1}} \left\{\mathbf{a}' \in \mathcal{V}(c_n) \cap \mathcal{X}_\tau \times \mathcal{U}_\tau \,|\, u_n(\mathbf{a}) + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \leq c_n\right\}. \tag{5.5}$$

The set $\mathcal{S}_n$ contains state-action pairs that we can safely evaluate under the current policy π_n without leaving the region of attraction, see Figure 5.1 (top, white shaded).

What remains is to define a strategy for collecting data points within $\mathcal{S}_n$ to effectively decrease the model uncertainty. We specifically focus on the high-level requirements for any exploration scheme without committing to a specific method. In practice, any (model-based) exploration strategy that aims to decrease the model uncertainty by driving the system to specific states may be used. Safety can be ensured by picking actions according to π_n whenever the exploration strategy reaches the boundary of the safe region $\mathcal{V}(c_n)$; that is, when $u_n(\mathbf{x},\mathbf{u}) > c_n$. This way, we can use π_n as a backup policy for exploration.

The high-level goal of the exploration strategy is to shrink the confidence intervals at state-action pairs $\mathcal{S}_n$ in order to expand the safe region. Specifically, the exploration strategy should aim to visit state-action pairs in $\mathcal{S}_n$ at which we are the most uncertain about the dynamics; that is, where the confidence interval is the largest:

$$(\mathbf{x}_n,\mathbf{u}_n) = \operatorname*{argmax}_{(\mathbf{x},\mathbf{u})\in\mathcal{S}_n} u_n(\mathbf{x},\mathbf{u}) - l_n(\mathbf{x},\mathbf{u}). \tag{5.6}$$

As we keep collecting data points according to (5.6), we decrease the uncertainty about the dynamics for different actions throughout the region of attraction and adapt the policy, until eventually we have gathered enough information in order to expand it. While (5.6) implicitly assumes that any state within $\mathcal{V}(c_n)$ can be reached by the exploration policy, it achieves the high-level goal of any exploration algorithm that aims to reduce model

uncertainty. In practice, any safe exploration scheme is limited by unreachable parts of the state space.

We compare the active learning scheme in (5.6) to an oracle baseline that starts from the same initial safe set $\mathcal{S}_0$ and knows $v(f(\mathbf{x}, \mathbf{u}))$ up to ϵ accuracy within the safe set. The oracle also uses knowledge about the Lipschitz constants and the optimal policy in Π_L at each iteration. We denote the set that this baseline manages to determine as safe with $\overline{R}_\epsilon(\mathcal{S}_0)$ and provide a detailed definition in Appendix C.1.

Theorem 6. *Under the assumptions of Theorem 4 and Corollary 3, with β_n as in Corollary 4, and with measurements collected according to (5.6), let n^* be the smallest positive integer so that*

$$\frac{n^*}{\beta_{n^*}^2 \gamma_{n^*}} \geq \frac{Cp(|\overline{R}(\mathcal{S}_0)| + 1)}{L_v^2 \epsilon^2}, \tag{5.7}$$

where $C = 8/\log(1+\sigma^{-2})$. Let $\mathcal{R}_\pi$ be the true region of attraction of (4.1) under a policy π. For any $\epsilon > 0$, and $\delta \in (0, 1)$, the following holds jointly with probability at least $(1 - \delta)$ for all $n > 0$:

(i) $\mathcal{V}(c_n) \subseteq \mathcal{R}_{\pi_n}$,

(ii) $\mathcal{V}(c_n) \subseteq \mathcal{X}$,

(iii) $f(\mathbf{x}, \mathbf{u}) \in \mathcal{R}_{\pi_n} \, \forall (\mathbf{x}, \mathbf{u}) \in \mathcal{S}_n$,

(iv) $\overline{R}_\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_n \subseteq \overline{R}_0(\mathcal{S}_0)$.

Theorem 6 states that, when selecting data points according to (5.6), the estimated region of attraction $\mathcal{V}(c_n)$ is (i) contained in the true region of attraction under the current policy. Moreover, as a consequence of Corollary 10 we have in (ii) that the resulting region of attraction is a subset of the state constraints. In particular, this means that for any n the policy π_n never violates the state constraints when it starts in $\mathcal{V}(c_n)$.

Next to the safety of the policy, (iii) ensures that the selected data points do not cause the system to leave the region of attraction. This means that any exploration method that considers the safety constraint (5.5) is able to safely learn about the system without leaving the region of attraction. The last part of Theorem 6, (iv), states that after a finite number of data points n^* we achieve at least the exploration performance of the oracle baseline, while we do not classify unsafe state-action pairs as safe. This means that the

Algorithm 2 SafeLyapunovLearning

Input: Initial safe policy π_0, Gaussian process dynamics model $\mathcal{GP}(\mu(\mathbf{a}), k(\mathbf{a}, \mathbf{a}'))$
for all $n = 1, \dots$ **do**
  Compute policy π_n via SGD on (5.8)
  $c_n = \operatorname{argmax}_{c<c_{\max}} c$, such that $\forall \mathbf{x} \in \mathcal{V}(c_n) \cap \mathcal{X}_\tau \colon u_n(\mathbf{x}, \pi_n(\mathbf{x})) - v(\mathbf{x}) < -L_{\Delta v}\tau$
  $\mathcal{S}_n = \{(\mathbf{x}, \mathbf{u}) \in \mathcal{V}(c_n) \times \mathcal{U}_\tau \,|\, u_n(\mathbf{x}, \mathbf{u}) \leq c_n\}$
  Select $(\mathbf{x}_n, \mathbf{u}_n)$ within $\mathcal{S}_n$ using (5.6) and drive system there with backup policy π_n
  Update Gaussian process with measurements $f(\mathbf{x}_n, \mathbf{u}_n) + \epsilon_n$

algorithm explores the largest region of attraction possible for a given Lyapunov function with residual uncertain about $v(f(\cdot))$ smaller than ϵ. Details of the comparison baseline are given in the appendix. In practice, this means that any exploration method that manages to reduce the maximal uncertainty about the dynamics within $\mathcal{S}_n$ is able to expand the region of attraction.

An example run of repeatedly evaluating (5.6) for a one-dimensional state-space is shown in Figure 5.1. The algorithm starts with the initial policy in Figure 5.1a, which encodes only a small region of attraction (red lines) under the Lyapunov function (blue lines, bottom) due to the significant model uncertainty (background top) that leads to significant uncertainty about the decrease in the future value (bottom plot, blue shaded). Nonetheless, this small region is sufficient to collect initial data points within $\mathcal{S}_0$ (blue shaded, top plot). As we gather more and more data, the model improves and the safe region expands in Figure 5.1b. At the same time the policy is optimized for performance, which additionally increases the safe set. Any Lipschitz-continuous policy with state-action pairs in $\mathcal{F}_n$ (red region) can be chosen without decreasing the region of attraction. Eventually in Figure 5.1c, the uncertainty about the model has decreased significantly and we find a close-to-optimal policy. Thus, by only selecting data points within the current estimate of the region of attraction, the algorithm can efficiently optimize the policy and expand the safe region over time and simultaneously optimize the performance of the policy.

5.1.3 Practical Implementation and Experiments

In the previous section, we have given strong theoretical results on safety and exploration for an idealized algorithm that can solve (5.2). In this section, we provide a practical variant of the theoretical algorithm in the previous section. In particular, while we retain

safety guarantees, we sacrifice exploration guarantees to obtain a more practical algorithm. This is summarized in Algorithm 2.

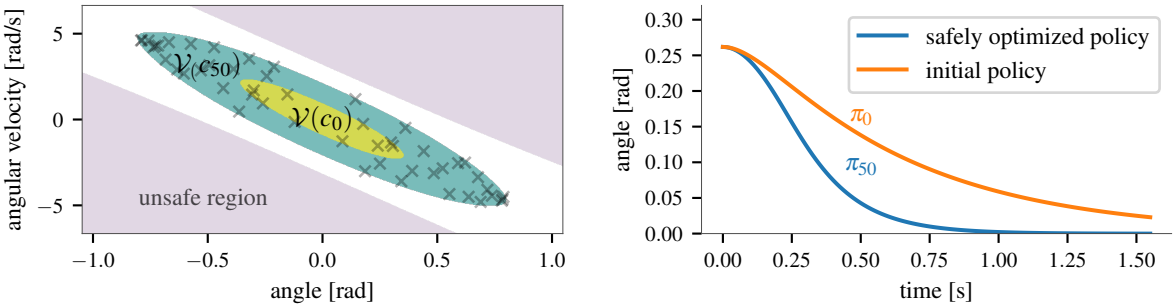
The policy optimization problem in (5.2) is intractable to solve and only considers safety, rather than a performance metric. We propose to use an approximate policy update that that maximizes approximate performance while providing stability guarantees. It proceeds by optimizing the policy first and then computes the region of attraction $\mathcal{V}(c_n)$ for the new, fixed policy. This does not impact safety, since data is still only collected inside the region of attraction. Moreover, should the optimization fail and the region of attraction decrease, one can always revert to the previous policy, which is guaranteed to be safe.

In our experiments, we use approximate dynamic programming (Powell, 2007) to capture the performance of the policy. Given a policy π_θ with parameters θ, we compute an estimate of the value function $J_{\pi_\theta}(\cdot)$ for the mean dynamics μ_n based on the reward $r(\mathbf{x}, \mathbf{u}) \leq 0$. In principle, one could also optimize the expected performance over the epistemic uncertainty by using the reparameterization trick, see Section 2.3. At each state, $J_{\pi_\theta}(\mathbf{x})$ is the sum of γ-discounted rewards encountered when following the policy π_θ. The goal is to adapt the parameters of the policy for maximum performance as measured by J_{π_θ}, while ensuring that the safety constraint on the worst-case decrease on the Lyapunov function in Theorem 4 is not violated. A Lagrangian formulation to this constrained optimization problem is

$$\pi_n = \underset{\pi_\theta \in \Pi_L}{\operatorname{argmax}} \int_{\mathbf{x} \in \mathcal{X}} r(\mathbf{x}, \pi_\theta(\mathbf{x})) + \gamma J_{\pi_\theta}(\mu_{n-1}(\mathbf{x}, \pi_\theta(\mathbf{x})) - \lambda \Big(u_n(\mathbf{x}, \pi_\theta(\mathbf{x})) - v(\mathbf{x}) + L_{\Delta v} \tau \Big), \tag{5.8}$$

where the first term measures long-term performance and $\lambda \geq 0$ is a Lagrange multiplier for the safety constraint from Theorem 4. In our experiments, we use the negative value function as a Lyapunov function candidate, $v = -J$ with $r(\cdot, \cdot) \leq 0$, and set $\lambda = 1$. In this case, (5.8) corresponds to an high-probability lower bound on the performance given the uncertainty in the dynamics. This is similar to worst-case performance formulations found in robust MDPs (Tamar et al., 2014; Wiesemann et al., 2012), which consider worst-case value functions given parametric uncertainty in MDP transition model. Moreover, since $L_{\Delta v}$ depends on the Lipschitz constant of the policy, this simultaneously serves as a regularizer on the parameters θ. Note that, in practice, one could also pose an outer optimization problem to optimize λ.

To verify safety, we use the Gaussian process confidence intervals l_n and u_n directly, as in (5.1). We also use the confidence intervals to compute $\mathcal{S}_n$ for the active learning scheme, see Algorithm 2, Line 5. In practice, we do not need to compute the entire set $\mathcal{S}_n$ to



(a) Estimated safe set.

(b) State trajectory (lower is better).

Figure 5.2: Optimization results for an inverted pendulum. Figure 5.2a shows the initial safe set (yellow) under the policy π_0, while the green region represents the estimated region of attraction under the optimized neural network policy. It is contained within the true region of attraction (white). Figure 5.2b shows the improved performance of the safely learned policy over the policy for the prior model.

solve (5.2), but can use a global optimization method or even a random sampling scheme within $\mathcal{V}(c_n)$ to find suitable state-actions.

Moreover, measurements for actions that are far away from the current policy are unlikely to expand $\mathcal{V}(c_n)$, since we optimize (5.8) via gradient descent and the policy therefore only changes locally, see Figure 5.1c. Thus, we can achieve better data and computational efficiency by restricting the exploratory actions $\mathbf{u}$ with $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}_n$ to be close to π_n, $\mathbf{u} \in [\pi_n(\mathbf{x}) - \bar{u}, \pi_n(\mathbf{x}) + \bar{u}]$ for some constant $\bar{u}$.

Computing the region of attraction by verifying the stability condition on a discretized domain suffers from the curse of dimensionality. However, it is not necessary to update policies in real time. In particular, since any policy that is returned by the algorithm is provably safe within some level set, any of these policies can be used safely for an arbitrary number of time steps. To scale this method to higher-dimensional system, one would have to consider an adaptive discretization for the verification as in (Bobiti and Lazar, 2016).

Experiments A Python implementation of Algorithm 2 and the experiments based on TensorFlow (Abadi et al., 2016) and GPflow (Matthews et al., 2017) is available at `https://github.com/befelix/safe_learning`.

We verify our approach on an inverted pendulum benchmark problem. The true, continuous-time dynamics are given by $ml^2\ddot{\psi} = gml\sin(\psi) - \lambda\dot{\psi} + u$, where ψ is the angle, m the

mass, g the gravitational constant, and u the torque applied to the pendulum. The control torque is limited, so that the pendulum necessarily falls down beyond a certain angle. We use a Gaussian process model for the *discrete-time* dynamics, where the mean dynamics are given by a linearized and discretized model of the true dynamics that considers a wrong, lower mass and neglects friction. As a result, the optimal policy for the mean dynamics does not perform well and has a small region of attraction as it underactuates the system. We use a combination of linear and Matérn kernels in order to capture the model errors that result from parameter and integration errors.

For the policy, we use a neural network with two hidden layers and 32 neurons with ReLU activations each. We compute a conservative estimate of the Lipschitz constant as in (Szegedy et al., 2014). We use standard approximate dynamic programming with a quadratic, normalized reward $r(\mathbf{x}, \mathbf{u}) = -\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} - \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}$, where $\mathbf{Q}$ and $\mathbf{R}$ are positive-definite, to compute the value function J_{π_θ}. Specifically, we use a piecewise-linear triangulation of the state-space as to approximate J_{π_θ}, see (Davies, 1996). In practice, one may use other function approximators. We optimize the policy via stochastic gradient descent on (5.8), where we sample a finite subset of $\mathcal{X}$ and replace the integral in (5.8) with a sum.

The theoretical confidence intervals for the Gaussian process model are conservative. To enable more data-efficient learning, we fix $\beta_n = 2$. This corresponds to a marginal high-probability decrease condition per-state, rather than jointly over the state space. Moreover, we use local Lipschitz constants of the Lyapunov function rather than the global one. While this does not affect guarantees, it greatly speeds up exploration.

For the initial policy, we use approximate dynamic programming to compute the optimal policy for the prior mean dynamics. This policy is unstable for large deviations from the initial state and has poor performance, as shown in Figure 5.2b. Under this initial, suboptimal policy, the system is stable within a small region of the state-space in Figure 5.2a. Starting from this initial safe set, the algorithm proceeds to collect safe data points and improve the policy. As the uncertainty about the dynamics decreases, the policy improves and the estimated region of attraction increases. The region of attraction after 50 data points is shown in Figure 5.2a. The resulting set $\mathcal{V}(c_n)$ is contained within the true safe region of the optimized policy π_n. At the same time, the control performance improves drastically relative to the initial policy, as can be seen in Figure 5.2b. Overall, the approach enables safe learning about dynamic systems, as all data points collected during learning are safely collected under the current policy.

5.2 Safe Exploration with Model Predictive Control

In Section 5.1, we provided safety and exploration guarantees for an idealized exploration scheme that can safely drive the system to any desired state and must return to the safe set in one step. In this section, we introduce a model predictive control scheme that can be used as a safe exploration scheme to drive the system to the desired states over multiple steps.

We take the safe return policy π_n from the previous section as a starting point. For generality, we explicitly assume that we are given a safe controller that renders some part of the state space, $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$ forward invariant and thus safe.

Assumption 8. We are given a controller $\pi_{\text{safe}}(\cdot)$ and a polytopic safe region

$$\mathcal{X}_{\text{safe}} := \{\mathbf{x} \in \mathbb{R}^p \,|\, \mathbf{H}_s \mathbf{x} \leq \mathbf{h}_s\} \subseteq \mathcal{X}, \tag{5.9}$$

where $\mathbf{H}_s \in \mathbb{R}^{m_s \times p}$ and $\mathbf{h}_s \in \mathbb{R}^{m_s}$ so that $\mathcal{X}_{\text{safe}}$ forms the intersection of $m_s \in \mathbb{N}$ individual half-spaces. We denote with $\mathbf{x}_{t+1} = f_{\pi_{\text{safe}}} = f(\mathbf{x}_t, \pi_{\text{safe}}(\mathbf{x}_t))$ the closed-loop system under π_{safe} and assume for any $t' \in \mathbb{N}$ that

$$\mathbf{x}_{t'} \in \mathcal{X}_{\text{safe}} \Rightarrow f_{\pi_{\text{safe}}}(\mathbf{x}_t) \in \mathcal{X}, \quad \forall t \geq t'. \tag{5.10}$$

For example, any inner approximation to $\mathcal{V}(c_n)$ together with $\pi_{\text{safe}} = \pi_n$ from the previous section fulfill these requirements. Note that Assumption 8 is weaker than assuming that $\mathcal{X}_{\text{safe}}$ is positive invariant, since it only requires the state constraints not to be violated. We can use the results from the finite-time trajectory analysis in Section 4.3 directly to define a safe exploration scheme given any objective function $J(\cdot)$:

$$\max_{\mathbf{u}_0,\ldots,\mathbf{u}_{T-1}} J(\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}) \tag{5.11a}$$

$$\text{s.t. } \mathcal{R}_{t+1} = \tilde{m}(\mathcal{R}_t, \mathbf{u}_t),\ t = 0, \ldots, T-1, \tag{5.11b}$$

$$\mathcal{R}_t \subset \mathcal{X},\ t = 1, \ldots, T-1, \tag{5.11c}$$

$$\mathbf{u}_t(\mathcal{R}_t) \subset \mathcal{U},\ t = 0, \ldots, T-1, \tag{5.11d}$$

$$\mathcal{R}_T \subset \mathcal{X}_{\text{safe}}, \tag{5.11e}$$

where $\mathcal{R}_0 := \{\mathbf{x}_t\}$ is the current state of the system and the intermediate state and control constraints are defined in (4.40)and (4.41), respectively. Since the terminal set $\mathcal{X}_{\text{safe}}$ is a

Algorithm 3 Safe Exploration with model predictive control

Inputs: Safe policy π_{safe},
dynamics model h,
statistical model (μ_0, Σ_0).

$\Pi_0 \leftarrow \{\pi_{\text{safe}}, \dots, \pi_{\text{safe}}\}$ with $|\Pi_0| = T$
for $t = 0, 1, \dots$ **do**
  feasible, $\Pi' \leftarrow$ solve model predictive control problem (5.11)
  if feasible **then**
    $\Pi_t \leftarrow \Pi'$
  else
    $\Pi_t \leftarrow (\Pi_{t-1,1:T-1}, \pi_{\text{safe}})$
  $\mathbf{x}_{t+1} \leftarrow$ apply $\mathbf{u}_t = [\Pi_t]_0(\mathbf{x}_t)$ to the system (2.2)
  Update statistical model with transition $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$.

polytope according to Assumption 8, the constraint $\mathcal{R}_T \subset \mathcal{X}_{\text{safe}}$ has has the same form as the state constraints in (4.40) and can be formulated the same way. For now, we assume an arbitrary objective function J and discuss how to choose it to solve a reinforcement learning task in Sections 5.2.1 and 5.3.

Due to the terminal constraint $\mathcal{R}_T \subset \mathcal{X}_{\text{safe}}$, a solution to (5.11) provides a sequence of feedback controllers $\mathbf{u}_0, \dots, \mathbf{u}_T$ that steers the system back to the safe set $\mathcal{X}_{\text{safe}}$. However, since the dynamics are learned online we cannot show that the model predictive control problem is recursively feasible so that a solution exists for all t and n. In particular, Corollary 4 only ensures that the the true dynamics are contained in the confidence intervals, not that the confidence intervals are strictly decreasing. As a consequence, a model update based on measurements might mean that $\mathcal{R}_t$ contains new states that render the problem infeasible. Moreover, we would have to carefully deal with the nonlinearity and non-convexity of our model predictive control problem, as e.g. Simon et al. (2013), and the fact that our terminal set is not necessarily robust control positive invariant, which is a pre-requisite in many robust model predictive control approaches (Rawlings and Mayne, 2009).

We employ a control scheme similar to standard robust model predictive control to guarantee recursively that, at any time step t, there exists a sequence of control inputs (control laws) that steer the system back to $\mathcal{X}_{\text{safe}}$. In particular, given a feasible solution $\Pi_t = (\mathbf{u}_t^0, \dots, \mathbf{u}_t^{T-1})$ to (5.11) at time t, we apply the first control input $\mathbf{u}_t^0$. In case

we do not find a feasible solution to (5.11) at the next time step, we shift the previous solution in a receding horizon fashion and append π_{safe} to the sequence to obtain $\Pi_{t+1} = (\mathbf{u}_t^1, \ldots, \mathbf{u}_t^{T-1}, \pi_{\text{safe}})$. We repeat this process until a new feasible solution exists that replaces the previous input sequence. This procedure is summarized in Algorithm 3 and provide safety guarantees in the following theorem.

Theorem 7. *Under the assumptions in Corollary 6, let the controller defined as in Algorithm 3,* $\mathbf{x}_0 \in \mathcal{X}_{\text{safe}}$. *The following hold jointly with probability at least* $(1-\delta)$ *for all* $t \geq 0$ *and* $n \geq 0$

(i) $\mathbf{x}_t \in \mathcal{X}$, $\mathbf{u}_t \in \mathcal{U}$

(ii) *After applying the sequence of control inputs/laws in* Π_t *we have* $\mathbf{x}_{t+T} \in \mathcal{X}_{\text{safe}}$.

Proof. From Corollary 6, the ellipsoidal outer approximations hold uniformly with probability at least $(1-\delta)$. As a consequence, any feasible solution to (5.11) satisfies (ii) and the constraints over the next T steps. We prove the result by induction.

If (5.11) is infeasible, (i) and (ii) follow from the properties of the backup controller π_{safe} in Assumption 8. Otherwise, the controller returned from (5.11) satisfies (i) and (ii) as a consequence of Corollary 6 and the terminal set constraint that leads to $\mathbf{x}_{t+T} \in \mathcal{X}_{\text{safe}}$.

Induction step: Let the previous controller Π_t satisfy (i) and (ii). At time step $t+1$, if (5.11) is infeasible then Π_t leads to a state $\mathbf{x}_{t+T} \in \mathcal{X}_{\text{safe}}$, from which the backup-controller satisfies (i) and (ii) by Assumption 8. If (5.11) is feasible, then the return path satisfies (i) and (ii) by Corollary 6. □

Theorem 7 ensures that the control algorithm in Corollary 6 satisfies both the state and input constraints for all time steps and has access to a safe return strategy back to the safe set $\mathcal{X}_{\text{safe}}$ at all times.

5.2.1 Safety and Performance

So far we have considered only safety. However, in practice ensuring safety over long time-horizons can be difficult, as the size of ellipsoids tends to increase as the prediction horizon increases. At the same time, reinforcement learning often requires planning over longer time horizons in order to achieve the best performance. To avoid these issues, we

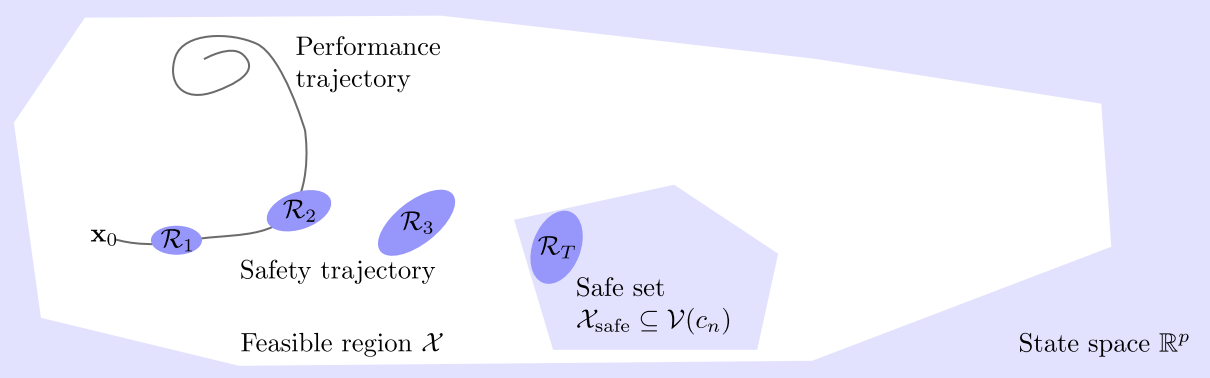


Figure 5.3: Illustration of the model predictive control problem in (5.12) for $r = 2$. As in Algorithm 3, we plan a safety trajectory based on outer approximations $\mathcal{R}$ of the states over T steps. To improve performance, we plan a performance trajectory over a different time horizon of H steps, but only enforce that the first $r \geq 1$ control inputs coincide in order to guarantee safety.

propose to optimize performance over a different time horizon and control inputs H. In particular, we extend the safety-constrained model predictive control problem in (5.11) with additional control inputs $\mathbf{u}_0^{\text{perf}}, \ldots, \mathbf{u}_{H-1}^{\text{perf}}$ that are used to optimize performance. Note that the corresponding objective $J(\mathbf{u}_0^{\text{perf}}, \ldots, \mathbf{u}_{H-1}^{\text{perf}})$ does not have to use the learned model from Corollary 4 or the uncertainty propagation, but instead can define its own performance criteria. As such, without further constraints this setting incorporates any unsafe model predictive control scheme.

In order to guarantee safety in this setting as in Theorem 7, we must ensure that the first o control inputs correspond to the ones in the safety-constrained setting, with $1 \leq o < H$. That is, we apply the best possible actions that maximize the performance criterion J, subject to the constraint that after r time steps a feasible, safe trajectory to $\mathcal{X}_{\text{safe}}$ must exist.

Intuitively, since we apply only the first control input to the system before replanning, even for $o = 1$ the recursive safety properties of Theorem 7 hold. This is illustrated in Figure 5.3. The performance criterion J and the corresponding trajectory (gray line) can be arbitrary. However, simultaneously planning a safety trajectory based on the ellipsoids $\mathcal{R}_t$ that coincides with the performance trajectory for the first o steps, we can recursively guarantee safety as in Theorem 7. The corresponding safety-constrained model predictive

control problem is given by

$$\max_{\substack{\mathbf{u}_0,\dots,\mathbf{u}_{T-1} \\ \mathbf{u}_0^{\text{perf}},\dots,\mathbf{u}_{H-1}^{\text{perf}}}} J(\mathbf{u}_0^{\text{perf}},\dots,\mathbf{u}_{H-1}^{\text{perf}}) \tag{5.12a}$$

$$\text{s.t. } (5.11\text{b})\text{–}(5.11\text{e}) \tag{5.12b}$$

$$\mathbf{u}_t = \mathbf{u}_t^{\text{perf}},\ t = 0,\dots,o-1 \tag{5.12c}$$

For example, this formulation can extend existing exploration schemes that are based on Gaussian processes, e.g. (Boedecker et al., 2014; Xie et al., 2016; Kamthe and Deisenroth, 2018), to the safety-constrained setting.

5.2.2 Practical Considerations

Algorithm 3 theoretically guarantees that the system remains safe, while actively optimizing for performance via the model predictive control problem (5.12). This problem can be solved by commonly used, nonlinear programming solvers, such as the *Interior Point OPTimizer* (Ipopt) by Wächter and Biegler (2006). We consider possible design choices that could improve the performance in a practical application.

Optimizing over affine feedback policies In practice, the affine feedback control structure introduced in Section 4.3.2.3 improves performance significantly. However, optimizing over mbK_t in (4.34) seems to be challenging, both in terms of numerical stability and computational complexity. Hence, we pre-specify the feedback terms in all of our experiments and only optimize over the feed-forward terms.

Lipschitz constants and eigenvalue computations In our multi-step ahead predictions (4.30), we need to solve a generalized eigenvalue problem for every step in the planning horizon. We use the *inverse power iteration*, an iterative method that asymptotically converges to the largest generalized eigenvalue of a pair of matrices (Golub and Loan, 2012). We run the algorithm for a fixed number of p^2 iterations to solve these intermediate eigenvalue problems. In practice, this seems to result in sufficiently accurate estimations.

Eigenvalue problem Due to the generalized eigenvalue problem, the uncertainty propagation (4.30) cannot be solved in closed-form. However, we can still obtain exact derivative

information by means of algorithmic differentiation, that is provided in many state-of-the-art optimization software libraries (Andersson, 2013).

5.2.3 Experiments

In this section, we evaluate the proposed model predictive control scheme in Algorithm 3 to safely learn about the dynamics of an inverted pendulum system. We provide the code to run all experiments detailed in this section in the following Github repository: https://github.com/befelix/safe-exploration.

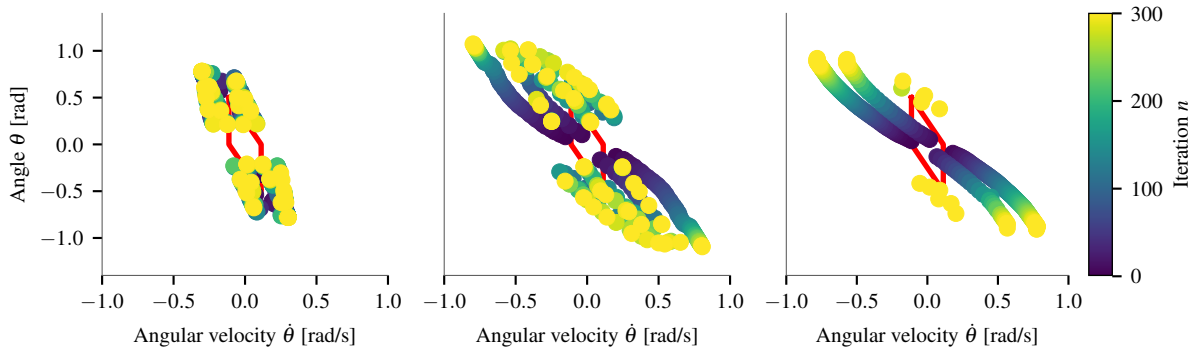
The continuous-time dynamics of the pendulum are given by

$$ml^2\ddot{\theta} = gml\sin(\theta) - \nu\dot{\theta} + \mathbf{u}, \tag{5.13}$$

where $m = 0.15\,\mathrm{kg}$ and $l = 0.5\,\mathrm{m}$ are the mass and length of the pendulum, respectively, $\nu = 0.1\,{}^{\mathrm{Nms}}/_{\mathrm{rad}}$ is a friction parameter, and $g = 9.81\,{}^{\mathrm{m}}/_{\mathrm{s}^2}$ is the gravitational constant. The state of the system $\mathbf{x} = (\theta, \dot{\theta})$ consists of the angle θ and angular velocity $\dot{\theta}$ of the pendulum. The origin with $\theta = 0$ corresponds to the pendulum standing upright. The system is underactuated with control constraints $\mathcal{U} = \{\mathbf{u} \in \mathbb{R} \,|\, |\mathbf{u}| \leq 1\}$. Due to these limits, the pendulum becomes unstable and falls down beyond a certain angle independently of the control policy.

