# Safe Control of Robotic Systems under Uncertainty: Reconciling Model-based and Data-driven Methods

by

Fernando Castaneda Garcia-Rozas

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Koushil Sreenath, Chair
Professor Claire J. Tomlin
Professor Francesco Borrelli

Spring 2023

The dissertation of Fernando Castaneda Garcia-Rozas, titled Safe Control of Robotic Systems under Uncertainty: Reconciling Model-based and Data-driven Methods, is approved:

Chair _____ Date _____

_____ Date _____

_____ Date _____

University of California, Berkeley

Safe Control of Robotic Systems under Uncertainty: Reconciling Model-based and Data-driven Methods

Abstract

Safe Control of Robotic Systems under Uncertainty: Reconciling Model-based and Data-driven Methods

by

Fernando Castaneda Garcia-Rozas

Doctor of Philosophy in Mechanical Engineering

University of California, Berkeley

Professor Koushil Sreenath, Chair

Safety is a primary concern for deploying autonomous robots in the real world. Model-based control theoretic tools provide formal safety guarantees when a mathematical model of the dynamics of the system is available. However, constructing analytical models that accurately predict future states from sensory measurements and designing controllers that use such models can be costly, labor-intensive, or even unattainable for intricate real robotic systems. In contrast, learning-based control strategies have demonstrated the ability for robots to execute complex tasks directly from data, eliminating the need for analytical dynamics models. However, these learning-based control policies are typically trained intensively on vast datasets collected through simulations, which may not encompass the full range of complexities found in real-world interactions. Furthermore, the resulting policies often lack interpretability, making it hard to formally analyze their robustness properties.

This dissertation focuses on establishing core principles for developing reliable and intelligible controllers for real-world autonomous systems. Specifically, it introduces formal techniques that leverage approximate model knowledge when available, and utilize data to nimbly adapt to the intricacies of the real world. The thesis unfolds in two main sections, each examining how control-theoretic model-based strategies and data-driven methodologies can mutually enhance each other. In the first section, the emphasis is on the application of control theory principles to enhance the interpretability and trustworthiness of data-driven approaches. Conversely, the second section flips this narrative, investigating how data can boost control-theoretic approaches and transfer the assurances provided by model-based controllers on approximate models to the actual system being controlled.

Delving into the first part of the dissertation, a principled reward design methodology for model-free policy optimization problems is presented. By exploiting the underlying geometric structures of the system, the proposed reward functions guide the policy search towards safe and stabilizing control policies. It is demonstrated that these policies can be learned directly

on hardware from only a few minutes of experimental data. Following this, the dissertation draws on nonlinear control methods to propose an end-to-end distributional shift prevention mechanism for learning-based policies. Preventing distributional shift is, in fact, a critical matter for assuring the safety of data-driven controllers, as operating in unexplored regions can lead to the unexpected behavior of these policies. Taking raw high-dimensional sensory observations as input, the proposed mechanism constitutes an effective safety layer for a wide variety of applications, from robotic manipulation to autonomous driving.

The second part of the dissertation introduces several approaches, ranging from reinforcement learning to Bayesian inference methods, able to safely bridge the gap between an approximate dynamics model and the real system when using model-based controllers. By quantifying the uncertainty within the learning model, it also presents a safe online learning strategy that empowers the system to assess whether its current information is adequate for ensuring safety, or if acquiring new measurements is necessary. Additionally, it puts forward a data selection method that ascertains the impact of individual data points on the overall decision-making process.

To my parents, Antonio and Concha, and to Rosario.

# Contents

**Contents** **ii**

**List of Figures** **v**

**List of Tables** **vii**

**1 Introduction** **1**
1.1 Thesis Motivation . . . . . 1
1.2 Challenges in Safe Robotic Control . . . . . 2
1.3 Contributions . . . . . 3
1.4 Dissertation Outline . . . . . 4

**2 Mathematical Background** **6**
2.1 System Dynamics . . . . . 6
2.2 Model-based Control . . . . . 7
2.3 Data-driven Control . . . . . 15

**I Model-based Supervision for Safe Robotic Learning** **19**

**3 Learning Min-norm Stabilizing Controllers** **20**
3.1 Introduction . . . . . 20
3.2 Learning Min-norm Stabilizing Controllers . . . . . 23
3.3 Examples . . . . . 28
3.4 Chapter Summary . . . . . 33

**4 Learning Min-norm Safe Stabilizing Controllers** **34**
4.1 Introduction . . . . . 35
4.2 Learning Min-norm Safe Stabilizing Controllers . . . . . 36
4.3 Theoretical Analysis . . . . . 38
4.4 Reinforcement Learning Implementation . . . . . 40
4.5 Examples . . . . . 41
4.6 Chapter Summary . . . . . 43

**5 Lyapunov-based Cost Design for Robust and Efficient Robotic Reinforcement Learning 45**
5.1 Introduction . . . 46
5.2 Problem Setting . . . 48
5.3 Lyapunov Design for Infinite Horizon Reinforcement Learning . . . 51
5.4 Connection to Stability of Model Predictive Control . . . 54
5.5 Overview of Experimental Results . . . 55
5.6 Examples and Comparison Studies . . . 58
5.7 Chapter Summary . . . 67

**6 Constraining Distributional Shift with Nonlinear Control Self-Supervision 68**
6.1 Introduction . . . 68
6.2 Problem Statement . . . 71
6.3 In-Distribution Barrier Functions . . . 73
6.4 Learning iDBFs from High-Dimensional Observations . . . 75
6.5 Examples . . . 76
6.6 Chapter Summary . . . 79

**II Leveraging Data to Safely Bridge the Reality Gap 81**

**7 Reinforcement Learning for Feedback Linearization Controllers under Model Uncertainty 82**
7.1 Introduction . . . 82
7.2 Feedback Linearization . . . 84
7.3 Effects of Uncertainty on the Feedback Linearization . . . 85
7.4 Reinforcement Learning for Uncertainty Compensation . . . 86
7.5 Implementation for Bipedal Walking . . . 87
7.6 Numerical Validation . . . 88
7.7 Chapter Summary . . . 91

**8 Reinforcement Learning for CLF and CBF-based Controllers under Model Uncertainty 92**
8.1 Introduction . . . 92
8.2 Control Lyapunov Functions and Control Barrier Functions under Feedback Linearization . . . 94
8.3 Reinforcement Learning for CLF-QP Based Controllers under Model Mismatch 97
8.4 Reinforcement Learning for CBF-CLF-QP Based Controllers under Model Mismatch . . . 99
8.5 Reinforcement Learning-based Framework . . . 101
8.6 Application to Bipedal Robots . . . 103
8.7 Numerical Validation . . . 104

8.8 Chapter Summary . . . . . 108

**9 Probabilistic Stabilizing Control under Uncertainty 110**
9.1 Introduction . . . . . 111
9.2 Problem Setting . . . . . 112
9.3 Gaussian Process Regression for Affine Target Functions . . . . . 113
9.4 Uncertainty-Aware Min-norm Stabilizing Controller . . . . . 117
9.5 Data Collection . . . . . 119
9.6 Examples . . . . . 122
9.7 Chapter Summary . . . . . 124

**10 Recursively Feasible Probabilistic Safe Control under Uncertainty 126**
10.1 Introduction . . . . . 127
10.2 Problem Statement . . . . . 130
10.3 Gaussian Process Regression to Estimate the Safety-Critical Uncertainty . . 132
10.4 Probabilistic Safety Filter . . . . . 133
10.5 Analysis of Pointwise Feasibility . . . . . 135
10.6 Probabilistic Safe Online Learning . . . . . 139
10.7 Examples . . . . . 144
10.8 Chapter Summary . . . . . 149

**11 Online Data Selection for Scalable Probabilistic Safe Control under Uncertainty 151**
11.1 Introduction . . . . . 151
11.2 Certifying Filters for Uncertain Systems . . . . . 154
11.3 Constraint-Guided Online Data Selection . . . . . 161
11.4 Examples . . . . . 171
11.5 Chapter Summary . . . . . 180

**12 Conclusions and Future Vision 181**

**Bibliography 183**

**A Proofs and Intermediate Results 199**
A.1 Chapter 3 Proofs . . . . . 199
A.2 Chapter 4 Proofs . . . . . 200
A.3 Chapter 5 Proofs and Intermediate Results . . . . . 201
A.4 Chapter 10 Proofs and Intermediate Results . . . . . 203
A.5 Chapter 11 Proofs . . . . . 208

# List of Figures

3.1 RABBIT, a five-link bipedal robot. . . . . 22
3.2 Learning curve for the double pendulum. . . . . 29
3.3 Double pendulum results of learning a min-norm stabilizing controller . . . . . 29
3.4 Snapshots of the walking gait with the min-norm CLF controller using the nominal (inaccurate) model. . . . . 30
3.5 Snapshots of the walking gait with the learned controller. . . . . 30
3.6 Learning curve of the RABBIT environment . . . . . 32
3.7 Learning a min-norm stabilizing controller results for RABBIT . . . . . 32

4.1 RABBIT stepping stones snapshots using the learned policy. . . . . 34
4.2 Double pendulum trajectory under the learned safe stabilizing controller . . . . 42
4.3 Step length plot for the stepping stone simulation of RABBIT using our learned controller . . . . . 43

5.1 Summary of the experimental results of Chapter 5 . . . . . 45
5.2 Results of the fine-tuned policy for the A1 velocity tracking experiment using our cost-shaping method. . . . . 56
5.3 Experimental results of the fine-tuned policy for the A1 quadruped carrying a load using our cost-shaping method. . . . . 60
5.4 Tracking error at different learning stages for the A1 quadruped using our novel cost. . . . . 60
5.5 Plots of cart-pole experimental results before and after finetuning with our novel cost. . . . . 61
5.6 Ablation study of cart-pole finetuning simulations using our proposed cost-shaping method . . . . . 63
5.7 Bipedal walking simulation comparison results with our cost-shaping method. . 65
5.8 Inverted pendulum learning curves comparison with our novel cost-shaping method. 66

6.1 In-Distribution Barrier Functions: inference diagram. . . . . 69
6.2 iDBF top-down visual navigation example results. . . . . 78
6.3 Egocentric driving simulation results using the iDBF policy filter. . . . . 79

7.1 Results of reinforcement learning for model uncertainty in feedback linearization. 89

7.2 Results of reinforcement learning for torque saturation in feedback linearization. 90
7.3 Results of tracking untrained trajectories using our RL approach for feedback linearization. . . . . 91

8.1 RL-CBF-CLF-QP framework diagram. . . . . 93
8.2 Tracking error of RABBIT walking simulations under the RL-CLF-QP with scaled masses and inertias. . . . . 105
8.3 Tracking error of RABBIT walking simulations under the RL-CLF-QP with additional torso mass. . . . . 106
8.4 Depiction of the RABBIT stepping stone CBF constraint. . . . . 106
8.5 Results of the RL-CBF-CLF-QP controller for the simulation of RABBIT walking on stepping stones. . . . . 107

9.1 Greedy data collection algorithm to estimate the region of attraction based on a CLF. . . . . 121
9.2 Results of the inverted pendulum simulation using the GP-CLF-SOCP . . . . . 123
9.3 Results of the kinematic vehicle simulation using the GP-CLF-SOCP . . . . . 124

10.1 Geometric interpretation of the feasibility conditions of the GP-CBF-SOCP. . . 138
10.2 Feasibility region expansion using the proposed safe learning algorithm on an adaptive cruise control system. . . . . 144
10.3 Summary of results of the proposed safe learning algorithm applied to an adaptive cruise control system. . . . . 145
10.4 Illustration of the safe set of the kinematic vehicle example. . . . . 146
10.5 Trajectory of the kinematic vehicle under our proposed safe online learning algorithm. . . . . 147
10.6 Summary of results of the kinematic vehicle example under our safe learning algorithm. . . . . 148

11.1 Comparison of simulation results of the polynomial running example using different stability filters. . . . . 162
11.2 Comparison between naive data selection and our proposed data selection strategy for the polynomial running example. . . . . 169
11.3 Configuration variables of RABBIT and the Quanser cart-pole systems. . . . . 172
11.4 Rabbit simulation results with our proposed online data selection method. . . . 173
11.5 Visualization of the data selection for the RABBIT example using our proposed approach. . . . . 174
11.6 Training data collected in the cart-pole experiments. . . . . 177
11.7 Experimental results of our proposed safety filter with online data selection on the Quanser cart-pole system. . . . . 178

# List of Tables

6.1 iDBF results compared to other approaches to prevent distributional shift. . . . 77

11.1 Comparison of execution time of the safety filter using our data selection strategy vs using the whole dataset. . . . . 175

## Acknowledgments

One of the most important journeys of my life comes to an end with this dissertation. Almost five years ago, when I first arrived at Berkeley, I had high expectations of what this period could bring. What I can say now is that I fell short when envisioning how enjoyable and fruitful, both personally and academically, this journey would be. In fact, I cannot describe in words how much I have learned during these past five years, all thanks to the brilliant people that I have encountered during this time, and without whom I would not have been able to finish this PhD. These final words are devoted to them.

First of all, I would like to thank my advisor, Professor Koushil Sreenath, for the continual support, guidance and kindness over these years. Thank you, Koushil, for being a great mentor and a true role model. I hope to inspire others with the same passion for this field that you've sparked in me.

I am grateful to the other two members of my dissertation committee: Professor Claire Tomlin, who has acted almost as a second advisor during my PhD, providing crucial guidance and feedback on my ideas; and Professor Francesco Borrelli, whose life wisdom and encouragement to undertake ambitious research have been indispensable.