For the prior model h we use the linearized and discretized dynamics that neglect friction and with a pendulum mass that is lower than the one in true system. The safety controller π_{safe} is a discrete-time, infinite horizon linear quadratic regulator (Kwakernaak and Sivan, 1972) of the *true* system f linearized and discretized around the origin with cost matrices $\mathbf{Q} = \mathrm{diag}([1, 2])$, $\mathbf{R} = 20$. The corresponding safety region $\mathcal{X}_{\text{Safe}}$ is given by a conservative polytopic inner-approximation of the true region of attraction of π_{safe}. We do not impose state constraints, i.e. $\mathcal{X} = \mathbb{R}^2$. However the terminal set constraint (5.11e) of the model predictive control problem (5.11) acts as a stability constraint and prevents the pendulum from falling.

To model the dynamical system, we use a Gaussian process with a mixture of linear and Matérn kernels for both systems. Since the theoretical scaling parameter β_n for the confidence intervals in Corollary 4 can be conservative and we choose a fixed value of $\beta_n = 2$ instead, as in Section 5.1. To start our experiment, we initially control the system with the safety controller π_{safe} to gather $n = 25$ initial data points. Moreover, to increase computational efficiency, we limit the number of training points used to update



(a) $T = 1$. (b) $T = 4$. (c) $T = 5$.

Figure 5.4: Visualization of the samples acquired in the static exploration setting for $T \in \{1, 4, 5\}$. The algorithm plans informative paths to the safe set $\mathcal{X}_{\text{safe}}$ (red polytope in the center). The baseline sample set for $T = 1$ (left) is dense around origin of the system. For $T = 4$ (center) we get the optimal trade-off between cautiousness due to a long horizon and limited length of the return trajectory due to a short horizon. The exploration for $T = 5$ (right) is too cautious, since the propagated uncertainty at the final state is too large.

the Gaussian process to 150 and use the *maximum variance* selection procedure (Jain et al., 2018) to sub-select the most informative observations whenever more samples are available. More sophisticated, provably near-optimal selection procedures could be used instead (Krause, A. Singh, et al., 2008).

Since we aim to learn about the model error g, we quantify exploration performance through the mutual information between the observed state transitions and the Gaussian process prior on the unknown model-error g, which can be computed in closed-form, see (2.31).

Static Exploration For a first experiment, we assume that the system is *static*, so that we can reset the system to an arbitrary state $\mathbf{x}_n \in \mathbb{R}^2$ in every iteration. In the static case and without terminal set constraints, a provably close-to-optimal exploration strategy is to, at each iteration n, select state-action pair $\mathbf{a}_{n+1}$ with the largest predictive standard deviation (Srinivas et al., 2012)

$$\mathbf{a}_{n+1} = \arg\max_{z \in \mathcal{X} \times \mathcal{U}} \|\boldsymbol{\sigma}_n(\mathbf{a})\|_2, \tag{5.14}$$

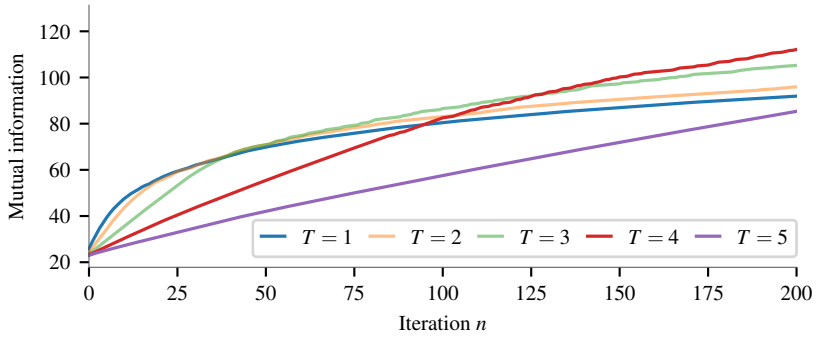


Figure 5.5: Mutual information $I(y_{\mathcal{A}_n}, g)$ after $n = 1, \ldots, 200$ iterations of static exploration with horizon lengths $T \in \{1, \ldots, 5\}$. Exploration settings with shorter horizon gather more informative samples at the beginning, but the short horizon limits the amount of information in the long term. In contrast, longer horizon lengths result in less informative samples at the beginning, due to uncertainties being propagated over long horizons. However, after having gathered some knowledge they quickly outperform the smaller horizon settings.

where $\boldsymbol{\sigma}_n(\cdot)$ is the predictive standard deviation (4.7) at the nth iteration. Inspired by this, at each iteration we collect samples by solving the model predictive control problem (5.11) with objective function $J_n = \|\boldsymbol{\sigma}_n(\mathbf{x}_0, \mathbf{u}_0)\|_2$, where we additionally optimize over the initial state $\mathbf{x}_0 \in \mathcal{X}$. Hence, we visit states where the Gaussian process model is uncertain about the dynamics, but only allow for state-action pairs $(\mathbf{x}_0, \mathbf{u}_0)$ that are part of a feasible return trajectory to the safe set $\mathcal{X}_{\text{Safe}}$.

Since optimizing over the initial state is highly non-convex, we solve the problem iteratively with 25 random initializations to obtain a good approximation of the global minimizer. After every iteration, we update the Gaussian process model with the collected observations.

We apply this procedure for varying horizon lengths. The resulting sample sets are visualized for varying horizon lengths $T \in \{1, \ldots, 5\}$ with 300 iterations in Figure 5.4, while Figure 5.5 shows how the mutual information of the sample sets for different values of T. For short time horizons ($T = 1$), the algorithm can only slowly explore, since it can only move one step outside of the safe set, see Figure 5.4a. This is also reflected in the mutual information gained, which levels off quickly. For a horizon length of $T = 4$ in Figure 5.4b, the algorithm is able to explore a larger part of the state-space, which means that more information is gained. For larger horizons in Figure 5.4c, the predictive uncertainty of the

final state is too large to explore effectively, which slows down exploration initially, when we do not have much information about our system. Note that this is only true over a a finite number of time steps. If the algorithms were to operator for more than 300 iterations, eventually the longest time horizon would perform best. However, in practice the algorithm can probably benefit from adaptively choosing the horizon during operation, e.g. by employing a variable horizon model predictive control approach (A. Richards and How, 2006), or by increasing the horizon when the mutual information saturates for the current horizon. That way, we can initially plan short horizons and only predict longer horizons once the model is confident enough to do so without leading to large over-approximations.

Dynamic Exploration In the previous experiment we considered a system that can be placed at any desired state. Next, we evaluate the algorithm when the system has to be operated continuously, without resetting.

We consider two settings. In the first, we solve the model predictive control problem (5.11) with J_n given by (5.14), similar to the previous experiments. In the second setting, we additionally plan a performance trajectory as proposed in Section 5.2.1. We define the objective-function $J(\cdot) = \sum_{t=0}^{T} \operatorname{trace}(\mathbf{S}_t^{1/2}) - \sum_{t=1}^{T} (\mathbf{m}_t - \mathbf{p}_t)^{\mathrm{T}} \mathbf{Q}_{\mathrm{perf}} (\mathbf{m}_t - \mathbf{p}_t)$, which maximizes the sum of predictive confidence intervals along the trajectory $m_1, \ldots, m_H$, while penalizing deviation from the safety trajectory. We choose $r = 1$ in the problem (5.12), i.e. the first action of the safety trajectory and performance trajectory are the same. We start at $x_0 \in \mathcal{X}_{\mathrm{safe}}$, and apply Algorithm 3 over 200 iterations, where the Gaussian process model is updated after every iteration.

We evaluate both settings for varying $T \in \{1, \ldots, 5\}$ and fixed $T = 5$ in terms of their mutual information in Figure 5.6. We observe a similar behavior as in the static exploration experiments and get the best exploration performance for $T = 4$, with a slight degradation of performance for $T = 5$ after 200 iterations. By comparing the exploration performance between iteration 50 and 200, we can see that influence of longer return trajectories on the exploration performance only comes into play after a certain number of iterations. This can be seen by comparing the similar performance of $T = 3$ and $T = 4$ after 50 iterations with the significantly improved performance for $T = 4$ after 200 iterations. The setting $T = 3$ during the same period only sees modest performance improvements. We can see that, except for $T = 1$, the performance trajectory decomposition setting consistently outperforms the standard setting. Planning a performance trajectory (orange) provides the algorithm with an additional degree of freedom, which leads to drastically improved

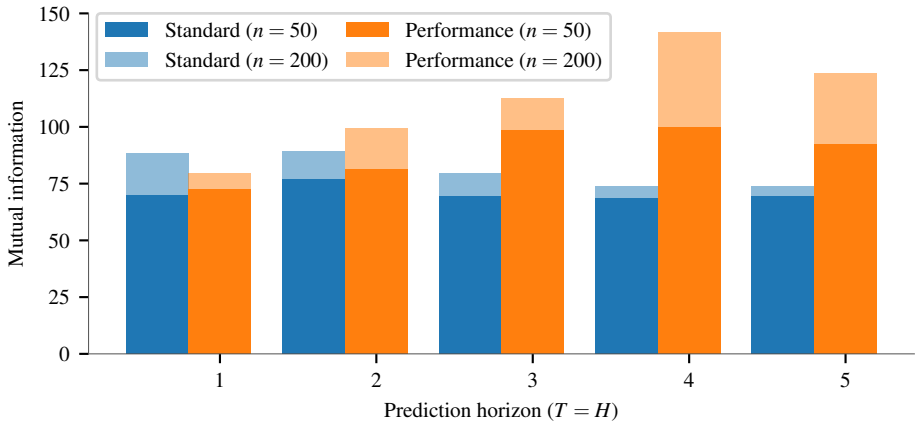


Figure 5.6: Comparison of the information gathered from the system after 50 (dark colors) and 200 (light colors) iterations for the standard setting (blue) and the setting where we plan an additional performance trajectory (orange).

exploration performance.

5.3 Task-driven Exploration

So far, in Section 5.1 we showed that safe system identification can effectively reduce the uncertainty about the transition dynamics and to improve the policy, while in Section 5.2 we introduced an algorithm that can be used to gather data safely. While a pure exploration objective naturally allows us to solve a given control task by learning the dynamics everywhere, it is data-inefficient. Instead, we want to safely learn about the system dynamics only when it is required for the control task.

In this section, we discuss how to effectively and safely explore in order to solve a task. We start by introducing two toy examples that illustrate the challenges faced by safe exploration schemes that do not rely on system identification. Next, as a first step, we introduce an optimistic exploration scheme that achieves efficient exploration without safety constraints. Lastly, we discuss how this exploration scheme can be used to effectively *and safely* explore.

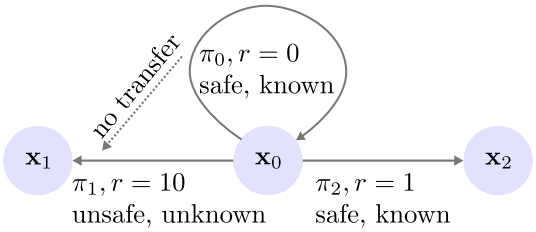


Figure 5.7: Challenge for safe exploration with a discrete set of policies $\Pi = \{\pi_1, \pi_2, \pi_3\}$. Consider a safe exploration method that aims to learn about the optimal policy. While the policy π_1 is actually unsafe, this is unknown and an optimistic exploration scheme would want to learn about it. In this case, a safe exploration scheme would instead select the 'similar' safe policy π_0 in order to learn about the safety of π_1. However, if we can never gain sufficient information to determine that π_1 is safe from data generated by π_0 or π_2, then this exploration process never terminates.

5.3.1 Challenges for Safe Exploration

In Section 5.1, we showed that learning about the system model throughout the parameter space can yield exploration guarantees. In this section, we introduce two canonical examples that illustrate the challenges in safe exploration when we do not identify the system everywhere, but instead focus on learning for a specific task. For simplicity, we consider a discrete set of policies.

5.3.1.1 Unlearnable, yet Desirable Decisions

As a first scenario, we consider using a goal-directed exploration scheme that aims to maximize reward for the safe reinforcement learning problem in Figure 5.7. There, we have an episodic control problem with three possible control policies. The first one, π_1 is the optimal policy without safety constraints and yields the highest return. While it is unsafe on the true system, this is *not* known under the current model; that is, the policy π_1 could be considered as potentially safe given the uncertainty in our model. The other two policies π_0 and π_2 are safe and this is known under the model. Since π_1 is unsafe, π_2 is the optimal *safe* policy with a reward of one.

The problem structure is such that we cannot safely learn about π_1. That is, we cannot gain sufficient information about the safety of π_1 by evaluating π_2 or π_0. Moreover, the trajectories induced by π_0 provide some information about the safety of π_1. This problem

is challenging for safe exploration, since π_1 seems desirable for any safe reinforcement learning algorithm as it achieves the highest reward and could potentially be safe. Since the goal is to solve the task, the learning algorithm cannot ignore π_1. In particular, the algorithm might be tempted to evaluate π_0 for all time steps, since we can always gain a small amount of information about the safety of π_1. Due to the aleatoric noise this information never drops to zero, but diminishes quickly.

Thus, in order to not get stuck evaluating the *suboptimal* policy π_0, any safe learning algorithm *must know when to stop attempting to learn about state-actions that are not safely learnable*. It is not easy to quantify whether it is possible to safely learn about something in continuous state-action spaces. Moreover, a policy might be safe, but just marginally above the safety threshold. To determine that it is safe, we would have to learn the dynamics up to arbitrary precision, which is generally not possible, or requires exponentially many data points, in the noisy setting.

Remark 3. The same problem is faced by algorithms that modify the actions suggested by an unsafe reinforcement learning algorithm and select the 'closest' safe action. In our example, the reinforcement learning algorithm is likely to suggest π_1 since it is high reward, while π_0 is the closest safe action. Since the reinforcement learning algorithm is not aware of the safety constraints, it keeps suggesting π_1 indefinitely.

5.3.1.2 Safe and Informative, yet Undesirable Decisions

For the second example, consider a reinforcement learning algorithm that optimizes performance subject to worst-case safety constraints. For example, one might use (5.12) with an objective that maximizes expected or optimistic performance. These kind of algorithms avoid the problem in the previous section. For example, restricted to only actions that are safe in the *worst-case*, π_2 in Figure 5.7 is the optimal action.

However, this strategy can fail to find optimal solutions due to a lack of exploration. Consider the example in Figure 5.8. As before, we have three discrete policies that correspond to trajectories. In this example, all trajectories are safe. Moreover, except for the safe, optimal policy π_2, all policies are known to be safe given the uncertainty about the transitions probability. In contrast to the previous example, we can learn about the safety of π_2 by evaluating π_0 once. The optimal safe exploration strategy would be to evaluate π_0 in order to learn that π_2 is safe and evaluate π_2 for all future iterations.

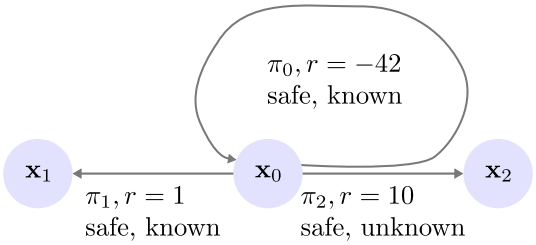


Figure 5.8: Example for safe exploration with a discrete set of policies $\Pi = \{\pi_1, \pi_2, \pi_3\}$. Consider a method that optimizes performance subject to safety constraints as in Section 5.2.1. In this example, we can learn about the safety of the optimal, safe action π_2 by evaluating π_0. Since π_2 is not known to be safe, the joint optimization criterion would select the highest-reward policy that fulfills the safety constraints, π_1. However, since we cannot learn about the safety of π_2 by evaluating π_1, this repeats infinitely and we never converge to the optimal safe parameters. The optimal strategy would be to evaluate π_0 once in order to learn about the safety of π_2, then evaluate π_2 for all other iterations.

The challenge in this problem is that we have to evaluate π_0 to learn about the safety of π_2, but that this comes at the cost of a large negative reward. If the safe exploration scheme is restricted to optimizing performance subject to the worst-case safety constraint, it cannot select π_2. Moreover, it is clear that π_1 performs significantly better than π_0. Thus, the optimal policy subject to the worst-case safety constraint is to evaluate π_1. Since this does not provide information about the safety of π_2, this repeats indefinitely.

This highlights the increased difficulty of safe exploration relative to standard exploration. We cannot focus only on learning about the trajectories with high reward that are known to be safe, but sometimes have to evaluate trajectories with high high-regret (low reward), but which allow us to learn about the safety of trajectories with high reward.

As a consequence of this example, it is clear that naively extending *unsafe* exploration schemes with worst-case safety constraints does not retain the exploration guarantees.

5.3.2 Exploration Without Safety Constraints

Given the challenges of safe exploration in the previous section, we first discuss how to efficiently explore without safety constraints. We discuss how to extend this to safe exploration in Section 5.3.3.

5.3.2.1 Problem Definition

We start by defining the *unsafe* control objective. The true, stochastic system dynamics are given by (4.1). From Assumption 2, we know that the distribution of the noise $\boldsymbol{\omega}_t$ is σ-sub-Gaussian. We consider an *episodic* setting over a finite time horizon T, so that the system is reset to a *known* state $\mathbf{x}_0$ every T time steps. Formally, the performance of specific parameters $\boldsymbol{\theta} \in \mathcal{D}$ is specified through the expected reward collected under the corresponding Lipschitz-continuous policy $\pi_{\boldsymbol{\theta}} \in \Pi$ as in Section 2.3,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\omega}_{0:T-1}}\left[\sum_{t=0}^{T} r(\mathbf{x}_t, \mathbf{u}_t) \,\middle|\, \mathbf{x}_0\right] \tag{5.15a}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}_t \tag{5.15b}$$

$$\mathbf{u}_t = \pi_{\boldsymbol{\theta}}(\mathbf{x}_t). \tag{5.15c}$$

Given the definition of the performance of parameter $\boldsymbol{\theta}$ according to $J(\boldsymbol{\theta})$ in (5.15), we aim to find the optimal parameters that maximize performance,

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} J(\boldsymbol{\theta}). \tag{5.16}$$

If the dynamics f are known, (5.16) is a standard stochastic optimal control problem over a finite time horizon. As discussed in Section 2.5, we do not know the dynamics. As in Chapter 4, we assume that the model error has bounded RKHS norm so that we can construct confidence intervals on the one-step prediction error according to Corollary 4. Moreover, we require the Lipschitz properties in Assumptions 3 and 4.

We consider an algorithm that iteratively selects parameters $\boldsymbol{\theta}_n$ at each iteration/episode n and evaluates the performance of the corresponding policy $\pi_{\boldsymbol{\theta}_n}$ on the real system. That is, at each iteration n, we observe one realization of a trajectory $\tau_n = \{\mathbf{x}_{t,n}\}_{t=0}^{T}$ that starts at $\mathbf{x}_0$ and evolves over time according to a realization of $\boldsymbol{\omega}_{t,n}$ and the selected policy. We use the resulting transition data in order to improve our model and select new parameters $\boldsymbol{\theta}_{n+1}$ for the next iteration.

To analyze the performance of our exploration strategy (5.22), we must define a quality criterion. A natural notion of regret in this setting, which is also used by Chowdhury and A. Gopalan (2019), is the difference in performance on the true system between $\boldsymbol{\theta}^*$ in (5.16) and $\boldsymbol{\theta}_n$ in expectation over the aleatoric uncertainty caused by the transition noise $\boldsymbol{\omega}$,

$$R_N = \sum_{n=0}^{N} |J(\boldsymbol{\theta}^*) - J(\boldsymbol{\theta}_n)| \tag{5.17}$$

We want to select a sequence of parameters $\boldsymbol{\theta}_n$ that ensure that the cumulative regret is sublinear, which implies convergence to close-to-optimal parameters.

5.3.2.2 Expected Performance

In practice, one of the most commonly used exploration strategies is to select $\boldsymbol{\theta}_n$ in order to maximize the expected performance over the aleatoric uncertainty *and epistemic uncertainty* induced by the Gaussian process model.

$$\boldsymbol{\theta}_n^{\text{exp}} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \mathbb{E}_{\boldsymbol{\eta}_{0:T-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \boldsymbol{\omega}_{0:T-1}} \left[\sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \,\middle|\, \mathbf{x}_0 \right] \tag{5.18a}$$

$$\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} = h(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \boldsymbol{\eta}_t + \boldsymbol{\omega}_t, \tag{5.18b}$$

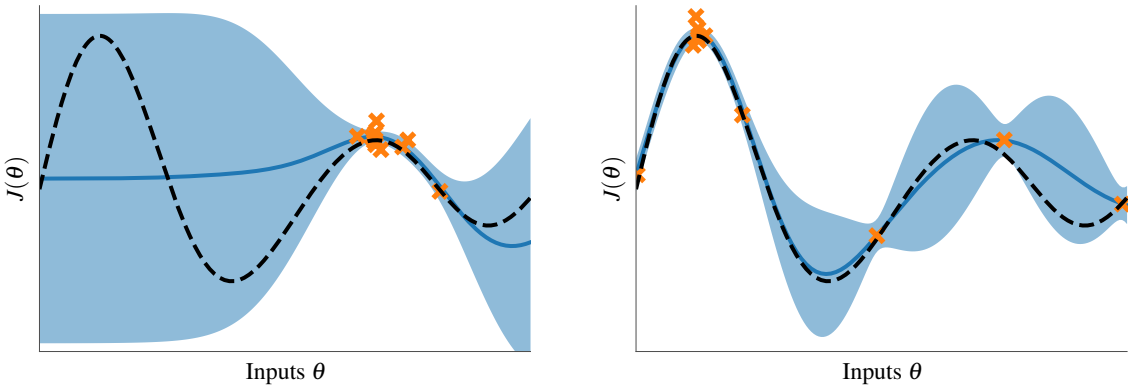
$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{5.18c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}). \tag{5.18d}$$

That is, conditioned on the previous state $\tilde{\mathbf{x}}_t$, the next state $\tilde{\mathbf{x}}_t$ is distributed according to the Normal distribution $\mathcal{N}(\mu_{n-1}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t), \boldsymbol{\Sigma}_{n-1}(\mathbf{x}_t, \tilde{\mathbf{u}}_t))$ together with additive noise $\boldsymbol{\omega}_t$. This methodology lies at the core of commonly used probabilistic model-based exploration algorithms, which employ different approximations for the intractable expectation in (5.18) for $T > 1$ and different optimization methods to solve the resulting optimal control problem. For example, Deisenroth and Carl E. Rasmussen (2011) and Kamthe and Deisenroth (2018) use moment matching to estimate a distribution over trajectories and solve the control problem with gradient descent on the parameters of a parametric policy and model predictive control, respectively. Ostafew et al. (2016) use an unscented Kalman filter and Chua et al. (2018) use sampling with the re-parametrization trick to propagate uncertainty. While these methods have different empirical performance due to the different assumptions employed in order to solve (5.18), they use the same exploration strategy in order to acquire data. In the following, we provide a simple example that illustrates that the expected performance is *not* a general exploration scheme for reinforcement learning.

Consider a system without aleatoric uncertainty, i.e., the deterministic system in (2.2). In this case, the only uncertainty originates from the transition model. For simplicity, we consider a one-dimensional system, $p = 1$, with a linear (concave) reward function $r(\mathbf{x}, \mathbf{u}) = \mathbf{x}$, a constant feedback policy $\pi_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}$, and a time horizon of one step, $T = 1$. This is the simplest possible scenario and reduces the optimal control problem in (5.18) to the bandit problem,

$$\max_{\boldsymbol{\theta} \in \mathcal{D}} f(\mathbf{x}_0, \boldsymbol{\theta}). \tag{5.19}$$



(a) $\beta_n = 0$. (b) $\beta_n = 2$.

Figure 5.9: Comparison of the GP-UCB algorithm in (2.34) with two different constants for β_n. The expected performance objective in (5.20) is equivalent setting to $\beta = 0$ in Figure 5.9a. The algorithm gets stuck and repeatedly evaluates inputs (orange crosses) at a local optimum of the true objective function (black dashed). This is due to the mean function (blue line) achieving higher values than the prior expected performance of zero. In contrast, an optimistic algorithm with $\beta = 2$ in Figure 5.9b determines close-to-optimal parameters after few evaluations.

The corresponding exploration strategy in terms of expected performance in (5.18) can be written as

$$\operatorname*{argmax}_{\boldsymbol{\theta}\in\mathcal{D}} \mathbb{E}_{\mathbf{x}_1\sim\mathcal{N}(\mu_n(\boldsymbol{\theta}),\sigma_n^2(\boldsymbol{\theta}))}[\mathbf{x}_1] = \operatorname*{argmax}_{\boldsymbol{\theta}\in\mathcal{D}} \mu_n(\boldsymbol{\theta}), \tag{5.20}$$

where we have omitted the dependence on $\mathbf{x}_0$ from the Gaussian process transition model since it is fixed for all iterations. The second term in (5.20) shows that the expected performance objective optimizes the mean of the Gaussian process transition model. This may seem natural, since the linear reward function encourages states that are as large as possible. However, when we view (5.19) as a Bayesian optimization problem, it becomes clear that (5.20) is equivalent to the GP-UCB strategy in (2.34) with $\beta_n = 0$. It is well-known in the literature that this kind of greedy exploration gets stuck in local optima (Srinivas et al., 2012).

This is illustrated in Figure 5.9. If we use (5.20) and set $\beta = 0$ in the GP-UCB algorithm, we obtain optimization behaviors as in Figure 5.9a. The first evaluation that achieves performance higher than the expected prior performance (in our case, zero), is evaluated repeatedly (orange crosses). However, this can correspond to a local optimum of the true, unknown objective function (black dashed). In contrast, if we use an optimistic algorithm

and set $\beta = 2$, GP-UCB evaluates parameters with close-to-optimal performance.

As a consequence of this example, it is clear that we cannot expect the expected performance to yield regret guarantees for exploration. Note that, since $T = 1$, this result is independent of the uncertainty propagation scheme used. This means that expected performance cannot be used as a general exploration scheme without additional assumptions. For example, (Deisenroth, Fox, et al., 2014) discuss in Section 6.1 how to choose specific reward functions that tend to encourage high-variance transitions and thus exploration. However, it is unclear how such an approach can be analyzed theoretically and we would prefer to be able to select general cost functions.

In the following, we propose an exploration scheme that does not require specific reward functions in order to encourage exploration.

5.3.2.3 Optimistic Performance

Rather than optimizing for the 'greedy' expected performance as in the previous section, we instead consider an optimistic strategy similar to GP-UCB in the Bandit setting. This class of algorithms constructs an optimistic estimate of the performance $J(\theta)$ and use it to select optimistic policy parameters $\boldsymbol{\theta}_n$.

Optimistic policy exploration algorithms in our setting were recently analyzed theoretically by Chowdhury and A. Gopalan (2019). However, they analyze an algorithm that is not implementable. In particular, their algorithm optimizes performance with respect ao all transition models that are compatible with the confidence intervals in Corollary 4 *and* encode Lipschitz-continuous value functions. This is intractable for the general system (4.1) and they do not propose a specific algorithm to do so. Moreover, their analysis requires that the state space is a compact subset of $\mathbb{R}^p$, which is not the case under the sub-Gaussian additive noise in (4.1).

Instead, we extend a practical strategy that was heuristically suggested by Moldovan, Levine, et al. (2015) for deterministic systems. They propose to construct an optimistic performance estimate by selecting optimistic dynamics within the one-step confidence intervals at every time step. We extend this idea to the stochastic system in (4.1) and

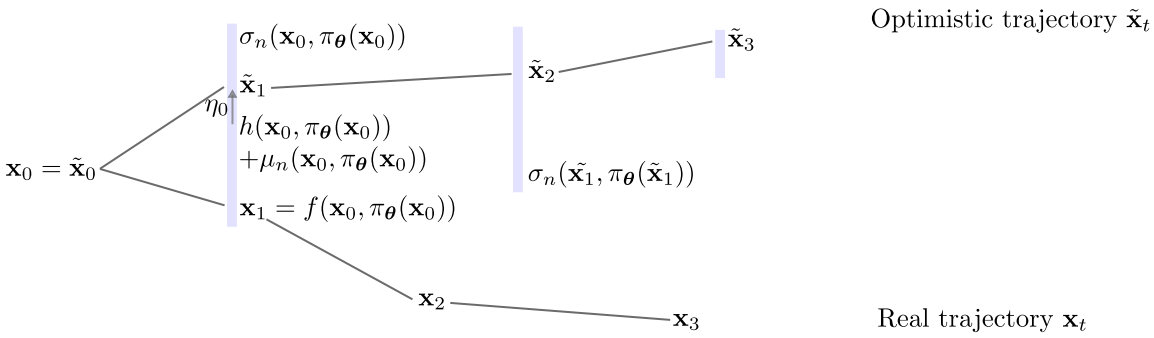


Figure 5.10: Illustrative comparison of the true state trajectory $\mathbf{x}_t$ under the policy $\pi_{\boldsymbol{\theta}}$ and the optimistic trajectory $\tilde{\mathbf{x}}_t$ from (5.22). After one step, $\mathbf{x}_1$ is contained within the confidence intervals (grey bars). The optimistic dynamics are chosen within this confidence interval to maximize performance. Since the optimistic dynamics are constructed iteratively based on the previous state $\tilde{\mathbf{x}}_t$, beyond one step the true dynamics are not contained in the confidence intervals.

construct the optimistic performance estimate

$$\tilde{J}_n(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\omega}_{0:T-1,n}}\left[\max_{\boldsymbol{\eta}_{0:T-1},\, \boldsymbol{\eta}_t \in [-1,1]^p} \sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \,\middle|\, \mathbf{x}_0\right] \tag{5.21a}$$

$$\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} = h(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \beta_{n-1}\boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n})\boldsymbol{\eta}_t + \boldsymbol{\omega}_{t,n}, \tag{5.21b}$$

$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{5.21c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}), \tag{5.21d}$$

where $\boldsymbol{\Sigma}_n^{1/2}(\mathbf{x}, \mathbf{u}) = \operatorname{diag}(\boldsymbol{\sigma}_n(\mathbf{x}, \mathbf{u}))$ is the diagonal covariance matrix that contains the bound on the model errors. Unlike the true performance $J(\boldsymbol{\theta})$ in (5.15), the optimistic performance estimate $\tilde{J}_n(\boldsymbol{\theta})$ in (5.21) is constructed using the Gaussian process model based on all data available up to iteration n. The additional optimization variables $\boldsymbol{\eta}_t \in [-1, 1]^p$ allow the optimization problem to pick optimistic dynamics within the confidence intervals. In particular, it follows directly from Corollary 4 that, with probability at least $(1 - \delta)$, for each sequence of $\boldsymbol{\omega}_{0:T-1}$, there exists a sequence of $\boldsymbol{\eta}_{0:T-1}$ with $\boldsymbol{\eta}_t \in [-1, 1]^p$ such that $g(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) = \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) + \beta_{n-1}\boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)\boldsymbol{\eta}_t$. As a result, we have $\tilde{J}_n(\boldsymbol{\theta}) \geq J(\boldsymbol{\theta}^*)$ for all $\boldsymbol{\theta} \in \mathcal{D}$; that is, $\tilde{J}_n$ is an *optimistic* estimate of the performance.

Unlike the uncertainty propagation scheme in Section 4.3 and methods like PILCO by Deisenroth and Carl E. Rasmussen (2011), the optimistic performance estimate $\tilde{J}$ in

(5.22) does not explicitly propagate uncertainty over the T time steps. Instead, it uses the pointwise uncertainty estimates from the Gaussian process model to recursively plan an optimistic trajectory. This can be thought of as an implicit uncertainty propagation scheme and is illustrated in Figure 5.10. The two trajectories start at the same state $\mathbf{x}_0$. As expected from Corollary 4, after one step the true state $\mathbf{x}_1$ is contained within the confidence intervals (blue shaded), while $\tilde{\mathbf{x}}_1$ is optimistically chosen within the confidence intervals to maximize performance. However, the prediction for $\tilde{\mathbf{x}}_2$ uses the Gaussian process model evaluated at $\tilde{\mathbf{x}}_1$, so that the true state $\mathbf{x}_2$ is not contained in the confidence intervals. While not having to construct explicit confidence intervals is a significant advantage over existing methods, this implicit construction renders the analysis significantly more challenging.

Lastly, the maximization over $\boldsymbol{\eta}_t$ inside the expectation in (5.21) might seem daunting. However, we show in Section 5.3.2.4 that this maximization can be replaced by an outer maximization over Lipschitz-continuous functions, which leads to practical algorithms. In the noise free case, (5.21) is a standard deterministic optimal control problem. For now, we assume that solving (5.21) is tractable and analyze an optimistic exploration strategy.

Exploration strategy Given the optimistic estimate of the performance in (5.21), we consider an exploration strategy that, at each iteration n, selects the parameters

$$\boldsymbol{\theta}_n = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \tilde{J}_n(\boldsymbol{\theta}) \tag{5.22}$$

and evaluates them in experiments. That is, at each iteration n we select parameters $\boldsymbol{\theta}_n$ that maximize the optimistic performance estimate $\tilde{J}_n(\boldsymbol{\theta})$. This performance estimate plans an optimistic trajectory $\tilde{\mathbf{x}}_{t,n}$ based on the Gaussian process model with data from the previous $n-1$ iterations. We then evaluate the performance of the policy $\pi_{\boldsymbol{\theta}_n}$ on the real system and observe a trajectory $\mathbf{x}_{t,n}$ of the real system over T time steps. At the end of each episode, we update the Gaussian process model with the observed data. This procedure is summarized in Algorithm 4.

To bound the cumulative regret, we bound the deviation of the optimistic trajectory $\tilde{\mathbf{x}}_t$ from the true trajectory $\mathbf{x}_t$. The posterior standard deviation $\|\boldsymbol{\sigma}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)\|$ at states of the optimistic trajectory can be significantly larger than that at the true states, $\mathbf{x}_t$. This is especially true since we may never observe state transitions from the planned optimistic $\tilde{\mathbf{x}}$, but only from the true states $\mathbf{x}$. Thus, the generalization properties of our statistical model determine how far the two state distributions diverge. If the posterior variance does not increase too quickly as we deviate from $\mathbf{x}_t$, then we can still bound the deviations

Algorithm 4 Optimistic Model-based Policy Search

Inputs: Gaussian process transition model,
reward function $r(\cdot, \cdot)$,
horizon T,
Initial state $\mathbf{x}_0$

1: **for** $n = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{\theta}_n \leftarrow \operatorname{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \tilde{J}_n(\boldsymbol{\theta})$
3: $\quad$ Reset the system to $\mathbf{x}_{0,n} = \mathbf{x}_0$
4: $\quad$ **for** $t = 1, \ldots, T$ **do**
5: $\quad\quad \mathbf{x}_{t+1,n} = f(\mathbf{x}_{t-1,n}, \pi_{\boldsymbol{\theta}_n}(\mathbf{x}_{t-1,n})) + \boldsymbol{\omega}_{t-1,n}$
6: $\quad$ Update Gaussian process model with T observed state transitions at iteration n.

between $\tilde{\mathbf{x}}$ and $\mathbf{x}$. To this end, we require that the posterior standard deviation is Lipschitz continuous. We make the following assumption.

Assumption 9. The kernel metric (2.33) is Lipschitz continuous.

This assumption is not trivially fulfilled by Assumption 4, since the square root function has unbounded derivatives at zero. However, for many commonly used kernels, e.g., the linear and squared exponential kernels, the kernel metric is Lipschitz continuous. This property immediately implies Lipschitz continuity of the Gaussian process posterior standard deviation.

Corollary 7. *Under Assumption 9, the Gaussian process posterior standard deviation* $\boldsymbol{\sigma}_n(\cdot)$ *is* L_σ*-Lipschitz continuous with respect to the 2-norm for all* $n \geq 0$.

Proof. It is trivial to show that the posterior Gaussian process variance is Lipschitz continuous as a function of the number of data points under Assumption 4, see, for example, (Lederer et al., 2019). We show in Lemma 50 that the posterior variance is in fact Lipschitz continuous with constant one with respect to the kernel metric in (2.33). The result then follows from Assumption 9. □

We use this additional assumption to bound the regret of (5.22) in the following theorem.

Theorem 8. *Under the assumption of Corollaries 4 and 7 and Assumptions 4, 5 and 7, let* $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g) + 1 + \ln(2/\delta)}$ *and* $b_n = L_f^{T-1} T \left(B_g + B_h + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T\pi^2 n^2}{3\delta}} \right)$.

At each iteration, select parameters according to (5.22), then the following holds with probability at least $(1-\delta)$ *for all* $n \geq 1$

$$R_n \leq \tilde{\mathcal{O}}\Big(\sqrt{nT^5 p L_f^T L_\sigma^T \beta_n^T \gamma_{pnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})}\Big), \tag{5.23}$$

where $\mathbb{B}(\mathbf{x}_0, b_n) = \{\mathbf{x} \in \mathbb{R}^p \mid \|\mathbf{x} - \mathbf{x}_0\|_2 \leq b_n\}$ *and* $\gamma_{pnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})$ *is the information capacity after* (pnT) *observations within the domain* $\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}$.

We provide a proof of Theorem 8 in Appendix D. The theorem ensures that, if we evaluate optimistic parameters according to (5.22), we eventually achieve performance $J(\boldsymbol{\theta}_n)$ arbitrarily close to the optimal performance of $J(\boldsymbol{\theta}^*)$ if $\beta_n^T \gamma$ grows at a rate smaller than $\sqrt{n}$. As one would expect, the regret bound in (D.91) depends on constant factors like the prediction horizon T, the Lipschitz constant of the closed-loop dynamics L_f, and the dimensionality of the state space p.

In contrast to the bound by Chowdhury and A. Gopalan (2019) for an algorithm that is not implementable, the regret bound in Theorem 8 additionally depends on the Lipschitz constant L_σ and an additional factor of β_n^{T-1}. This is the cost of potentially allowing 'discontinuous' optimistic trajectories by not explicitly enforcing a continuity constraint. However, the bound remains sublinear for commonly used kernels. Moreover, we show in Appendix D.4 that we obtain the same bound as Chowdhury and A. Gopalan (2019) if we assume that we can optimize over all dynamics functions that are compatible with Corollary 4 and L_f-Lipschitz continuous. Empirically, we often assume calibrated confidence intervals for a constant β_n, in which case we match their bound up to constants.

In contrast to Chowdhury and A. Gopalan (2019), we do not assume that the state space is compact. Instead, we use the model properties together with a high-probability bound on the noise norm to bound the domain. As a consequence, the bound on the mutual information, the information capacity $\gamma(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})$, is computed over a norm-ball with an increasing radius b_n. We show in the following that this is still a sublinear function for commonly used kernels and compact sets $\mathcal{U}$.

Information capacity For the bound in Theorem 8 to be meaningful, we must show that the information capacity (largest mutual information) over the domain $\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}$ is sublinear. For constant b_n, Srinivas et al. (2012) bound the information capacity by bounding the tail of the eigenspectrum of the empirical kernel matrix over a tight discretization. By increasing the domain, we increase both the size of the discretization

and, potentially, the bound on the eigenspectrum. In Appendix D.3, we analyze this dependence and obtain the following results.

Lemma 7. *For the squared exponential kernel we have*

$$\gamma_n(\mathbb{B}(\mathbf{x}_0, b_n)) = \mathcal{O}\Big(b_n^d (\log(n))^{d+1}\Big) = \tilde{\mathcal{O}}\Big(T \log(n^2) \log(n)^{d+1}\Big) \tag{5.24}$$

That is, as we increase the radius of the norm ball over iterations, the mutual information only grows at a rate logarithmic in n. For parametric kernels, the bound is even independent of the size of the domain.

Lemma 8 (Srinivas et al. (2012))**.** *For the linear kernel $k(\mathbf{a}, \mathbf{a}') = \mathbf{a}^{\mathrm{T}}\mathbf{a}'$ with $\mathbf{a} \in \mathbb{R}^d$ we have*

$$\gamma_n(\mathbb{B}(\mathbf{x}_0, b_n)) = \mathcal{O}(d \log(n)) \tag{5.25}$$

As a consequence, the regret bound is in fact sublinear for these kernels. Lastly, we note that we are using a composite kernel based on the kernel matrix in Section 3.2. The information capacity of these composite kernels can be bounded using the methodology by Krause and Ong (2011).

5.3.2.4 Practical Implementation

In practice, (5.21) is difficult to solve due to the expectation over an optimization problem. However, we show in Appendix D.6 that we can exploit the boundedness of $\boldsymbol{\sigma}$ over any compact domain and the Lipschitz assumptions to obtain a practical algorithm. In particular, let Ξ be the class of functions that are bounded in $[-1, 1]^p$ with finite Lipschitz constant. Then (5.21) attains the same regret bound as

$$\tilde{J}_n(\boldsymbol{\theta}) = \max_{\eta(\cdot) \in \Xi} \mathbb{E}_{\boldsymbol{\omega}_{0:T-1,n}} \left[\sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \,\middle|\, \mathbf{x}_0 \right] \tag{5.26a}$$

$$\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} = h(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \beta_{n-1} \boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \boldsymbol{\eta}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) + \boldsymbol{\omega}_{t,n} \tag{5.26b}$$

$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{5.26c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}), \tag{5.26d}$$

That is, we can optimize jointly over the expected performance and a Lipschitz continuous, bounded *function* $\eta(\cdot)$. Effectively, $\eta(\cdot)$ parametrizes the optimistic transition functions that we optimize. In practice, this can be implemented by, for example, specifying the class of functions Ξ through a neural network with a tanh nonlinearity on the last layer.

5.3.3 Safe Exploration

So far we have only considered input constraints, $\mathcal{U}$, that were enforced through the definition of the policy class in Assumption 6. In this section, we discuss how these results can be used for safe exploration with state constraints.

We can extend both the control problem (5.15) and the optimistic performance estimate in (5.21) with additional state (chance-)constraints. However, the regret bounds only imply that the *performance* converges. This does not imply safety, since constraint satisfaction of the optimistic trajectory planned by (5.22) does not imply constraint satisfaction of the true system (4.1). Thus, while eventually $\tilde{\mathbf{x}}_{t,n}$ converges arbitrarily close to $\mathbf{x}_{t,n}$, the true trajectory induced by the parameters $\boldsymbol{\theta}_n$ may violate the safety constraints.

Instead of optimistic constraints, one could instead use (5.12) with worst-case safety constraints over the epistemic uncertainty together with the optimistic performance objective. However, by enforcing worst-case constraints we can no longer guarantee that $\tilde{J}_n(\boldsymbol{\theta}_n) > J(\boldsymbol{\theta}^*)$, which is required for our safety guarantees. As a consequence, our exploration analysis, and Theorem 9 in particular, *does not hold* in the safety-constrained setting. Moreover, the examples in Section 5.3.1 illustrate that such an algorithm can get stuck in local optima.

Thus enforcing 'optimistic' safety constraints on $\tilde{\mathbf{x}}_t$ yields the desired convergence guarantees but no safety guarantees, while worst-case safety constraints that consider the epistemic uncertainty generally do not converge to the global optimum. As a middle-ground, we now consider an algorithm that combines both approaches. In particular, rather than extending the optimal control problem in Section 2.3 with state constraints $\mathcal{X}$, we instead compare against a slightly more conservative baseline with tightened constraints

$$\mathcal{X}_\epsilon = \{\mathbf{x} \in \mathcal{X} \mid \min_{\mathbf{x}' \in \mathbb{R}^p \setminus \mathcal{X}} \|\mathbf{x} - \mathbf{x}'\|_2 \geq \epsilon_x\}, \tag{5.27}$$

so that the optimal trajectories cannot lie on the boundary of the set $\mathcal{X}$, but can at most be ϵ_x-close. We consider a problem with state chance-constraints that must hold with

probability at least $(1-\delta_x)$ over the aleatoric uncertainty induced by the transition noise,

$$\boldsymbol{\theta}^{s,*} = \underset{\boldsymbol{\theta}\in\mathcal{D}}{\operatorname{argmax}}\ \mathbb{E}_{\boldsymbol{\omega}_{0:T-1}}\left[\sum_{t=0}^{T} r(\mathbf{x}_t, \mathbf{u}_t)\,\middle|\,\mathbf{x}_0\right] \tag{5.28a}$$

$$\text{s.t. } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}_t, \tag{5.28b}$$

$$\mathbf{u}_t = \pi_{\boldsymbol{\theta}}(\mathbf{x}_t), \tag{5.28c}$$

$$\mathbb{P}(\mathbf{x}_t \in \mathcal{X}_\epsilon\, \forall t \in 1,\dots,T) \geq 1-\delta_x. \tag{5.28d}$$

Note that the $(1-\delta_x)$ probability is only over the aleatoric uncertainty, since (5.28) assumes a *known transition model*, so that there is no epistemic uncertainty.

We consider an exploration algorithm that is analogous to the unsafe setting, but additionally consider the chance-constraints in (5.28):

$$\boldsymbol{\theta}_n^s = \underset{\boldsymbol{\theta}\in\mathcal{D}}{\operatorname{argmax}}\ \max_{\eta(\cdot)\in\Xi}\ \mathbb{E}_{\boldsymbol{\omega}_{0:T-1,n}}\left[\sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n})\,\middle|\,\mathbf{x}_0\right] \tag{5.29a}$$

$$\begin{aligned}\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} &= h(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \\ &\quad + \beta_{n-1}\boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n})\boldsymbol{\eta}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) + \boldsymbol{\omega}_{t,n}\end{aligned} \tag{5.29b}$$

$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{5.29c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}), \tag{5.29d}$$

$$\mathbb{P}(\tilde{\mathbf{x}}_{t,n} \in \mathcal{X}_\epsilon\, \forall t \in 1,\dots,T) \geq 1-\delta_x. \tag{5.29e}$$

It is important to note that the parameters $\boldsymbol{\theta}_n^s$ do *not* impose safe trajectories $\mathbf{x}_t$ in general, since the chance constraints are only enforced over the optimistic trajectory $\tilde{\mathbf{x}}_{t,n}$. However, if we were to evaluate the parameters $\boldsymbol{\theta}_n^s$ we retain exploration guarantees relative to (5.28):

Lemma 9. *Under the assumption of Theorem 8. At each iteration, select parameters according to (5.29), then the following holds with probability at least* $(1-\delta)$ *for all* $n \geq 1$

$$\sum_{n=0}^{N} J(\boldsymbol{\theta}^{s,*}) - J(\boldsymbol{\theta}_n^s) \leq \tilde{\mathcal{O}}\Big(\sqrt{NT^5 p L_f^T L_\sigma^T \beta_n^T \gamma_{pnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})}\Big). \tag{5.30}$$

Proof. The proof follows analogous to that of Theorem 8. In particular, since both the reference (5.28) and the exploration strategy (5.29) are subject to the same constraints, we retain that (5.29) is an optimistic performance estimate. □

That is, we obtain the same regret bounds as in Theorem 8. Since we employ tightened state-constraints $\mathcal{X}_\epsilon$ to ensure safety, we additionally ensure that once the model uncertainty decreases sufficiently, (5.29) eventually proposes safe parameters.

Lemma 10. *Under the assumptions of Lemma 9 and with* $\bar{L}_f = 1 + L_f + 2\beta_{n-1}L_\sigma$*, if*

$$\sum_{t=0}^{T-1} \|\boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_{t,n})\|_2 \leq \frac{\epsilon_x}{2\beta_{n-1}\bar{L}_f^{T-1}} \tag{5.31}$$

holds at iteration n*, then, with probability at least* $(1-\delta)$*, we have that*

$$\mathbb{P}(\mathbf{x}_{t,n} \in \mathcal{X}\, \forall t \in 1, \ldots, T) \geq 1 - \delta_x. \tag{5.32}$$

Proof. This is a direct consequence of Lemma 35 in the appendix. □

That is, if the optimization problem (5.29) plans an optimistic trajectory through states with sufficiently small uncertainty as in (5.31), with probability at least $(1-\delta)$ over the epistemic uncertainty, the resulting trajectory satisfies the state constraints on the true system with probability at least $(1-\delta_x)$ over the aleatoric uncertainty. Note that one could employ a union bound over the two sources of uncertainty in order to obtain a single probability over both sources of uncertainty. Moreover, here we use a $(1-\delta_x)$ guarantee *per iteration*, rather than over all iterations. This is not a limitation of the proof strategy, since one could us a bound as in Lemma 4 to obtain bounds for every step. However, this would mean that exploration gets more and more conservative over the course of iterations and we would have to adapt the optimal parameters in (5.28) to consider safety over many iterations.

This means that, after learning about the system so that the uncertainty is sufficiently small, the parameters $\boldsymbol{\theta}_n^s$ are safe on the real system. Thus, as the uncertainty about the system reduces, (5.29) eventually stops proposing unsafe parameters. Moreover, according to Lemma 9 we know that if we repeatedly evaluate $\boldsymbol{\theta}_n^s$, then we eventually converge to close-to-optimal parameters. What remains is to ensure safety throughout the learning process. We propose Algorithm 5, which at each iteration n computes optimistic parameters $\boldsymbol{\theta}_n^s$ according to (5.29). However, we only evaluate them if they satisfy the safety constraints on the real system. To verify this, we can use confidence intervals on the trajectory as in Section 4.3. If the parameters are unsafe, we overrule the decision on $\boldsymbol{\theta}_n$ and instead aim to safely learn about the safety of the optimistic trajectory $\tilde{\mathbf{x}}_t$. We then evaluate the parameters $\boldsymbol{\theta}_n$ on the real system and update the GP model with the corresponding observations.

This algorithm clearly satisfies the safety constraints, since only safe parameters are evaluated. However, for the regret bounds to be meaningful in this setting, we must bound

Algorithm 5 Safe Optimistic Model-based Policy Search

Inputs: Gaussian process transition model,
reward function $r(\cdot, \cdot)$,
horizon T,
Initial state $\mathbf{x}_0$

1: **for** $n = 1, 2, \ldots$ **do**
2: $\quad \boldsymbol{\theta}_n^s \leftarrow \operatorname{argmax}_{\boldsymbol{\theta} \in \mathcal{D}} \tilde{J}_n(\boldsymbol{\theta})$ s.t. $\mathbb{P}(\tilde{\mathbf{x}}_{t,n} \in \mathcal{X}_\epsilon \, \forall t \in 1, \ldots, T) \geq 1 - \delta_x$
3: $\quad$ Evaluate aleatoric and epistemic safety of $\boldsymbol{\theta}_n^s$ as in Section 4.3
4: $\quad$ **if** unsafe **then**
5: $\quad\quad \boldsymbol{\theta}_n^s \leftarrow$ compute safe trajectory to learn about safety of $\tilde{\mathbf{x}}$ from $\boldsymbol{\theta}_n^s$.
6: $\quad$ Reset the system to $\mathbf{x}_{0,n} = \mathbf{x}_0$
7: $\quad$ **for** $t = 1, \ldots, T$ **do**
8: $\quad\quad \mathbf{x}_{t+1,n} = f(\mathbf{x}_{t-1,n}, \pi_{\boldsymbol{\theta}_n^s}(\mathbf{x}_{t-1,n})) + \boldsymbol{\omega}_{t-1,n}$
9: $\quad$ Update Gaussian process model with the T observed state transitions.

the number of iterations where the safety filter kicks in. Unfortunately, this is generally only possible for a scheme that eventually reduces uncertainty about the state space as in Section 5.1. Such a scheme was proposed heuristically by Lew et al. (2019). A more practical exploration scheme that does not require full exploration is to actively learn about $\tilde{\mathbf{x}}$, e.g., by exploring a trajectory that maximizes the mutual information gained about this trajectory. Once the uncertainty is reduced sufficiently, Lemma 10 ensures that the suggested parameters are safe to evaluate. Alas, it is difficult to analyze such an algorithm in continuous spaces.

In related work by Turchetta et al. (2019) that is not part of this dissertation (see Section 5.4), we propose a similar safe exploration scheme for discrete spaces, where it is possible to quantify exploration *and* provide exploration guarantees. However, at the heart of the analysis we exploit the fact that, in the worst case, the proposed algorithm explores the entire space. In particular, in discrete spaces we can identify and avoid situations like the one in Section 5.3.1.1.

While it is not clear how to transfer this analysis to continuous spaces, we would like to point out that we expect Algorithm 5 to work well empirically; that it provides rigorous safety guarantees; and that its exploration scheme is *based on* a theoretically rigorous approach.

5.4 Conclusion

In this chapter, we introduced several safe exploration schemes. We first showed how to compute safe parametric policies and showed that an idealized exploration scheme that can sample within the current region of attraction can lead to effective exploration. Next, we constructed a model predictive control scheme that can be used to safely collect data. Lastly, we proposed and analyzed an unsafe exploration scheme and provided regret bounds. This scheme can be combined heuristically with the safe model predictive control scheme in order to yield an effective safe policy search algorithm.

Other Related Work There are several publications which are not part of this dissertation, but that were written during the course of the PhD and are relevant to this chapter.

1. In (Turchetta et al., 2016), we provide safe exploration guarantees in discrete state spaces.
2. In (S. M. Richards et al., 2018) we investigate how to use neural networks in order to learn Lyapunov functions from data.
3. In (Turchetta et al., 2019), we extend (Turchetta et al., 2016) with goal-directed optimistic exploration scheme.

6

Conclusion

In this dissertation, we derived several safe reinforcement learning algorithms both in the model-free and the model-based setting.

We first considered a direct policy search method in Chapter 3 based on Bayesian optimization. We extended existing work in order to obtain a practical and safe algorithm that can be used to optimize policy parameters in an episodic setting. Moreover, we investigated how contexts can be used in order to safely transfer knowledge between multiple tasks. We conducted thorough experiments on a quadrotor robot and demonstrated high-performance learning behavior.

Next, in Chapter 4, we showed how to reliably learn models of dynamical systems. Given these reliable models, we introduced a method to analyze the underlying deterministic transition function for stability based on Lyapunov theory. We extended these results to construct reliable confidence intervals over trajectories, in order to verify system properties over longer time horizons. Together, these methods provide the basic tools to analyze uncertain systems for safety.

Lastly, we considered safe model-based reinforcement learning based on the safety analysis tools introduced before in Chapter 5. We showed how one can learn policies that are provably safe and introduced an idealized exploration scheme that used the analysis tools derived perviously in order to safely collect data. Next, we constructed a model predictive control algorithm that can be used to collect data in practice and enjoys safety guarantees

throughout the learning process. In order to collect data that is relevant for the reinforcement learning task, we analyzed an optimistic exploration scheme and showed that it is provably no-regret. Lastly, we combined this optimistic objective with the worst-case safety guarantees of model predictive control in order to obtain a practical safe reinforcement learning algorithms.

6.1 Future Work

There are several promising directions for future work.

The safe model-free Bayesian optimization algorithm in Chapter 3 is mostly restricted to low-dimensional problems due to the computational burden of optimizing (3.14) and the statistical problem of defining suitable Gaussian process priors in high-dimensions. In unsafe Bayesian optimization, there has be interesting progress on both learning better statistical models from data and scaling up these methods to high-dimensional problems. Investigating both these directions in the safety-constrained setting could lead to more practical algorithms. Moreover, it would be interesting to explore other settings, for example, multi-fidelity Bayesian optimization, in the safety constrained setting.

The safety analysis tools derived in Chapter 4 are provably reliable, but can be conservative. Learning better reliable, statistical models and providing more fine-grained and computationally less expensive analysis tools are interesting directions for future work.

In the space of safe model-based reinforcement learning large amounts of work remain to be done. In fact, safe exploration in continuous state-action spaces is not yet fully understood. For example, formally analyzing an algorithm similar to Algorithm 5 that does not rely on full system identification is a significant challenge that is left for future work. Moreover, many of the algorithms rely on solving constrained policy optimization problems that might be difficult to solve in practice, e.g., (5.8). A better understanding how approximate dynamic programming problems can be solved under constraints would be an interesting future direction.

Lastly, while we have focused on guaranteeing safety with respect to statistical models here, safety can also be ensured by providing additional external knowledge. For example, learning in the presence of humans, who can be asked for advice infrequently, could be an interesting direction.

A

Proofs for Safe Direct Policy Optimization

In this section, we provide the proofs for Theorem 2 and Lemma 3 in Chapter 3. The results in this section were previously published in (Berkenkamp, A. P. Schoellig, and Krause, 2016) and (Berkenkamp, Krause, et al., 2016).

In our setting, we obtain $|\mathcal{I}| = m + 1$ measurements at every iteration step n, each with different, independent noise. The mutual information with regards to these multiple measurements at parameters $\mathcal{D}_n \subset \mathcal{D}$ can be bounded with

$$\begin{aligned} \mathrm{I}(\mathbf{y}_{\mathcal{D}_n \times \mathcal{I}}; f') &\leq \max_{\mathcal{A} \subset \mathcal{D}, |\mathcal{A}| \leq n} \mathrm{I}(\mathbf{y}_{\mathcal{A} \times \mathcal{I}}; f'), && \text{(A.1)} \\ &\leq \max_{\mathcal{A} \subset \mathcal{D} \times \mathcal{I}, |\mathcal{A}| \leq n|\mathcal{I}|} \mathrm{I}(\mathbf{y}_{\mathcal{A}}; f'), && \text{(A.2)} \\ &= \gamma_{|\mathcal{I}|n}, && \text{(A.3)} \end{aligned}$$

where $\mathcal{D} \times \mathcal{I}$ is the cartesian product that means we obtain one measurement for every function indexed by $i \in \mathcal{I}$ at each parameter in $\mathcal{D}$. The first inequality bounds the mutual information gained by Algorithm 1 by the worst-case mutual information, while the second inequality bounds this again by the worst-case mutual information when optimizing over the $|\mathcal{I}|$ measurements at each iteration step individually. Intuitively, obtaining multiple optimal samples does not fundamentally change the properties of the information gain, but accelerates the rate at which information can be obtained in the worst case by $|\mathcal{I}|$.

In the following, we assume that $f'(\boldsymbol{\theta}, i)$ has bounded RKHS norm. Lemma 1 provides requirements for β_n, which we use in the following to prove the results.

Lemma 2 (based on Chowdhury and A. Gopalan (2017)). *Assume that $f'(\boldsymbol{\theta}, i) = [f'(\boldsymbol{\theta})]_i$ has RKHS norm bounded by B and that measurements are corrupted by σ-sub-Gaussian noise. Let $\mathcal{A}_n = \mathcal{D}_n \times \mathcal{I}$ denote the measurements obtained up to iteration n. If $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f') + 1 + \ln(1/\delta)}$, then the following holds for all parameters $\boldsymbol{\theta} \in \mathcal{D}$, function indices $i \in \mathcal{I}$, and iterations $n \geq 0$ jointly with probability at least $1 - \delta$:*

$$\left| f'(\boldsymbol{\theta}, i) - \mu_n(\boldsymbol{\theta}, i) \right| \leq \beta_n \sigma_n(\boldsymbol{\theta}, i) \tag{3.5}$$

Proof. Directly follows from Chowdhury and A. Gopalan (2017). The only difference is that we obtain $|\mathcal{I}|$ measurements at every iteration, which causes the information capacity γ to grow at a faster rate. □

Note Where needed in the following lemmas, we implicitly assume that the assumptions of Lemma 1 hold, and that β_n is defined as above.

Corollary 8. *For β_n as above, the following holds with probability at least $1 - \delta$:*

$$\forall n \geq 1,\ \forall i \in \mathcal{I},\ \forall \boldsymbol{\theta} \in \mathcal{A},\ f'(\boldsymbol{\theta}, i) \in \mathcal{C}_n(\boldsymbol{\theta}, i).$$

Proof. From Lemma 1 we know that the true functions are contained in $Q_n(\boldsymbol{\theta}, i)$ for all iterations n with probability at least $1 - \delta$. As a consequence, the true functions will be contained in the intersection of these sets with the same probability. □

Corollary 8 gives a choice of β_n, which ensures that all the function values of f' are contained within their respective confidence intervals with high probability. In the remainder of the paper, we follow the outline of the proofs in Sui et al., 2015, but extended them to account for multiple constraints.

We start by showing the dynamics of important sets and functions. Most importantly, the upper confidence bounds are decreasing, lower confidence bounds increasing with the number of iterations, since the sets $\mathcal{C}_{n+1} \subseteq \mathcal{C}_n$ for all iterations n.

Lemma 11. *The following hold for any $n \geq 1$:*

(i) $\forall \boldsymbol{\theta} \in \mathcal{A}, \forall i \in \mathcal{I},\ u^i_{n+1}(\boldsymbol{\theta}) \leq u^i_n(\boldsymbol{\theta}),$

(ii) $\forall \boldsymbol{\theta} \in \mathcal{A}, \forall i \in \mathcal{I},\ l^i_{n+1}(\boldsymbol{\theta}) \geq l^i_n(\boldsymbol{\theta})$,

(iii) $\forall \boldsymbol{\theta} \in \mathcal{A}, \forall i \in \mathcal{I},\ w_{n+1}(\boldsymbol{\theta}, i) \leq w_n(\boldsymbol{\theta}, i)$,

(iv) $\mathcal{S}_{n+1} \supseteq \mathcal{S}_n \supseteq \mathcal{S}_0$,

(v) $\mathcal{S} \subseteq \mathcal{R} \Rightarrow R_\epsilon(\mathcal{S}) \subseteq R_\epsilon(\mathcal{R})$,

(vi) $\mathcal{S} \subseteq \mathcal{R} \Rightarrow \bar{R}_\epsilon(\mathcal{S}) \subseteq \bar{R}_\epsilon(R)$.

Proof. (i), (ii), and (iii) follow directly from their definitions and the definition of $\mathcal{C}_n(\boldsymbol{\theta})$.

(iv) Proof by induction. Consider the initial safe set, $\mathcal{S}_0$. By definition of $\mathcal{C}_0$ we have for all $\boldsymbol{\theta} \in \mathcal{S}_0$ and $i \in \mathcal{I}$ that

$$l^i_1(\boldsymbol{\theta}) - Ld(\boldsymbol{\theta}, \boldsymbol{\theta}) = l^i_1(\boldsymbol{\theta}) \geq l^i_0(\boldsymbol{\theta}) \geq 0.$$

It then follows from the definition of $\mathcal{S}_n$ that $\boldsymbol{\theta} \in \mathcal{S}_1$.

For the induction step, assume that for some $n \geq 2$, $\mathcal{S}_{n-1} \subseteq \mathcal{S}_n$ and let $\boldsymbol{\theta} \in \mathcal{S}_n$. This means that for all $i \in \mathcal{I}_c$, $\exists \mathbf{z}_i \in \mathcal{S}_{n-1}, l^i_n(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0$ by the definition of the safe set. But, since $\mathcal{S}_{n-1} \subseteq \mathcal{S}_n$, this implies that $\mathbf{z}_i \in \mathcal{S}_n, \forall i \in \mathcal{I}_c$. Furthermore, by part (ii), $l^i_{n+1}(\mathbf{z}) \geq l^i_n(\mathbf{z}_i)$. Therefore, we conclude that for all $i \in \mathcal{I}_c$, $l^i_{n+1}(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0$, which implies that $\boldsymbol{\theta} \in \mathcal{S}_{n+1}$.