The work presented in this dissertation is the result of many fruitful collaborations with some of the brightest individuals I have ever met. Thank you, Ayush and Tyler, for the countless discussions about the future of reinforcement learning for locomotion, for the brainstorms and joint discoveries, and for the hours we spent together debugging code and running experiments. Thank you, Jason, for being an incredible collaborator all these years, and for all the joyful eureka moments we had when doing math together. Thank you, Bike, for being the most generous lab-mate and collaborator that I could ask for, always willing to give a helping hand with simulations and experiments. Thank you, Mathias for being my first collaborator in the project that ultimately shaped what my PhD would look like. Thank you, Wonsuhk, for all your help with our GP project. I also want to thank all the other members of the Hybrid Robotics Group: Prasanth, Zhongyu, Jun, Shu, Matthew and Akshay, each one of you contributed to making the time I spent at the lab truly enjoyable.

I owe much gratitude to Haruki Nishimura and all the members of the Machine Learning Research division at Toyota Research Institute, for giving me the opportunity to work with them as an intern. The wealth of knowledge I gained about deep learning from this experience is invaluable.

Many other individuals have indirectly contributed to my research through insightful discussions and idea sharing. While I cannot acknowledge everyone, I must thank Andrew Taylor and Victor Dorobantu for the unending conversations on data-driven control. I am also grateful to Alonso Marco for his always relevant suggestions.

My work would not have come to fruition without the financial backing I received during my PhD. Truly, my journey towards this PhD commenced the day when I was granted the esteemed "la Caixa" Foundation Fellowship. This award was a game-changer for me, opening the door to further my studies in the USA. I am equally grateful to the Rafael del Pino Foundation for their supportive endorsement of my work. Lastly, the National

Science Foundation deserves acknowledgment for their financial assistance through Grant CMMI-1931853.

I must also extend my gratitude to the extraordinary friends I have made during this journey. In particular, my two housemates, Jason and Drew, who have shared nearly the entirety of the PhD with me, deserve special recognition. Not to forget Tony, who, though not officially living with us, was always an integral part of our household. Thank you to Matt, Eloïse, Mathias, Joe and all the visiting students who were part of the unforgettable moments I experienced during my initial years at Berkeley. Thank you to my Spanish fellows Arturo, Raquel, Marta, and Cris, with whom I have embarked on some truly unforgettable trips. Thanks to Vicky for being an incredibly good friend and to all the other amazing people I have met during the last years of the PhD: Rui, Pier, Inês, Giulio, Diana, Carol, Sebastian, Alessio, Gaia, Ale, Francesca, Jodi and many more. Thank you to my ME PhD fellows Cyndia, Tony, Michael and to all the EH 1115 folks, who never failed to lift my spirits when research demanded a pause.

I do not want to miss out on acknowledging my group of friends from back home: Palao, both Diegos, Jorge, Alex, Cayetano, Llamas and Manu. Regardless of my distance from home, they continue to be the most supportive companions in life.

My deepest and final gratitude is reserved for my family, for their unconditional support and love, always prioritizing my best interests, even if this means being far away from them. I cannot thank enough my dearest girlfriend Irene. Since she became a part of my life, she has emerged as the most significant source of joy. She keeps instilling in me the belief that the most daunting challenges can indeed be overcome. I also want to thank Rosario, who mentored me during the early phases of my life and instilled in me the virtue of generosity. She stands as one of my primary inspirations. Thank you to my brother Toni, and to my sister Almudena, who have always held faith in me and encouraged my aspirations. Finally, I want to thank my father Antonio, and my mother Concha. Ultimately, it is the unconditional love they have shown me that has made the realization of this dissertation possible.

# Chapter 1

# Introduction

## 1.1 Thesis Motivation

In this era of swift technological advancement, human society is witnessing an unprecedented rise in automation. This growth is driven by factors such as greater accessibility to computing resources, high-speed internet, and remarkable progress in artificial intelligence. Automation has spread far beyond the industrial sphere, now affecting various aspects of our daily lives. For instance, a vast range of software applications running on our smartphones constantly generates information and recommendations tailored to our needs.

Robotics has emerged as a crucial area of innovation in the automation landscape. Robotic systems, once exclusive to factories, now consist of electromechanical devices capable of sensing, decision-making, and action-taking, steadily becoming integrated into our everyday lives. Robotic vacuum cleaners are now a household item [63], while robotic prosthetics are already enhancing the quality of life for individuals with disabilities [137, 33]. Drones are being massively sold worldwide for both recreational and professional applications, including photography, environmental monitoring, and search and rescue. Legged robots are starting to be commercialized to private users and industries alike [22, 195], and the deployment of autonomous vehicles has already begun [158], with the promise to revolutionize urban transportation and significantly improve road safety. It is evident that the potential of robotic technologies to empower human endeavors and enhance our collective well-being is immense.

However, as these complex automated systems become more widespread, new concerns about safety and reliability emerge. The improper functioning of an autonomous vehicle, for example, could lead to severe accidents affecting both the passengers and other road users [74]. Thus, as we continue to develop and implement safety-critical robotic systems in the coming years, it is crucial to prioritize avoiding catastrophic failures.

This dissertation seeks to explore methods for systematically assessing the safety of existing and future robotic systems, determining the extent to which these technologies can be assured, and incorporating safety guarantees into their automated decision-making pro-

cesses. A special emphasis is placed on the safety of learning-based and data-driven algorithms that control robotic systems. Reinforcement learning has showcased remarkable results in controlling robotic systems [174, 117, 2, 142], enabling them to adapt and excel in complex tasks. Learning-based approaches, in general, are highly promising due to the ever-increasing amounts of data available to train these algorithms. However, in contrast to studies that exclusively use data for control, this dissertation illustrates how established model-based control methods can effectively augment data-driven techniques, providing a solid foundation for evaluating the safety and performance of the learned control policies. Ultimately, this combined approach allows us to propose feasible, rigorous, and efficient methods that enable robotic systems to uphold safety with a high degree of confidence, even in the face of considerable uncertainties.

## 1.2 Challenges in Safe Robotic Control

### 1.2.1 Model-based Control

Model-based controllers play a crucial role in various domains, leveraging mathematical models to predict outcomes and guide decision-making. However, the effectiveness of these models is often hindered by model uncertainty and the complexity inherent in real-world systems. One of the primary concerns is that the assurances provided by mathematical models may not hold in practice when the model fails to accurately represent the underlying system. The potential consequences of relying on an inaccurate model are numerous and could lead to undesirable outcomes, particularly in safety-critical applications like autonomous vehicles or robotic surgery.

Another challenge is designing accurate models for multi-agent systems, particularly those involving human interactions. In the real world, robotic systems cannot be expected to operate in isolation. Instead, they must coexist and cooperate with other agents, including humans, whose behavior is often unpredictable and complex. Accurately modeling such multi-agent systems is essential for ensuring safe and efficient collaboration between agents. However, the process of developing a robust model that takes into account the intricacies of human behavior and effectively bridges the reality gap is not an easy task.

Lastly, even when a dynamics model is available, model-based optimal controllers can face difficulties related to computational complexity and numerical conditioning, making it challenging to find good solutions using common optimization solvers. High-dimensional and nonlinear problems present particular challenges, which become even more pronounced when dealing with high-dimensional sensory data like images. State estimation for such data is a complex issue in itself, and integrating this information into model-based methods presents additional challenges.

### 1.2.2 Learning-based Control

The prospect of using data-driven approaches for designing controllers has become increasingly appealing, given the vast amounts of data now available. However, ensuring the safety of learning-based controllers is a complex and daunting task. One key issue is that data-driven controllers often involve highly overparameterized neural networks, which are difficult to understand and tend to be treated as black boxes. This lack of interpretability makes it particularly challenging to guarantee the safety of these methods when they are deployed in real-world systems. Ensuring that learning-based controllers operate safely and effectively requires a deep understanding of their behavior, which remains an ongoing challenge for researchers and engineers.

A further complication arises in anticipating the behavior of learning-based controllers when they operate beyond the scope of their training data distribution. It is often unfeasible to predict how these controllers will respond to situations not encountered during training. To tackle this issue, it is necessary to develop reliable methods that either keep the systems within the bounds of the training data distribution, promoting well-founded decision-making, or that manage to safely explore unvisited regions.

Furthermore, developing accurate uncertainty estimation techniques for the learned policies and models is crucial, as this allows the controllers to assess when their understanding of the world is inadequate to take action, potentially prompting human intervention. Uncertainty quantification should also enhance the understanding of which data is vital to acquire, and therefore facilitate the development of life-long learning approaches that effectively utilize the most valuable collected data. In addition, it is important for learning-based controllers to be able to safely gather missing data when needed, enabling them to adapt and improve their performance over time. Developing strategies that strike the right balance between exploiting existing knowledge and safely exploring new information is an essential aspect of ensuring the safety of learning-based controllers, ultimately contributing to their successful deployment in real-world applications.

## 1.3 Contributions

This thesis takes a first principles approach for the design of reliable and interpretable controllers for real-world uncertain systems. In particular, we present formal methods that exploit approximate model knowledge when available, and use data to flexibly adapt to the complexities of real systems. With this, we provide evidence about the potential of combining both model-based and data-driven control paradigms to *get the best of both worlds*. The key contributions of this dissertation are the following:

1. *Supervising reinforcement learning with structural model knowledge:* We propose novel methods to exploit approximate model knowledge in the design of reinforcement learning reward functions, leading to better conditioned and more sample efficient policy learning problems. By exploiting the underlying geometric structure of the system, we

guide the policy search towards safe and stabilizing solutions. As such, these methods constitute principled approaches for reward-shaping. We apply these techniques to rapidly learn safe and stabilizing control policies directly on real robotic systems, including the A1 quadrupedal robot.

2. *Long-term distributional shift prevention with a safety certificate self-supervision:* We propose a self-supervised learning approach to constrain learning-based control policies to remain in-distribution with respect to the training data. Precisely, we study how to embed the mathematical property of set invariance into a deep learning-based framework. We use these ideas to build safety filters that, taking high-dimensional observations as inputs, constrain autonomous vehicles and robotic systems from entering out-of-distribution regions of the state space.

3. *Augmenting model-based controllers with data to safely bridge the reality gap:* We propose several supervised learning frameworks to learn the uncertainty gap between an approximate dynamics model and the real system when using model-based controllers. Our decision making module is able to reason about its prediction quality confidence based on the current available information. This allows us to robustify model-based controllers in order to account for the model mismatch.

4. *Control theoretic safe online learning:* Finally, we show that model-based knowledge serves to add meaning to data collected from real systems, and to understand which kind of information is required to prevent failure. We use these insights to propose safe active learning strategies, in which the system can safely collect data when needed in order to stay safe. Furthermore, we propose data selection approaches which quantify the importance that each datapoint has on the decision making process.

## 1.4 Dissertation Outline

First, in Chapter 2 we introduce some important concepts that constitute the theoretical foundation for the results of this dissertation. In particular, we present necessary background on the different model-based and data-driven control techniques that we use throughout the thesis. Afterwards, the main body of the dissertation is broken up into two parts, as follows.

**Part I: Model-based Supervision for Safe Robotic Learning.** The first part of the thesis provides guidance on how to use tools from the nonlinear control literature to supervise deep-learning algorithms for decision making. In particular, Chapters 3 and 4 show that it is possible to guide policy optimization algorithms for systems with unknown dynamics towards safe and stabilizing solutions. Then, in Chapter 5, we propose a Lyapunov-based reward shaping strategy for reinforcement learning problems that greatly simplifies the search for stabilizing policies. As a result, our method can be used to efficiently learn policies from real experimental data. Finally, in Chapter 6, we present an framework whose objective

is to avoid distributional shift for visuomotor control tasks. We prove that, even for end-to-end deep learning control frameworks, incorporating control-theoretic inductive biases in the learning process can serve to encourage the satisfaction of desired properties, such as set invariance.

**Part II: Leveraging Data to Safely Bridge the Reality Gap.** The second part of this dissertation tackles the central challenge of ensuring the safe operation of real systems despite the use of inevitably inaccurate mathematical descriptions of their dynamics. Specifically, Chapters 7 and 8 propose the use of neural network function approximation schemes to learn the discrepancy between the real system and the approximate mathematical model. As shown in these chapters, after properly accounting for the reality gap, a variety of model-based control methods are effective at controlling complex legged robots. Then, in Chapters 9 and 10, we replace these neural network approximators by Gaussian Process regression models, which provide probabilistic high-confidence bounds of their predictions. By formulating robust control problems that take into account these worst-case bounds, we show that it is possible to ensure the safe operation of the real system. Additionally, in Chapter 10, we propose a safe online learning strategy that empowers the system with the ability to autonomously and safely collect meaningful data when needed. Taking further these ideas, Chapter 11 presents a data selection method able to identify and choose data that offers valuable insights into maintaining safety.

Finally, Chapter 12 provides concluding remarks and identifies critical technical obstacles that must be overcome in the pursuit of safely deploying advanced autonomous systems.

# Chapter 2

# Mathematical Background

## 2.1 System Dynamics

In this thesis, we will focus on understanding the functioning and evolution of different robotic systems. We will approach this problem through the mathematical framework of dynamical systems theory, and investigate an extensive array of robotic platforms, such as legged robots and autonomous vehicles, among others. To study their evolution over time, a key notion is that of the *state* of the system—a minimal set of variables required to define the current condition of the system at any given time and predict its future behavior in the absence of external inputs. For robotic systems, there is another important set of variables, the *control input*, which consists of those that an external entity, usually called a controller, can manipulate to alter the state of the system in a desirable way. For example, the motion of a humanoid robot depends on the applied motor torques, which a user can adjust at any given instant.

### 2.1.1 Continuous-Time Dynamics

We can describe the evolution over time of robotic systems using differential equations. For instance, continuous-time models of the evolution of a robot derived from the Lagrangian or from the Newton-Euler equations have been extensively used to predict and control robots for a wide variety of applications.