(v) Let $\boldsymbol{\theta} \in R_\epsilon(\mathcal{S}S)$. Then, by definition, for all $i \in \mathcal{I}_c$, $\exists \mathbf{z}_i \in S, c_i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0$. But, since $\mathcal{S} \subseteq \mathcal{R}$, it means that $\mathbf{z}_i \in \mathcal{R}\, \forall i \in \mathcal{I}_c$, and, therefore, $c_i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0$ for all $i \in \mathcal{I}_c$ also implies that $\boldsymbol{\theta} \in R_\epsilon(\mathcal{R})$.

(vi) This follows directly by repeatedly applying the result of part (v).

□

Using the previous results, we start by showing that, after a finite number of iterations, the safe set has to expand if possible. As a first step, note that the set of expanders and maximizers are contained in each other as well if the safe set does not increase:

Lemma 12. *For any $n_1 \geq n_0 \geq 1$, if $\mathcal{S}_{n_1} = \mathcal{S}_{n_0}$, then, for any n, such that $n_0 \leq n < n_1$, it holds that*

$$\mathcal{G}_{n+1} \cup \mathcal{M}_{n+1} \subseteq \mathcal{S}_n \cup \mathcal{S}_n.$$

Proof. Given the assumption that $\mathcal{S}_n$ does not change, both $\mathcal{G}_{n+1} \subseteq \mathcal{S}_n$ and $\mathcal{M}_{n+1} \subseteq \mathcal{S}_n$ follow directly from the definitions of $\mathcal{S}_n$ and $\mathcal{S}_n$. In particular, for $\mathcal{S}_n$, note that for any $\boldsymbol{\theta} \in \mathcal{S}_n$, $e_n^i(\boldsymbol{\theta})$ is decreasing in n for all $i \in \mathcal{I}_c$, since $u_n^i(\boldsymbol{\theta})$ are decreasing in n. For $\mathcal{S}_n$, note that $\max_{\boldsymbol{\theta}' \in \mathcal{S}_n} l_n^J(\boldsymbol{\theta}')$ is increasing in n, while $u_n^J(\boldsymbol{\theta})$ is decreasing in n (see Lemma 11 (i), (ii)). □

When running the SafeOpt-MC algorithm, we repeatedly choose the most uncertain element from $\mathcal{S}_n$ and $\mathcal{S}_n$. Since these sets are contained in each other if the safe set does not expand, we gain more information about these sets with each sample. Since the information gain is bounded, this allows us to bound the uncertainty in terms of the information gain over the entire set:

Lemma 13. *For any $n_1 \geq n_0 \geq 1$, if $\mathcal{S}_{n_1} = \mathcal{S}_{n_0}$ and $\mathcal{C}_1 := 8/\log(1+\sigma^{-2})$, then, for any n, such that $n_0 \leq t \leq n_1$, it holds for all $i \in \mathcal{I}$ that*

$$w_n(\boldsymbol{\theta}_n, i) \leq \sqrt{\frac{\mathcal{C}_1 \beta_n^2 \gamma_{|\mathcal{I}|n}}{n - n_0}}.$$

Proof. Given Lemma 12, the definition of $\boldsymbol{\theta}_n := \operatorname{argmax}_{\boldsymbol{\theta} \in \mathcal{S}_n \cup \mathcal{S}_n}(w_n(\boldsymbol{\theta}))$, and the fact that, $w_n^i(\boldsymbol{\theta}_n) \leq 2\beta_n \max_{\epsilon \in \mathcal{I}} \sigma_{n-1}(\boldsymbol{\theta}_n, i) = 2\beta_n(\boldsymbol{\theta}_n, i_n)$, the proof is completely analogous to that of Lemma 5.3 by Srinivas et al., 2012. We only highlight the main differences here, which results from having several functions.

$$w_n^i(\boldsymbol{\theta}_n) \leq 2\beta_n \max_{i \in \mathcal{I}} \sigma_{n-1}(\boldsymbol{\theta}_n, i), \tag{A.4}$$

$$= 2\beta_n \sigma_{n-1}(\boldsymbol{\theta}_n, i_j), \tag{A.5}$$

which following Srinivas et al., 2012, Lemma 5.4 leads to

$$\sum_{j=1}^{n} w_j^2(\boldsymbol{\theta}_j, i_j) \leq \beta_n^2 \mathcal{C}_1 \frac{1}{2} \sum_{j=1}^{n} \log(1 + \sigma^{-2} \sigma_{n-1}^2(\boldsymbol{\theta}_n, i_j)) \tag{A.6}$$

$$\leq \beta_n^2 \mathcal{C}_1 \frac{1}{2} \sum_{j=1}^{n} \sum_{i \in \mathcal{I}} \log(1 + \sigma^{-2} \sigma_{n-1}^2(\boldsymbol{\theta}_n, i)) \tag{A.7}$$

$$\leq \beta_n^2 \mathcal{C}_1 \, \mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f'), \tag{A.8}$$

$$\leq \mathcal{C}_1 \beta_n^2 \gamma_{|\mathcal{I}|n}, \tag{A.9}$$

where $\mathcal{A}_n = \{\boldsymbol{\theta}_n, i_n\}$ and the last inequality follows from (A.3). □

Corollary 9. *For any $n \geq 1$, if $\mathcal{C}_1$ is defined as above, N_n is the smallest positive integer satisfying $\dfrac{N_n}{\beta_{n+N_n}\gamma_{|\mathcal{I}|(n+N_n)}} \geq \dfrac{\mathcal{C}_1}{\epsilon^2}$, and $\mathcal{S}_{n+N_n} = \mathcal{S}_n$, then, for any $\boldsymbol{\theta} \in \mathcal{G}_{n+N_n} \cup \mathcal{M}_{n+N_n}$, and for all $i \in \mathcal{I}$ it holds that*

$$w_{n+N_n}(\boldsymbol{\theta}, i) \leq \epsilon.$$

Note Where needed in the following lemmas, we assume that $\mathcal{C}_1$ and N_n are defined as above.

That is, after a finite number of evaluations N_n the most uncertain element within these sets is at most ϵ. Given that the reachability operator in (3.7) is defined in terms of the same accuracy, it allows us to show that after at most N_n evaluations, the safe set has to increase unless it is impossible to do so:

Lemma 14. *For any $n \geq 1$, if $\bar{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \varnothing$, then $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n \neq \varnothing$.*

Proof. Assume, to the contrary, that $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n = \varnothing$. By definition, $R_\epsilon(\mathcal{S}_n) \supseteq \mathcal{S}_n$, therefore $R_\epsilon(\mathcal{S}_n) = \mathcal{S}_n$. Iteratively applying R_ϵ to both sides, we get in the limit $\bar{R}_\epsilon(\mathcal{S}_n) = \mathcal{S}_n$. But then, by Lemma 11 (iv) and (vi), we get

$$\bar{R}_\epsilon(\mathcal{S}_0) \subseteq \bar{R}_\epsilon(\mathcal{S}_n) = \mathcal{S}_n, \tag{A.10}$$

which contradicts the lemma's assumption that $\bar{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \varnothing$. □

Lemma 15. *For any $n \geq 1$, if $\bar{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \varnothing$, then the following holds with probability at least $1 - \delta$:*

$$\mathcal{S}_{n+N_n} \supsetneq \mathcal{S}_n.$$

Proof. By Lemma 14, we get that, $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n \neq \varnothing$, Equivalently, by definition, for all $i \in \mathcal{I}_c$

$$\exists \boldsymbol{\theta} \in R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n, \exists \mathbf{z}_i \in \mathcal{S}_n \colon c_i(\mathbf{z}_i) - \epsilon - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0. \tag{A.11}$$

Now, assume, to the contrary, that $\mathcal{S}_{n+N_n} = \mathcal{S}_n$ (see Lemma 11 (iv)), which implies that $\boldsymbol{\theta} \in \mathcal{A} \setminus \mathcal{S}_{n+N_n}$ and $\mathbf{z}_I \in \mathcal{S}_{n+N_n} \forall i \in \mathcal{I}_c$. Then, we have for all $i \in \mathcal{I}_c$

$$\begin{aligned} u^i_{n+N_n}(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) &\geq c_i(\mathbf{z}_i) - Ld(\mathbf{z}, \boldsymbol{\theta}) && \text{by Lemma 1} \\ &\geq c_i(\mathbf{z}_i) - \epsilon - Ld(\mathbf{z}, \boldsymbol{\theta}) \\ &\geq 0. && \text{by (A.11)} \end{aligned}$$

Therefore, by definition, $e_{n+N_n}(\mathbf{z}_i) > 0$, which implies $\mathbf{z}_i \in \mathcal{G}_{n+N_n}$, $\forall i \in \mathcal{I}_c$.

Finally, since $\mathcal{S}_{n+N_n} = \mathcal{S}_n$ and $\mathbf{z}_i \in \mathcal{G}_{n+N_n} \forall i \in \mathcal{I}_c$, we know that for all $i \in \mathcal{I}$, $w_{n+N_n}(\boldsymbol{\theta}', i) \leq \epsilon$. (Corollary 9). Hence, for all $i \in \mathcal{I}_c$,

$$\begin{aligned} l^i_{n+N_n}(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) &\geq c_i(\mathbf{z}_i) - w(\mathbf{z}_i, i) - Ld(\boldsymbol{\theta}, \mathbf{x}_i) && \text{by Lemma 1} \\ &\geq c_i(\mathbf{z}) - \epsilon - Ld(\boldsymbol{\theta}, \mathbf{x}_i) && \text{by Corollary 9} \\ &\geq 0. && \text{by (A.11)} \end{aligned}$$

This means we get $\boldsymbol{\theta} \in \mathcal{S}_{n+N_n}$, which is a contradiction. □

Intuitively, repeatedly applying the previous result leads to full safe exploration within a finite domain $\mathcal{A}$. In particular, it follows that if $\mathcal{S}_{n+N_n} = \mathcal{S}_n$, then the safely reachable set has been fully explored to the desired accuracy. From this it follows, that the pessimistic estimate in (3.16) is also ϵ-close to the optimum value within the safely reachable set, $\bar{R}_\epsilon(\mathcal{S}_0)$:

Lemma 16. *For any $n \geq 1$, if $\mathcal{S}_{n+N_n} = \mathcal{S}_n$, then the following holds with probability at least $1 - \delta$:*

$$J(\boldsymbol{\theta}_{n+N_n}) \geq \max_{\boldsymbol{\theta} \in \bar{R}_\epsilon(\mathcal{S}_0)} J(\boldsymbol{\theta}) - \epsilon.$$

Proof. Let $\boldsymbol{\theta}^* := \operatorname{argmax}_{\boldsymbol{\theta} \in \mathcal{S}_{n+N_n}} J(\boldsymbol{\theta})$. Note that $\boldsymbol{\theta}^* \in \mathcal{M}_{n+N_n}$, since

$$\begin{aligned} u^J_{n+N_n}(\boldsymbol{\theta}^*) &\geq J(\boldsymbol{\theta}^*) && \text{by Lemma 1} \\ &\geq J(\boldsymbol{\theta}) && \text{by definition of } \boldsymbol{\theta}^* \\ &\geq l^J_{n+N_n}(\boldsymbol{\theta}) && \text{by Lemma 1} \\ &\geq \max_{\boldsymbol{\theta} \in \mathcal{S}_{n+N_n}} l^J_{n+N_n}(\boldsymbol{\theta}). && \text{by definition of } \boldsymbol{\theta} \end{aligned}$$

We will first show that $J(\boldsymbol{\theta}_{n+N_n}) \geq J(\boldsymbol{\theta}^*) - \epsilon$. Assume, to the contrary, that

$$J(\boldsymbol{\theta}_{n+N_n}) < J(\boldsymbol{\theta}^*) - \epsilon. \tag{A.12}$$

Then, we have

$$\begin{aligned}
l^J_{n+N_n}(\boldsymbol{\theta}^*) &\leq l^J_{n+N_n}(\boldsymbol{\theta}) && \text{by definition of } \boldsymbol{\theta} \\
&\leq J(\boldsymbol{\theta}) && \text{by Lemma 1} \\
&< J(\boldsymbol{\theta}^*) - \epsilon && \text{by (A.12)} \\
&\leq u^J_{n+N_n}(\boldsymbol{\theta}^*) - \epsilon && \text{by Lemma 1} \\
&\leq l^J_{n+N_n}(\boldsymbol{\theta}^*), && \text{by Corollary 9 and } \boldsymbol{\theta}^* \in \mathcal{M}_{n+N_n}
\end{aligned}$$

which is a contradiction.

Finally, since $\mathcal{S}_{n+N_n} = \mathcal{S}_n$, Lemma 15 implies that $\bar{R}_\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_n = \mathcal{S}_{n+N_n}$. Therefore,

$$\begin{aligned}
\max_{\boldsymbol{\theta} \in \bar{R}_\epsilon(\mathcal{S}_0)} J(\boldsymbol{\theta}) - \epsilon &\leq \max_{\boldsymbol{\theta} \in \mathcal{S}_{n+N_n}} J(\boldsymbol{\theta}) - \epsilon && \bar{R}_\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_{n+N_n} \\
&= J(\boldsymbol{\theta}^*) - \epsilon && \text{by definition of } \boldsymbol{\theta}^* \\
&\leq J(\boldsymbol{\theta}_{n+N_n}). && \text{proven above}
\end{aligned}$$

□

Corollary 10. *For any $n \geq 1$, if $\mathcal{S}_{n+N_n} = \mathcal{S}_n$, then the following holds with probability at least $1 - \delta$:*

$$\forall n' \geq 0, J(\boldsymbol{\theta}_{n+N_n+n'}) \geq \max_{\boldsymbol{\theta} \in \bar{R}_\epsilon(\mathcal{S}_0)} J(\boldsymbol{\theta}) - \epsilon.$$

Proof. This is a direct consequence of the proof of the preceding lemma, combined with the facts that both $\mathcal{S}_{n+N_n+n'}$ and $l^J_{n+N_n+n'}(\boldsymbol{\theta}_{n+N_n+n'})$ are increasing in n' (by Lemma 11 (iv) and (ii) respectively), which imply that $\max_{\boldsymbol{\theta} \in \mathcal{S}_{n+N_n+n'}} l^J_{n+N_n+n'}(\boldsymbol{\theta})$ can only increase in n'. □

Moreover, since we know the true function is contained within the confidence intervals, we cannot go beyond the safe set if we knew the function perfectly everywhere, $\bar{R}_0$:

Lemma 17. *For any $n \geq 0$, the following holds with probability at least $1 - \delta$:*

$$\mathcal{S}_n \subseteq \bar{R}_0(\mathcal{S}_0).$$

Proof. Proof by induction. For the base case, $n = 0$, we have by definition that $\mathcal{S}_0 \subseteq \bar{R}_0(\mathcal{S}_0)$.

For the induction step, assume that for some $n \geq 1$, $\mathcal{S}_{n-1} \subseteq \bar{R}_0(\mathcal{S}_0)$. Let $\boldsymbol{\theta} \in \mathcal{S}_n$, which, by definition, means that for all $i \in \mathcal{I}_c$ $\exists \mathbf{z}_i \in \mathcal{S}_{n-1}$, such that

$$\begin{aligned} & l_n^i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0 \\ \Rightarrow \quad & c_i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \geq 0. \qquad \text{by Lemma 1} \end{aligned}$$

Then, by definition of $\bar{R}_0$ and the fact that $\mathbf{z}_i \in \bar{R}_0(\mathcal{S}_0)$ for all $i \in \mathcal{I}_c$, it follows that $\boldsymbol{\theta} \in \bar{R}_0(\mathcal{S}_0)$. □

The previous results is enough to show that we eventually explore the full safe set by repeatedly applying Lemma 15:

Lemma 18. *Let n^* be the smallest integer, such that $n^* \geq |\bar{R}_0(\mathcal{S}_0)| N_{n^*}$. Then, there exists $n_0 \leq n^*$, such that $\mathcal{S}_{n_0+N_{n_0}} = \mathcal{S}_{n_0}$.*

Proof. Assume, to the contrary, that for any $n \leq n^*$, $\mathcal{S}_n \subsetneq \mathcal{S}_{n+N_n}$. (By Lemma 11 (iv), we know that $\mathcal{S}_n \subseteq \mathcal{S}_{n+N_n}$.) Since N_n is increasing in n, we have

$$\mathcal{S}_0 \subsetneq \mathcal{S}_{n_0} \subseteq \mathcal{S}_{N_{n^*}} \subsetneq \mathcal{S}_{N_{n^*}+N_{N_{n^*}}} \subseteq \mathcal{S}_{2N_{n^*}} \subsetneq \cdots,$$

which implies that, for any $0 \leq k \leq |\bar{R}_0(\mathcal{S}_0)|$, it holds that $|\mathcal{S}_{kN_{n^*}}| > k$. In particular, for $k^* := |\bar{R}_0(\mathcal{S}_0)|$, we get

$$|\mathcal{S}_{k^*T}| > |\bar{R}_0(\mathcal{S}_0)|$$

which contradicts $\mathcal{S}_{k^*T} \subseteq \bar{R}_0(\mathcal{S}_0)$ by Lemma 17. □

Corollary 11. *Let n^* be the smallest integer, such that $\dfrac{n^*}{\beta_{n^*}^2 \gamma_{|\mathcal{I}|n^*}} \geq \dfrac{\mathcal{C}_1 |\bar{R}_0(\mathcal{S}_0)|}{\epsilon^2}$. Then, there exists $n_0 \leq n^*$, such that $\mathcal{S}_{n_0+N_{n_0}} = \mathcal{S}_{n_0}$.*

Proof. This is a direct consequence of combining Lemma 18 and Corollary 9. □

Since we showed that we completely explore the safe set and that we remain safe throughout the exploration procedure, we are ready to state the main results:

Lemma 19. *If f' is L-Lipschitz continuous, then, for any $n \geq 0$, the following holds with probability at least $1 - \delta$ for all $i \in \mathcal{I}_c$:*

$$\forall \boldsymbol{\theta} \in \mathcal{S}_n, c_i(\boldsymbol{\theta}) \geq 0.$$

Proof. We will prove this by induction. For the base case $n = 0$, by definition, for any $\boldsymbol{\theta} \in \mathcal{S}_0$ and $i \in \mathcal{I}_c$, $c_i(\boldsymbol{\theta}) \geq 0$.

For the induction step, assume that for some $n \geq 1$, for any $\boldsymbol{\theta} \in \mathcal{S}_{n-1}$ and for all $i \in \mathcal{I}_c$, $c_i(\boldsymbol{\theta}) \geq 0$. Then, for any $\boldsymbol{\theta} \in \mathcal{S}_n$, by definition, for all $i \in \mathcal{I}_c$, $\exists \mathbf{z}_i \in \mathcal{S}_{n-1}$,

$$\begin{aligned} 0 &\leq l_n^i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) \\ &\leq c_i(\mathbf{z}_i) - Ld(\mathbf{z}_i, \boldsymbol{\theta}) && \text{by Lemma 1} \\ &\leq c_i(\boldsymbol{\theta}). && \text{by } L\text{-Lipschitz-continuity} \end{aligned}$$

□

Theorem 2. *Under the assumptions of Lemma 2, also assume that $\mathcal{S}_0 \neq \emptyset$ and $c_i(\boldsymbol{\theta}) \geq 0$ for all $\boldsymbol{\theta} \in \mathcal{S}_0$ and $i \in \mathcal{I}_c$. Choose β_n as in Lemma 2, define $\hat{\boldsymbol{\theta}}_n$ as in (3.16), and let $n^*(\epsilon, \delta)$ be the smallest positive integer satisfying*

$$\frac{n^*}{\beta_{n^*}^2 \gamma_{|\mathcal{I}|n^*}} \geq \frac{C_1(|\bar{R}_0(\mathcal{S}_0)| + 1)}{\epsilon^2}, \tag{3.17}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$. For any $\epsilon > 0$ and $\delta \in (0, 1)$, when running Algorithm 1 the following inequalities jointly hold with probability at least $1 - \delta$:

1. Safety: $\forall n \geq 1, \forall i \in \mathcal{I}_c \colon c_i(\boldsymbol{\theta}_n) \geq 0$
2. Optimality: $\forall n \geq n^*,\ J(\hat{\boldsymbol{\theta}}_n) \geq J_\epsilon^* - \epsilon$

Proof. The first part of the theorem is a direct consequence of Lemma 19. The second part follows from combining Corollary 10 and Corollary 11. □

B

Proofs for Model Analysis

In this chapter, we provide the proofs for the theoretical claims in Chapter 4. Some of the results in this chapter have been previously published in (Berkenkamp, Moriconi, et al., 2016; Berkenkamp, Turchetta, et al., 2017; Koller, Berkenkamp, Turchetta, Boedecker, et al., 2019). Partial results of the last paper were shown in (Koller, Berkenkamp, Turchetta, and Krause, 2018).

B.1 Noise Bound

We start by bounding the norm of the noise vector $\boldsymbol{\omega}_t$ over all time steps t.

We know that the $\boldsymbol{\omega}_t$ are i.i.d. sub-Gaussian vectors. We exploit the basic properties of sub-Gaussian random variables and refer to Vershynin (2012, Chapter 5) for a concise review.

Lemma 20. *Vershynin (2010, Corollary 5.17) Let $X_1, \ldots, X_p$ be independent centered sub-exponential random variables, and let $2\sigma = \max_i \|X_i\|_{\phi_1}$ be the largest, sub-exponential norm. Then, for every $\epsilon \geq 0$, we have*

$$\mathbb{P}\left\{\left|\sum_{i=1}^{N} X_i\right| \geq \epsilon p\right\} \leq 2\exp\left[\frac{-\mathrm{e}N}{2}\min\left(\frac{\epsilon^2}{4\sigma^2}, \frac{\epsilon}{2\sigma}\right)\right] \tag{B.1}$$

This allows us to bound the 2-norm of the noise vectors in (4.1).

Lemma 21. *Let $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_p)$ be a vector with i.i.d. elements $[\boldsymbol{\omega}]_i = \omega_i$ that are σ-sub-Gaussian. Then, with probability at least $1 - \delta$, we have that*

$$\|\boldsymbol{\omega}\|_2^2 \leq 2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2}{\delta} \tag{B.2}$$

Proof. Since the ω_i are σ-sub-Gaussian, we have the ω_i^2 are 2σ-sub-exponential Vershynin, 2010, Lemma 5.14. Thus we have

$$\|\boldsymbol{\omega}\|_2^2 = \sum_{i=1}^{p} \omega_i^2,$$

where the ω_i^2 are i.i.d. 2σ-sub-exponential. Following Lemma 20, we have

$$\mathbb{P}\left\{\|\boldsymbol{\omega}\|_2^2 \geq \epsilon p\right\} \leq 2\exp\left[\frac{-\mathrm{e}p}{2} \min\left(\frac{\epsilon^2}{4\sigma^2}, \frac{\epsilon}{2\sigma}\right)\right] \tag{B.3}$$

Now for $\epsilon \geq 2\sigma$ we have $\epsilon^2/(4\sigma^2) \geq \epsilon/(2\sigma)$. Thus

$$\mathbb{P}\left\{\|\boldsymbol{\omega}\|_2^2 \geq (2\sigma + \epsilon)p\right\} \leq 2\exp\left[\frac{-\mathrm{e}p}{2} \frac{(2\sigma + \epsilon)}{2\sigma}\right] \leq 2\exp\left[\frac{-\mathrm{e}p}{2} \frac{\epsilon}{2\sigma}\right] \tag{B.4}$$

We want to upper bound the right hand side by δ. so

$$2\exp\left[\frac{-\mathrm{e}p\epsilon}{4\sigma}\right] \leq \delta, \tag{B.5}$$

$$\frac{-\mathrm{e}p\epsilon}{4\sigma} \leq \log(\delta/2), \tag{B.6}$$

$$\frac{\mathrm{e}p\epsilon}{4\sigma} \geq \log(2/\delta), \tag{B.7}$$

$$\epsilon \geq \frac{4\sigma}{\mathrm{e}p} \log(2/\delta). \tag{B.8}$$

the result follows by plugging the bound for ϵ into (B.4),

$$(2\sigma + \epsilon)p = (2\sigma + \frac{4\sigma}{\mathrm{e}p} \log(2/\delta))p \tag{B.9}$$

$$= 2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2}{\delta} \tag{B.10}$$

□

As the last step, we apply the union bound to obtain confidence intervals over multiple steps.

Lemma 4. *Let $\boldsymbol{\omega}_0, \boldsymbol{\omega}_1, \ldots$ be i.i.d. random vectors with $\boldsymbol{\omega}_t \in \mathbb{R}^p$ such that each entry of the vector is i.i.d. σ-sub-Gaussian. Then, with probability at least $(1-\delta)$,*

$$\|\boldsymbol{\omega}_t\|_2^2 \leq 2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{(t+1)^2\pi^2}{3\delta} \tag{4.11}$$

holds jointly for all $t \geq 0$.

Proof. At each time step t, we apply a probability budget of δ/π_t to the bound in Lemma 21, where $\pi_t \geq 0$ and $\sum_{t\geq 0} \pi_t^{-1} = 1$. In particular, we use $\pi_t = \frac{(t+1)^2\pi^2}{6}$ as in Srinivas et al., 2012, Lemma 5.1, so that we apply monotonically decreasing probability thresholds as t increases. We obtain the result by applying a union bound over t, since $\sum_{t\geq 0} \delta/\pi_t = \delta$. □

B.2 Lyapunov Stability

In this section, we prove the results for Theorem 4. In the following we write $f(\mathbf{a}) = f(\mathbf{x}, \mathbf{u})$ to be concise.

Lemma 22. *Using Corollary 3 and Lemma 2, let $\mathcal{X}_\kappa$ be a discretization of $\mathcal{X}$ such that $\|\mathbf{x} - [\mathbf{x}]_\kappa\|_2 \leq \kappa$ for all $\mathbf{x} \in \mathcal{X}$. Then, for all $\mathbf{x} \in \mathcal{X}$, we have with probability at least $1-\delta$ for all $n \geq 0$ that*

$$\Big|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v([\mathbf{x}]_\kappa) - \Big(v(f(\mathbf{a})) - v(\mathbf{x})\Big)\Big| \leq L_v\beta_n\|\boldsymbol{\sigma}_n([\mathbf{a}]_\kappa)\|_2 + (L_vL_f(L_\pi+1)+L_v)\kappa, \tag{B.11}$$

where $\mathbf{a} = (\mathbf{x}, \pi(\mathbf{x}))$ and $[\mathbf{a}]_\kappa = ([\mathbf{x}]_\kappa, \pi([\mathbf{x}]_\kappa))$.

Proof. Let $\mathbf{a} = (\mathbf{x}, \pi(\mathbf{x}))$, $[\mathbf{a}]_\kappa = ([\mathbf{x}]_\kappa, \pi([\mathbf{x}]_\kappa))$. Then we have that

$$\begin{aligned}
&\Big|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v([\mathbf{x}]_\kappa) - \Big(v(f(\mathbf{a})) - v(\mathbf{x})\Big)\Big|, \\
&= \Big|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v([\mathbf{x}]_\kappa) - v(f(\mathbf{a})) + v(\mathbf{x})\Big|, \\
&= \Big|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v(f([\mathbf{a}]_\kappa)) + v(f([\mathbf{a}]_\kappa)) - v(f(\mathbf{a})) + v(\mathbf{x}) - v([\mathbf{x}]_\kappa)\Big|, \\
&\leq \Big|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v(f([\mathbf{a}]_\kappa))\Big| + \Big|v(f([\mathbf{a}]_\kappa)) - v(f(\mathbf{a}))\Big| + \Big|v(\mathbf{x}) - v([\mathbf{x}]_\kappa)\Big|, \\
&\leq L_v\|\boldsymbol{\mu}_n([\mathbf{a}]_\kappa) - f([\mathbf{a}]_\kappa)\|_2 + L_v\|f([\mathbf{a}]_\kappa) - f(\mathbf{a})\|_2 + L_v\|\mathbf{x} - [\mathbf{x}]_\kappa\|_2, \\
&\leq L_v\beta_n\|\boldsymbol{\sigma}_n([\mathbf{a}]_\kappa)\|_2 + L_vL_f\|[\mathbf{a}]_\kappa - \mathbf{a}\|_2 + L_v\|\mathbf{x} - [\mathbf{x}]_\kappa\|_2,
\end{aligned}$$

where the last three inequalities follow from Corollaries 3 and 5. The result holds with probability at least $1-\delta$. By definition of the discretization and the policy class Π in Assumption 5, we have on each grid cell that

$$\begin{aligned}\|\mathbf{a}-[\mathbf{a}]_\kappa\|_2 &\leq \|\mathbf{x}-[\mathbf{x}]_\kappa\|_2 + \|\pi(\mathbf{x})-\pi([\mathbf{x}]_\kappa)\|_2,\\ &\leq \kappa + L_\pi\|\mathbf{x}-[\mathbf{x}]_\kappa\|_2,\\ &\leq (L_\pi+1)\kappa,\end{aligned}$$

where the equality in the first step follows from the definition of the norm. Plugging this into the previous bound yields

$$\left|v(\boldsymbol{\mu}_n([\mathbf{a}]_\kappa)) - v([\mathbf{x}]_\kappa) - \Big(v(f(\mathbf{a})) - v(\mathbf{x})\Big)\right| \leq L_v\beta_n\|\boldsymbol{\sigma}_n([\mathbf{a}]_\kappa)\|_2 + \left(L_vL_f(1+L_\pi)+L_v\right)\kappa,$$

which completes the proof. □

Lemma 23. *$v(f(\mathbf{x},\mathbf{u})) \in \mathcal{Q}_n$ holds for all $\mathbf{x}\in\mathcal{X}$, $\mathbf{u}\in\mathcal{U}$, and $n>0$ with probability at least $(1-\delta)$.*

Proof. The proof is analogous to Lemma 22 and follows from Corollaries 3 and 5. □

Corollary 12. *$v(f(\mathbf{x},\mathbf{u})) \in \mathcal{C}_n$ holds for all $\mathbf{x}\in\mathcal{X}$, $\mathbf{u}\in\mathcal{U}$, and $n>0$ with probability at least $(1-\delta)$.*

Proof. Direct consequence of the fact that Lemma 23 holds jointly for all $n>0$ with probability at least $1-\delta$. □

Lemma 22 show that the decrease on the Lyapunov function on the discrete grid $\mathcal{X}_\kappa$ is close to that on the continuous domain $\mathcal{X}$. Given these confidence intervals, we can now establish the region of attraction using Corollary 1:

Theorem 4. *Under Corollaries 3 and 4 with $L_{\Delta v} := L_vL_f(L_\pi+1)+L_v$, let $\mathcal{X}_\kappa$ be a discretization of $\mathcal{X}$ such that $\|\mathbf{x}-[\mathbf{x}]_\kappa\|_2 \leq \kappa$ for all $\mathbf{x}\in\mathcal{X}$. If, for all $\mathbf{x}\in\mathcal{V}(c)\cap\mathcal{X}_\kappa$ with $c>0$, $\mathbf{u}=\pi(\mathbf{x})$, and for all $n\geq 0$ it holds that*

$$u_n(\mathbf{x},\mathbf{u}) < v(\mathbf{x}) - L_{\Delta v}\kappa, \tag{4.13}$$

then $v(f(\mathbf{x},\pi(\mathbf{x}))) < v(\mathbf{x})$ holds for all $\mathbf{x}\in\mathcal{V}(c)$ with probability at least $(1-\delta)$ and $\mathcal{V}(c)$ is a positive invariant region of attraction for (2.2) under the policy π.