In a very general form, we can model the dynamics of the robot through the differential equation

$$\dot{x} = f(x, u, t), \tag{2.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state of the system, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ the control input, and $t \in \mathbb{R}_+$[1] the time.

[1]Throughout this thesis, we use $\mathbb{R}_{++}$ to denote the set of positive real numbers, and $\mathbb{R}_+$ for the set of non-negative real numbers.

However, as will be shown in later sections, while (2.1) is very general in form, for robotic systems it is often more convenient to exploit the specific structure that results from the Lagrangian equations of motion that describe these systems. In fact, for most robotic systems, the derivative of the state does not depend explicitly on time and the control input appears linearly, resulting in the dynamics

$$\dot{x} = f(x) + g(x)u. \tag{2.2}$$

Because of their structure, these are called *time-invariant control-affine nonlinear systems*. Throughout this thesis, we will assume that the real robotic system we want to control is of the form (2.2), and we will use the term *true plant* to refer to it. While nonlinear, the functions $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^{n\times m}$ are assumed to be locally Lipschitz continuous.

The main focus of this dissertation will be on how to design feedback control policies for system (2.2) to achieve a behaviour with desirable properties. These policies take the current state as input and should apply corrective control actions to accomplish the desired outcome. We will let $\Pi$ denote the space of all control polices $\pi \colon \mathcal{X} \to \mathcal{U}$ for the system.

### 2.1.2 Discrete-Time Dynamics

There are cases in which it can be more convinient to describe the evolution of the system over discrete time steps $k = 0, 1, 2, ....$ These time steps are typically chosen to be regularly spaced, with interval $\Delta t > 0$. In this case, we will consider the following representation of the true plant's dynamics in discrete time:

$$x_{k+1} = F(x_k, u_k), \tag{2.3}$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state at time step $k$, $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$ is the input applied to the system at that time, and $F \colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is the discrete-time state-transition function.

Note that this representation is not meant to imply that the system actually evolves in discrete increments. It is, however, sometimes convenient to use as it allows us to work with sequences of states and inputs, instead of having to treat them as continuous functions of time.

## 2.2 Model-based Control

Now that we have formalized how to describe the evolution of a robotic system, both using continuous-time representations (2.2) and discrete-time ones (2.3), we can introduce different methods that use model knowledge to decide which control inputs should be applied at each time step.

The methods presented in this section assume that perfect knowledge of the true plant's dynamics is available, and use this information to design controllers to achieve specific objectives.

### 2.2.1 Optimal Control

Optimal control studies the problem of optimal decision-making regarding the evolution of a dynamical system. In order to define which decisions are desirable, a notion of cost is introduced in these methods. Then, the optimal decisions will be those that lead to the least cumulative cost over a trajectory.

Here, we present a brief introduction to a subclass of optimal control problems for discrete-time systems of the form (2.3).

We consider a running cost $\ell\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ of the form

$$\ell(x, u) = Q(x) + R(u), \tag{2.4}$$

where $Q\colon \mathcal{X} \to \mathbb{R}$ is the state cost and $R\colon \mathcal{U} \to \mathbb{R}$ is the input cost. Both $Q$ and $R$ are assumed to be positive definite functions (in practice, both are usually quadratic). Given a policy $\pi \in \Pi$, discount factor $\gamma \in [0, 1]$, and initial condition $x_0 \in \mathcal{X}$, the associated long-run cost is:

$$\begin{aligned} V_\gamma^\pi(x_0) = &\sum_{k=0}^{\infty} \gamma^k \ell(x_k, \pi(x_k)) \\ &\text{s.t. } x_{k+1} = F(x_k, \pi(x_k)), \end{aligned} \tag{2.5}$$

where $V_\gamma^\pi\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is the *value function* associated with policy $\pi$. Small discount factors incentivize policies which greedily optimize a small number of time-steps into the future, while larger discount factors promote policies which reduce the cost in the long-run. We say that a policy $\pi_\gamma^* \in \Pi$ is *optimal* if it achieves the smallest cost from each $x \in \mathcal{X}$:

$$V_\gamma^{\pi_\gamma^*}(x) = V_\gamma^*(x) := \inf_{\pi \in \Pi} V_\gamma^\pi(x), \quad \forall x \in \mathcal{X}, \tag{2.6}$$

where $V_\gamma^*\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is the *optimal value function*. Together $V_\gamma^*$ and $\pi_\gamma^*$ capture the 'ideal' behavior induced by the cost function (2.5). It is well-known [20] that the optimal value function will satisfy the Bellman equation:

$$V_\gamma^*(x) = \inf_{u \in \mathcal{U}} \big[\gamma V_\gamma^*(F(x, u)) + \ell(x, u)\big], \quad \forall x \in \mathcal{X}, \tag{2.7}$$

and an optimal policy $\pi_\gamma^*$ will satisfy:

$$\pi_\gamma^*(x) \in \arg\min_{u \in \mathcal{U}} \big[\gamma V_\gamma^*(F(x, u)) + \ell(x, u)\big], \quad \forall x \in \mathcal{X}. \tag{2.8}$$

While it is a well-defined optimization problem, directly solving (2.5) is usually impractical, as it requires optimizing over the infinite-dimensional space of all control policies $\Pi : \mathcal{X} \to \mathcal{U}$ for the system. In contrast, (2.7) and (2.8) present a very interesting method to solve this problem by working backwards through time: if the optimal value function and

control policy are known for trajectories starting at time step $k$, then we can find the corresponding optimal value function and policy for time step $k-1$ by solving a much simpler optimization problem over control inputs (2.8). This is an embodiment of the *dynamic programming principle of optimality* [17], which constitutes one of the most important results in decision theory. However, this approach requires computing the optimal value function and control policy at each state $x \in \mathcal{X}$, which presents a remarkable challenge for systems with high-dimensional state spaces. In particular, by discretizing the state space $\mathcal{X}$ and solving (2.8) at each discrete state, the number of optimization problems that need to be solved and, therefore, the overall required computation in general increase exponentially with the number of state variables. This problem is known as the *curse of dimensionality*.

Most modern reinforcement learning algorithms essentially try to obtain approximate solutions of optimal control problems from data. By directly collecting data from interactions and using highly expressive function approximators, they aim to overcome the curse of dimensionality and the need to know a priori the dynamics model $F$. An introduction to reinforcement learning methods is provided later in this chapter.

Finally, even though solving for an optimal control policy is a very challenging problem for complex systems, optimality in the most strict sense is not required for many robotic applications. It is often sufficient to obtain control policies that lead to the system satisfying some desired properties, such as being *stable*, or that prevent the state of the system from entering undesired regions—keeping it *safe*. We will now introduce these notions of stability and safety, together with model-based controllers that are able to synthesize stabilizing and safe control policies. These controllers often require far less computation efforts than the optimal control problem.

### 2.2.2 Stability and Control Lyapunov Functions

#### Stability Theory

We now introduce the notion of stability of a system, and how Control Lyapunov Functions can be used to synthesize stabilizing control policies. Since these controllers were originally formulated for continuous-time control-affine systems, we adhere to this formulation. Thus, the systems that we consider in this section are of the form (2.2).

Let us consider a locally Lipschitz continuous control policy $\pi \in \Pi$ for system (2.2). Then, the closed loop dynamics of the system can be expressed as

$$\dot{x} = f(x) + g(x)\pi(x) = f_{cl}(x). \tag{2.9}$$

Suppose $x_e \in \mathcal{X}$ is an equilibrium point of the closed-loop system (2.9); that is $f_{cl}(x_e) = 0$. We now can introduce the definition of stability of the equilibrium point $x_e$. However, for simplicity, unless otherwise stated, throughout the thesis we present all the stability-related definitions and theorems for the case in which the equilibrium point is at the origin of $\mathbb{R}^n$; that is, $x_e = 0$. This is without loss of generality, as any equilibrium point can be shifted to the origin via a change of variables.

We will denote $x(t)$ the solution of system (2.2) under a control policy $\pi \in \Pi$ starting at $x(0) = x_0 \in \mathcal{X}$. Since we assume that the vector fields $f$, $g$ and the control policy $\pi$ are all locally Lipschitz continuous in $\mathcal{X} \subseteq \mathbb{R}^n$, then the solution $x(t)$ exists and is unique for some time $t \in [0, \tau_{max})$ [102], where $\tau_{max}$ is known as the maximum time of existence and uniqueness of $x(t)$.

**Definition 2.1** (Stability in the sense of Lyapunov)**.** *Suppose that there exists some $r > 0$ such that, if $\|x(0)\| < r$, the solution $x(t)$ of the autonomous system* (2.9) *uniquely exists for all $t \geq 0$. Then, the equilibrium point $x_e = 0$ of the closed-loop system* (2.9) *is said to be stable in the sense of Lyapunov (SISL) if for each $\epsilon > 0$ there exists a $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\|_2 < \delta \implies \|x(t)\|_2 < \epsilon, \quad \forall t \geq 0. \tag{2.10}$$

**Definition 2.2** (Unstable equilibrium)**.** *We say that the equilibrium point $x_e = 0$ of the closed-loop system* (2.9) *is unstable if it is not stable in the sense of Lyapunov.*

Stability in the sense of Lyapunov is a desirable property since, it intuitively means that trajectories that start close to the equilibrium never leave a neighborhood of it. If the origin of the closed-loop system (2.9) is SISL, we say that the policy $\pi$ is *stabilizing*.

However, the equilibrium being SISL does not imply that the trajectories of the system will converge towards it. Because of this, we now present the definition of asymptotic stability, which constitutes a more practical and desirable property than SISL.

**Definition 2.3** (Asymptotic stability)**.** *The equilibrium point $x_e = 0$ of the closed-loop system* (2.9) *is asymptotically stable if it is SISL and $\delta$ can be chosen such that*

$$\|x(0)\|_2 < \delta \implies \lim_{t \to \infty} x(t) = 0. \tag{2.11}$$

If this is the case, we say that the policy $\pi$ is asymptotically stabilizing to the origin. This property still lacks practicality for some applications, as it only guarantees convergence to the equilibrium as the time goes to infinity, and does not give any notion of how fast this convergence is. This can be overcome by imposing a more strict notion of stability.

**Definition 2.4** (Exponential stability)**.** *The equilibrium point $x_e = 0$ of the closed-loop system* (2.9) *is said to be exponentially stable with rate of convergence $\alpha$, if it is SISL and there exist positive constants $M, \alpha > 0$ such that*

$$\|x(t)\|_2 \leq M e^{-\alpha t} \|x(0)\|_2, \quad \forall t \geq 0. \tag{2.12}$$

### Control Lyapunov Functions

Now that we have introduced the different forms of stability of an equilibrium point, we tackle the problem of designing stabilizing control policies. For this, we can use the model-based control design paradigm of Control Lyapunov Functions (CLFs). We will first introduce the definition of CLF for system (2.2), in the case of unconstrained control inputs; that is, $\mathcal{U} = \mathbb{R}^m$.

**Definition 2.5** (Control Lyapunov Function [11]). *Let $V\colon \mathcal{X} \to \mathbb{R}_+$ be a positive definite, continuously differentiable, and radially unbounded function. $V$ is a* Control Lyapunov Function *(CLF) for system* (2.2) *if there exists a class $\mathcal{K}_\infty$ function $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ such that for each $x \in \mathcal{X} \setminus \{0\}$ the following holds:*

$$\inf_{u\in\mathbb{R}^m} \underbrace{L_f V(x) + L_g V(x) u}_{=\dot{V}(x,u)} + \lambda\left(V(x)\right) \leq 0, \tag{2.13}$$

where the functions $L_f V(x) \coloneqq \nabla V(x) \cdot f(x)$ and $L_g V(x) \coloneqq \nabla V(x) \cdot g(x)$ are known as Lie derivatives. We say that a function $V$ is radially unbounded if $V(x) \to \infty$ as $\|x\| \to \infty$. Furthermore, a continuous function $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ is class $\mathcal{K}_\infty$ if it is strictly increasing and $\lambda(0) = 0$.

If such a CLF exists, the system is globally asymptotically stabilizable to the origin [11]. In this case, it would be desirable to find a locally Lipschitz continuous feedback control law $\pi\colon \mathcal{X} \to \mathbb{R}^m$ such that the condition

$$L_f V(x) + L_g V(x)\pi(x) + \lambda\left(V(x)\right) \leq 0 \tag{2.14}$$

holds for any $x \in \mathcal{X} \setminus \{0\}$. It should be noted that not all systems which satisfy (2.13) admit such a controller [169], but a number of important systems such as the ones considered in this document are continuously stabilizable.

Note that the key benefit of using a CLF for stabilization is that it condenses into a simple condition (2.14) the inherently long-horizon problem of finding a stabilizing control policy.

A simple way of synthesizing a stabilizing control law is by enforcing (2.14) as a constraint in a min-norm optimization problem. Then, the asymptotically stabilizing control law is defined point-wise by:

$$\pi_{\text{CLF}}(x) = \arg\min_{u\in\mathbb{R}^m} \|u\|_2^2 \tag{2.15}$$

$$\text{s.t. } L_f V(x) + L_g V(x)\pi(x) + \lambda\left(V(x)\right) \leq 0. \tag{2.16}$$

At every point, this controller greedily selects the smallest control input which satisfies the CLF constraint. This greatly simplifies the policy synthesis problem compared to typical optimal control methods (2.5), as the use of the CLF removes the need to plan far into the future.

If $V$ is a CLF for the system, a sufficient condition for $\pi_{\text{CLF}}$ to be locally Lipschitz continuous is that $f$, $g$ and the gradient of $V$ are each locally Lipschitz continuous [62]. Furthermore, if $u$ is unconstrained, this min-norm stabilizing controller can be expressed in closed-form [177] as:

$$\pi_{\text{CLF}}(x) = \begin{cases} -\frac{\left(L_f V(x) + \lambda(V(x))\right) L_g V(x)}{L_g V(x)^T L_g V(x)} & \text{if } L_f V(x) + \lambda\left(V(x)\right) > 0 \\ 0 & \text{if } L_f V(x) + \lambda\left(V(x)\right) \leq 0 \end{cases} \tag{2.17}$$

However, many real-world systems require the addition of input constraints due to actuator limitations, i.e., $u \in \mathcal{U} \subset \mathbb{R}^m$, in which case condition (2.13) becomes $\inf_{u\in\mathcal{U}} L_f V(x) + L_g V(x)u + \lambda(V(x)) \leq 0$, which might not be satisfied at every $x \in \mathcal{X} \setminus \{0\}$ even if $V$ is a valid CLF for system (2.2). This fact motivates the following lemma.

**Lemma 2.1** (Stability from a CLF [11])**.** *Let $V\colon \mathcal{X} \to \mathbb{R}_+$ be a CLF for system (2.2) and let $\mathcal{U} \subset \mathbb{R}^m$ be the compact set of admissible control inputs. For each $c \in \mathbb{R}_+$ let $\Omega_c$ be the sublevel set of $V$ such that $\Omega_c \coloneqq \{x \in \mathcal{X} \subseteq \mathbb{R}^n \colon V(x) \leq c\}$. If there exists a $c_i > 0$ such that*

$$\inf_{u\in\mathcal{U}} L_f V(x) + L_g V(x)u + \lambda(V(x)) \leq 0 \tag{2.18}$$

*is satisfied $\forall x \in \Omega_{c_i} \setminus \{0\}$, then the origin is locally asymptotically stabilizable from $x \in \Omega_{c_i}$, and $\Omega_{c_i}$ is a subset of the Region of Attraction (RoA) of the origin.*

*Proof.* See [123, Proposition 2.2]. □

We can now take $c_{max}$ as the maximum value of $c_i \in \mathbb{R}_+$ such that (2.18) holds for any $x \in \Omega_{c_i} \setminus \{0\}$. Then, $\Omega_{c_{max}}$ is the largest sublevel set of $V$ from which the system (2.2) is asymptotically stabilizable.

We can also consider a stronger notion of stabilizability by imposing exponential convergence to the origin. We now restrict the function $\lambda$ to be linear, and with a slight abuse of notation, we denote its coefficient also as $\lambda$. It is well-known that if there exists a compact subset $D \subseteq \Omega_{c_{max}}$ such that $\forall x \in D$ the following holds for some constant $\lambda > 0$,

$$\inf_{u\in\mathcal{U}} L_f V(x) + L_g V(x)u + \lambda V(x) \leq 0, \tag{2.19}$$

then the state of the system can be driven exponentially fast to the origin from any initial state $x_0 \in \Omega_{c_{exp}} \subseteq D$ [9]. If such $c_{exp} > 0$ exists, we will say that $V$ is a *locally exponentially stabilizing* CLF. Similarly to the asymptotic stability case, the condition (2.19) can be incorporated as a constraint into a min-norm optimization problem:

---

**CLF-QP**:

$$\pi_{\text{CLF}}(x) = \quad \arg\min_{u\in\mathcal{U}} \quad \|u\|_2^2 \tag{2.20a}$$

$$\text{s.t. } L_f V(x) + L_g V(x)u + \lambda V(x) \leq 0. \tag{2.20b}$$

---

In this thesis, we always assume that the input constraints that define $\mathcal{U}$ are linear, which makes problem (2.20) a quadratic program (QP). We will refer to constraint (2.20b) as the *exponential CLF constraint*. This optimization problem defines a feedback control law $\pi_{\text{CLF}}\colon \mathcal{X} \to \mathcal{U}$ selecting the min-norm input such that the system state converges to the origin exponentially quickly. Note that, in practice, constraint (2.20b) is typically relaxed

by adding a slack variable in order to guarantee the feasibility of the problem if condition (2.19) is not locally satisfied [65].

The CLF-QP constitutes a model-based controller that synthesizes exponential stabilizing control policies when perfect knowledge of the dynamics of the system $f$ and $g$ is available.

### 2.2.3 Safety and Control Barrier Functions

#### Safety as a Set Invariance Problem

We now formalize the mathematical definition of safety for a continuous-time system (2.2) that we use in this thesis. In particular, we consider that the system is safe when it never leaves a set of safe states $\mathcal{X}_{\text{safe}} \subset \mathcal{X}$.

**Definition 2.6** (Safety as Forward Invariance)**.** *We say that a control law $\pi : \mathcal{X} \to \mathcal{U}$ guarantees the safety of system* (2.2) *with respect to a safe set $\mathcal{X}_{safe} \subset \mathcal{X}$, if the set $\mathcal{X}_{safe}$ is forward invariant under the control law $\pi$, i.e., for any $x_0 \in \mathcal{X}_{safe}$, the solution $x(t)$ of system* (2.2) *under the control law $\pi$ remains in $\mathcal{X}_{safe}$ for all $t \in [0, \tau_{max})$.*

Here, $\tau_{max}$ is the maximum time of existence and uniqueness of $x(t)$, which is guaranteed to exist and be strictly greater than zero if the control law $\pi$ is locally Lipschitz [102].

#### Control Barrier Functions

Similar to the use case of CLFs being synthesizing stabilizing policies, the model-based control design paradigm of Control Barrier Functions (CBFs) serves to build *safe control policies*.

**Definition 2.7** (Control Barrier Function [8])**.** *Let $\mathcal{X}_{safe} := \{x \in \mathcal{X} : B(x) \geq 0\}$ be the zero-superlevel set of a continuously differentiable function $B : \mathcal{X} \to \mathbb{R}$. Then, $B$ is a* Control Barrier Function *(CBF) for system* (2.2) *if there exists an extended class $\mathcal{K}_\infty$ function $\gamma$ such that for all $x \in \mathcal{X}$ the following holds:*

$$\sup_{u \in \mathcal{U}} \underbrace{L_f B(x) + L_g B(x) u}_{=\dot{B}(x,u)} + \gamma(B(x)) \geq 0, \tag{2.21}$$

where $L_f B(x) \coloneqq \nabla B(x) \cdot f(x)$ and $L_g B(x) \coloneqq \nabla B(x) \cdot g(x)$ are the Lie derivatives of $B$ with respect to $f$ and $g$. Furthermore, a continuous function $\gamma : \mathbb{R} \to \mathbb{R}$ is extended class $\mathcal{K}_\infty$ if it is strictly increasing and $\gamma(0) = 0$.

The following result states that the existence of a CBF guarantees that the control system can be rendered safe.

**Lemma 2.2** (Safety from a CBF [8])**.** *Let system* (2.2) *admit a CBF $B : \mathcal{X} \to \mathbb{R}$. Let $\mathcal{X}_{safe} = \{x \in \mathcal{X} : B(x) \geq 0\}$ be its associated safe set, with boundary $\partial\mathcal{X}_{safe} = \{x \in \mathcal{X} :$*

$B(x) = 0\}$. *If for all* $x \in \partial\mathcal{X}_{safe}$ *it holds that* $\nabla B(x) \neq 0$, *then any Lipschitz continuous control law* $\pi : \mathcal{X} \to \mathcal{U}$ *satisfying*

$$\pi(x) \in \{u \in \mathcal{U} : \dot{B}(x, u) + \gamma\left(B(x)\right) \geq 0\} \tag{2.22}$$

*renders the set* $\mathcal{X}_{safe}$ *forward invariant.*

Given a safety-agnostic reference controller $\pi_{\text{ref}} : \mathcal{X} \to \mathcal{U}$, the condition in (2.22) can be used to formulate a minimally-invasive safety-filter [8]:

---

**CBF-QP**:

$$\pi_{\text{CBF}}(x) = \quad \underset{u \in \mathcal{U}}{\arg\min} \quad \|u - \pi_{\text{ref}}(x)\|_2^2 \tag{2.23a}$$

$$\text{s.t.} \quad L_f B(x) + L_g B(x) u + \gamma(B(x)) \geq 0, \tag{2.23b}$$

---

which is a quadratic program (QP) if the input bounds are linear. This problem is solved pointwise in time to obtain a safety-critical control law $\pi_{\text{CBF}} : \mathcal{X} \to \mathcal{U}$ that only deviates from the reference controller $\pi_{\text{ref}}$ when safety is compromised. However, note that this optimization problem requires perfect knowledge of the dynamics of the system, since the Lie derivatives of $B$ appear in the constraint. Therefore, throughout this thesis we will refer to (2.23b) as the *true CBF constraint*, since it depends on the dynamics of the true plant given by $f$ and $g$.

Note that CBFs condense the problem of long-term constraint satisfaction into a single time step condition (2.22). This is a very attractive property, as it removes the need to plan far-ahead into the future to make sure that the system never violates safety and leaves the safe set.

### 2.2.4 Combining Control Lyapunov Functions and Control Barrier Functions

In [8], both CLF and CBF conditions are incorporated in a combined min-norm QP. Similar to the individual CBF and CLF cases, this QP is solved pointwise in time to obtain a safety-critical stabilizing control law $\pi_{\text{CBF-CLF}} : \mathcal{X} \to \mathcal{U}$.

---

**CBF-CLF-QP**:

$$\pi_{\text{CBF-CLF}}(x) = \quad \underset{u \in \mathcal{U},\ d \in \mathbb{R}}{\arg\min} \quad \|u\|_2^2 + pd^2 \tag{2.24a}$$

$$\text{s.t.} \quad L_f V(x) + L_g V(x) u + \lambda V(x) \leq d, \tag{2.24b}$$

$$L_f B(x) + L_g B(x) u + \gamma(B(x)) \geq 0. \tag{2.24c}$$

---

Here, $d \in \mathbb{R}$ is a slack variable that is introduced to relax the CLF constraint, giving preference to safety over stability in case of conflict. Similar to the previous CBF-QP and CLF-QP, this optimization program requires perfect knowledge of the dynamics of (2.2), since the Lie derivatives of $B$ and $V$ with respect to $f$ and $g$ appear in the constraints.

### 2.2.5 Model Uncertainty

So far, all of the presented controllers are designed using a particular mathematical model of the dynamics of the system we want to control. However, perfectly modelling complex real-world systems is in general not possible. Since what we ultimately want to study is the evolution of the real system, not of the approximate mathematical model that we use, it is important to explicitly account for this gap.

Because of the reasons explained above, in this thesis we never assume we know the dynamics of the true plant (2.2). Instead, for many of the methods that we introduce, we exploit approximate model knowledge through a *nominal model* of the system's dynamics:

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u, \tag{2.25}$$

where we also assume that $\tilde{f}$ and $\tilde{g}$ are Lipschitz continuous vector fields. As explained above, the nominal model vector fields ($\tilde{f}$, $\tilde{g}$) do not perfectly match the true plant vector fields ($f$, $g$) due to the model mismatch.

Obviously, the problem of model uncertainty is especially relevant for model-based controllers, such as the ones that have just been introduced. The theoretical properties that these controllers hold for the model they are based on might not hold on the real system if the gap is not accounted for appropriately.

## 2.3 Data-driven Control

Data-driven controllers, on the other hand, get rid of the need for a mathematical model. By directly collecting data from the real system, these approaches *learn* how the system reacts to different control signals and decide which are the most desirable to apply based on this information.

Data-driven controllers can be classified into two main groups: those that follow the system identification paradigm, and those that are completely model-free. (1) The first group includes approaches that explicitly fit a dynamics model or, more generally, a dynamics-dependent function (such as the CLF or CBF derivative) from data, and then design a control law based on the identified model. Thus, once all the required model knowledge has been obtained from the collected data, for the control synthesis stage they can use one of the approaches that have been presented in the previous section. (2) The second group includes all of the model-free reinforcement learning algorithms that have been recently proposed and are achieving very promising results. By using the data directly for policy optimization, these later approaches have a simpler decision pipeline, typically with fewer components

than alternative approaches. However, these methods in general require very large amounts of data to obtain well-behaved control policies.

### 2.3.1 System Identification through Gaussian Process Regression

We now present background on the use of Gaussian Process (GP) regression to fit an unknown function from data. The main role of GP regression in this thesis is to fit functions that explicitly depend on the dynamics of the system (2.2), when we do not have perfect knowledge of $f$ and $g$.

Gaussian Process regression is a powerful method for conducting nonparametric Bayesian inference on an unknown function $h : \bar{\mathcal{X}} \to \mathbb{R}$, where $\bar{\mathcal{X}}$ is the domain of $h$. Essentially, a Gaussian Process can be understood as the extension of the multivariate Gaussian distribution to the infinite-dimensional space of functions.

More formally, a Gaussian Process (GP) is a random process for which any finite collection of samples have a joint Gaussian distribution. A GP is fully characterized by its mean $q : \bar{\mathcal{X}} \to \mathbb{R}$ and covariance (or kernel) $k : \bar{\mathcal{X}} \times \bar{\mathcal{X}} \to \mathbb{R}$ functions. In GP regression, the unknown function $h$ is assumed to be a sample from a GP, and through a set of $N$ noisy measurements $\mathbb{D}_N = \{x_j, h(x_j) + \epsilon_j\}_{j=1}^N$ at points $x_j \in \bar{\mathcal{X}}$, a prediction of $h(\cdot)$ at an unseen query point $x_* \in \bar{\mathcal{X}}$ can be derived from the joint distribution of $[h(x_1), \cdots, h(x_N), h(x_*)]^T$ conditioned on the dataset $\mathbb{D}_N$. Throughout this thesis, $\epsilon_j \sim \mathcal{N}(0, \sigma_n^2)$ is white measurement noise, with variance $\sigma_n^2 > 0$.