Proof. Using Lemma 22 it holds that $v(f(\mathbf{x}, \pi(\mathbf{x})) - v(\mathbf{x}) < 0$ for all continuous states $\mathbf{x} \in \mathcal{V}(c)$ with probability at least $1 - \delta$, since all discrete states $\mathbf{x}_\kappa \in \mathcal{V}(c) \cap \mathcal{X}$ fulfill the condition (4.13). Thus we can use Corollary 1 to conclude that $\mathcal{V}(c)$ is a region of attraction for (2.2). □

C

Proofs for Safe Exploration

In this chapter, we prove the safe exploration results in Chapter 5. Some of the results in this chapter have been previously published in (Berkenkamp, Moriconi, et al., 2016; Berkenkamp, Turchetta, et al., 2017; Koller, Berkenkamp, Turchetta, Boedecker, et al., 2019). Partial results of the last paper were shown in (Koller, Berkenkamp, Turchetta, and Krause, 2018).

We start by restating the stability result from Theorem 4 in terms of the set $\mathcal{F}_n$ from (5.4).

Theorem 5. *Let $\mathcal{R}_{\pi_n}$ be the true region of attraction of (4.1) under the policy π_n. For any $\delta \in (0, 1)$, we have with probability at least $(1 - \delta)$ that $\mathcal{V}(c_n) \subseteq \mathcal{R}_{\pi_n}$ for all $n > 0$. Moreover, for any $\mathbf{x}_0 \in \mathcal{V}(c_n)$, we have for all $t \geq 0$ that*

$$\mathbf{x}_t \in \mathcal{X}, \qquad \pi(\mathbf{x}_t) \in \mathcal{U} \tag{5.3}$$

Proof. Following the definition of $\mathcal{F}_n$ in (5.1), it is clear from the constraint in the optimization problem (5.2) that for all $\mathbf{x} \in \mathcal{F}_n$ it holds that $(\mathbf{x}, \pi_n(\mathbf{x})) \in \mathcal{F}_n$ or, equivalently, that $u_n(\mathbf{x}, \pi(\mathbf{x})) - v(\mathbf{x}) < -L_{\Delta v}\kappa$, see (5.1). The result $\mathcal{V}(c_n) \subseteq \mathcal{R}_{\pi_n}$ then follows from Theorem 4. Since $c < c_{\max}$ by definition in (5.2), the state constraints follow from Lemma 5. The input constraints are satisfied by assumption in Assumption 6. □

Note that the initialization of the confidence intervals $\mathcal{Q}_0$ ensures that the decrease condition is always fulfilled for the initial policy.

C.1 Safe Exploration

Remark 4. In the following we assume that $\mathcal{F}_n$ and $\mathcal{S}_n$ are defined as in (5.4) and (5.5).

Baseline As a baseline, we consider a class of algorithms that know about the Lipschitz continuity properties of v, f, and π. In addition, we can learn about $v(f(\mathbf{x}, \mathbf{u}))$ up to some arbitrary statistical accuracy ϵ by visiting state $\mathbf{x}$ and obtaining a measurement for the next state after applying action $\mathbf{u}$, but face the safety restrictions defined in Section 5.1. Suppose we are given a set $\mathcal{S}$ of state-action pairs about which we can learn safely. Specifically, this means that we have a policy such that, for any state-action pair $(\mathbf{x}, \mathbf{u})$ in $\mathcal{S}$, if we apply action $\mathbf{u}$ in state $\mathbf{x}$ and then apply actions according to the policy, the state converges to the origin. Such a set can be constructed using the initial policy π_0 from Section 5.1 as $\mathcal{S}_0 = \{(\mathbf{x}, \pi_0(\mathbf{x})) \,|\, \mathbf{x} \in \mathcal{S}_0^x\}$.

The goal of the algorithm is to expand this set of states that we can learn about safely. Thus, we need to estimate the region of attraction by certifying that state-action pairs achieve the $-L_{\Delta v}\kappa$ decrease condition in Theorem 4 by learning about state-action pairs in $\mathcal{S}$. We can then generalize the gained knowledge to unseen states by exploiting the Lipschitz continuity,

$$R^{\text{dec}}(\mathcal{S}) = \mathcal{S}_0 \cup \Big\{ \mathbf{a} \in \mathcal{X}_\kappa \times \mathcal{U}_\kappa \,|\, \exists (\mathbf{x}, \mathbf{u}) \in \mathcal{S} \colon v(f(\mathbf{x}, \mathbf{u})) - v(\mathbf{x}) + \epsilon + L_{\Delta v} \|\mathbf{a} - (\mathbf{x}, \mathbf{u})\|_2 < -L_{\Delta v}\kappa \Big\}, \tag{C.1}$$

where we use that we can learn $v(f(\mathbf{x}, \mathbf{u}))$ up to ϵ accuracy within $\mathcal{S}$. We specifically include $\mathcal{S}_0$ in this set, to allow for initial policies that are safe, but does not meet the strict decrease requirements of Theorem 4. Given that all states in $R^{\text{dec}}(\mathcal{S})$ fulfill the requirements of Theorem 4, we can estimate the corresponding region of attraction by committing to a control policy $\pi \in \Pi_L$ and estimating the largest safe level set of the Lyapunov function. With $\mathcal{D} = R^{\text{dec}}(\mathcal{S})$, the operator

$$R^{\text{lev}}(\mathcal{F}) = \mathcal{V}\Big(\operatorname{argmax} c, \quad \text{such that } \exists \pi \in \Pi_L \colon \forall \mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa, \, (\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{F} \Big) \tag{C.2}$$

encodes this operation. It optimizes over safe policies $\pi \in \Pi_L$ to determine the largest level set, such that all state-action pairs $(\mathbf{x}, \pi(\mathbf{x}))$ at discrete states $\mathbf{x}$ in the level set $\mathcal{V}(c) \cap \mathcal{X}_\kappa$ fulfill the decrease condition of Theorem 4. As a result, $R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}))$ is an estimate of the largest region of attraction given the ϵ-accurate knowledge about state-action pairs in $\mathcal{S}$. Based on this increased region of attraction, there are more states that we can safely

learn about. Specifically, we again use the Lipschitz constant and statistical accuracy ϵ to determine all states that map back into the region of attraction,

$$R_\epsilon(\mathcal{S}) = \mathcal{S} \cup \Big\{ \mathbf{a}' \in R_\kappa^{\text{lev}}(R^{\text{dec}}(\mathcal{S})) \times \mathcal{U}_\kappa \,|\, \exists \mathbf{a} \in \mathcal{S} \colon v(f(\mathbf{a})) + \epsilon + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \leq \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}))} v(\mathbf{x}) \Big\}, \tag{C.3}$$

where $R_\kappa^{\text{lev}}(\mathcal{F}) = R^{\text{lev}}(\mathcal{F}) \cap \mathcal{X}_\kappa$. Thus, $R_\epsilon(\mathcal{S}) \supseteq \mathcal{S}$ contains state-action pairs that we can visit to learn about the system. Repeatedly applying this operator leads the largest set of state-action pairs that any safe algorithm with the same knowledge and restricted to policies in Π_L could hope to reach. Specifically, let $R_\epsilon^0(\mathcal{S}) = \mathcal{S}$ and $R_\epsilon^{i+1}(\mathcal{S}) = R_\epsilon(R_\epsilon^i(\mathcal{S}))$. Then $\overline{R}_\epsilon(\mathcal{S}) = \lim_{i \to \infty} R_\epsilon^i(\mathcal{S})$ is the set of all state-action pars on the discrete grid that any algorithm could hope to classify as safe without leaving this safe set. Moreover, $R^{\text{lev}}(\overline{R}_\epsilon(\mathcal{S}))$ is the largest corresponding region of attraction that any algorithm can classify as safe for the given Lyapunov function.

Proofs In the following we implicitly assume that the assumptions of Corollary 4 hold and that β_n is defined as specified within Corollary 4. Moreover, for ease of notation we assume that $\mathcal{S}_0^x$ is a level set of the Lyapunov function $v(\cdot)$.

Lemma 24. *$\mathcal{V}(c_n) = R^{\text{lev}}(\mathcal{F}_n)$ and $c_n = \max_{\mathbf{x} \in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x})$*

Proof. Directly by definition, compare (5.2) and (C.2). □

Remark 5. Lemma 24 allows us to write the proofs entirely in terms of operators, rather than having to deal with explicit policies. In the following and in Algorithm 6 we replace $\mathcal{V}(c_n)$ and c_n according to Lemma 24. This moves the definitions closer to the baseline and makes for an easier comparison.

We roughly follow the proof strategy in (Sui et al., 2015), but deal with the additional complexity of having safe sets that are defined in a more difficult way (indirectly through the policy). This is non-trivial and the safe sets are carefully designed in order to ensure that the algorithm works for general nonlinear systems.

We start by listing some fundamental properties of the sets that we defined below.

Lemma 25. *It holds for all $n \geq 1$ that*

(i) $\forall \mathbf{a} \in \mathcal{X}_\kappa \times \mathcal{U}_\kappa,\ u_{n+1}(\mathbf{a}) \leq u_n(\mathbf{a})$

Algorithm 6 Theoretical algorithm

Input: Initial safe policy $\mathcal{S}_0$, dynamics model $\mathcal{GP}(\mu(\mathbf{a}), k(\mathbf{a}, \mathbf{a}'))$
for all $n = 1, \dots$ **do**
  $\mathcal{F}_n = \bigcup_{(\mathbf{x},\mathbf{u})\in\mathcal{S}_{n-1}} \left\{\mathbf{a}' \in \mathcal{X}_\kappa \times \mathcal{U}_\kappa \,|\, u_n(\mathbf{x},\mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|\mathbf{a}' - (\mathbf{x},\mathbf{u})\|_2 < -L_{\Delta v}\kappa\right\}$,
  $\pi_n, c_n = \operatorname{argmax}_{\pi\in\Pi_L, c\in\mathbb{R}_{>0}} c$, such that for all $\mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa$: $(\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{F}_n$

  $\mathcal{S}_n = \bigcup_{\mathbf{a}\in\mathcal{S}_{n-1}} \left\{\mathbf{a}' \in \mathcal{V}(c_n) \cap \mathcal{X}_\kappa \times \mathcal{U}_\kappa \,|\, u_n(\mathbf{a}) + L_v L_f\|\mathbf{a} - \mathbf{a}'\|_2 \le c_n\right\}$
    $= \bigcup_{\mathbf{a}\in\mathcal{S}_{n-1}} \left\{\mathbf{a}' \in R^{\text{lev}}_\kappa(\mathcal{F}_n) \times \mathcal{U}_\kappa \,|\, u_n(\mathbf{a}) + L_v L_f\|\mathbf{a} - \mathbf{a}'\|_2 \le \max_{\mathbf{x}\in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x})\right\}$
  $(\mathbf{x}_n, \mathbf{u}_n) = \operatorname{argmax}_{(\mathbf{x},\mathbf{u})\in\mathcal{S}_n} u_n(\mathbf{x},\mathbf{u}) - l_n(\mathbf{x},\mathbf{u})$
  $\mathcal{S}_n = \{\mathbf{a} \in \mathcal{V}(c_n) \times \mathcal{U}_\kappa \,|\, u_n(\mathbf{a}) \le c_n\}$
  Update GP with measurements $f(\mathbf{x}_n, \mathbf{u}_n) + \epsilon_n$

(ii) $\forall \mathbf{a} \in \mathcal{X}_\kappa \times \mathcal{U}_\kappa,\ l_{n+1}(\mathbf{a}) \ge l_n(\mathbf{a})$

(iii) $\mathcal{S} \subseteq \mathcal{R} \implies R^{\text{lev}}(\mathcal{S}) \subseteq R^{\text{lev}}(\mathcal{R})$

(iv) $\mathcal{S} \subseteq \mathcal{R} \implies R^{\text{dec}}(\mathcal{S}) \subseteq R^{\text{dec}}(\mathcal{R})$

(v) $\mathcal{S} \subseteq \mathcal{R} \implies R_\epsilon(\mathcal{S}) \subseteq R_\epsilon(\mathcal{R})$

(vi) $\mathcal{S} \subseteq \mathcal{R} \implies \overline{R}_\epsilon(\mathcal{S}) \subseteq \overline{R}_\epsilon(\mathcal{R})$

(vii) $\mathcal{S}_n \supseteq \mathcal{S}_{n-1} \implies \mathcal{F}_{n+1} \supseteq \mathcal{F}_n$

(viii) $\mathcal{F}_1 \supseteq \mathcal{S}_0$

(ix) $\mathcal{S}_n \supseteq \mathcal{S}_{n-1}$

(x) $\mathcal{F}_n \supseteq \mathcal{F}_{n-1}$

Proof. (i) and (ii) follow directly form the definition of $\mathcal{C}_n$.

(iii) Let $\pi \in \Pi_L$ be a policy such that for some $c > 0$ it holds for all $\mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa$ that $(\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{S}$. Then we have that $(\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{R}$, since $\mathcal{S} \subseteq \mathcal{R}$. Thus it follows that with

$$c_s = \operatorname{argmax} c \quad \text{s.t. } \exists \pi \in \Pi_L : \forall \mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa,\ (\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{S} \tag{C.4}$$

and

$$c_r = \operatorname{argmax} c \quad \text{s.t. } \exists \pi \in \Pi_L : \forall \mathbf{x} \in \mathcal{V}(c) \cap \mathcal{X}_\kappa,\ (\mathbf{x}, \pi(\mathbf{x})) \in \mathcal{R} \tag{C.5}$$

we have that $c_r \ge c_s$. This implies $\mathcal{V}(c_r) \supseteq \mathcal{V}(c_s)$. The result follows.

(iv) Let $\mathbf{a} \in R^{\text{dec}}(\mathcal{S})$. Then there exists $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}$ such that $v(f(\mathbf{x}, \mathbf{u})) - v(\mathbf{x}) + \epsilon + L_{\Delta v}\|\mathbf{a} - (\mathbf{x}, \mathbf{u})\|_2 < -L_{\Delta v}\kappa$. Since $\mathcal{S} \subseteq \mathcal{R}$ we have that $(\mathbf{x}, \mathbf{u}) \in \mathcal{R}$ as well and thus $\mathbf{a} \in R^{\text{dec}}(\mathcal{R})$.

(v) $\mathcal{S} \subseteq \mathcal{R} \implies R^{\text{lev}}(R^{\text{dec}}(\mathcal{S})) \subseteq R^{\text{lev}}(R^{\text{dec}}(\mathcal{R}))$ due to (iii) and (iv). Since $\mathbf{a}' \in R_\epsilon(\mathcal{S})$, there must exist an $\mathbf{a} \in \mathcal{S}$ such that $v(f(\mathbf{a})) + \epsilon + L_v L_f\|\mathbf{a} - \mathbf{a}'\|_2 \leq \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}))} v(\mathbf{x})$. Since $\mathcal{S} \subseteq \mathcal{R}$ it follows that $\mathbf{a} \in \mathcal{R}$. Moreover,

$$\max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}))} v(\mathbf{x}) \leq \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{R}))} v(\mathbf{x}) \tag{C.6}$$

follows from $R^{\text{lev}}(R^{\text{dec}}(\mathcal{S})) \subseteq R^{\text{lev}}(R^{\text{dec}}(\mathcal{R}))$, so that we conclude that $\mathbf{a}' \in R_\epsilon(\mathcal{R})$.

(vi) This follows directly by repeatedly applying the result of (v).

(vii) Let $\mathbf{a}' \in \mathcal{F}_n$. Then $\exists (\mathbf{x}, \mathbf{u}) \in \mathcal{S}_{n-1}\colon u_n(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|\mathbf{a}' - (\mathbf{x}, \mathbf{u})\|_2 < -L_{\Delta v}\kappa$. Since $\mathcal{S}_n \supseteq \mathcal{S}_{n-1}$ it follows that $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}_n$ as well. Moreover, we have

$$\begin{aligned} &u_{n+1}(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|\mathbf{a}' - (\mathbf{x}, \mathbf{u})\|_2 \\ \leq\ &u_n(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|\mathbf{a}' - (\mathbf{x}, \mathbf{u})\|_2 < -L_{\Delta v}\kappa \end{aligned}$$

since u_{n+1} is non-increasing, see (i). Thus $\mathbf{a}' \in \mathcal{F}_{n+1}$.

(viii) By definition of $\mathcal{C}_0$ we have for all $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}_0$ that $u_0(\mathbf{x}, \mathbf{u}) < v(\mathbf{x}) - L_{\Delta v}\kappa$. Now we have that

$$\begin{aligned} &u_1(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}) + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}, \mathbf{u})\|_2, \\ =\ &u_1(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}), \\ \leq\ &u_0(\mathbf{x}, \mathbf{u}) - v(\mathbf{x}), \quad \text{by Lemma 25 (i)} \\ <\ &- L_{\Delta v}\kappa, \end{aligned}$$

which implies that $(\mathbf{x}, \mathbf{u}) \in \mathcal{F}_1$.

(ix) Proof by induction. We consider the base case, $\mathbf{a} \in \mathcal{S}_0$, which implies that $\mathbf{a} \in \mathcal{F}_1$ by (viii). Moreover, since $\mathcal{S}_0^x$ is a level set of the Lyapunov function v by assumption, we have that $R^{\text{lev}}(\mathcal{S}_0) = \mathcal{S}_0^x$. The previous statements together with (iii) imply that $\mathbf{a} \in R_\kappa^{\text{lev}}(\mathcal{F}_1) \times \mathcal{U}_\kappa$, since $\mathcal{F}_1 \supseteq \mathcal{S}_0$ by (viii). Now, we have that

$$u_1(\mathbf{a}) + L_v L_f\|\mathbf{a} - \mathbf{a}\|_2 = u_1(\mathbf{a}) \overset{(i)}{\leq} u_0(\mathbf{a}).$$

Moreover, by definition of $\mathcal{C}_0$, we have for all $(\mathbf{x}, \mathbf{u}) \in \mathcal{S}_0$ that

$$u_0(\mathbf{x}, \mathbf{u}) < v(\mathbf{x}) - L_{\Delta v}\kappa < v(\mathbf{x}). \tag{C.7}$$

As a consequence,

$$\begin{aligned} u_0(\mathbf{x}, \mathbf{u}) &\leq \max_{(\mathbf{x},\mathbf{u})\in\mathcal{S}_0} v(\mathbf{x}), && \text{(C.8)} \\ &= \max_{\mathbf{x}\in R^{\text{lev}}(\mathcal{S}_0)} v(\mathbf{x}), && \text{(C.9)} \\ &\leq \max_{\mathbf{x}\in R^{\text{lev}}(\mathcal{F}_1)} v(\mathbf{x}), && \text{(C.10)} \end{aligned}$$

where the last inequality follows from (iii) and (viii). Thus we have $\mathbf{a} \in \mathcal{S}_1$.

For the induction step, assume that for $n \geq 2$ we have $\mathbf{a}' \in \mathcal{S}_n$ with $\mathcal{S}_n \supseteq \mathcal{S}_{n-1}$. Now since $\mathbf{a}' \in \mathcal{S}_n$ we must have that $\mathbf{a}' \in R^{\text{lev}}_\kappa(\mathcal{F}_n) \times \mathcal{U}_\kappa$. This implies that $\mathbf{a}' \in R^{\text{lev}}_\kappa(\mathcal{D}_{n+1}) \times \mathcal{U}_\kappa$, due to Lemma 25 (iii) and (vii) together with the induction assumption of $\mathcal{S}_n \supseteq \mathcal{S}_{n-1}$. Moreover, there must exist a $\mathbf{a} \in \mathcal{S}_{n-1} \subseteq \mathcal{S}_n$ such that

$$\begin{aligned} u_{n+1}(\mathbf{a}) + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2, &\leq u_n(\mathbf{a}) + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2, && \text{(C.11)} \\ &\leq \max_{\mathbf{x}\in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x}), && \text{(C.12)} \\ &\leq \max_{\mathbf{x}\in R^{\text{lev}}(\mathcal{F}_{n+1})} v(\mathbf{x}), && \text{(C.13)} \end{aligned}$$

which in turn implies $\mathbf{a} \in \mathcal{S}_{n+1}$. The last inequality follows from Lemma 25 (iii) and (vii) together with the induction assumption that $\mathcal{S}_n \supseteq \mathcal{S}_{n-1}$.

(x) This is a direct consequence of (vii), (viii), and (ix).

□

Given these set properties, we first consider what happens if the safe set $\mathcal{S}_n$ does not expand after collecting data points. We use these results later to conclude that the safe set must either expand or that the maximum level set is reached. We denote by

$$\mathbf{a}_n = (\mathbf{x}_n, \mathbf{u}_n) \tag{C.14}$$

the data point the is sampled according to (5.6).

Lemma 26. *For any $n_1 \geq n_0 \geq 1$, if $\mathcal{S}_{n_1} = \mathcal{S}_{n_0}$, then for any n such that $n_0 \leq n < n_1$, it holds that*

$$2\beta_n \sigma_n(\mathbf{a}_n) \leq \sqrt{\frac{C_1 q \beta_n^2 \gamma_n}{n - n_0}}, \tag{C.15}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Proof. We modify the results for $q = 1$ by (Srinivas et al., 2012) to this lemma, but use the different definition for β_n from Chowdhury and A. Gopalan, 2017. Even though the goal of (Srinivas et al., 2012, Lemma 5.4) is different from ours, we can still apply their reasoning to bound the amplitude of the confidence interval of the dynamics. In particular, in (Srinivas et al., 2012, Lemma 5.4), we have $r_n = 2\beta_n \sigma_{n-1}(\mathbf{a}_n)$ with $\mathbf{a}_n = (\mathbf{x}_n, \mathbf{u}_n)$ according to Corollary 4. Then

$$\begin{aligned}
r_n^2 &= 4\beta_n^2 \sigma_{n-1}^2(\mathbf{a}_n), && \text{(C.16)}\\
&= 4\beta_n^2 \left(\sum_{i=1}^{q} \sigma_{n-1}(\mathbf{a}_n, i) \right)^2, && \text{(C.17)}\\
&\leq 4\beta_n^2 q \sum_{i=1}^{q} \sigma_{n-1}^2(\mathbf{a}_n, i) \qquad \text{(Jensen's inequality)}, && \text{(C.18)}\\
&\leq 4\beta_n^2 q \sigma^2 C_2 \sum_{i=1}^{q} \log(1 + \sigma^{-2} \sigma_{n-1}^2(\mathbf{a}_n, i)), && \text{(C.19)}
\end{aligned}$$

where $C_2 = \sigma^{-2}/\log(1 + \sigma^{-2})$. The result then follows analogously to (Srinivas et al., 2012, Lemma 5.4) by noting that

$$\sum_{j=1}^{n} r_j^2 \leq C_1 \beta_n^2 q \gamma_n \qquad \forall n \geq 1 \tag{C.20}$$

according to the definition of γ_n in this paper and using the Cauchy-Schwartz inequality. □

The previous result allows us to bound the width of the confidence intervals:

Corollary 13. *For any $n_1 \geq n_0 \geq 1$, if $\mathcal{S}_{n_1} = \mathcal{S}_{n_0}$, then for any n such that $n_0 \leq n < n_1$, it holds that*

$$u_n(\mathbf{a}_n) - l_n(\mathbf{a}_n) \leq L_v \sqrt{\frac{C_1 q \beta_n^2 \gamma_n}{n - n_0}}, \tag{C.21}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Proof. Direct consequence of Lemma 26 together with the definition of $\mathcal{C}$ and $\mathcal{Q}$. □

Corollary 14. *For any $n \geq 1$ with C_1 as defined in Lemma 26, let N_n be the smallest integer satisfying $\frac{N_n}{\beta^2_{n+N_n}\gamma_{n+N_n}} \geq \frac{C_1 L_v^2 q}{\epsilon^2}$ and $\mathcal{S}_{n+N_n} = \mathcal{S}_{N_n}$, then, for any $\mathbf{a} \in \mathcal{S}_{n+N_n}$ it holds that*

$$u_n(\mathbf{a}) - l_n(\mathbf{a}) \leq \epsilon. \tag{C.22}$$

Proof. The result trivially follows from substituting N_n in the bound in Corollary 13. □

Lemma 27. *For any $n \geq 1$, if $\overline{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \emptyset$, then $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n \neq \emptyset$.*

Proof. As in (Sui et al., 2015, Lemma 6). Assume, to the contrary, that $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n = \emptyset$. By definition $R_\epsilon(\mathcal{S}_n) \supseteq \mathcal{S}_n$, therefore $R_\epsilon(\mathcal{S}_n) = \mathcal{S}_n$. Iteratively applying R_ϵ to both sides, we get in the limit $\overline{R}_\epsilon(\mathcal{S}_n) = \mathcal{S}_n$. But then, by Lemma 25,(vi) and (ix), we get

$$\overline{R}_\epsilon(\mathcal{S}_0) \subseteq \overline{R}_\epsilon(\mathcal{S}_n) = \mathcal{S}_n, \tag{C.23}$$

which contradicts the assumption that $\overline{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \emptyset$. □

Lemma 28. *For any $n \geq 1$, if $\overline{R}_\epsilon(\mathcal{S}_0) \setminus \mathcal{S}_n \neq \emptyset$, then the following holds with probability at least $1 - \delta$:*

$$\mathcal{S}_{n+N_n} \supset \mathcal{S}_n. \tag{C.24}$$

Proof. By Lemma 27, we have that $R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n \neq \emptyset$. By definition, this means that there exist $\mathbf{a} \in R_\epsilon(\mathcal{S}_n) \setminus \mathcal{S}_n$ and $\mathbf{a}' \in \mathcal{S}_n$ such that

$$v(f(\mathbf{a}')) + \epsilon + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \leq \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}_n))} v(\mathbf{x}) \tag{C.25}$$

Now we assume, to the contrary, that $\mathcal{S}_{n+N_n} = \mathcal{S}_n$ (the safe set cannot decrease due to Lemma 25 (ix)). This implies that $\mathbf{a} \in \mathcal{X}_\kappa \times \mathcal{U}_\kappa \setminus \mathcal{S}_{n+N_n}$ and $\mathbf{a}' \in \mathcal{S}_{n+N_n} = \mathcal{S}_{n+N_n-1}$. Due to Corollary 13, it follows that

$$\begin{aligned}
& u_{n+N_n}(\mathbf{a}') + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 && && \text{(C.26)}\\
\leq\ & v(f(\mathbf{a}')) + \epsilon + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 && && \text{(C.27)}\\
\leq\ & \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}_n))} v(\mathbf{x}) && \text{by (C.25)} && \text{(C.28)}\\
=\ & \max_{\mathbf{x} \in R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}_{n+N_n}))} v(\mathbf{x}) && \text{by (iii), (iv) and (ix)} && \text{(C.29)}
\end{aligned}$$

Thus, to conclude that $\mathbf{a} \in \mathcal{S}_{n+N_n}$ according to (5.5), we need to show that $R^{\text{lev}}(\mathcal{F}_{n+N_n}) \supseteq R^{\text{lev}}(R^{\text{dec}}(\mathcal{S}_n))$. To this end, we use Lemma 25 (iii) and show that $\mathcal{F}_{n+N_n} \supseteq R^{\text{dec}}(\mathcal{S}_{n+N_n})$. Consider $(\mathbf{x}, \mathbf{u}) \in R^{\text{dec}}(\mathcal{S}_{n+N_n})$, we know that there exists a $(\mathbf{x}', \mathbf{u}') \in \mathcal{S}_{n+N_n} = \mathcal{S}_{n+N_n-1}$ such that

$$-L_{\Delta v}\kappa > v(f(\mathbf{x}', \mathbf{u}')) - v(\mathbf{x}') + \epsilon + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2, \tag{C.30}$$

$$\geq u_{n+N_n}(\mathbf{x}', \mathbf{u}') - v(\mathbf{x}') + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2, \tag{C.31}$$

where the second inequality follows from Corollary 13. This implies that $(\mathbf{x}, \mathbf{u}) \in \mathcal{F}_n$ and thus $\mathcal{F}_{n+N_n} \supseteq R^{\text{dec}}(\mathcal{S}_{n+N_n})$. This, in turn, implies that $\mathbf{a} \in \mathcal{S}_{n+N_n}$, which is a contradiction. □

Lemma 29. *For any $n \geq 0$, the following holds with probability at least $1 - \delta$:*

$$\mathcal{S}_n \subseteq \overline{R}_0(\mathcal{S}_0). \tag{C.32}$$

Proof. Proof by induction. For the base case, $n = 0$, we have $\mathcal{S}_0 \subseteq \overline{R}_0(\mathcal{S}_0)$ by definition. For the induction step, assume that for some $n \geq 1$, $\mathcal{S}_{n-1} \subseteq \overline{R}_0(\mathcal{S}_0)$. Let $\mathbf{a} \in \mathcal{S}_n$. Then, by definition, $\exists \mathbf{a}' \in \mathcal{S}_{n-1}$ such that

$$u_n(\mathbf{a}') + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \leq \max_{\mathbf{x} \in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x}), \tag{C.33}$$

which, by Corollary 12, implies that

$$v(f(\mathbf{a}')) + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \leq \max_{\mathbf{x} \in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x}) \tag{C.34}$$

Now since $\mathbf{a}' \in \overline{R}_0(\mathcal{S}_0)$ by the induction hypothesis, in order to conclude that $\mathbf{a} \in \overline{R}_0(\mathcal{S}_0)$ we need to show that $R^{\text{lev}}(\mathcal{F}_n) \subseteq R^{\text{lev}}(R^{\text{dec}}(\overline{R}(\mathcal{S}_0)))$.

Let $(\mathbf{x}, \mathbf{u}) \in \mathcal{F}_n$, then there exist $(\mathbf{x}', \mathbf{a}') \in \mathcal{S}_{n-1}$ such that

$$u_{n-1}(\mathbf{x}', \mathbf{u}') - v(\mathbf{x}') + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2 < -L_{\Delta v}\kappa, \tag{C.35}$$

which, by Corollary 12, implies that

$$v(f(\mathbf{x}', \mathbf{u}')) - v(\mathbf{x}') + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2 < -L_{\Delta v}\kappa, \tag{C.36}$$

which means that $(\mathbf{x}, \mathbf{u}) \in R^{\text{dec}}(\overline{R}_0(\mathcal{S}_0))$ since $\mathcal{S}_{n-1} \subseteq \overline{R}_0(\mathcal{S}_0)$ and therefore $(\mathbf{x}', \mathbf{u}') \in \overline{R}_0(\mathcal{S}_0)$ holds by the induction hypothesis. We use (iii) to conclude that $R^{\text{lev}}(\mathcal{F}_n) \subseteq R^{\text{lev}}(R^{\text{dec}}(\overline{R}(\mathcal{S}_0)))$, which concludes the proof. □

Lemma 30. *Let n^* be the smallest integer, such that $n^* \geq |\overline{R}_0(\mathcal{S}_0)|N_{n^*}$. Then, there exists $n_0 \leq n^*$ such that $\mathcal{S}_{n_0+N_{n_0}} = \mathcal{S}_{n_0}$ holds with probability $1-\delta$.*

Proof. By contradiction. Assume, to the contrary, that for all $n \leq n^*$, $\mathcal{S}_n \subset \mathcal{S}_{n+N_n}$. From Lemma 25 (ix) we know that $\mathcal{S}_n \subseteq \mathcal{S}_{n+N_n}$. Since N_n is increasing in n, we have that $N_n \leq N_{n^*}$. Thus, we must have

$$\mathcal{S}_0 \subset \mathcal{S}_{N_{n^*}} \subset \mathcal{S}_{2N_{n^*}} \cdots, \tag{C.37}$$

so that for any $0 \leq j \leq |\overline{R}_0(\mathcal{S}_0)|$, it holds that $|S_{jT_{n^*}}| > j$. In particular, for $j = |\overline{R}_0(\mathcal{S}_0)|$, we get

$$|\mathcal{S}_{jN_{n^*}}| > |\overline{R}_0(\mathcal{S}_0)|, \tag{C.38}$$

which contradicts $\mathcal{S}_{jN_{n^*}} \subseteq \overline{R}_0(\mathcal{S}_0)$ from Lemma 29. □

Corollary 15. *Let n^* be the smallest integer such that*

$$\frac{n^*}{\beta_{n^*}\gamma_{n^*}} \geq \frac{C_1 L_v^2 q |\overline{R}_0(\mathcal{S}_0)|}{\epsilon^2}, \tag{C.39}$$

then there exists a $n_0 \leq n^*$ such that $\mathcal{S}_{n_0+N_{n_0}} = \mathcal{S}_{n_0}$.