Setting the mean function of the prior GP $q$ to zero, the mean and variance of the prediction of $h(x_*)$ given the dataset $\mathbb{D}_N$ are:

$$\mu(x_*|\mathbb{D}_N) = \mathbf{z}^T (K + \sigma_n^2 I)^{-1} K_*^T, \tag{2.26}$$

$$\sigma^2(x_*|\mathbb{D}_N) = k(x_*, x_*) - K_*(K + \sigma_n^2 I)^{-1} K_*^T, \tag{2.27}$$

where $K \in \mathbb{R}^{N \times N}$ is the GP Kernel matrix, whose $(i,j)^{th}$ element is $k(x_i, x_j)$, $K_* = [k(x_*, x_1), \cdots, k(x_*, x_N)] \in \mathbb{R}^N$, and $\mathbf{z} \in \mathbb{R}^N$ is the vector containing the noisy measurements of $h$, $z_j := h(x_j) + \epsilon_j$.

Equations (2.26) and (2.27) provide, respectively, the mean estimate of the value of the unknown function $h$ at a query point $x_*$ and a measure of the uncertainty of this prediction. This later uncertainty estimate constitutes an extremely valuable measure of confidence on the model's prediction, and it is possible to obtain thanks to the nonparametric nature of GPs. The trade-off for this nonparametric adaptability is substantial computational demands, as the required matrix inversion expense escalates approximately cubically with the number of measurements $N$.

### 2.3.2 Model-Free Reinforcement Learning

Reinforcement learning (RL) focuses on training algorithms to make decisions based on the concept of learning from interaction with an environment. The objective of the RL agent

is to learn an optimal policy that dictates the best action to take in each state in order to maximize the expected cumulative reward over time.

Note that RL is essentially trying to solve the same problem as in Section 2.2.1. However, the model-based approaches presented in that Section rely on an accurate model of the system dynamics and may not be tractable for high-dimensional, nonlinear, or uncertain systems. RL offers an alternative, data-driven approach for approximating the solution to optimal control problems, even when the system model is not known a priori.

One of the main components of RL is the agent-environment interaction, which is typically modeled as a Markov Decision Process (MDP). An MDP consists of a finite set of states, actions, and rewards, and is characterized by the state-transition probability function and the reward function. More formally, an MDP is a tuple $(S, A, P, r, \gamma)$, where:

- $S$ is a finite set of states.
- $A$ is a finite set of actions.
- $P : S \times A \times S \to [0, 1]$ is the state-transition probability function, such that $P(x_{k+1}|x_k, u_k)$ denotes the probability of transitioning to state $x_{k+1}$ when taking action $u_k$ in state $x_k$.
- $r : S \times A \to \mathbb{R}$ is the reward function, with $r(x_k, u_k)$ denoting the expected immediate reward for taking action $u_k$ at state $x_k$.
- $\gamma \in [0, 1]$ is the discount factor, representing the preference for immediate rewards over delayed rewards.

Note that this problem setting is, in fact, closely related to the one of Section 2.2.1, in which we defined the cumulative cost (2.5) and the optimal control problem (2.6) for a deterministic discrete-time dynamic system (2.3). In fact, we can construct an equivalent Markov Decision Process (MDP) with the following components:

- State space: $S = \mathcal{X}$.
- Action space: $A = \mathcal{U}$.
- State-transition probability function: $P(x_{k+1}|x_k, u_k) = \delta\left(x_{k+1} - F(x_k, u_k)\right)$, where $\delta(\cdot)$ is the Dirac delta function.
- Reward function: $r(x_k, u_k) = -\ell(x_k, u_k)$.
- Discount factor: $\gamma \in [0, 1]$.

To learn an approximate solution to the optimal control problem in a data-driven manner, we can employ RL algorithms that approximate the optimal policy and/or the value function. Highly expressive function approximators, such as neural networks, are typically used for

this purpose. While this approach enables these methods to optimize high-dimensional policies, they are data-hungry, can display high-variance and thus frequently return highly sub-optimal policies when data is limited. This is, in part, due to the fact that these approaches do not exploit any structural knowledge of the system. Part I of this thesis will present principled methods that exploit structural model knowledge when formulating reinforcement learning problems, resulting in improved sample efficiency.

# Part I

# Model-based Supervision for Safe Robotic Learning

# Chapter 3

# Learning Min-norm Stabilizing Controllers

*This chapter is based on the paper titled "Learning Min-Norm Stabilizing Control Laws for Systems with Unknown Dynamics" [210], co-authored by Tyler Westenbroek, Ayush Agrawal, S. Shankar Sastry and Koushil Sreenath.*

This chapter introduces a framework for learning a minimum-norm stabilizing controller for a system with unknown dynamics using model-free reinforcement learning algorithms. The approach begins by first designing a Control Lyapunov Function (CLF) for a (possibly inaccurate) dynamics model for the system. Treating the CLF condition as a constraint on the desired closed-loop behavior of the real-world system, we use penalty methods to formulate an unconstrained optimization problem over the parameters of a learned controller, which can be solved using model-free policy optimization algorithms using data collected from the plant. We discuss when the optimization learns a stabilizing controller for the real-world system and derive conditions on the structure of the learned controller which ensure that the optimization is strongly convex, meaning the globally optimal solution can be found reliably. We validate the approach in simulation, first for a double pendulum, and then generalize the framework to learn stable walking controllers for underactuated bipedal robots using the Hybrid Zero Dynamics framework. By encoding a large amount of structure into the learning problem, we are able to learn stabilizing controllers for both systems with only minutes or even seconds of training data.

## 3.1 Introduction

Recently, the literature has displayed a renewed interest in data-driven methods for controller design [186, 19, 135, 3]. Much of this excitement has been driven by recent advances in the model-free reinforcement learning literature [86, 119]. Despite their generality, model-free policy optimization methods are known to suffer from poor sample complexity, as they

generally are unable to take advantage of known structure in the control system. This chapter bridges the gap between model-based and model-free design paradigms by embedding Lyapunov-based design techniques into a model-free reinforcement learning problem. By encoding basic information about the structure of the system into the learning problem through a Control Lyapunoc Function (CLF), our approach is able to learn optimal stabilizing controllers for highly uncertain systems with as little as seconds or a few minutes of data.

Specifically, we propose a framework for learning a min-norm stabilizing control law for an unknown system using model-free policy optimization techniques. Our approach begins by first designing a CLF for a nominal dynamics model of the system. We then formulate a continuous-time optimization problem over the parameters of a learned controller which treats the CLF energy dissipation condition as a constraint. The cost function for the optimization encourages choices of parameters which minimize control effort, but uses a penalty term to ensure that the CLF dissipation constraint is satisfied, if possible, when the penalty term is chosen to be large enough. The terms in the optimization depend on the dynamics of the unknown system, but discrete-time approximations to the problem can be solved using policy-optimization algorithms and data collected from the plant. In general, the problem may be non-convex, but when the learned controller is linear in its parameters the problem becomes (strongly) convex, meaning the globally optimal solutions for the problem can be found using standard policy gradient [183] or random search techniques [134].

To demonstrate the utility of the proposed framework, we apply the method in simulation to a double pendulum and a high-dimensional model of a bipedal robot. For the double pendulum example, the learned controller is comprised of a linear combination of radial basis functions so that the convexity result discussed above applies, and we demonstrate empirically that the learned controller is able to closely match the true min-norm controller performance. The walking example demonstrates how to extend our results in the body of the chapter to encompass the Hybrid Zero Dynamics framework as in [9]. For this high-dimensional system, a feed-forward neural network is used for the learned controller. While we cannot guarantee that the optimal set of parameters is found, the learned controller still produces a stable walking motion in the face of high model uncertainty.

### 3.1.1 Related Work

CLF-based controllers [11, 177] have been proved to be effective for a wide variety of complex robotic tasks, such as bipedal walking [65],[9], manipulation [7] and multi-agent coordination [150]. In [65] and [7] quadratic programs (CLF-QP), which integrate the CLF condition as a constraint, are used to get optimal min-norm stabilizing controllers. The CLF-QP is solved online and additional constraints, such as input saturation, can be added.

However, the dynamics of many real-world systems have nonlinearities that might be difficult to model correctly and/or physical parameters which could be difficult to identify. Input-to-state stability has been used to tackle this problem in [82],[178]. Also, adaptive [145] and robust [14, 146] versions of CLF-based controllers have been developed in recent years.

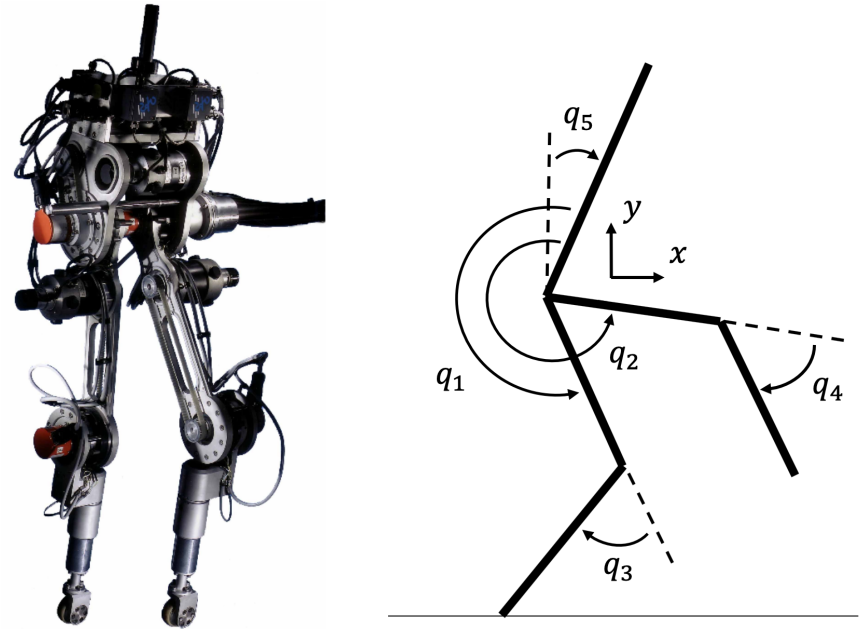



Figure 3.1: RABBIT, a planar five-link bipedal robot with nonlinear, hybrid and underactuated dynamics. $q_1$, $q_2$ are the relative stance and swing leg thigh angles referenced to the torso, $q_3$, $q_4$ are the relative stance and swing leg knee angles, $q_5$ is the absolute torso angle in the world frame, and $x$ and $y$ are the position of the hip in the world frame. Here $q = [x, y, q_1, q_2, q_3, q_4, q_5]^\top$.

However, these approaches sometimes fail to account for the correct amount of uncertainty due to the typical assumptions they make on the uncertainties' structures and bounds.

Our work most closely aligns with recent research that use data-driven approaches to tackle the issue of model uncertainty in nonlinear controllers. Our works builds on [209, 28], where reinforcement learning is used to account for uncertainty when performing feedback linearization of nonlinear systems. In contrast to recently proposed approaches [186, 38] which focus on learning the uncertain terms in a CLF-QP in order to indirectly improve the optimization-based controller, the framework proposed in this chapter directly learns the optimal stabilizing controller. By directly learning the desired controller, our approach removes the need for solving a real-time optimization problem involving a potentially complex learned component, which may take a non-trivial amount of time to process during real-time applications. On hardware, CLF-based controllers frequently need to be updated at frequencies exceeding 1000 Hertz to maintain the stability of the system [9], placing strict timing requirements on the rate at which the CLF-based controller must be updated. Thus, the direct approach has meaningful advantages in applications.

## 3.2 Learning Min-norm Stabilizing Controllers

### 3.2.1 Learning a Min-norm Stabilizing Controller for a System with Unknown Dynamics

Despite the wide-spread utility of the CLF-based controllers introduced in Section 2.2.2, the primary drawback of these methods is that they require an accurate dynamics model to implement. Our present objective is to learn a min-norm stabilizing controller for the plant with unknown control-affine dynamics (2.2) while ensuring that the learned controller adheres to the dissipation constraint imposed by some candidate CLF $V\colon \mathcal{X} \to \mathbb{R}_+$. In this chapter, we will consider systems of the form (2.2) without input constraints, i.e., $\mathcal{U} = \mathbb{R}^m$.

We will focus on learning the min-norm controller for the plant on a compact subset of the state-space. Specifically, we will focus on learning the min-norm stabilizing controller for the system on the set

$$\Omega_c \coloneqq \{x \in \mathcal{X} \colon V(x) \leq c\}, \tag{3.1}$$

where $c > 0$ is a design parameter.

We will make the following technical assumptions throughout the chapter unless otherwise specified:

**Assumption 3.1.** *The components $f$, $g$ and $\nabla V$ are each locally Lipschitz continuous.*

**Assumption 3.2.** *There exists a locally Lipschitz continuous control law $\tilde{\pi}\colon \mathcal{X} \to \mathbb{R}^m$ such that for each $x \in \Omega_c$*

$$\nabla V(x)[f(x) + g(x)\tilde{\pi}(x)] \leq -\lambda V(x). \tag{3.2}$$

**Remark 3.1.** *Assumption 3.2 ensures that $V$ is a true exponential CLF for the system with associated dissipation rate $\lambda$. While our approach does not explicity require a nominal dynamics model for the plant, in practice, our candidate CLF for the plant is constructed using a nominal dynamics model of the form* (2.25) *which incorporates any information we have about the plant, but may be inaccurate due to nonlinearities which are difficult to model or dynamics parameters which are challenging to identify. However, despite model mismatch between* (2.2) *and* (2.25)*, we can often design a CLF for the model and reasonably expect it to also be a CLF for the plant. For example, our two numerical examples systematically construct CLF's for the unknown system using feedback-linearizing coordinates. In these examples Assumption 3.2 is tantamount to knowing the relative degree of the system, a rather mild structural assumption.*

Since we do not know the terms in (2.2), we now propose a method to learn a stabilizing CLF-based controller for the system using data collected from the plant. Under the preceding assumptions, and recalling the discussion from the background Section 2.2.2, we know that

there is a well-defined control law $\pi_{\text{CLF}}\colon \Omega_c \to \mathbb{R}_+$ which exponentially stabilizes the plant on $\Omega_c$ and is given point-wise by

$$\pi_{\text{CLF}}(x) = \arg\min_{u\in\mathbb{R}^m} \|u\|_2^2 \quad \text{s.t. } \nabla V(x)[f(x) + g_p(x)u] \leq -\lambda V(x).$$

We will denote our learned approximation for $\pi_{\text{CLF}}$ by $\hat{\pi}\colon \mathcal{X}\times\Theta \to \mathbb{R}^m$. For each choice of parameter $\theta \in \Theta \subset \mathbb{R}^K$ the control law $\hat{\pi}(\cdot,\theta)\colon \mathcal{X} \to \mathbb{R}^m$ defines the learned control law supplied to the plant, with $\Theta \subset \mathbb{R}^K$ a convex set of allowable learned parameters. It is assumed that $\hat{\pi}$ is locally Lipschitz continuous in its first argument and continuously differentiable in its second argument. Common function approximators such as feed-forward neural networks, radial basis functions or bases of polynomials can be used to construct the learned controller.

**Remark 3.2.** *In general, the learned controller can incorporate information from a nominal dynamics model by giving it the structure*

$$\hat{\pi}(x,\theta) = \pi_m(x) + \delta\pi(x,\theta), \tag{3.3}$$

*where $\pi_m$ is a nominal model-based controller and $\delta\pi\colon \mathcal{X}\times\mathbb{R}^K \to \mathbb{R}^m$ is the learned component.*

Next, in order to find parameters for the learned controller which satisfy the dissipation constraint (3.2), we will solve optimizations over the parameters of the learned controller of the form

$$(\mathbf{P}_\rho) : \min_{\theta\in\Theta} L_\rho(\theta), \tag{3.4}$$

where for each $\rho \in \mathbb{R}_+$ we define the loss function

$$L_\rho(\theta) = E_{x\sim X}\left[\|\pi(x,\theta)\|_2^2 + \rho\left[\Psi(x,\theta)\right]^+\right], \tag{3.5}$$

where $X$ is the uniform probability distribution over $\Omega_c$, the mapping $\Psi\colon \mathbb{R}^n\times\Theta\to\mathbb{R}$ is defined by

$$\Psi(x,\theta) = \nabla V(x)[f(x) + g_p(x)\hat{\pi}(x,\theta)] + \lambda V(x) \tag{3.6}$$

and finally $[\cdot]^+$ is defined for each $y\in\mathbb{R}$ by

$$[y]^+ = \begin{cases} y & \text{if } y \geq 0, \\ 0 & \text{if } y < 0. \end{cases} \tag{3.7}$$

The first term in the loss $L_\rho$ encourages small control efforts while the second term penalizes violations of the CLF dissipation constraint, with $\rho \in \mathbb{R}_+$ used to control the magnitude of the penalty. While we do not know $\Psi(x,\theta)$ *a priori*, we can measure this quantity by applying the control $\hat{\pi}(x,\theta)$ to the plant at the point $x$ and measuring the resulting time

derivative of $V$. Then, equation (3.6) can be used to compute the desired quantity. Thus, any stochastic optimization algorithm can be used to solve $\mathbf{P}_\rho$ by running experiments to evaluate the terms in $L_\rho$. We will discuss this in further detail when we present practical approaches for solving $\mathbf{P}_\rho$ below.

**Remark 3.3.** *The uniformity of the distribution $X$ ensures that all points in $\Omega_c$ are considered when optimizing over the parameters of $\hat{\pi}$. This requirement is seen to be analogous to the persistency of excitation conditions which are common in the adaptive control literature [171]. In the proofs of the following theoretical results this condition ensures that each component of the learned controller is activated sufficiently during the learning process.*

### 3.2.2 Theoretical Results

We now study how the solution set of $\mathbf{P}_\rho$ changes as the penalty parameter $\rho$ is increased and derive conditions under which the problem is convex, meaning that it can be solved reliably to global optimality using iterative gradient-based optimization algorithms. To simplify the statement of our results, for each $\rho \in \mathbb{R}_+$ we define

$$S_\rho \coloneqq \left\{ \theta \in \Theta \colon \theta \in \arg\min_{\theta \in \Theta} L_\rho(\theta) \right\} \tag{3.8}$$

to capture the set of global minimizers for $\mathbf{P}_\rho$. We also define

$$\Xi \coloneqq \{\theta \in \Theta \colon \Psi(x, \theta) \leq 0,\ \forall x \in \Omega_c\} \subset \Theta \tag{3.9}$$

to be the set of parameters for which the corresponding learned controller satisfies the desired CLF dissipation constraint at every point in $\Omega_c$. Next, we present our theoretical results in Lemma 3.1 and Theorems 3.1 and 3.2, whose proofs can be found in Appendix A.1.

First, we compare the sets $\Xi$ and $S_\rho$ as the penalty term $\rho$ is increased:

**Lemma 3.1.** *Assume that $\Xi$ is non-empty so that there exists at least one choice of learned parameters which satisfy the desired CLF constraint. Then there exists $\bar{\rho} \in \mathbb{R}_+$ such that for each $\rho > \bar{\rho}$ all global optimizers of $\boldsymbol{P}_\rho$ also satisfy the dissipation constraint, namely, $S_\rho \subset \Xi$.*

In other words, if the penalty parameter $\rho \in \mathbb{R}_+$ is chosen to be large enough then $\mathbf{P}_\rho$ recovers the set of learned parameters which stabilize the plant and satisfy the CLF constraint. Note that if $\theta^* \in \Xi$ is one such choice of parameters then it must be the case that $\mathbb{E}_{x \sim X}\left[\rho\left[\Psi(x, \theta^*)\right]^+\right] = 0$. Thus, when $\Xi$ is non-empty and $\rho$ is chosen to be large enough the minimizers of $\mathbf{P}_\rho$ are selected by the set of parameters which minimize the term $\mathbb{E}_{x \sim X}\left[\|\hat{\pi}(x, \theta)\|_2^2\right]$, which is the average control effort exerted over the state-space by the corresponding learned controller. By definition, the min-norm stabilizing controller $\pi_{\text{CLF}}$ minimizes the control effort needed to satisfy the CLF dissipation constraint at every point in the state-space. Thus, if $\rho$ is large enough and $\pi_{\text{CLF}}$ is in the space of learned controllers spanned by $\hat{\pi}$, it must be recovered by the optimization:

**Theorem 3.1.** *Assume that there exists $\bar{\theta} \in \Theta$ such that $\hat{\pi}(x, \bar{\theta}) = \pi_{CLF}(x)$ for each $x \in \Omega_c$. Then there exists $\bar{\rho} \in \mathbb{R}_+$ such that for each $\rho > \bar{\rho}$ and $\theta^* \in S_\rho$ we have $\hat{\pi}(x, \theta^*) = \pi_{CLF}(x)$ for each $x \in \Omega_c$.*

However, the family of optimization problems we have formulated over the parameters of the learned controller will generally be non-convex, meaning that we cannot efficiently find their globally optimal solutions. Thus, we seek conditions under which $\mathbf{P}_\rho$ becomes convex so that we can reliably find its global minimizers using iterative methods. Towards this end we will now assume that

$$\hat{\pi}(x, \theta) = \sum_{k=1}^{K} \theta_k \pi_k(x), \tag{3.10}$$

where $\{\pi_k\}$ is a set of locally Lipschitz continuous mappings from $\mathcal{X}$ to $\mathbb{R}^m$ and $\theta_k$ is the $k$-th entry of the learned parameter[1]. Linearity assumptions of this sort are common in convergence proofs found in both the adaptive control and reinforcement learning literature. In the statement of the following result we informally view each basis element $\pi_k$ as a subset of $C(\Omega_c, \mathbb{R}^m)$, the vector space of continuous functions from $\Omega_c$ to $\mathbb{R}^m$.

**Lemma 3.2.** *Assume that $\hat{\pi}$ is of the form (3.10) and that the set $\{\pi_k\}_{k=1}^K$ is linearly independent on $C(\Omega_c, \mathbb{R}^m)$. Then for each $\rho \in \mathbb{R}_+$ the loss $L_\rho$ is strongly convex.*

This leads to the main theoretical result of this chapter, which follows from an immediate application of Theorem 3.1 and Lemma 3.2.

**Theorem 3.2.** *Assume that $\hat{\pi}$ is of the form (3.10) and that the set $\{\pi_k\}_{k=1}^K$ is linearly independent on $C(\Omega_c, \mathbb{R}^m)$. Further assume that $\Theta \subset \mathbb{R}^K$ is convex and that there exists $\theta^* \in \Theta$ such that $\hat{\pi}(x, \theta^*) = \pi_{CLF}(x)$ for each $x \in \Omega_c$. Then there exists $\bar{\rho} \in \mathbb{R}_+$ such that for each $\rho > \bar{\rho}$ the problem $\boldsymbol{P}_\rho$ is a strongly convex optimization problem with $\theta^*$ its unique global minimizer.*

In practice, since we do not have a parametric model for the system, it is unlikely that there exists a set of learned parameters which exactly reconstructs the true min-norm controlller for the plant. However, many model-free learning schemes [121] make use of function approximation schemes which can recover a continuous function up to a desired level of accuracy, if enough terms are included in the basis of features. However, in practice the number of elements required in such an expansion can quickly become prohibitively large as the dimension of the state grows. Thus, for high dimensional systems, such as the bipedal robots we consider below, practical implementations may require the use of more compactly represented function approximation schemes, such as multi-layer feed-forward neural networks,

[1] Alternatively, one could also assume that the learned controller is of the form $\hat{\pi} = \pi_m(x) + \sum_{k=1}^K \theta_k \pi_k(x)$ if the system designer wishes to augment a known model-based controller as in (3.3). The statement and proof of Theorem 3.2 go through with minor modifications in this case.

which can also approximate continuous functions to a desired degree of accuracy (Universal Approximation Theorem [49, 85]), but lead to non-convexities in our optimization problem (3.10).

### 3.2.3 Solving Discrete-time Approximations with Reinforcement Learning

Many real-world systems have digital sensors and actuators which can only be updated at some maximum frequency, meaning we can only obtain finite difference approximations of $\dot{V}$ when different control signals are applied to the plant. Thus, in this section we introduce discrete-time approximations to $\mathbf{P}_\rho$ and discuss how they can be solved with standard reinforcement learning algorithms [173, 78]. Our description of this process will be brief, since the approach is similar to the one described in [209].

For the reinforcement learning problem we will assume that the control supplied to the plant can only be updated at a fixed minimum sampling period $\Delta t > 0$. We will let $t_k = k \times \Delta t$ for each $k \in \mathbb{N}$ denote the set of sampling intervals. When the control $\hat{\pi}(x, \theta) \in \mathbb{R}^m$ is applied over the interval $[t_k, t_{k+1}]$ a Taylor expansion can be used to show that

$$\Psi(x, \theta) = \underbrace{\frac{V(x(t_{k+1})) - V(x(t_k))}{\Delta t} + \lambda V(x(t_k))}_{:=\tilde{\Psi}(x,\theta)} + O(\Delta t^2). \tag{3.11}$$

Thus for small $\Delta t$ we use the loss

$$\tilde{l}_\rho(x, \theta) = \|\hat{\pi}(x, \theta)\|_2^2 + \rho \left[\tilde{\Psi}(x, \theta)\right]^+. \tag{3.12}$$

We use this approximate pointwise loss to define the following reinforcement learning problem, which serves as an approximation to $\mathbf{P}_\rho$:

$$\begin{aligned} \tilde{\mathbf{P}}_\rho \colon \min_{\theta \in \Theta} \; & E_{x_0 \sim X} \left[ \sum_{k=0}^{N} \tilde{l}(x_k, \theta) \right] \\ \text{s.t. } & x_{k+1} = x_k + \int_{t_k}^{t_{k+1}} [f(x(t)) + g(x(t))u_k] \, dt \\ & u_k = \hat{\pi}(x_k, \theta). \end{aligned} \tag{3.13}$$

Here, the curve $x \colon \mathbb{R} \to \mathcal{X}$ is the trajectory of the plant starting from initial condition $x(0) = x_0$, and $N \in \mathbb{N}$ is the number of time steps in each rollout. Probing noise can be added to the input to encourage exploration, e.g, by instead setting $u_k = \hat{\pi}(x_k, \theta) + w_k$, where $w_k \sim \mathcal{N}(0, \sigma_w^2 I)$ is zero mean random noise. The policy optimization problem (3.13) is in a standard form for reinforcement learning problems [183], and in the following section we demonstrate how these discrete-time approximations can be used to successfully learn stabilizing controllers for unknown systems.

## 3.3 Examples

### 3.3.1 Double Pendulum

As depicted in Figure 3.3a) the system is parameterized by the masses of the two arms $m_1, m_2$ as well as their lengths, $l_1, l_2 \in \mathbb{R}$. For the purposes of simulation, we set $m_1 = m_2 = l_1 = l_2 = 1$. To set up the learning problem, we assume that we are given an inaccurate dynamics model with inaccurate estimates $\hat{m}_1, \hat{m}_2, \hat{l}_1, \hat{l}_2$. Specifically, we set $\hat{m}_1 = \hat{m}_2 = \hat{l}_1 = \hat{l}_2 = \frac{1}{2}$ so that each of the parameter estimates are half of their true value.