Proof. A direct consequence of Lemma 30 and Corollary 14. □

C.2 Safety and Policy Adaptation

In the following, we denote the true region of attraction of (4.1) under a policy π by $\mathcal{R}_\pi$.

Lemma 31. *$R^{\text{lev}}(\mathcal{F}_n) \subseteq \mathcal{R}_{\pi_n}$ for all $n \geq 0$.*

Proof. By definition, we have for all $(\mathbf{x}, \mathbf{u}) \in \mathcal{F}_n$ that the exists $(\mathbf{x}', \mathbf{u}') \in \mathcal{S}_{n-1}$ such that

$$\begin{aligned} -L_{\Delta v}\kappa &\geq u_n(\mathbf{x}', \mathbf{u}') - v(\mathbf{x}') + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2, \\ &\geq v(f(\mathbf{x}', \mathbf{u}')) - v(\mathbf{x}') + L_{\Delta v}\|(\mathbf{x}, \mathbf{u}) - (\mathbf{x}', \mathbf{u}')\|_2, \\ &\geq v(f(\mathbf{x}, \mathbf{u})) - v(\mathbf{x}), \end{aligned}$$

where the first inequality follows from Corollary 12 and the second one by Lipschitz continuity, see Lemma 22.

By definition of R^{lev} in (C.2), it follows that for all $\mathbf{x} \in R^{\text{lev}}(\mathcal{F}_n) \cap \mathcal{X}_\kappa$ we have that $(\mathbf{x}, \pi_n(\mathbf{x})) \in \mathcal{F}_n$. Moreover, $R^{\text{lev}}(\mathcal{F}_n)$ is a level set of the Lyapunov function by definition. Thus the result follows from Theorem 4. □

Lemma 32. $f(\mathbf{x}, \mathbf{u}) \in \mathcal{R}_{\pi_n} \forall (\mathbf{x}, \mathbf{u}) \in \mathcal{S}_n$.

Proof. This holds for $\mathcal{S}_0$ by definition. For $n \geq 1$, by definition, we have for all $\mathbf{a} \in \mathcal{S}_n$ there exists an $\mathbf{a}' \in \mathcal{S}_{n-1}$ such that

$$\begin{aligned}\max_{\mathbf{x} \in R^{\text{lev}}(\mathcal{F}_n)} v(\mathbf{x}) &\geq u_n(\mathbf{a}') + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \\ &\geq v(f(\mathbf{a}')) + L_v L_f \|\mathbf{a} - \mathbf{a}'\|_2 \\ &\geq v(f(\mathbf{a}))\end{aligned}$$

where the first inequality follows from Corollary 12 and the second one by Lipschitz continuity, see Lemma 22. Since $R^{\text{lev}}(\mathcal{F}_n) \subseteq \mathcal{R}_{\pi_n}$ by Lemma 31, we have that $f(\mathbf{a}) \in \mathcal{R}_{\pi_n}$. □

Theorem 6. *Under the assumptions of Theorem 4 and Corollary 3, with β_n as in Corollary 4, and with measurements collected according to (5.6), let n^* be the smallest positive integer so that*

$$\frac{n^*}{\beta_{n^*}^2 \gamma_{n^*}} \geq \frac{Cp(|\overline{R}(\mathcal{S}_0)| + 1)}{L_v^2 \epsilon^2}, \tag{5.7}$$

where $C = 8/\log(1+\sigma^{-2})$. Let $\mathcal{R}_\pi$ be the true region of attraction of (4.1) under a policy π. For any $\epsilon > 0$, and $\delta \in (0, 1)$, the following holds jointly with probability at least $(1 - \delta)$ for all $n > 0$:

(i) $\mathcal{V}(c_n) \subseteq \mathcal{R}_{\pi_n}$,

(ii) $\mathcal{V}(c_n) \subseteq \mathcal{X}$,

(iii) $f(\mathbf{x}, \mathbf{u}) \in \mathcal{R}_{\pi_n} \forall (\mathbf{x}, \mathbf{u}) \in \mathcal{S}_n$,

(iv) $\overline{R}_\epsilon(\mathcal{S}_0) \subseteq \mathcal{S}_n \subseteq \overline{R}_0(\mathcal{S}_0)$.

Proof. See Lemmas 31 and 32 for (i) and (iii), respectively. Part (iv) is a direct consequence of Corollary 15 and Lemma 29. □

D

Proofs for Exploration Regret Bound

Some of the results in this chapter have been previously published in (Berkenkamp, Moriconi, et al., 2016; Berkenkamp, Turchetta, et al., 2017; Koller, Berkenkamp, Turchetta, Boedecker, et al., 2019). Partial results of the last paper were shown in (Koller, Berkenkamp, Turchetta, and Krause, 2018).

In the following, we implicitly denote with $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_t$ the states visited under the true and optimistic dynamics under the policy parameterized by $\boldsymbol{\theta}$, whenever it is clear from context. Moreover, we drop the argument $\pi_{\boldsymbol{\theta}}(\cdot)$ and write $\boldsymbol{\sigma}_n(\mathbf{x}) = \boldsymbol{\sigma}_n(\mathbf{x}, \pi_{\boldsymbol{\theta}}(\mathbf{x}))$ for ease of notation. Moreover, we drop the subscript n from $\mathbf{x}_{t,n}$ whenever it is clear that we refer to the nth episode. Lastly, when no norm is specified, $\|\cdot\| = \|\cdot\|_2$ refers to the two-norm.

Lemma 33. *Under the assumptions of Corollary 4, with probability at least* $(1-\delta)$ *we have for all* $n \geq 0$ *that the regret* r_n *is bounded by*

$$r_n = J(\boldsymbol{\theta}^*) - J(\boldsymbol{\theta}_n) \leq \tilde{J}_n(\boldsymbol{\theta}_n) - J(\boldsymbol{\theta}_n) \tag{D.1}$$

Proof. By Corollary 4, we know that the true dynamics under $\boldsymbol{\theta}^*$ are contained within the feasible region of (5.21).

As a consequence, we have $\tilde{J}_n(\boldsymbol{\theta}) \geq J(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \mathcal{D}$ and $n \geq 1$. Now since $\boldsymbol{\theta}^*$ is in the feasible region of (5.22), we must have that $\tilde{J}_n(\boldsymbol{\theta}_n) \geq J(\boldsymbol{\theta}^*)$. The result follows. □

Thus, to bound the instantaneous regret r_n, we must bound the difference between $\tilde{J}(\theta_n)$ and $J(\boldsymbol{\theta}_n)$. We can use the Lipschitz continuity properties to obtain

Lemma 34. *Under Assumptions 5 and 7, then*

$$|J(\boldsymbol{\theta}_n) - \tilde{J}_n(\boldsymbol{\theta}_n)| \leq L_r(1 + L_\pi) \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{\omega}=\tilde{\boldsymbol{\omega}}}[\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\|_2] \tag{D.2}$$

Proof.

$$|J(\boldsymbol{\theta}_n) - \tilde{J}_n(\boldsymbol{\theta}_n)| = \left| \mathbb{E}_{\boldsymbol{\omega}} \left[\sum_{t=0}^{T} r(\mathbf{x}_t, \pi_{\boldsymbol{\theta}_n}(\mathbf{x}_t)) \right] - \mathbb{E}_{\tilde{\boldsymbol{\omega}}} \left[\sum_{t=0}^{T} r(\tilde{\mathbf{x}}_t, \pi_{\boldsymbol{\theta}_n}(\tilde{\mathbf{x}}_t)) \right] \right| \tag{D.3}$$

$$= \left| \mathbb{E}_{\boldsymbol{\omega}=\tilde{\boldsymbol{\omega}}} \left[\sum_{t=0}^{T} r(\mathbf{x}_t, \pi_{\boldsymbol{\theta}_n}(\mathbf{x}_t)) - r(\tilde{\mathbf{x}}_t, \pi_{\boldsymbol{\theta}_n}(\tilde{\mathbf{x}}_t)) \right] \right| \tag{D.4}$$

$$\leq L_r(1 + L_\pi) \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{\omega}=\tilde{\boldsymbol{\omega}}}[\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|] \tag{D.5}$$

□

Lemma 35. *Under the assumptions of Corollaries 4 and 7, let* $\bar{L}_f = 1 + L_f + 2\beta_{n-1}L_\sigma$. *Then, for any sequence of* $\boldsymbol{\eta}_t \in [-1, 1]^p$, *any sequence of* $\boldsymbol{\omega}_t$ *with* $\tilde{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t$, $\boldsymbol{\theta} \in \mathcal{D}$, *and* $1 \leq t \leq T$ *we have that*

$$\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\| \leq 2\beta_{n-1}\bar{L}_f^{T-1} \sum_{i=0}^{t-1} \|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.6}$$

Proof. We start by showing that, for any $t \geq 1$ we have

$$\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\| \leq 2\beta_{n-1} \sum_{i=0}^{t-1} (L_f + 2\beta_{n-1}L_\sigma)^{t-1-i} \|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.7}$$

by induction. For the base case we have $\tilde{\mathbf{x}}_0 = \mathbf{x}_0$. Consequently, at n we have

$$\|\mathbf{x}_{1,n} - \tilde{\mathbf{x}}_{1,n}\| = \|h(\mathbf{x}_0) + g(\mathbf{x}_0) + \boldsymbol{\omega}_0 - h(\mathbf{x}_0) - \boldsymbol{\mu}_{n-1}(\mathbf{x}_0) - \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\mathbf{x}_0)\boldsymbol{\eta}_0 - \tilde{\boldsymbol{\omega}}_0\| \tag{D.8}$$

$$\leq \|g(\mathbf{x}_0) - \boldsymbol{\mu}_{n-1}(\mathbf{x}_0)\| + \beta_{n-1}\|\boldsymbol{\Sigma}_{n-1}(\mathbf{x}_0)\boldsymbol{\eta}_0\| \tag{D.9}$$

$$\leq \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| + \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| \tag{D.10}$$

$$= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| \tag{D.11}$$

For the induction step assume that (D.7) holds at time step t. Subsequently we have at iteration n that

$$\begin{aligned}
\|\mathbf{x}_{t+1,n} - \tilde{\mathbf{x}}_{t+1,n}\| &= \|h(\mathbf{x}_t) + g(\mathbf{x}_t) + \boldsymbol{\omega}_t - h(\tilde{\mathbf{x}}_t) - \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t) - \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\tilde{\mathbf{x}}_t)\boldsymbol{\eta}_t - \tilde{\boldsymbol{\omega}}_t\| \\
&= \|h(\mathbf{x}_t) + g(\mathbf{x}_t) - h(\tilde{\mathbf{x}}_t) - \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t) - \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\tilde{\mathbf{x}}_t)\boldsymbol{\eta}_t + g(\tilde{\mathbf{x}}_t) - g(\tilde{\mathbf{x}}_t)\| \\
&= \|g(\tilde{\mathbf{x}}_t) - \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t) - \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\tilde{\mathbf{x}}_t)\boldsymbol{\eta}_t + h(\mathbf{x}_t) + g(\mathbf{x}_t) - h(\tilde{\mathbf{x}}_t) - g(\tilde{\mathbf{x}}_t)\| \\
&= \|g(\tilde{\mathbf{x}}_t) - \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t)\| + \|\beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\tilde{\mathbf{x}}_t)\boldsymbol{\eta}_t\| + \|h(\mathbf{x}_t) + g(\mathbf{x}_t) - h(\tilde{\mathbf{x}}_t) - g(\tilde{\mathbf{x}}_t)\| \\
&\leq \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_t)\| + \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_t)\| + L_f\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_t)\| + L_f\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t) + \boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_t) - \boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + L_f\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&\leq 2\beta_{n-1}\left(\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + L_\sigma\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|\right) + L_f\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + (L_f + 2\beta_{n-1}L_\sigma)\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&\leq 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\tilde{\mathbf{x}}_t)\| + (L_f + 2\beta_{n-1}L_\sigma)2\beta_{n-1}\sum_{i=0}^{t-1}(L_f + 2\beta_{n-1}L_\sigma)^{t-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_i)\| \\
&= 2\beta_{n-1}\sum_{i=0}^{(t+1)-1}(L_f + 2\beta_{n-1}L_\sigma)^{(t+1)-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_i)\|
\end{aligned}$$

Thus (D.7) holds. Now since $t \leq T$ we have

$$\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\| \leq 2\beta_{n-1}\sum_{i=0}^{t-1}(L_f + 2\beta_{n-1}L_\sigma)^{t-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.12}$$

$$\leq 2\beta_{n-1}\sum_{i=0}^{t-1}(1 + L_f + 2\beta_{n-1}L_\sigma)^{t-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.13}$$

$$\leq 2\beta_{n-1}\underbrace{(1 + L_f + 2\beta_{n-1}L_\sigma)}_{:=\bar{L}_f}^{T-1}\sum_{i=0}^{t-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.14}$$

$$\tag{D.15}$$

□

Corollary 16. *For any sequence of $\eta_t \in [-1, 1]$, $\boldsymbol{\theta} \in \mathcal{D}$, and $t \geq 1$, $n \geq 1$ we have that*

$$\mathbb{E}_{\boldsymbol{\omega}=\tilde{\boldsymbol{\omega}}}[\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\|] \leq 2\beta_{n-1}\bar{L}_f^{T-1}\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{t-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|\right] \tag{D.16}$$

Proof. This is a direct consequence of Lemma 49. □

As a direct consequence of these lemmas we can bound the regret in terms of the GP variance in expectation over the states visited under the true dynamics.

Lemma 36. *Under the assumption of Corollary 4 and Assumptions 5 and 7, let* $L_J = 2L_r(1+L_\pi)\beta_{n-1}\bar{L}_f^{T-1}$. *Then, with probability at least* $(1-\delta)$ *it holds for all* $n \geq 0$ *that*

$$r_n^2 \leq L_J^2 T^3 \mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{t=0}^{T-1}\sum_{i=1}^{p} \sigma_{n-1}(\mathbf{x}_{t,n}, \pi_{\boldsymbol{\theta}}(\mathbf{x}_{t,n}), i)^2\right] \tag{D.17}$$

Proof.

$$\begin{aligned}
r_n &\leq |J(\boldsymbol{\theta}_n) - \tilde{J}(\boldsymbol{\theta}_n)| && \text{(D.18)}\\
&\leq L_r(1+L_\pi)\sum_{t=0}^{T}\mathbb{E}_{\boldsymbol{\omega}=\tilde{\boldsymbol{\omega}}}[\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\|_2] && \text{(D.19)}\\
&\leq 2L_r(1+L_\pi)\beta_{n-1}\bar{L}_f^{T-1}\sum_{t=0}^{T}\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{t-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2\right] && \text{(D.20)}\\
&\leq 2L_r(1+L_\pi)\beta_{n-1}\bar{L}_f^{T-1}T\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{T-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2\right] && \text{(D.21)}
\end{aligned}$$

where the third inequality follows from Corollary 16. Now, let $L_J = 2L_r(1+L_\pi)\beta_{n-1}\bar{L}_f^{T-1}$, so that

$$\begin{aligned}
r_n &\leq L_J T\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{T-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2\right] && \text{(D.22)}\\
r_n^2 &\leq L_J^2 T^2\left(\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{T-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2\right]\right)^2 && \text{(D.23)}\\
&\leq L_J^2 T^2\mathbb{E}_{\boldsymbol{\omega}}\left[\left(\sum_{i=0}^{T-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2\right)^2\right] && \text{(D.24)}\\
&\leq L_J^2 T^3\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{T-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|_2^2\right] && \text{(D.25)}\\
&\leq L_J^2 T^3\mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{i=0}^{T-1}\sum_{j=1}^{p}\sigma_{n-1}(\mathbf{x}_{i,n}, j)^2\right] && \text{(D.26)}
\end{aligned}$$

□

To obtain regret bounds, we must bound the expectation in Lemma 36 by the sublinear information capacity, as in the proofs for GP-UCB in (Srinivas et al., 2012).

Lemma 37 (Srinivas et al. (2012))**.** $s^2 \leq \frac{s_{\max}^2}{\log(1+s_{\max}^2)}\log(1+s^2)$ *for all* $s \in [0, s_{\max}^2]$

Lemma 38. *Let* $\max_{\mathbf{a}\in\mathcal{A}} k(\mathbf{a},\mathbf{a}) \leq k_{\max}$. *Then*

$$\boldsymbol{\sigma}_n^2(\mathbf{a}) \leq \frac{k_{\max}}{\log(1+\sigma^{-2}k_{\max})}\log(1+\sigma^{-2}\sigma_n^2(\mathbf{a})) \tag{D.27}$$

Proof.

$$\sigma_n^2(\mathbf{a}) \leq \sigma^2(\sigma^{-2}\sigma_n^2(\mathbf{a})) \tag{D.28}$$

Now $\sigma^{-2}\sigma_n^2(\mathbf{a}) \leq \sigma^{-2}k(\mathbf{a},\mathbf{a}) \leq \sigma^{-2}k_{\max}$. Thus, we can use Lemma 37 to obtain

$$\sigma_n^2(\mathbf{a}) \leq \sigma^2\frac{\sigma^{-2}k_{\max}}{\log(1+\sigma^{-2}k_{\max})}\log(1+\sigma^{-2}\sigma_n^2(\mathbf{a})) \tag{D.29}$$

$$= \frac{k_{\max}}{\log(1+\sigma^{-2}k_{\max})}\log(1+\sigma^{-2}\sigma_n^2(\mathbf{a})) \tag{D.30}$$

□

Lemma 39. *Let* $\mathcal{A}_n = \{\mathbf{x}_{t,i}\}_{t=0,i=0}^{T-1,n} \times \mathcal{I}_p$ *denote the states visited up to iteration* n *and* $\mathbf{y}_{\mathcal{A}_n}$ *the corresponding observations. Then*

$$\frac{1}{2}\sum_{n=1}^{N}\sum_{t=0}^{T-1}\sum_{j=1}^{p}\log(1+\sigma^{-2}\sigma_{n-1}^2(\mathbf{x}_{t,n},j)) \leq Tp\,\mathrm{I}(\mathbf{y}_{\mathcal{A}_N};f_{\mathcal{A}_N}) \tag{D.31}$$

Proof.

$$\frac{1}{2}\sum_{n=1}^{N}\sum_{t=0}^{T-1}\sum_{j=1}^{p}\log(1+\sigma^{-2}\sigma_{n-1}^2(\mathbf{x}_{t,n},j)) \tag{D.32}$$

$$= \sum_{t=0}^{T-1}\sum_{j=1}^{p}\frac{1}{2}\sum_{n=1}^{N}\log(1+\sigma^{-2}\sigma_{n-1}^2(\mathbf{x}_{t,n},j)) \tag{D.33}$$

$$\leq \sum_{t=0}^{T-1}\sum_{j=1}^{p}\mathrm{I}(\mathbf{y}_{\mathcal{A}_N};f_{\mathcal{A}_N}) \tag{D.34}$$

$$= Tp\,\mathrm{I}(\mathbf{y}_{\mathcal{A}_N};f_{\mathcal{A}_N}) \tag{D.35}$$

Where the second to last step follows from Srinivas et al., 2012, Lemma 2 together with $\log(1+x) \geq 0$ for $x \geq 0$ and the properties of the mutual information. In particular, the inner sum conditions on $(n-1)Tp$ measurements, but sums only over the one element $(\mathbf{x}_{t,n},j)$. The mutual information in Srinivas et al., 2012, Lemma 2 instead sums over every element that we condition on in the next step. By adding the missing non-negative terms together with the fact that the mutual information is independent of the order of the observations we obtain the result. □

We can use this in conjunction with Lemma 36 to obtain

Lemma 40. *Under the assumption of Corollary 4 and Assumptions 5 and 7, let* $\max_{\mathbf{x}\in\mathbb{R}^p} k(\mathbf{x},\mathbf{x}) \leq k_{\max}$. *Then, with probability at least* $(1-\delta)$ *it holds for all* $n \geq 0$ *that*

$$R_n^2 \leq \frac{k_{\max} N L_J^2 T^4 p}{\log(1+\sigma^{-2}k_{\max})} \mathbb{E}_{\boldsymbol{\omega}}[\,\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f_{\mathcal{A}_n})] \tag{D.36}$$

Proof.

$$\begin{aligned}
R_n^2 &\leq N \sum_{n=1}^{N} r_n^2 && \text{Jensen's} && \text{(D.37)}\\
&\leq N \sum_{n=1}^{N} L_J^2 T^3 \mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{t=0}^{T-1}\sum_{i=1}^{p} \sigma_{n-1}(\mathbf{x}_{t,n}, \mathbf{u}_{t,n}, i)^2\right] && \text{Lemma 36} && \\
& && && \text{(D.38)}\\
&\leq \frac{k_{\max} N L_J^2 T^3}{\log(1+\sigma^{-2}k_{\max})} \mathbb{E}_{\boldsymbol{\omega}}\left[\sum_{n=1}^{N}\sum_{t=0}^{T-1}\sum_{i=1}^{p} \log(1+\sigma^{-2}\sigma_{n-1}(\mathbf{x}_{t,n}, \mathbf{u}_{t,n}, i)^2\right] && \text{Lemma 38} && \text{(D.39)}\\
&\leq \frac{k_{\max} N L_J^2 T^4 p}{\log(1+\sigma^{-2}k_{\max})} \mathbb{E}_{\boldsymbol{\omega}}[\,\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f_{\mathcal{A}_n})] && \text{Lemma 39} && \\
& && && \text{(D.40)}
\end{aligned}$$

□

Where $\mathbb{E}_{\boldsymbol{\omega}}[\,\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; f_{\mathcal{A}_n})]$ is the expected mutual information obtained by visiting state/action pairs in $\mathcal{A}_n$, in expectation over the transition noise of all rounds.

To obtain an instance-independent bound, we must bound the mutual information by the worst-case mutual information as in (Srinivas et al., 2012). One subtle problem is that the domain over which we obtain measurements is unbounded due to the sub-Gaussian transition noise, which can have unbounded support. In the following, we first provide a high-probability bound on the domain over all iterations and then bound the mutual information.

D.1 Bounding the Domain Under Aleatoric Uncertainty

We exploit the σ-sub-Gaussian property of the transition noise and build on Lemmas 20 and 21 to obtain a bound over the domain. We start by applying a union bound on Lemma 21 over the time horizon T.

Lemma 41. *Let $\boldsymbol{\omega}_0, \dots, \boldsymbol{\omega}_{T-1}$ be vectors with $\boldsymbol{\omega}_i \in \mathbb{R}^p$ such that each entry of the vector is i.i.d. σ-sub-Gaussian. Then, with probability at least $(1-\delta)$,*

$$\sum_{t=0}^{T-1} \|\boldsymbol{\omega}_i\|_2 \leq T\sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T}{\delta}} \tag{D.41}$$

Proof. Now using Lemma 21 with probability threshold δ/T and applying the union bound we, get that $\|\boldsymbol{\omega}_i\|_2^2 \leq 2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T}{\delta}$ holds for all $0 \leq i \leq T-1$ with probability at least $1-\delta$.

Now, first using Jensen's inequality and then plugging in the bound for $\|\boldsymbol{\omega}_i\|_2^2$, we obtain

$$\begin{aligned}
\sum_{t=1}^{T} \|\boldsymbol{\omega}_t\|_2 &= \sum_{i=0}^{T-1} \sqrt{\|\boldsymbol{\omega}_t\|_2^2} && \text{(D.42)}\\
&\leq \sqrt{T}\sqrt{\sum_{t=0}^{T-1} \|\boldsymbol{\omega}_t\|_2^2} && \text{(D.43)}\\
&\leq \sqrt{T}\sqrt{\sum_{t=0}^{T-1} \left(2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T}{\delta}\right)} && \text{(D.44)}\\
&= T\sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T}{\delta}} && \text{(D.45)}
\end{aligned}$$

□

Lastly, we use a union bound over all iterations similar to Srinivas et al., 2012, Lemma 5.1.

Lemma 42. *Let $\boldsymbol{\omega}_{t,n}$ be the random vectors as in Lemma 41 at iteration n. Then, with probability $(1-\delta)$ we have for all $n \geq 1$ that*

$$\sum_{t=1}^{T} \|\boldsymbol{\omega}_{t,n}\|_2 \leq T\sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{T\pi^2 n^2}{3\delta}} \tag{D.46}$$

Proof. At each iteration n, we apply a probability budget of δ/ρ_n to the bound in Lemma 41, where $\rho_n \geq 0$ and $\sum_{n\geq 1} \rho_n^{-1} = 1$. In particular, we use $\rho_n = \frac{n^2\pi^2}{6}$ as in Srinivas et al., 2012, Lemma 5.1, so that we apply monotonically decreasing probability thresholds as n increases. We obtain the result by applying a union bound over n, since $\sum_{n\geq 1} \delta/\rho_n = \delta$. □

Now that we can bound the noise over all iterations, we can bound the domain over which the system acts with a compact set.

Lemma 43. *Let f be L_f-Lipschitz continuous with respect to the norm $\|\cdot\|$. Then we have for all $t \geq 1$ that*

$$\|\mathbf{x}_t - \mathbf{x}_0\| \leq \sum_{i=0}^{t-1} L_f^i \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{t-1} L_f^{t-1-i} \|\boldsymbol{\omega}_i\| \tag{D.47}$$

$$\leq (1 + L_f)^{t-1} \left(t\|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{t-1} \|\boldsymbol{\omega}_i\| \right) \tag{D.48}$$

Proof. We first proof (D.47) by induction. For the base case we have

$$\|\mathbf{x}_1 - \mathbf{x}_0\| = \|f(\mathbf{x}_0) + \boldsymbol{\omega}_0 - \mathbf{x}_0\| \tag{D.49}$$

$$\leq \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|\boldsymbol{\omega}_0\|, \tag{D.50}$$

$$= L_f^0 \|f(\mathbf{x}_0) - \mathbf{x}_0\| + L_f^0 \|\boldsymbol{\omega}_0\|. \tag{D.51}$$

For the induction step, assume that the assumption holds for some t. Then,

$$\|\mathbf{x}_{t+1} - \mathbf{x}_0\| = \|f(\mathbf{x}_t) + \boldsymbol{\omega}_t - \mathbf{x}_0\| \tag{D.52}$$

$$= \|f(\mathbf{x}_t) - f(\mathbf{x}_0) + f(\mathbf{x}_0) - \mathbf{x}_0 + \boldsymbol{\omega}_t\| \tag{D.53}$$

$$\leq \|f(\mathbf{x}_t) - f(\mathbf{x}_0)\| + \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|\boldsymbol{\omega}_t\| \tag{D.54}$$

$$\leq L_f \|\mathbf{x}_t - \mathbf{x}_0\| + \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|\boldsymbol{\omega}_t\| \tag{D.55}$$

$$\leq L_f \left(\sum_{i=0}^{t-1} L_f^i \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{t-1} L_f^{t-1-i} \|\boldsymbol{\omega}_i\| \right) \tag{D.56}$$

$$+ \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|\boldsymbol{\omega}_t\| \tag{D.57}$$

$$= \sum_{i=1}^{(t-1)+1} L_f^i \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|f(\mathbf{x}_0) - \mathbf{x}_0\|$$

$$+ \sum_{i=0}^{t-1} L_f^{(t+1)-1-i} \|\boldsymbol{\omega}_i\| + \|\boldsymbol{\omega}_t\| \tag{D.58}$$

$$= \sum_{i=0}^{(t-1)+1} L_f^i \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{(t+1)-1} L_f^{(t+1)-1-i} \|\boldsymbol{\omega}_i\| \tag{D.59}$$

Which concludes the proof. For (D.48), note that $L_f^i \leq (1 + L_f)^t$ for all $i \leq t$. Thus we have

$$\sum_{i=0}^{t-1} L_f^i \|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{t-1} L_f^{t-1-i} \|\boldsymbol{\omega}_i\| \tag{D.60}$$

$$\leq L_f^{t-1} \sum_{i=0}^{t-1} \left(\|f(\mathbf{x}_0) - \mathbf{x}_0\| + \|\boldsymbol{\omega}_i\| \right) \tag{D.61}$$

$$= L_f^{t-1} \left(t\|f(\mathbf{x}_0) - \mathbf{x}_0\| + \sum_{i=0}^{t-1} \|\boldsymbol{\omega}_i\| \right) \tag{D.62}$$

□

Lemma 44. *Let* $b_n = L_f^{T-1} T \left(B_g + B_h + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{T\pi^2 n^2}{3\delta}} \right)$. *Then, with probability at least* $(1-\delta)$, *we have for all iterations* $n \geq 1$ *and corresponding time steps* $0 \leq t \leq T$ *that*

$$\mathbf{x}_{t,n} \in \mathbb{B}(\mathbf{x}_0, b_n), \tag{D.63}$$

where $\mathbb{B}(\mathbf{x}_0, b_n) = \{\mathbf{x} \in \mathbb{R}^p \mid \|\mathbf{x} - \mathbf{x}_0\|_2 \leq b_n\}$ *is a norm-ball centered around* $\mathbf{x}_0$ *with radius* b_n.

Proof. From Lemma 43, we have for all $n \geq 1$, $0 \leq t \leq T$ that

$$\|\mathbf{x}_{t,n} - \mathbf{x}_0\|_2 \leq (1 + L_f)^{t-1} \left(t \|f(\mathbf{x}_0) - \mathbf{x}_0\|_2 + \sum_{i=0}^{t-1} \|\boldsymbol{\omega}_i\|_2 \right) \tag{D.64}$$

Now by Assumption 3 and Combined with Lemma 42, we obtain

$$\begin{aligned} \|\mathbf{x}_{t,n} - \mathbf{x}_0\|_2 &\leq (1 + L_f)^{t-1} \left(t \|f(\mathbf{x}_0) - \mathbf{x}_0\|_2 + t\sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{t\pi^2 n^2}{3\delta}} \right) && \text{(D.65)} \\ &\leq (1 + L_f)^{T-1} T \left(\|f(\mathbf{x}_0) - \mathbf{x}_0\|_2 + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{T\pi^2 n^2}{3\delta}} \right) && \text{(D.66)} \\ &:= b_n && \text{(D.67)} \end{aligned}$$

Lastly, we have $\|f(\mathbf{x}_0) - \mathbf{x}_0\|_2 \leq B_h + B_g$ since $\|g\|_\infty \leq B_g$ by Assumption 4 and $\|h(\mathbf{x}_0) - \mathbf{x}_0\|_2 \leq B_h$ by Assumption 3. □

The domain bound holds with a $(1-\delta)$ probability that is different from the one in Corollary 4. We now apply a union bound for them to hold jointly.

Lemma 45. *Under the assumptions of Corollary 4. Let* $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g) + 1 + \ln(2/\delta)}$ *and* $b_n = L_f^{T-1} T \left(B_g + B_h + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T\pi^2 n^2}{3\delta}} \right)$. *Then the following hold jointly with probability at least* $(1-\delta)$ *for all* $n \geq 1$ *and* $0 \leq t < T$

i) $\|f(\mathbf{x}, \mathbf{u}) - h(\mathbf{x}, \mathbf{u}) - \boldsymbol{\mu}_n(\mathbf{x}, \mathbf{u})\|_2 \leq \beta_n \|\boldsymbol{\sigma}_n(\mathbf{x}, \mathbf{u})\|_2$ *for all* $\mathbf{x} \in \mathbb{R}^p$ *and* $\mathbf{u} \in \mathbb{R}^q$

ii) $\mathbf{x}_{t,n} \in \mathbb{B}(\mathbf{x}_0, b_n)$

Proof. This follows directly from applying a union bound over Lemma 44 and Corollary 5 with a probability budget of $\delta/2$ for each. □

D.2 Regret Bound

In Lemma 39 we assumed that the kernel is bounded over $\mathbb{R}^p$. This is a restrictive assumption for modeling dynamic systems. For example, it does not even hold for linear kernels. However, we can exploit the Lipschitz continuity of the kernel to bound it over any compact domain.

Lemma 46. *Let k be a L_k-Lipschitz continuous kernel function with respect to the 2-norm with that $k(\mathbf{x}_0, \mathbf{x}_0) \leq 1$ for some $\mathbf{x}_0 \in \mathbb{R}^p$. Then, for all $\mathbf{x} \in \mathbb{B}(\mathbf{x}_0, b_n)$ with any radius b_n,*

$$k(\mathbf{x}, \mathbf{x}) \leq 1 + \sqrt{2} L_k b_n \tag{D.68}$$

Proof. This follows directly from the Lipschitz continuity of the kernel function. In particular, for all $\mathbf{x} \in \mathbb{B}(\mathbf{x}_0, b_n)$, we have

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}) &\leq k(\mathbf{x}_0, \mathbf{x}_0) + |k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}_0, \mathbf{x}_0)| && \text{(D.69)} \\
&\leq 1 + |k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}_0, \mathbf{x}_0)| && \text{(D.70)} \\
&\leq 1 + L_k \|(\mathbf{x}, \mathbf{x}) - (\mathbf{x}_0, \mathbf{x}_0)\|_2 && \text{(D.71)} \\
&\leq 1 + \sqrt{2} L_k \|\mathbf{x} - \mathbf{x}_0\|_2 && \text{(D.72)} \\
&\leq 1 + \sqrt{2} L_k b_n && \text{(D.73)}
\end{aligned}$$

□

We are now ready to bound the expected mutual information

Lemma 47. *Under the assumptions of Lemma 45 and Assumption 4, we have with probability at least $(1 - \delta)$ that*

$$\mathbb{E}_{\boldsymbol{\omega}}[\, \mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g_{\mathcal{A}_n})] \leq \gamma_{qnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}) \tag{D.74}$$

Proof. By the properties of the expectation together with Lemma 45 and that $|\mathcal{A}_n| \leq Tpn$ we have

$$\mathbb{E}_{\boldsymbol{\omega}}[\, \mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g_{\mathcal{A}_n})] \leq \max_{\mathcal{A} \subseteq \mathbb{B}(b_n, \mathbf{x}_0) \times \mathcal{U} \times \mathcal{I}_p, |\mathcal{A}| \leq Tpn} \mathrm{I}(\mathbf{y}_{\mathcal{A}}; g) := \gamma_{qnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}). \tag{D.75}$$

□

Theorem 8. *Under the assumption of Corollaries 4 and 7 and Assumptions 4, 5 and 7, let* $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g) + 1 + \ln(2/\delta)}$ *and* $b_n = L_f^{T-1} T \left(B_g + B_h + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}} \log \frac{2T\pi^2 n^2}{3\delta}} \right)$. *At each iteration, select parameters according to (5.22), then the following holds with probability at least* $(1-\delta)$ *for all* $n \geq 1$

$$R_n \leq \tilde{\mathcal{O}}\Big(\sqrt{nT^5 p L_f^T L_\sigma^T \beta_n^T \gamma_{pnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})}\Big), \tag{5.23}$$

where $\mathbb{B}(\mathbf{x}_0, b_n) = \{\mathbf{x} \in \mathbb{R}^p \,|\, \|\mathbf{x} - \mathbf{x}_0\|_2 \leq b_n\}$ *and* $\gamma_{pnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})$ *is the information capacity after* (pnT) *observations within the domain* $\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}$.

Proof. □

Proof. From Lemma 45 we know that with probability $(1-\delta)$ the confidence intervals hold and all states are in $\mathbb{B}(\mathbf{x}_0, b_n)$. From Lemmas 40 and 47 we have

$$R_n^2 \leq \frac{k_{\max} N L_J^2 T^4 p}{\log(1 + \sigma^{-2} k_{\max})} \gamma_{qnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U}) \tag{D.76}$$

where $L_J = 2L_r(1 + L_\pi)\beta_{n-1}\bar{L}_f^{T-1}$ and $k_{\max} = 1 + \sqrt{2} L_k b_n$ by Lemma 46. Plugging in we get

$$R_n^2 \leq \tilde{\mathcal{O}}\Big(NT^4 p \beta_n \bar{L}_f^T b_n \gamma_{qnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})\Big) \tag{D.77}$$

Since $b_n = \mathcal{O}\Big(T\sqrt{\log(Tn^2/\delta)}\Big) = \tilde{\mathcal{O}}(T)$ by Lemma 44 the result follows. □

D.3 Bounding the Mutual Information

Lemma 48 (Srinivas et al. (2012))**.** *For the linear kernel* $k(\mathbf{a}, \mathbf{a}') = \mathbf{a}^{\mathrm{T}}\mathbf{a}'$ *with* $\mathbf{a} \in \mathbb{R}^d$ *we have*

$$\gamma_n(\mathbb{B}(\mathbf{x}_0, b_n)) = \mathcal{O}(d \log(n)) \tag{D.78}$$

Lemma 7. *For the squared exponential kernel we have*

$$\gamma_n(\mathbb{B}(\mathbf{x}_0, b_n)) = \mathcal{O}\Big(b_n^d (\log(n))^{d+1}\Big) = \tilde{\mathcal{O}}\Big(T \log(n^2) \log(n)^{d+1}\Big) \tag{5.24}$$

Proof. The proof is the same as in (Srinivas et al., 2012). In their notation, we have $n_T = \mathcal{O}\Big(b_n^d \log(b_n^d)\Big)$ while analyzing the terms in the eigenvalue bound leads to $B_k(T^*) \sim b_n^d$. Thus the proof follows exactly along the same lines, which leads to the result. □

D.4 Bound With Lipschitz Constraint

Now, we show that if we assume that the optimistic dynamics are Lipschitz, which implies the Lipschitz continuity of the value function that is assumed by Chowdhury and A. Gopalan (2019), we obtain the same regret bounds.

Let

$$\mathcal{M} = \Big\{ f' \,|\, \|\boldsymbol{\mu}(\mathbf{x},\mathbf{u}) - f'(\mathbf{x},\mathbf{u})\| \leq \beta \|\boldsymbol{\sigma}(\mathbf{x},\mathbf{u})\| \,\forall \mathbf{x},\mathbf{u} \in \mathbb{R}^p \times \mathbb{R}^q, \\ \|f'(\mathbf{x},\mathbf{u}) - f'(\mathbf{x}',\mathbf{u}')\| \leq L_f \|(\mathbf{x},\mathbf{u}) - (\mathbf{x}',\mathbf{u}')\| \,\forall (\mathbf{x},\mathbf{u}),(\mathbf{x}',\mathbf{u}') \in \mathbb{R}^p \times \mathbb{R}^q, \Big\}$$

be the set of all Lipschitz continuous dynamics that are compatible with the uncertainty representation in Corollary 4. The following optimistic performance objective optimizes over this set,

$$\tilde{J}_n(\boldsymbol{\theta}) = \max_{f' \in \mathcal{M}} \sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \tag{D.79a}$$

$$\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} = f'(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\omega}_{t,n} \tag{D.79b}$$

$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{D.79c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}), \tag{D.79d}$$

Note that this is not tractable in the noisy case. However, in the deterministic case we can construct the following algorithm that optimized over Lipschitz-continuous dynamics.

$$\tilde{J}_n(\boldsymbol{\theta}) = \max_{\boldsymbol{\eta}_{0:T-1},\, \boldsymbol{\eta}_t \in [-1,1]^p} \sum_{t=0}^{T} r(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \tag{D.80a}$$

$$\text{s.t. } \tilde{\mathbf{x}}_{t+1,n} = h(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \\ + \beta_{n-1} \boldsymbol{\Sigma}_{n-1}^{1/2}(\tilde{\mathbf{x}}_{t,n}, \tilde{\mathbf{u}}_{t,n}) \boldsymbol{\eta}_t + \boldsymbol{\omega}_{t,n} \tag{D.80b}$$

$$\tilde{\mathbf{x}}_{0,n} = \mathbf{x}_0, \tag{D.80c}$$

$$\tilde{\mathbf{u}}_t = \pi_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t,n}), \tag{D.80d}$$

$$\|\tilde{\mathbf{x}}_{t+1,n} - \tilde{\mathbf{x}}_{t'+1,n}\| \leq L_f \|\tilde{\mathbf{x}}_{t,n} - \tilde{\mathbf{x}}_{t',n}\| \,\forall t, t' \in \{0, \ldots, T-1\} \tag{D.80e}$$

Note that the Lipschitz constraint imposes $T(T-1)/2 = \mathcal{O}(T^2)$ additional constraints. The constraint $\|\tilde{\mathbf{x}}_{t+1,n} - \tilde{\mathbf{x}}_{t'+1,n}\| \leq L_f \|\tilde{\mathbf{x}}_{t,n} - \tilde{\mathbf{x}}_{t',n}\|$ is equivalent to $\|\tilde{f}(\tilde{\mathbf{x}}_{t,n}) - \tilde{f}(\tilde{\mathbf{x}}_{t',n})\| \leq L_f \|\tilde{\mathbf{x}}_{t,n} - \tilde{\mathbf{x}}_{t',n}\|$ where, with abuse of notation, $\tilde{f}(\mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\Sigma}(\mathbf{x})\eta$ represents the optimistic dynamics.

For this (intractable) algorithm we have the following results that lead to improved regret bounds that match those in (Chowdhury and A. Gopalan, 2019) up to constant factors.

Lemma 49. *Under the assumptions of Corollary 4, let $\bar{L}_f = L_f$. Then, for any sequence of $\boldsymbol{\eta}_t \in [-1, 1]^p$, any sequence of $\boldsymbol{\omega}_t$ with $\tilde{\boldsymbol{\omega}}_t = \boldsymbol{\omega}_t$, $\boldsymbol{\theta} \in \mathcal{D}$, and $t \geq 1$ we have that*

$$\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\| \leq 2\beta_{n-1}\bar{L}_f^{T-1} \sum_{i=0}^{t-1} \|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.81}$$

Proof. Let

$$\tilde{f}(\tilde{\mathbf{x}}_{t,n}) = h(\tilde{\mathbf{x}}_t) + \boldsymbol{\mu}_{n-1}(\tilde{\mathbf{x}}_t) + \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\tilde{\mathbf{x}}_t)\boldsymbol{\eta}_t. \tag{D.82}$$

Then by design we have $\|\tilde{f}(\mathbf{x}) - \tilde{f}(\mathbf{x}')\| \leq L_f\|\mathbf{x} - \mathbf{x}'\|$.

We start by showing that, for any $t \geq 1$, we have

$$\|\mathbf{x}_{t,n} - \tilde{\mathbf{x}}_{t,n}\| \leq 2\beta_{n-1} \sum_{i=0}^{t-1} L_f^{t-1-i} \|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.83}$$

by induction.

For the base case we have $\tilde{\mathbf{x}}_0 = \mathbf{x}_0$. Consequently, at n we have

$$\begin{aligned}
\|\mathbf{x}_{1,n} - \tilde{\mathbf{x}}_{1,n}\| &= \|f(\mathbf{x}_0) + \boldsymbol{\omega}_0 - \tilde{f}(\mathbf{x}_0) - \tilde{\boldsymbol{\omega}}_0\| && \text{(D.84)}\\
&= \|f(\mathbf{x}_0) - \tilde{f}(\mathbf{x}_0)\| && \text{(D.85)}\\
&= \|h(\mathbf{x}_0) + g(\mathbf{x}_0) - h(\mathbf{x}_0) - \boldsymbol{\mu}_{n-1}(\mathbf{x}_0) - \beta_{n-1}\boldsymbol{\Sigma}_{n-1}(\mathbf{x}_0)\boldsymbol{\eta}_0\| && \text{(D.86)}\\
&\leq \|g(\mathbf{x}_0) - \boldsymbol{\mu}_{n-1}(\mathbf{x}_0)\| + \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\boldsymbol{\eta}_0\| && \text{(D.87)}\\
&\leq \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| + \beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| && \text{(D.88)}\\
&= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_0)\| && \text{(D.89)}
\end{aligned}$$

For the induction step assume that (D.83) holds at time step t. Subsequently we have at

iteration n that

$$\begin{aligned}
\|\mathbf{x}_{t+1,n} - \tilde{\mathbf{x}}_{t+1,n}\| &= \|f(\mathbf{x}_t) - \tilde{f}(\tilde{\mathbf{x}}_t)\| \\
&= \|f(\mathbf{x}_t) - \tilde{f}(\mathbf{x}_t) + \tilde{f}(\mathbf{x}_t) - \tilde{f}(\tilde{\mathbf{x}}_t)\| \\
&= \|f(\mathbf{x}_t) - \tilde{f}(\mathbf{x}_t)\| + \|\tilde{f}(\mathbf{x}_t) - \tilde{f}(\tilde{\mathbf{x}}_t)\| \\
&\leq 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + L_f\|\mathbf{x}_t - \tilde{\mathbf{x}}_t\| \\
&\leq 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + L_f 2\beta_{n-1}\sum_{i=0}^{t-1} L_f^{t-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \\
&= 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_t)\| + 2\beta_{n-1}\sum_{i=0}^{t-1} L_f^{t-1-i+1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \\
&= 2\beta_{n-1}\sum_{i=0}^{(t+1)-1} L_f^{(t+1)-1-i+1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \\
&= 2\beta_{n-1}\sum_{i=0}^{(t+1)-1} L_f^{(t+1)-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\|
\end{aligned}$$

Thus (D.83) holds. Now since $t \leq T$ we have

$$\|\mathbf{x}_{t+1,n} - \tilde{\mathbf{x}}_{t+1,n}\| \leq 2\beta_{n-1}\sum_{i=0}^{t-1} L_f^{t-1-i}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \quad \leq 2\beta_{n-1}L_f^{T-1}\sum_{i=0}^{t-1}\|\boldsymbol{\sigma}_{n-1}(\mathbf{x}_{i,n})\| \tag{D.90}$$

□

Theorem 9. *Under the assumption of Corollary 4 and Assumptions 4, 5 and 7, let* $\beta_n = B + 4\sigma\sqrt{\mathrm{I}(\mathbf{y}_{\mathcal{A}_n}; g) + 1 + \ln(2/\delta)}$ *and* $b_n = L_f^{T-1}T\left(B_g + B_h + \sqrt{2\sigma p + \frac{4\sigma}{\mathrm{e}}\log\frac{2T\pi^2 n^2}{3\delta}}\right)$. *At each iteration, select parameters according to (5.22), then the following holds with probability at least* $(1-\delta)$ *for all* $n \geq 1$

$$R_n \leq \tilde{\mathcal{O}}\left(\sqrt{nT^5 p L_f^T \beta_n \gamma_{qnT}(\mathbb{B}(\mathbf{x}_0, b_n) \times \mathcal{U})}\right) \tag{D.91}$$

Proof. Since the true dynamics are L_f-Lipschitz continuous, Lemma 34 holds with the additional constraint (D.80e) since no unrealistic dynamics are eliminated. As a consequence, this follows directly from the proof Theorem 9 by plugging in $\bar{L}_f = L_f$ into the proof from (D.83). □

D.5 Lipschitz Continuity of the Gaussian Process Variance

Lemma 50. *Let the kernel function $k(\cdot,\cdot)$ be L_k-Lipschitz continuous. Then the corresponding Gaussian process variance is L_σ-Lipschitz continuous with Lipschitz constant independent of the number of data points.*

Proof. It follows from (Kirschner and Krause, 2018, Appendix D) that we can write the Gaussian process posterior variance in terms of the weighted norm of the corresponding feature vector in the RKHS,

$$\sigma_n(\mathbf{a}) = \|k(\mathbf{a},\cdot)\|_{\mathbf{V}_n^{-1}}, \tag{D.92}$$

where $\mathbf{V}_n = \sigma^2\mathbf{M}^*\mathbf{M} + \mathbf{I}$ and $\mathbf{M}$ is a linear operator that corresponds to the infinite-dimensional feature vectors of the data points in $\mathcal{A}_n$ so that $\mathbf{M}\mathbf{M}^* = \mathbf{K}_{\mathcal{A}_n}$. Now we have that the minimum eigenvalue of $\mathbf{V}_n$ is larger or equal than one, so that the maximum eigenvalue of $\mathbf{V}_n^{-1}$ is less or equal than one. Thus,

$$\begin{aligned}
|\sigma_n^2(\mathbf{a}) - \sigma_n^2(\mathbf{a}')| &= \left| \|k(\mathbf{a},\cdot)\|^2_{\mathbf{V}_n^{-1}} - \|k(\mathbf{a}',\cdot)\|^2_{\mathbf{V}_n^{-1}} \right| && \text{(D.93)}\\
&\leq \|k(\mathbf{a},\cdot) - k(\mathbf{a}',\cdot)\|^2_{\mathbf{V}_n^{-1}}, && \text{(D.94)}\\
&\leq \|k(\mathbf{a},\cdot) - k(\mathbf{a}',\cdot)\|^2_{k}, && \text{(D.95)}\\
&= \langle k(\mathbf{a},\cdot) - k(\mathbf{a}',\cdot), k(\mathbf{a},\cdot) - k(\mathbf{a}',\cdot)\rangle_k, && \text{(D.96)}\\
&= k(\mathbf{a},\mathbf{a}) - k(\mathbf{a},\mathbf{a}') - k(\mathbf{a}',\mathbf{a}) + k(\mathbf{a}',\mathbf{a}'), && \text{(D.97)}\\
&= k(\mathbf{a},\mathbf{a}) + k(\mathbf{a}',\mathbf{a}') - 2k(\mathbf{a},\mathbf{a}'), && \text{(D.98)}
\end{aligned}$$

where $<\cdot,\cdot>_k$ is the RKHS norm as in Section 2.6.2. Since the kernel is Lipschitz continuous according to Assumption 4, the result follows. Note that this means that the GP standard deviation is Lipschitz continuous with respect to the kernel metric in (2.33). □

D.6 Practical Implementation

We now show that (5.26) achieves the same regret bound as (5.21). The only additional requirements for the previous proofs to hold, is to show that (5.26) yields an optimistic performance estimate; that is, that the true dynamics are feasible with high probability.

Lemma 51. *Under the assumptions of Theorem 8, with Assumption 4, and with $\sigma > 0$, we have with probability at least $(1-\delta)$ that there exists a Lipschitz-continuous function with $\|\eta(\cdot)\|_\infty = 1$ such that $f(\mathbf{x}) - \boldsymbol{\mu}_n(\mathbf{x}) = \beta_n \boldsymbol{\Sigma}_n(\mathbf{x})\eta(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^p$.*

Proof. By contradiction. Let $\eta(\cdot)$ be a function that is not Lipschitz continuous such that $f(\mathbf{x}) - \boldsymbol{\mu}(\mathbf{x}) = \beta\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x})$. Moreover, let L_μ denote the Lipschitz constant of the mean function, which exists according to (Lederer et al., 2019). Now, due to Assumption 4 we know that $\boldsymbol{\sigma}_n(\mathbf{x})$ is bounded element-wise by some constant over any compact domain, so that $\|\boldsymbol{\Sigma}_n(\mathbf{x})\| \leq c_\sigma$ for all $\mathbf{x} \in \mathbb{R}^p$. Now with $\sigma > 0$ we have $\boldsymbol{\sigma}_n(\mathbf{x}) > 0$ elementwise for all $\mathbf{x} \in \mathbb{R}^p$. As a consequence, $\boldsymbol{\Sigma}^{-1}(\mathbf{x})$ is $L_{\sigma-}$-Lipschitz continuous and its norm is bounded from below by some constant $c_{\sigma-}$. Thus, we have

$$
\begin{aligned}
\|\eta(\mathbf{x}) - \eta(\mathbf{x}')\|_2 &= \|\boldsymbol{\Sigma}^{-1}(\mathbf{x})\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x}) - \boldsymbol{\Sigma}^{-1}(\mathbf{x})\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x})\|_2 \\
&\leq \|\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x}) - \boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x})\|_2 c_{\sigma-} + \|\boldsymbol{\Sigma}^{-1}(\mathbf{x}) - \boldsymbol{\Sigma}^{-1}(\mathbf{x}')\|_2 \max_{\mathbf{x}\in\mathbb{R}^p} \|\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x})\|_2 \\
&\leq \|\boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x}) - \boldsymbol{\Sigma}(\mathbf{x})\eta(\mathbf{x})\|_2 c_{\sigma-} + L_{\sigma-}\|\mathbf{x} - \mathbf{x}'\|_2 \max_{\mathbf{x}\in\mathbb{R}^p} \|\boldsymbol{\Sigma}(\mathbf{x})\|_2 \|\eta(\mathbf{x})\|_2 \\
&= \frac{1}{\beta}\|f(\mathbf{x}) - \boldsymbol{\mu}(\mathbf{x}) - (f(\mathbf{x}') - \mu(\mathbf{x}'))\|_2 c_{\sigma-} + L_{\sigma-}\|\mathbf{x} - \mathbf{x}'\|_2 \max_{\mathbf{x}\in\mathbb{R}^p} \|\boldsymbol{\Sigma}(\mathbf{x})\|_2 \|\eta(\mathbf{x})\|_2 \\
&\leq \frac{1}{\beta}(L_f + L_\mu)\|\mathbf{x} - \mathbf{x}'\|_2 c_{\sigma-} + L_{\sigma-}\|\mathbf{x} - \mathbf{x}'\|_2 \max_{\mathbf{x}\in\mathbb{R}^p} \|\boldsymbol{\Sigma}(\mathbf{x})\|_2 \|\eta(\mathbf{x})\|_2 \\
&\leq \frac{1}{\beta}(L_f + L_\mu)\|\mathbf{x} - \mathbf{x}'\|_2 c_{\sigma-} + L_{\sigma-}\|\mathbf{x} - \mathbf{x}'\|_2 c_\sigma p
\end{aligned}
$$

Since $\beta_n \geq 0$ we have that $\eta(\mathbf{x})$ is Lipschitz continuous, which is a contradiction. □

Thus, it is sufficient to optimize over Lipschitz continuous functions in order to obtain the same regret bounds as in the optimistic case. Note that the Lipschitz constant increases as the predictive variance decreases.

Bibliography

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (14, 2016). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: arXiv: 1603.04467 [cs] (cit. on p. 98).

Abbasi-Yadkori, Yasin (2012). "Online learning of linearly parameterized control problems". PhD Thesis (cit. on p. 27).

Abbasi-Yadkori, Yasin and Csaba Szepesvári (2011). "Regret bounds for the adaptive control of linear quadratic systems". In: *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 1–26 (cit. on p. 33).

Abdelrahman, Hany, Felix Berkenkamp, and Andreas Krause (2016). "Bayesian optimization for maximum power point tracking in photovoltaic power plants". In: *2016 European Control Conference (ECC)*, pp. 2078–2083 (cit. on p. 70).

Achiam, Joshua, David Held, Aviv Tamar, and Pieter Abbeel (2017). "Constrained policy optimization". In: *Proc. of the International Conference on Machine Learning (ICML)*. arXiv: 1705.10528 (cit. on p. 36).

Afanas'ev, V. N., V. B. Kolmanovskii, and V. R. Nosov (1996). "Stability of Stochastic Systems". In: *Mathematical Theory of Control Systems Design*. Ed. by V. N. Afanas'ev, V. B. Kolmanovskii, and V. R. Nosov. Mathematics and Its Applications. Dordrecht: Springer Netherlands, pp. 73–103 (cit. on p. 19).

Akametalu, Anayo K., Shromona Ghosh, Jaime F. Fisac, and Claire J. Tomlin (3, 2018). "A Minimum Discounted Reward Hamilton-Jacobi Formulation for Computing Reachable Sets". In: arXiv: 1809.00706 [cs, math] (cit. on p. 20).

Alshiekh, Mohammed, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu (2018). "Safe reinforcement learning via shielding". In: *Thirty-Second AAAI Conference on Artificial Intelligence* (cit. on p. 40).

Altman, Eitan (30, 1999). *Constrained Markov Decision Processes*. CRC Press. 260 pp. (cit. on p. 19).

Álvarez, Mauricio A., Lorenzo Rosasco, and Neil D. Lawrence (2012). "Kernels for Vector-Valued Functions: A Review". In: *Foundations and Trends in Machine Learning* 4.3, pp. 195–266 (cit. on p. 45).

Ames, Aaron D., Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada (2019). "Control Barrier Functions: Theory and Applications". In: *arXiv preprint arXiv:1903.11199* (cit. on p. 18).

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané (21, 2016). "Concrete problems in AI safety". In: arXiv: 1606.06565 [cs] (cit. on p. 1).

Andersson, Joel (2013). "A general-purpose software framework for dynamic optimization". PhD Thesis. Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (cit. on p. 105).

Asselborn, L., D. Groß, and O. Stursberg (1, 2013). "Control of Uncertain Nonlinear Systems using Ellipsoidal Reachability Calculus". In: *IFAC Proceedings Volumes*. 9th IFAC Symposium on Nonlinear Control Systems 46.23, pp. 50–55 (cit. on p. 79).

Aswani, Anil, Humberto Gonzalez, S. Shankar Sastry, and Claire Tomlin (2013). "Provably safe and robust learning-based model predictive control". In: *Automatica* 49.5, pp. 1216–1226 (cit. on p. 40).

Atkeson, C. G. and J. C. Santamaria (1997). "A comparison of direct and model-based reinforcement learning". In: *Proceedings of International Conference on Robotics and Automation*. Proceedings of International Conference on Robotics and Automation. Vol. 4, 3557–3564 vol.4 (cit. on p. 21).

Auer, Peter (2002). "Using Confidence Bounds for Exploitation-Exploration Trade-offs". In: *Journal of Machine Learning Research* 3 (Nov), pp. 397–422 (cit. on p. 31).

Bäuerle, Nicole and Jonathan Ott (1, 2011). "Markov Decision Processes with Average-Value-at-Risk criteria". In: *Mathematical Methods of Operations Research* 74.3, pp. 361–379 (cit. on p. 19).

Bechtle, Sarah, Akshara Rai, Yixin Lin, Ludovic Righetti, and Franziska Meier (14, 2019). "Curious iLQR: Resolving Uncertainty in Model-based RL". In: arXiv: 1904.06786 [cs] (cit. on p. 34).

Bemporad, Alberto and Manfred Morari (1999). "Robust model predictive control: A survey". In: *Robustness in identification and control*. Ed. by A. Garulli and A. Tesi. Lecture Notes in Control and Information Sciences. Springer London, pp. 207–226 (cit. on p. 39).

Bennett, S. (1996). "A brief history of automatic control". In: *IEEE Control Systems Magazine* 16.3, pp. 17–25 (cit. on p. 5).

Berkenkamp, Felix, Andreas Krause, and Angela P. Schoellig (2016). "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics". In: *arXiv:1602.04450 [cs.RO]* (cit. on pp. 4, 43, 129).

Berkenkamp, Felix, Riccardo Moriconi, Angela P. Schoellig, and Andreas Krause (2016). "Safe learning of regions of attraction in nonlinear systems with Gaussian processes". In: *Proc. of the Conference on Decision and Control (CDC)*, pp. 4661–4666 (cit. on pp. 4, 71, 77, 87, 89, 139, 145, 157).

Berkenkamp, Felix and Angela P. Schoellig (2015). "Safe and robust learning control with Gaussian processes". In: *Proc. of the European Control Conference (ECC)*, pp. 2501–2506 (cit. on p. 40).

Berkenkamp, Felix, Angela P. Schoellig, and Andreas Krause (2016). "Safe controller optimization for quadrotors with Gaussian processes". In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 493–496 (cit. on pp. 3, 43, 56, 57, 129).

Berkenkamp, Felix, Angela P. Schoellig, and Andreas Krause (2019). "No-Regret Bayesian optimization with unknown hyperparameters". In: *Journal of Machine Learning Research (JMLR)* 20.50, pp. 1–24 (cit. on pp. 4, 69).

Berkenkamp, Felix, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause (2017). "Safe model-based reinforcement learning with stability guarantees". In: *Proc. of Neural Information Processing Systems (NeurIPS)*, pp. 908–918 (cit. on pp. 4, 71, 89, 139, 145, 157).

Bertsekas, Dimitri P., Dimitri P. Bertsekas, Dimitri P. Bertsekas, and Dimitri P. Bertsekas (1995). *Dynamic programming and optimal control*. Vol. 1. 2. Athena scientific Belmont, MA (cit. on p. 5).

Bıyık, Erdem, Jonathan Margoliash, Shahrouz Ryan Alimo, and Dorsa Sadigh (1, 2019). "Efficient and Safe Exploration in Deterministic Markov Decision Processes with Unknown Transition Models". In: arXiv: 1904.01068 [cs] (cit. on p. 39).

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* 112.518, pp. 859–877 (cit. on p. 23).

Bobiti, Ruxandra and Mircea Lazar (2016). "A sampling approach to finding Lyapunov functions for nonlinear discrete-time systems". In: *Proc. of the European Control Conference (ECC)*. 2016 European Control Conference (ECC), pp. 561–566 (cit. on p. 98).

Boedecker, Joschka, Jost Tobias Springenberg, Jan Wulfing, and Martin Riedmiller (2014). "Approximate real-time optimal control based on sparse Gaussian process models". In: *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). Orlando, FL, USA: IEEE, pp. 1–8 (cit. on p. 104).

Bof, Nicoletta, Ruggero Carli, and Luca Schenato (14, 2018). "Lyapunov Theory for Discrete Time Systems". In: arXiv: 1809.05289 [math] (cit. on pp. 15, 16).

Boggs, Paul T. and Jon W. Tolle (1995). "Sequential quadratic programming". In: *Acta numerica* 4, pp. 1–51 (cit. on p. 13).

Borrelli, Francesco, Alberto Bemporad, and Manfred Morari (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press (cit. on p. 13).

Brafman, Ronen I. and Moshe Tennenholtz (2003). "R-max - a General Polynomial Time Algorithm for Near-optimal Reinforcement Learning". In: *J. Mach. Learn. Res.* 3, pp. 213–231 (cit. on p. 33).

Breiman, Leo and Adele Cutler (1, 1993). "A deterministic algorithm for global optimization". In: *Mathematical Programming* 58.1, pp. 179–199 (cit. on p. 82).

Bull, Adam D. (2011). "Convergence rates of efficient global optimization algorithms". In: *Journal of Machine Learning Research* 12 (Oct), pp. 2879–2904 (cit. on pp. 30, 60).

Calandra, Roberto, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth (2014). "Bayesian gait optimization for bipedal locomotion". In: *Learning and Intelligent Optimization*. Springer, pp. 274–290 (cit. on pp. 30, 43).