Using the input-output linearization design technique from [9], we design a CLF for the system of the form $V \colon \mathbb{R}^4 \to \mathbb{R}$, $V := x^T P x$, with

$$P = \begin{bmatrix} 1.5I & 0.5I \\ 0.5I & 0.5I \end{bmatrix}, \tag{3.14}$$

where $I$ is the $2 \times 2$ identity matrix. This can be shown to be a valid CLF for both the inaccurate dynamics model and the true plant. We focus on learning the min-norm controller for the plant on the set $\Omega_c = \{V(x) \leq c\}$ with the design parameter $c = 2$ and construct our learned controller by setting

$$\hat{\pi}(x, \theta) = \pi_m(x) + \delta\pi(x, \theta), \tag{3.15}$$

where $\pi_m$ is the min-norm CLF controller computed using the inaccurate dynamic parameters and the learned augmentation $\delta\pi$ is comprised of a linear combination of 500 radial basis functions so as to match the assumptions of Lemma 3.2.

We trained the learned component using a policy-gradient algorithm with action conditioned baselines [183]. Each training epoch consisted of 50 1-step roll-outs and a total of 500 epoch were used. The time-step for the simulator was 0.05 seconds. The performance of the ultimate learned controller is depicted in Figure 3.3, where we see that the learned controller closely matches the behavior of the true min-norm controller for the system. To further evaluate the performance of the learned controller, we randomly selected 1000 states $\{x_i\}_{i=1}^{1000}$ in $\Omega_c$ and calculated the ratio

$$R = \sum_{i=1}^{1000} \frac{\|\hat{\pi}(x_i, \theta^*) - \pi_{\text{CLF}}(x_i)\|_2}{\|\pi_{\text{CLF}}(x_i)\|_2}, \tag{3.16}$$

where $\pi_{\text{CLF}}$ is the true min-norm controller for the system and $\theta^*$ is the parameter selected by the training process. We calculated $R = 0.044$, indicating that the learned controller was able to closely match the performance of the true min-norm controller for the system. As depicted in Figure 3.2, the learning converges in about 200 iterations, which corresponds to about eight minutes of data. Our implementation of the learning algorithm for this problem was hand-coded, and we believe the sample efficiency for this problem could match that of the walking example below by improving the implementation.

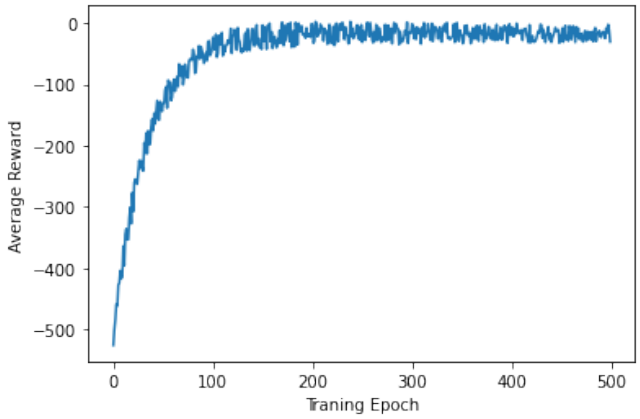



Figure 3.2: Learning curve for the double pendulum simulations using the proposed CLF-based reward function for reinforcement learning.

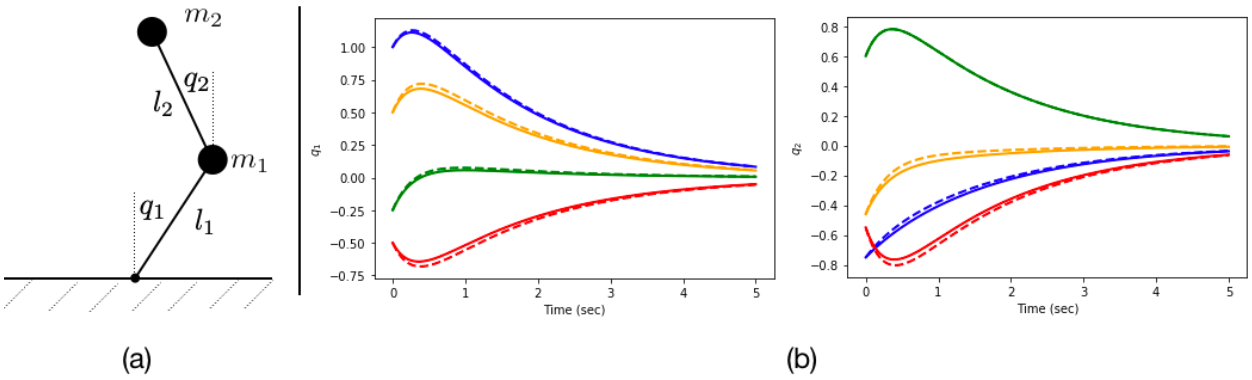



Figure 3.3: (a) Depiction of the double pendulum model with the states and physical parameters shown. (b) Trajectories corresponding to different initial conditions for the learned controller and true min-norm controller for the system. Each color represents trajectories starting from a specific initial condition. Solid lines denote the trajectories generated by the true min-norm controller for the system while the dashed lines correspond to the trajectories generated by the learned controller. Observe that the learned controller closely matches the desired closed-loop behavior. Note that the velocities of the trajectories are not depicted, which is why several of the plotted curves intersect.

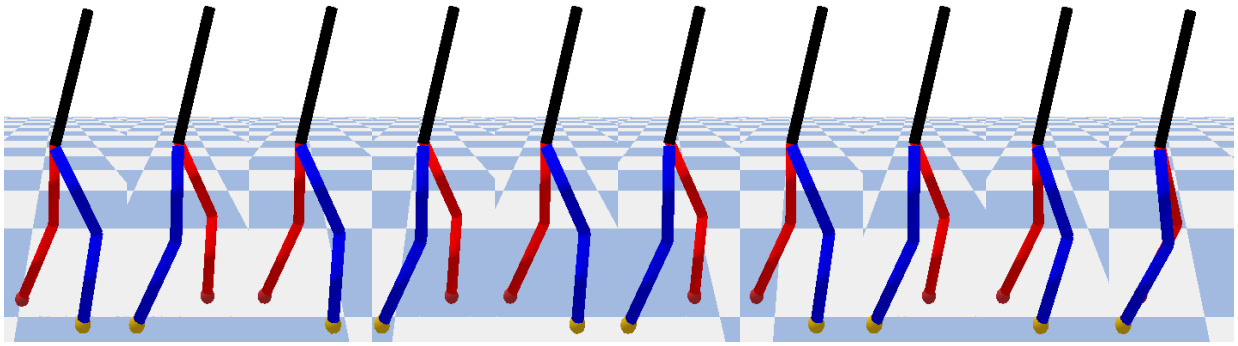


Figure 3.4: Snapshots of the walking gait with the min-norm CLF controller using the nominal (inaccurate) model. The robot falls after 10 walking steps.

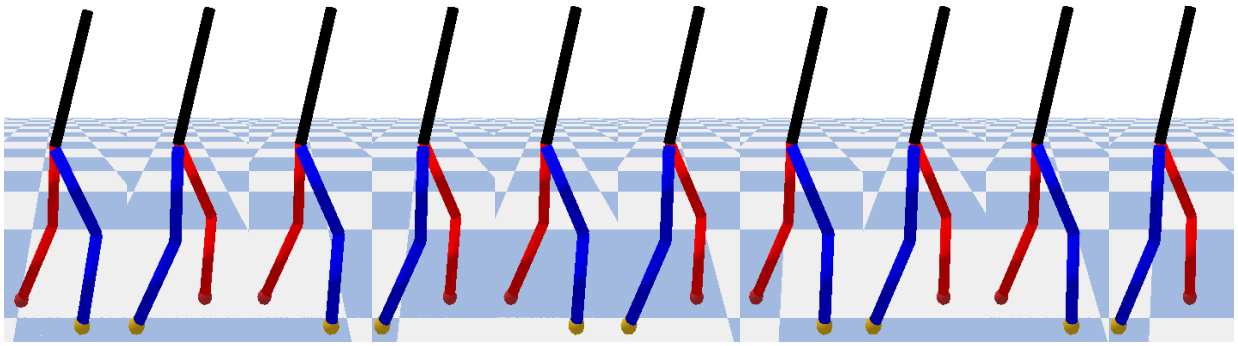


Figure 3.5: Snapshots of the walking gait with the learned controller. The robot walks indefinitely (the above figure shows the first ten walking steps).

### 3.3.2 Bipedal Walking

Next, we discuss how to apply our method to the Hybrid Zero Dynamics (HZD) framework using the CLF-based design approach proposed in [9] in order to learn an efficient, stable walking controller for a bipedal robot. We model the robot as a hybrid system with impulse effects as in [9],

$$\Sigma : \begin{cases} \dot{\eta} = f(\eta, z) + g(\eta, z)u, & \\ \dot{z} = h(\eta, z) & \text{when } (\eta, z) \notin \mathcal{S}, \\ \eta^+ = \Delta_X\left(\eta^-, z^-\right), & \\ z^+ = \Delta_Z\left(\eta^-, z^-\right) & \text{when } (\eta, z) \in \mathcal{S}, \end{cases} \tag{3.17}$$

where $\eta \in \mathcal{T} \subset \mathbb{R}^{n_a}$ represents the controlled (actuated) states, $z \in \mathcal{Z} \subset \mathbb{R}^{n_u}$ represents the uncontrolled states and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ represents the control inputs. The model assumes

alternating phases of single support, where one foot is off the ground (swing foot) and the other (stance foot) is assumed to remain at a fixed point without slipping. The impact between the swing foot and the ground is modelled as a rigid impact and occurs when $(\eta, z) \in \mathcal{S}$, where $\mathcal{S}$ is a smooth switching manifold. Here, $\eta^+ \in \mathcal{T}$ and $z^+ \in \mathcal{Z}$ represent the post-impact states while $\eta^- \in \mathcal{T}$ and $z^- \in \mathcal{Z}$ denote the pre-impact states.

Following the framework in [9], an input-output linearization based CLF is designed for the actuated coordinates during the continuous portion of the evolution of the state. Namely, we design a Lyapunov function $V\colon \mathbb{R}^{n_u} \to \mathbb{R}$ and dissipation rate $\lambda$ such that the following condition holds for each $(\eta, z) \in \mathcal{T} \times \mathcal{Z}$:

$$\inf_{u \in U} \nabla V(\eta)[f(\eta, z) + g(\eta, z)u] \leq -\lambda V(\eta). \tag{3.18}$$

Thus, the control objective is to drive only the actuated states to zero. As shown in [9], when the coordinates for the actuated and unactuated portions of the system are chosen correctly the condition $\eta \to 0$ corresponds to the robot converging to a periodic walking gait. The CLF and dissipation rate are designed so that the actuated coordinates are driven to zero fast enough to overcome shocks to the system introduced by the switching condition. We refer the readers to [9] for more details on this procedure.

To accommodate this new objective, our goal is to learn a control law $\hat{\pi}\colon \mathcal{T} \times \mathcal{Z} \times \Theta \to \mathcal{U}$ such that

$$\underbrace{\nabla V(\eta)[f(\eta, z) + g(\eta, z)\hat{\pi}(\eta, z, \theta)] + \lambda V(\eta)}_{:=\hat{\Psi}(\eta, z, \theta)} \leq 0 \tag{3.19}$$

for each $(\eta, z) \in \mathcal{T} \times \mathcal{Z}$ for our choice of learned parameters $\theta \in \Theta$. Here, $f$ and $g$ are the terms in true dynamics of the plant, which may differ from the nominal dynamics. To modify our approach to this new setting, for each $\rho \in \mathbb{R}_+$ we now define the loss

$$\hat{L}_\rho(\theta) = \mathbb{E}_{(\eta, z) \sim X} \left[ \|\hat{\pi}(\eta, z, \theta)\|_2^2 + \rho \left[ \hat{\Psi}(\eta, z, \theta) \right]^+ \right], \tag{3.20}$$

where $X$ is now the uniform distribution over $\mathcal{T} \times \mathcal{Z}$. Despite the fact that the CLF is defined only over the lower dimensional state $\eta$, the theoretical results from section 3.2.2 naturally extend to this case. Moreover, the techniques from section 3.2.3 can be used to find local minimizers of $\hat{L}_\rho$.

In particular, the proposed method is validated on a model for RABBIT (Fig. 3.1, [37]), an under-actuated five-link planar bipedal robot with seven degrees-of-freedom. Model uncertainty is introduced by scaling the mass of each of RABBIT's links by a factor of two, i.e., the real plant's masses are twice the nominal model's masses. Our learned controlled is

$$\hat{\pi}(\eta, z, \theta) = \pi_m(\eta, z) + \delta\pi(\eta, z, \theta), \tag{3.21}$$

where $\pi_m$ is the min-norm CLF controller obtained using the nominal model dynamics. The term $\delta\pi(\eta, z, \theta) \in \mathbb{R}^4$ takes the form of a Multi-Layer Perceptron (MLP) neural network

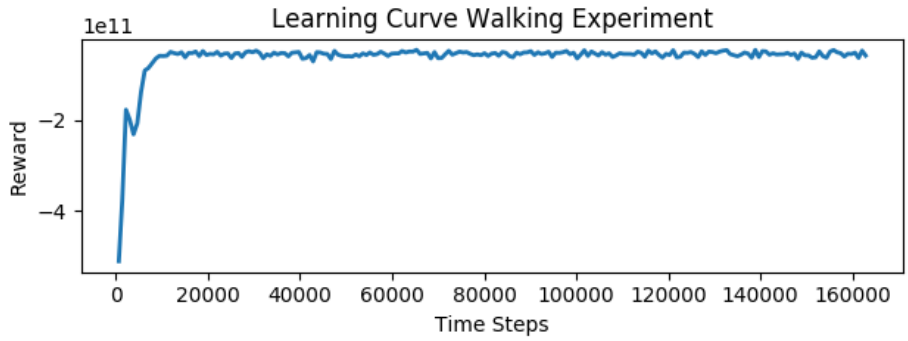



Figure 3.6: Learning curve of our learning approach for the RABBIT walking simulation.