Calandra, Roberto, André Seyfarth, Jan Peters, and Marc Peter Deisenroth (2014). "An experimental comparison of Bayesian optimization for bipedal locomotion". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1951–1958 (cit. on p. 30).

Cao, Gang, Edmund M.-K. Lai, and Fakhrul Alam (1, 2017). "Gaussian Process Model Predictive Control of an Unmanned Quadrotor". In: *Journal of Intelligent & Robotic Systems* 88.1, pp. 147–162 (cit. on p. 34).

Chatzilygeroudis, K., R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J. Mouret (2017). "Black-box data-efficient policy search for robotics". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 51–58 (cit. on p. 34).

Chen, S., K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari (2018). "Approximating Explicit Model Predictive Control Using Constrained Neural Networks". In: *2018 Annual American Control Conference (ACC)*. 2018 Annual American Control Conference (ACC), pp. 1520–1527 (cit. on p. 13).

Chentanez, Nuttapong, Andrew G. Barto, and Satinder P. Singh (2005). "Intrinsically motivated reinforcement learning". In: *Advances in neural information processing systems*, pp. 1281–1288 (cit. on p. 32).

Chow, Yinlam, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh (28, 2019). "Lyapunov-based Safe Policy Optimization for Continuous Control". In: arXiv: 1901.10031 [cs, stat] (cit. on p. 36).

Chowdhury, Sayak Ray and Aditya Gopalan (2017). "On kernelized multi-armed bandits". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 844–853 (cit. on pp. 27, 31, 47, 130, 151).

Chowdhury, Sayak Ray and Aditya Gopalan (2019). "Online Learning in Kernelized Markov Decision Processes". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. The 22nd International Conference on Artificial Intelligence and Statistics, pp. 3197–3205 (cit. on pp. 34, 113, 116, 120, 168, 169).

Christmann, Andreas and Ingo Steinwart (2008). *Support Vector Machines*. Information Science and Statistics. New York, NY: Springer (cit. on pp. 27, 73).

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach,

H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 4754–4765 (cit. on pp. 34, 114).

Cohen, Andrew, Lei Yu, and Robert Wright (22, 2018). "Diverse Exploration for Fast and Safe Policy Improvement". In: arXiv: 1802.08331 [cs] (cit. on p. 37).

Dalal, Gal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa (26, 2018). "Safe Exploration in Continuous Action Spaces". In: arXiv: 1801.08757 [cs] (cit. on p. 41).

Davidor, Yuval (1991). *Genetic algorithms and robotics: a heuristic strategy for optimization*. World Scientific. 192 pp. (cit. on p. 21).

Davies, Scott (1996). "Multidimensional triangulation and interpolation for reinforcement learning". In: *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pp. 1005–1011 (cit. on p. 99).

Dayan, Peter (1, 1992). "The convergence of TD(λ) for general λ". In: *Machine Learning* 8.3, pp. 341–362 (cit. on p. 11).

Dean, Sarah, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu (2018). "Regret Bounds for Robust Adaptive Control of the Linear Quadratic Regulator". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 4188–4197 (cit. on p. 33).

Dean, Sarah, Stephen Tu, Nikolai Matni, and Benjamin Recht (26, 2018). "Safely Learning to Control the Constrained Linear Quadratic Regulator". In: arXiv: 1809.10121 [cs, math, stat] (cit. on p. 41).

Deisenroth, Marc, Dieter Fox, and Carl Rasmussen (2014). "Gaussian processes for data-efficient learning in robotics and control". In: *Transactions on Pattern Analysis and Machine Intelligence* 37.2, pp. 1–1 (cit. on p. 116).

Deisenroth, Marc and Carl E. Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *Proc. of the International Conference on Machine Learning (ICML)*, pp. 465–472 (cit. on pp. 34, 114, 117).

Deisenroth, Marc, Carl Rasmussen, and Jan Peters (2009). "Gaussian process dynamic programming". In: *Neurocomputing* 72.7-9, pp. 1508–1524 (cit. on p. 33).

Der Kiureghian, Armen and Ove Ditlevsen (2009). "Aleatory or epistemic? Does it matter?" In: *Structural Safety* 31.2, pp. 105–112 (cit. on p. 22).

Ding, J., J. H. Gillula, H. Huang, M. P. Vitus, W. Zhang, and C. J. Tomlin (2011). "Hybrid Systems in Robotics". In: *IEEE Robotics Automation Magazine* 18.3, pp. 33–43 (cit. on p. 20).

Duffie, Darrell and Jun Pan (1997). "An overview of value at risk". In: *Journal of derivatives* 4.3, pp. 7–49 (cit. on p. 19).

Duivenvoorden, Rikky R.P.R., Felix Berkenkamp, Nicolas Carion, Andreas Krause, and Angela P. Schoellig (2017). "Constrained Bayesian optimization with particle swarms for adaptive controller tuning". In: *Proc. of the IFAC (International Federation of Automatic Control) World Congress*, pp. 12306–12313 (cit. on p. 70).

Engel, Yaakov, Shie Mannor, and Ron Meir (2005). "Reinforcement learning with Gaussian processes". In: *Proceedings of the 22nd international conference on Machine learning*. ACM, pp. 201–208 (cit. on p. 33).

Even-Dar, Eyal and Yishay Mansour (2002). "Convergence of optimistic and incremental Q-learning". In: *Advances in neural information processing systems*, pp. 1499–1506 (cit. on p. 33).

Faradonbeh, Mohamad Kazem Shirani, Ambuj Tewari, and George Michailidis (2017). "Finite time analysis of optimal adaptive policies for linear-quadratic systems". In: *arXiv preprint arXiv:1711.07230* (cit. on p. 33).

Filippova, Tatiana F. (1, 2017). "Ellipsoidal Estimates of Reachable Sets for Control Systems with Nonlinear Terms **The research was supported by Russian Science Foundation (RSF Project No.16-11-10146)". In: *IFAC-PapersOnLine*. 20th IFAC World Congress 50.1, pp. 15355–15360 (cit. on p. 79).

Fisac, J. F., A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin (2018). "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems". In: *IEEE Transactions on Automatic Control*, pp. 1–1 (cit. on p. 40).

Gal, Yarin (2016). "Uncertainty in deep learning". PhD Thesis. PhD thesis, University of Cambridge (cit. on pp. 22, 28).

Gal, Yarin, Rowan McAllister, and Carl Edward Rasmussen (2016). "Improving PILCO with Bayesian neural network dynamics models". In: *Data-Efficient Machine Learning workshop, ICML*. Vol. 4 (cit. on p. 34).

Garcia, J. and F. Fernandez (2012). "Safe exploration of state and action spaces in reinforcement learning". In: *Journal of Artificial Intelligence Research*, pp. 515–564 (cit. on p. 40).

García, Javier and Fernando Fernández (2015). "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research (JMLR)* 16, pp. 1437–1480 (cit. on pp. 2, 36).

Gelbart, Michael A., Jasper Snoek, and Ryan P. Adams (2014). "Bayesian optimization with unknown constraints". In: *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 250–259 (cit. on p. 37).

Ghosh, S., F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor (2018). "Verifying Controllers Against Adversarial Examples with Bayesian Optimization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7306–7313 (cit. on p. 69).

Giesl, Peter and Sigurdur Hafstein (2015). "Review on computational methods for Lyapunov functions". In: *Discrete and Continuous Dynamical Systems, Series B* 20.8, pp. 2291–2337 (cit. on p. 17).

Girard, Agathe, Carl Edward Rasmussen, J. Quinonero-Candela, R. Murray-Smith, J. Quinonero-Candela, O. Winther, J. Quinonero-Candela, A. Girard, J. Larsen, and C. Rasmussen (2002). "Multiple-step ahead prediction for non linear dynamic systems–a gaussian process treatment with propagation of the uncertainty". In: *Proc. of Neural Information Processing Systems (NIPS)*. Vol. 15, pp. 529–536 (cit. on p. 34).

Golub, Gene H. and Charles F. Van Loan (27, 2012). *Matrix Computations*. JHU Press. 738 pp. (cit. on p. 104).

Gopalan, Aditya and Shie Mannor (2015). "Thompson sampling for learning parameterized Markov decision processes". In: *Conference on Learning Theory*, pp. 861–898 (cit. on p. 33).

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (17, 2017). "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning, pp. 1321–1330 (cit. on p. 28).

Hans, Alexander, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft (2008). "Safe exploration for reinforcement learning." In: *Proc. of the European Symposium on Artificial Neural Networks (ESANN)*. European Symposium on Artificial Neural Networks (ESANN), pp. 143–148 (cit. on p. 40).

Hessem, D. H. van and O. H. Bosgra (2002). "Closed-loop stochastic dynamic process optimization under input and state constraints". In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301). Vol. 3, 2023–2028 vol.3 (cit. on p. 86).

Hewing, Lukas, Juraj Kabzan, and Melanie N. Zeilinger (30, 2017). "Cautious Model Predictive Control using Gaussian Process Regression". In: arXiv: 1705.10702 [cs, math] (cit. on p. 40).

Howard, Ronald A. and James E. Matheson (1972). "Risk-sensitive Markov decision processes". In: *Management science* 18.7, pp. 356–369 (cit. on p. 19).

Ibrahimi, Morteza, Adel Javanmard, and Benjamin V. Roy (2012). "Efficient Reinforcement Learning for High Dimensional Linear Quadratic Systems". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., pp. 2636–2644 (cit. on p. 34).

Jacobson, David H. and David Q. Mayne (1970). "Differential dynamic programming". In: (cit. on p. 13).

Jain, A., T. Nghiem, M. Morari, and R. Mangharam (2018). "Learning and Control Using Gaussian Processes". In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), pp. 140–149 (cit. on p. 106).

Jaksch, Thomas, Ronald Ortner, and Peter Auer (2010). "Near-optimal regret bounds for reinforcement learning". In: *Journal of Machine Learning Research* 11 (Apr), pp. 1563–1600 (cit. on p. 33).

Jin, Ming and Javad Lavaei (26, 2018). "Stability-certified reinforcement learning: A control-theoretic perspective". In: arXiv: 1810.11505 [cs] (cit. on p. 40).

Johnson, Roger W. (2001). "An Introduction to the Bootstrap". In: *Teaching Statistics* 23.2, pp. 49–54 (cit. on p. 28).

Jones, Donald R. (1, 2001). "A taxonomy of global optimization methods based on response surfaces". In: *Journal of Global Optimization* 21.4, pp. 345–383 (cit. on p. 29).

Kamthe, Sanket and Marc Deisenroth (31, 2018). "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". In: *International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics, pp. 1701–1710 (cit. on pp. 34, 104, 114).

Kanagawa, Motonobu, Philipp Hennig, Dino Sejdinovic, and Bharath K. Sriperumbudur (6, 2018). "Gaussian processes and kernel methods: a review on connections and equivalences". In: arXiv: 1807.02582 [stat.ML] (cit. on p. 27).

Kearns, Michael and Satinder Singh (1, 2002). "Near-Optimal Reinforcement Learning in Polynomial Time". In: *Machine Learning* 49.2, pp. 209–232 (cit. on p. 33).

Khalil, Hassan K. and J. W. Grizzle (1996). *Nonlinear systems*. Vol. 3. Prentice Hall (cit. on pp. 16, 19, 90).

Khasminskii, Rafail (2012). *Stochastic Stability of Differential Equations*. 2nd ed. Stochastic Modelling and Applied Probability. Berlin Heidelberg: Springer-Verlag (cit. on p. 75).

Kingma, Diederik P. and Max Welling (20, 2013). "Auto-Encoding Variational Bayes". In: arXiv: 1312.6114 [cs, stat] (cit. on p. 12).

Kirschner, Johannes and Andreas Krause (2018). "Information directed sampling and bandits with heteroscedastic noise". In: *Proceedings of the 31st Conference On Learning Theory*. Vol. 75. Proceedings of Machine Learning Research. PMLR, pp. 358–384. arXiv: 1801.09667 (cit. on pp. 31, 171).

Kober, Jens and Jan Peters (2014). "Reinforcement learning in robotics: a survey". In: (cit. on pp. 1, 21).

Koller, Torsten, Felix Berkenkamp, Matteo Turchetta, Joschka Boedecker, and Andreas Krause (27, 2019). "Learning-based model predictive control for safe exploration and reinforcement learning". In: arXiv: 1906.12189 [cs, eess] (cit. on pp. 4, 71, 89, 139, 145, 157).

Koller, Torsten, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause (2018). "Learning-Based model predictive control for safe exploration". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018 IEEE Conference on Decision and Control (CDC), pp. 6059–6066 (cit. on pp. 71, 89, 139, 145, 157).

Krause, Andreas and Cheng S. Ong (2011). "Contextual Gaussian process bandit optimization". In: *Proc. of Neural Information Processing Systems (NIPS)*, pp. 2447–2455 (cit. on pp. 31, 32, 54, 121).

Krause, Andreas, Ajit Singh, and Carlos Guestrin (2008). "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies". In: *Journal of Machine Learning Research* 9 (Feb), pp. 235–284 (cit. on p. 106).

Kurzhanskii, A. B. and István Vályi (1997). *Ellipsoidal Calculus for Estimation and Control*. Nelson Thornes. 354 pp. (cit. on p. 79).

Kwakernaak, Huibert and Raphael Sivan (1972). *Linear optimal control systems*. Vol. 1. Wiley-interscience New York (cit. on pp. 14, 105).

Lai, Tze Leung and Herbert Robbins (1985). "Asymptotically efficient adaptive allocation rules". In: *Advances in applied mathematics* 6.1, pp. 4–22 (cit. on p. 30).

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in*

Neural Information Processing Systems 30. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 6402–6413 (cit. on p. 28).

Langford, John and Tong Zhang (2007). "The epoch-greedy algorithm for contextual multi-armed bandits". In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Citeseer, pp. 817–824 (cit. on p. 31).

Lattimore, Tor and Csaba Szepesvári (2018). "Bandit algorithms". In: *preprint* (cit. on pp. 28, 29).

Lederer, Armin, Jonas Umlauft, and Sandra Hirche (4, 2019). "Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control". In: arXiv: 1906.01376 [cs, stat] (cit. on pp. 119, 172).

Lew, Thomas, Apoorva Sharma, James Harrison, and Marco Pavone (28, 2019). "Safe Learning and Control using Meta-Learning". In: Robust autonomy workshop at Robotics: Science and Systems (cit. on p. 125).

Li, Huijuan and Lars Grüne (15, 2016). "Computation of local ISS Lyapunov functions for discrete-time systems via linear programming". In: *Journal of Mathematical Analysis and Applications* 438.2, pp. 701–719 (cit. on p. 17).

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (9, 2015). "Continuous control with deep reinforcement learning". In: arXiv: 1509.02971 [cs, stat] (cit. on pp. 12, 21).

Liu, Anqi, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue (13, 2019). "Robust Regression for Safe Exploration in Control". In: arXiv: 1906.05819 [cs, eess, stat] (cit. on p. 28).

Lizotte, Daniel J., Tao Wang, Michael H. Bowling, and Dale Schuurmans (2007). "Automatic gait optimization with Gaussian process regression." In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*. Vol. 7, pp. 944–949 (cit. on p. 30).

Lu, Tyler, Martin Zinkevich, Craig Boutilier, Binz Roy, and Dale Schuurmans (29, 2017). "Safe Exploration for Identifying Linear Systems via Robust Optimization". In: arXiv: 1711.11165 [cs] (cit. on p. 41).

Luce, Robert Duncan and Howard Raiffa (1958). *Games and decisions: Introduction and critical survey*. Wiley New York (cit. on p. 19).

Lupashin, Sergei, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea (2014). "A platform for aerial robotics research and demonstration: The Flying Machine Arena". In: *Mechatronics* 24.1, pp. 41–54 (cit. on p. 59).

Lyapunov, Aleksandr Mikhailovich (1992). "The general problem of the stability of motion". In: *International journal of control* 55.3, pp. 531–534 (cit. on p. 16).

Maler, Oded and Dejan Nickovic (2004). "Monitoring Temporal Properties of Continuous Signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Ed. by Yassine Lakhnech and Sergio Yovine. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 152–166 (cit. on p. 15).

Malik, Ali, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon (24, 2019). "Calibrated Model-Based Deep Reinforcement Learning". In: *International Conference on Machine Learning*. International Conference on Machine Learning, pp. 4314–4323 (cit. on p. 34).

Marco, A., F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe (2017). "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization". In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1557–1563 (cit. on pp. 30, 69).

Marcus, S. I., E. Fernández-Gaucherand, D. Hernández-Hernández, S. Coraluppi, and P. Fard (1997). "Risk Sensitive Markov Decision Processes". In: *Systems and Control in the Twenty-First Century*. Ed. by Christopher I. Byrnes, Biswa N. Datta, Clyde F. Martin, and David S. Gilliam. Systems & Control: Foundations & Applications. Birkhäuser Boston, pp. 263–279 (cit. on p. 19).

Matthews, Alexander G. de G., Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman (2017). "GPflow: a Gaussian process library using TensorFlow". In: *Journal of Machine Learning Research* 18.40, pp. 1–6 (cit. on p. 98).

Melchior, Silvan, Felix Berkenkamp, Sebastian Curi, and Andreas Krause (2019). *Structured Variational Inference in Unstable Gaussian Process State Space Models*. (under review) (cit. on p. 87).

Mitchell, I. M., A. M. Bayen, and C. J. Tomlin (2005). "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on Automatic Control* 50.7, pp. 947–957 (cit. on p. 20).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski,

Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis (26, 2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533 (cit. on p. 1).

Mockus, Jonas (6, 2012). *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media. 267 pp. (cit. on p. 29).

Moldovan, Teodor Mihai and Pieter Abbeel (2012). "Safe exploration in Markov decision processes". In: *Proc. of the International Conference on Machine Learning (ICML)*, pp. 1711–1718 (cit. on p. 39).

Moldovan, Teodor Mihai, Sergey Levine, Michael I. Jordan, and Pieter Abbeel (2015). "Optimism-driven exploration for nonlinear systems". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, pp. 3239–3246 (cit. on pp. 34, 116).

Morari, Manfred and Jay H. Lee (1, 1999). "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4–5, pp. 667–682 (cit. on p. 12).

Munos, Rémi, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare (8, 2016). "Safe and Efficient Off-Policy Reinforcement Learning". In: arXiv: 1606.02647 [cs, stat] (cit. on p. 21).

Mutny, Mojmir and Andreas Krause (2018). "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features". In: *Advances in Neural Information Processing Systems*, pp. 9005–9016 (cit. on p. 31).

Nikolov, Nikolay, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause (2019). "Information-Directed Exploration for Deep Reinforcement Learning". In: *Proc. International Conference on Learning Representations (ICLR)* (cit. on p. 33).

Osband, Ian, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy (2016). "Deep exploration via bootstrapped DQN". In: *Advances in neural information processing systems*, pp. 4026–4034 (cit. on p. 33).

Osband, Ian, Dan Russo, and Benjamin Van Roy (2013). "(More) Efficient Reinforcement Learning via Posterior Sampling". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., pp. 3003–3011 (cit. on p. 33).

Osband, Ian and Benjamin Van Roy (2014). "Model-based Reinforcement Learning and the Eluder Dimension". In: *Advances in Neural Information Processing Systems 27*. Ed. by

Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 1466–1474 (cit. on p. 34).

Osband, Ian and Benjamin Van Roy (9, 2016). "Posterior Sampling for Reinforcement Learning Without Episodes". In: arXiv: 1608.02731 [cs, stat] (cit. on p. 34).

Osband, Ian, Benjamin Van Roy, Daniel Russo, and Zheng Wen (2017). "Deep exploration via randomized value functions". In: *arXiv preprint arXiv:1703.07608* (cit. on p. 33).

Osband, Ian, Benjamin Van Roy, and Zheng Wen (4, 2014). "Generalization and Exploration via Randomized Value Functions". In: arXiv: 1402.0635 [cs, stat] (cit. on p. 33).

Ostafew, Chris J., Angela P. Schoellig, and Timothy D. Barfoot (2016). "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking". In: *The International Journal of Robotics Research (IJRR)* 35.13, pp. 1547–1536 (cit. on pp. 40, 60, 114).

Pathak, Deepak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17 (cit. on p. 32).

Pecka, Martin and Tomas Svoboda (2014). "Safe exploration techniques for reinforcement learning – an overview". In: *Modelling and Simulation for Autonomous Systems*. Springer, pp. 357–375 (cit. on p. 36).

Peters, Jan and Stefan Schaal (2006). "Policy gradient methods for robotics". In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2219–2225 (cit. on p. 21).

Polymenakos, Kyriakos, Alessandro Abate, and Stephen Roberts (2019). "Safe Policy Search Using Gaussian Process Models". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada). AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1565–1573 (cit. on p. 41).

Powell, Warren B. (5, 2007). *Approximate dynamic programming: solving the curses of dimensionality*. John Wiley & Sons. 488 pp. (cit. on p. 97).

Powell, Warren B. (1, 2009). "What you should know about approximate dynamic programming". In: *Naval Research Logistics (NRL)* 56.3, pp. 239–249 (cit. on p. 11).

Rasmussen, Carl Edward and Christopher K.I Williams (2006). *Gaussian processes for machine learning*. Cambridge MA: MIT Press (cit. on pp. 23, 24, 59).

Rawlings, James Blake and David Q. Mayne (2009). *Model predictive control: Theory and design*. Nob Hill Pub. Madison, Wisconsin (cit. on pp. 19, 85, 101).

Richards, Arthur and Jonathan P. How (2006). "Robust variable horizon model predictive control for vehicle maneuvering". In: *International Journal of Robust and Nonlinear Control* 16.7, pp. 333–351 (cit. on p. 108).

Richards, Spencer M., Felix Berkenkamp, and Andreas Krause (23, 2018). "The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems". In: Conference on Robot Learning (CoRL). PMLR, pp. 466–476 (cit. on p. 126).

Rockafellar, R. Tyrrell and Stanislav Uryasev (1, 2002). "Conditional value-at-risk for general loss distributions". In: *Journal of Banking & Finance* 26.7, pp. 1443–1471 (cit. on p. 19).

Rosolia, U. and F. Borrelli (2018). "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework". In: *IEEE Transactions on Automatic Control* 63.7, pp. 1883–1896 (cit. on p. 13).

Rosolia, Ugo and Francesco Borrelli (12, 2019). "Sample-Based Learning Model Predictive Control for Linear Uncertain Systems". In: arXiv: 1904.06432 [cs] (cit. on p. 40).

Russo, Daniel J., Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen (12, 2018). "A Tutorial on Thompson Sampling". In: *Foundations and Trends® in Machine Learning* 11.1, pp. 1–96 (cit. on p. 31).

Russo, Daniel and Benjamin Van Roy (2014). "Learning to optimize via information-directed sampling". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., pp. 1583–1591 (cit. on p. 31).

Ruszczyński, Andrzej (1, 2010). "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical Programming* 125.2, pp. 235–261 (cit. on p. 19).

Sadigh, Dorsa and Ashish Kapoor (2016). "Safe control under uncertainty with Probabilistic Signal Temporal Logic". In: *Proc. of Robotics: Science and Systems*. arXiv: 1510.07313 (cit. on p. 39).

Schmidhuber, J. (1991). "Curious model-building control systems". In: *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*. [Proceedings] 1991 IEEE International Joint Conference on Neural Networks, 1458–1463 vol.2 (cit. on p. 32).

Schoellig, A.P., C. Wiltsche, and R. D'Andrea (2012). "Feed-forward parameter identification for precise periodic quadrocopter motions". In: *Proc. of the American Control Conference (ACC)*. American Control Conference (ACC), pp. 4313–4318 (cit. on p. 59).

Schoellig, Angela P., M. Hehn, S. Lupashin, and R. D'Andrea (2011). "Feasiblity of motion primitives for choreographed quadrocopter flight". In: *Proc. of the American Control Conference (ACC)*. American Control Conference (ACC), 2011, pp. 3843–3849 (cit. on p. 65).

Schreiter, Jens, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint (7, 2015). "Safe exploration for active learning with Gaussian processes". In: *Machine Learning and Knowledge Discovery in Databases*. 9286. Springer International Publishing, pp. 133–149 (cit. on pp. 37, 51).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (19, 2017). "Proximal Policy Optimization Algorithms". In: arXiv: 1707.06347 [cs] (cit. on p. 21).

Schwarm, Alexander T. and Michael Nikolaou (1999). "Chance-constrained model predictive control". In: *AIChE Journal* 45.8, pp. 1743–1752 (cit. on p. 19).

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (28, 2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489 (cit. on p. 1).

Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller (2014). "Deterministic policy gradient algorithms". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 387–395 (cit. on p. 21).

Simon, D., J. Löfberg, and T. Glad (2013). "Nonlinear model predictive control using Feedback Linearization and local inner convex constraint approximations". In: *2013 European Control Conference (ECC)*. 2013 European Control Conference (ECC), pp. 2056–2061 (cit. on p. 101).

Solak, E., R. Murray-smith, W. E. Leithead, D. J. Leith, and Carl E. Rasmussen (2003). "Derivative observations in Gaussian process models of dynamic systems". In: *Proc. of Neural Information Processing Systems (NIPS)*. Ed. by S. Becker, S. Thrun, and K. Obermayer. MIT Press, pp. 1057–1064 (cit. on p. 52).

Soloperto, Raffaele, Matthias A. Müller, Sebastian Trimpe, and Frank Allgöwer (1, 2018). "Learning-Based Robust Model Predictive Control with State-Dependent Uncertainty".

In: *IFAC-PapersOnLine*. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018 51.20, pp. 442–447 (cit. on p. 40).

Srinivas, Niranjan, Andreas Krause, Sham M. Kakade, and Matthias Seeger (2012). "Gaussian process optimization in the bandit setting: no regret and experimental design". In: *IEEE Transactions on Information Theory* 58.5, pp. 3250–3265. arXiv: 0912.3995 (cit. on pp. 23, 25, 26, 30, 106, 115, 120, 121, 132, 141, 151, 160–163, 167).

Sui, Yanan, Alkis Gotovos, Joel W. Burdick, and Andreas Krause (2015). "Safe exploration for optimization with Gaussian processes". In: *Proc. of the International Conference on Machine Learning (ICML)*, pp. 997–1005 (cit. on pp. 37, 47, 48, 53, 68, 130, 147, 152).

Sutton, Richard S. (1, 1988). "Learning to predict by the methods of temporal differences". In: *Machine Learning* 3.1, pp. 9–44 (cit. on p. 11).

Sutton, Richard S. (1, 1990). "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming". In: *Machine Learning Proceedings 1990*. Ed. by Bruce Porter and Raymond Mooney. San Francisco (CA): Morgan Kaufmann, pp. 216–224 (cit. on p. 32).

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement learning: an introduction*. MIT press (cit. on pp. 1, 5, 20).

Sutton, Richard S., David A. McAllester, Satinder P. Singh, Yishay Mansour, et al. (1999). "Policy gradient methods for reinforcement learning with function approximation." In: *NIPS*. Vol. 99, pp. 1057–1063 (cit. on p. 21).

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2014). "Intriguing properties of neural networks". In: *Proc. of the International Conference on Learning Representations (ICLR)*. arXiv: 1312.6199 (cit. on p. 99).

Tamar, Aviv, Shie Mannor, and Huan Xu (2014). "Scaling Up Robust MDPs by Reinforcement Learning". In: *Proc. of the International Conference on Machine Learning (ICML)*. arXiv: 1306.6189 (cit. on p. 97).

Tang, Haoran, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel (2017). "#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 2753–2762 (cit. on p. 32).

Tassa, Y., T. Erez, and E. Todorov (2012). "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4906–4913 (cit. on p. 13).

Tesch, Matthew, Jeff Schneider, and Howie Choset (2011). "Using response surfaces and expected improvement to optimize snake robot gait parameters". In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1069–1074 (cit. on p. 30).

The GPy authors (2012–2014). *GPy: A Gaussian process framework in Python* (cit. on p. 58).

Theodorou, E., J. Buchli, and S. Schaal (2010). "Reinforcement learning of motor skills in high dimensions: A path integral approach". In: *2010 IEEE International Conference on Robotics and Automation*. 2010 IEEE International Conference on Robotics and Automation, pp. 2397–2403 (cit. on p. 21).

Thomas, Philip S., Georgios Theocharous, and Mohammad Ghavamzadeh (2015a). "High confidence policy improvement". In: *International Conference on Machine Learning*, pp. 2380–2388 (cit. on p. 37).

Thomas, Philip S., Georgios Theocharous, and Mohammad Ghavamzadeh (2015b). "High-confidence off-policy evaluation". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence* (cit. on p. 37).

Turchetta, Matteo, Felix Berkenkamp, and Andreas Krause (2016). "Safe exploration in finite markov decision processes with gaussian processes". In: Proc. of Neural Information Processing Systems (NeurIPS), pp. 4305–4313 (cit. on pp. 39, 126).

Turchetta, Matteo, Felix Berkenkamp, and Andreas Krause (2019). *Safe Exploration for Interactive Machine Learning* (cit. on pp. 125, 126).

Vershynin, Roman (12, 2010). "Introduction to the non-asymptotic analysis of random matrices". In: arXiv: `1011.3027 [cs, math]` (cit. on pp. 27, 139, 140).

Vershynin, Roman (2012). *Compressed Sensing, Theory and Applications* (cit. on p. 139).

Vinogradska, Julia, Bastian Bischoff, Duy Nguyen-Tuong, and Jan Peters (2017). "Stability of controllers for Gaussian process dynamics". In: *The Journal of Machine Learning Research* 18.1, pp. 3483–3519 (cit. on p. 40).

Wabersich, K. P. and M. N. Zeilinger (2018). "Linear Model Predictive Safety Certification for Learning-Based Control". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018 IEEE Conference on Decision and Control (CDC), pp. 7130–7135 (cit. on p. 40).

Wachi, Akifumi, Yanan Sui, Yisong Yue, and Masahiro Ono (26, 2018). "Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. Thirty-Second AAAI Conference on Artificial Intelligence (cit. on p. 39).

Wächter, Andreas and Lorenz T. Biegler (1, 2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1, pp. 25–57 (cit. on p. 104).

Wang, Y., N. Matni, and J. C. Doyle (2019). "A System Level Approach to Controller Synthesis". In: *IEEE Transactions on Automatic Control*, pp. 1–1 (cit. on p. 41).

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8.3-4, pp. 279–292 (cit. on p. 33).

Wiesemann, Wolfram, Daniel Kuhn, and Berç Rustem (14, 2012). "Robust Markov Decision Processes". In: *Mathematics of Operations Research* 38.1, pp. 153–183 (cit. on pp. 39, 97).

Williams, G., N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou (2017). "Information theoretic MPC for model-based reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1714–1721 (cit. on p. 21).

Williams, Ronald J. (1, 1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3, pp. 229–256 (cit. on p. 20).

Woodcock, Jim, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald (2009). "Formal Methods: Practice and Experience". In: *ACM Comput. Surv.* 41.4, 19:1–19:36 (cit. on pp. 15, 19).

Xie, C., S. Patil, T. Moldovan, S. Levine, and P. Abbeel (2016). "Model-based reinforcement learning with parametrized physical models and optimism-driven exploration". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 504–511 (cit. on pp. 34, 104).

Zanette, Andrea and Emma Brunskill (1, 2019). "Tighter Problem-Dependent Regret Bounds in Reinforcement Learning without Domain Knowledge using Value Function Bounds". In: arXiv: 1901.00210 [cs, stat] (cit. on p. 33).

Zhou, Kemin and John Comstock Doyle (1998). *Essentials of robust control*. Vol. 104. Prentice Hall (cit. on pp. 19, 39).