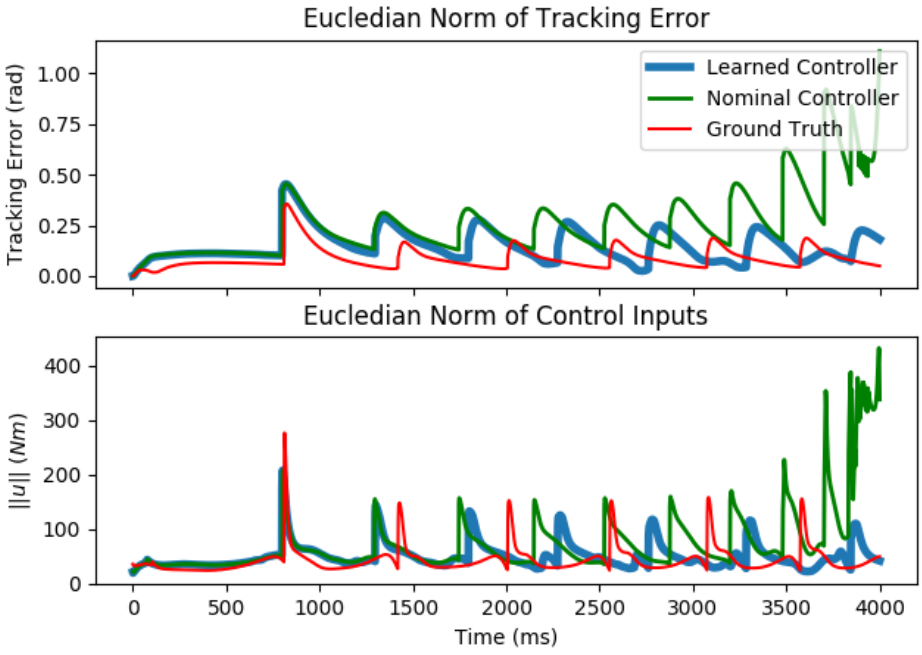



Figure 3.7: Tracking error (top) and norm of control inputs (bottom) of the learned min-norm controller (blue), the nominal controller (green) and the actual CLF-based controller of the plant computed using the true robot dynamics (red), each simulated for 4 seconds of walking.

with 2 hidden layers of width 64 each, tanh activation functions and layer normalization. We use the Soft Actor Critic algorithm [78], an off-policy method, for training the learned policy $\delta\pi(\eta, z, \theta)$. The training is done on episodes consisting of one walking step each. The simulations are conducted on the open-source physics simulator PyBullet [46] using a discrete time-step of one millisecond. As it can be seen in Figure 3.6, the training converges in about 20,000 time steps, which corresponds to roughly 50 steps of the biped and about 20 seconds of data collection from the system. Altogether, the simulations and training took about 10 minutes of computation using the six cores of an Intel(R) Core(TM) i7-8705G CPU (3.10GHz), without using a GPU.

Figure 3.7 shows a comparison between the proposed learned controller, the nominal controller $\pi_m$ and $\pi_{\text{CLF}}$, which is the CLF-based controller of the plant computed using the true (unknown) dynamics. This figure shows that while the nominal controller fails after ten walking steps making the robot fall, the learned controller achieves stable walking for an indefinite number of steps and gives good tracking error performance. It is also important to notice that the learned controller achieves this while using similar magnitudes of control inputs as the nominal and the true CLF-based controllers. However, the tracking error performance is not as good as with the actual CLF-based controller of the plant, as expected, and is likely due to the fact that the learner has converged to a local minima. Additionally, we note that the walking speeds for the learned controller and the true min-norm CLF controller for the plant are different. Underactuated robots such as RABBIT may contain multiple periodic orbits on the surface $\{(\eta, z) \in \mathcal{T} \times \mathcal{Z} \colon \eta = 0\}$. Thus, while both controllers successfully drive the system to this set, the periodic orbits the two controllers converge to are different.

## 3.4 Chapter Summary

This chapter presents a novel framework for learning minimum-norm stabilizing controllers for systems with unknown dynamics using model-free reinforcement learning algorithms. By incorporating significant structure into the learning problem, the proposed method enables the learning of stabilizing controllers for complex systems with only minutes or seconds of training data, showcasing its potential for real-world applications.

To demonstrate the effectiveness of the proposed framework, we validated the approach through simulations of a double pendulum and extend it to learning stable walking controllers for underactuated bipedal robots using the Hybrid Zero Dynamics framework.

# Chapter 4

# Learning Min-norm Safe Stabilizing Controllers

*This chapter is based on the paper titled "Combining Model-Based Design and Model-Free Policy Optimization to Learn Safe, Stabilizing Controllers" [208], co-authored by Tyler Westenbroek, Ayush Agrawal, S. Shankar Sastry and Koushil Sreenath.*

Compared to the previous chapter, we now consider the problem of learning from data a policy that not only needs to stabilize the system but, most importantly, needs to restrict it from entering unsafe regions of the state-space. We therefore want to synthesize controllers that are both safe and stabilizing. To do this, we again formulate a model-free unconstrained policy optimization problem. However, this time, we need to incorporate both the stability and safety constraints into the objective function using penalty methods. We demonstrate that when the penalty terms are scaled correctly, the optimization prioritizes the maintenance of safety over stability, and stability over optimality.

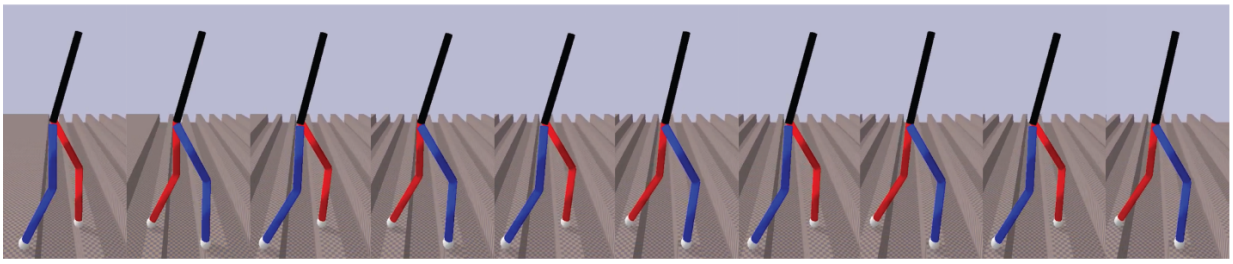


Figure 4.1: We apply our framework to rapidly learn a stable walking policy for the robot RABBIT [37] on a terrain on randomly-spaced stepping stones.

## 4.1 Introduction

Following recent empirical successes from the reinforcement learning (RL) literature [119], there has been a renewed interest in data-driven methods for controller design in the case of model uncertainty [19, 3]. However, despite the flexibility of model-free approaches, these methods are known to suffer from poor sample complexity since they do not take advantage of known structural properties of the control system. Moreover, the literature currently lacks constructive methods for designing learning problems which give the system designer fine-grained control over potentially competing global objectives, such as the rate of convergence to a desired operating point or the avoidance of an unsafe region of the state-space.

Fortunately, modern model-based control theory has developed many tools such as Control Lyapunov Functions (CLFs; [177]) and Control Barrier Functions (CBFs; [8]) which allow the system designer to constrain the pointwise closed-loop behavior of a given control system to ensure desired global properties (stability and safety, respectively) are achieved. When an accurate dynamics model is available, online optimization can be used to satisfy these pointwise constraints while minimizing a cost, such as control effort [8]. In effect, these approaches reduce the satisfaction of challenging global objectives to simple local decisions from the perspective of controller synthesis.

This chapter takes steps towards extending this design philosophy to the model-free setting by introducing a framework for systematically designing policy optimization problems over a parameterized learned controller which enforces a hierarchy of user-specified constraints on the closed-loop dynamics. To make the framework explicit, we focus on learning safe, stabilizing controllers using CLFs and CBFs and choose to prioritize safety over stability. We focus on the regime where the system designer has access to a dynamics model which may be highly inaccurate but is assumed to at least capture basic structural information about the real world plant. The model is used to construct a candidate CLF and CBF for the plant and a family of policy optimization problems are formulated which use penalty terms to discourage violations of the pointwise constraints imposed by these functions. This allows the system designer to carefully constrain the desired closed-loop behavior for the learned controller while also allowing for additional performance terms, such as minimizing control effort.

Our theoretical results demonstrate how to scale the penalty terms to control violations of the constraints and appropriately prioritize safety over stability and stability over performance. We first introduce the approach for classical control systems but then demonstrate how to extend the approach to the hybrid case via an application to a class of hybrid models which are frequently used in the dynamic walking literature [72]. We discuss how to synthesize numerical approximations to the family of learning problems which can be solved using standard machine learning techniques, including state of the art reinforcement learning algorithms. Simulation experiments are provided for both the continuous and hybrid cases, which demonstrate that our method is able to effectively learn safe, stabilizing controllers in the face of large amounts of dynamics uncertainty. We can reliably solve the policy optimization problems formulated over these systems using only a few minutes or even seconds

of simulated data, representing a sharp increase in the sample efficiency usually found in the reinforcement learning literature [86, 119]. We conjecture that this is due to the large amount of structure embedded in the learning problem through the incorporation of CLF and CBF constraints, which reduce the search for an optimal safe, stabilizing controller to a set of local criteria at each point in the state space.

### 4.1.1 Related Work

The unification of Control Barrier Functions and Control Lyapunov Functions to synthesize safe, stabilizing controllers was first proposed in [8] using online quadratic programming. In the case of model uncertainty, robust formulations have been proposed [147]. Learning based methods using supervised learning [187] or reinforcement learning [38] to learn the uncertain dynamics terms in the quadratic program have also been considered. These can be thought of as indirect learning methods, since they still require solving an optimization problem involving the learned components to calculate the desired controller. The primary downside of each of these approaches is that if the optimization is infeasible at a particular point then the control strategy will generally be undefined, which can be particularly difficult to rule out when learning unknown dynamics.

Building on the approach presented in the last chapter of this thesis, we now introduce a framework for directly learning a safe, stabilizing controller for the system using model-free policy optimization algorithms. By directly learning the desired controller, our approach removes the need for solving a real-time optimization problem involving a potentially complex learned component, which may take a non-trivial amount of time to process during real-time applications. At points where it is infeasible to satisfy the desired constraints, our method provides a "best effort" control strategy which satisfies the constraints to the greatest degree possible, bypassing issues of feasibility.

## 4.2 Learning Min-norm Safe Stabilizing Controllers

In this chapter, we consider a control-affine true plant of the form (2.2), without input constraints, i.e., $\mathcal{U} = \mathbb{R}^m$. Furthermore, we assume that the dynamics vector fields $f$ and $g$ are continuously differentiable.

As in the previous chapter, we take a nominal *dynamics model* (2.25) and design a CLF $V : \mathcal{X} \to \mathbb{R}_+$ based on it. Furthermore, since we now need to consider the problem of ensuring safety, we also use the nominal model to design a CBF $B : \mathcal{X} \to \mathbb{R}$.

**Assumption 4.1.** *We assume that the nominal dynamics model* (2.25) *has been used to synthesize a candidate exponential CLF $V$ (and rate $\lambda$) and CBF $B$ (and rate $\gamma$) for the unknown plant* (2.2).

Even though the dynamics of the plant are unknown, it is often reasonable to assume that the model captures enough basic structural information about the plant to guarantee

that these functions are also a valid CLF and CBF for the real-world system. For example, in our simulated applications we design the candidate CLF using feedback linearization, which is guaranteed to be a CLF for the true system as long as the relative degree of the plant matches that of the model, a relatively weak assumption.

As explained in Chapter 2, given a CLF and CBF, if the dynamics of the true system were known, it would be natural to search for a Lipschitz continuous control law which satisfies the pointwise constraints of the CLF (2.19) and CBF (2.21) simultaneously. One candidate control law $\pi_{\text{CBF-CLF}}$ is given by solving the pointwise quadratic program CBF-CLF-QP introduced in (2.24), which aims to minimize control effort while satisfying the two pointwise constraints.

As a reminder, note that even if $V$ and $B$ are an actual CLF and CBF for the system, it may be impossible to satisfy both constraints simultaneously leading to infeasibility issues. A common heuristic is to add slack terms to one or both of the constraints of the CBF-CLF-QP to ensure feasibility of the problem at the cost of some violation of the constraints [8].

While control laws similar to the CBF-CLF-QP of (2.24) have been successfully applied in a number of applications they have several practical limitations. Most importantly, these approaches require that an exact dynamics model is available to ensure that the pointwise constraints in (2.24) can be satisfied on the real-world system. Secondly, the infeasibility issues mentioned above mean that the controller may be undefined at certain points in the state-space, which can be highly problematic during real-time operation. This motivates the method detailed below, which uses the candidate CLF and CBF to learn an optimal safe, stabilizing controller for an uncertain system using data collected from the plant. The method prioritizes satisfaction of the CBF constraint over the CLF constraint and removes the need for real-time optimization.

As in the previous chapter, the learned controller $\hat{\pi}\colon \mathcal{X} \times \Theta \to \mathbb{R}^m$ is of the form

$$\hat{\pi}(x, \theta) = \pi_m(x) + \delta\pi(x, \theta). \tag{4.1}$$

Here, $\pi_m\colon \mathcal{X} \to \mathbb{R}^m$ is a nominal controller supplied by the system designer which is derived from the nominal dynamics model, and $\delta\pi\colon \mathcal{X} \times \Theta \to \mathbb{R}^m$ is a learned augmentation. The learned parameters $(\theta_1, \ldots, \theta_p) \in \Theta \subset \mathbb{R}^p$ are to be trained so as to select the optimal safe, stabilizing controller for the system.

**Assumption 4.2.** *The learned controller $\hat{\pi}\colon \mathcal{X} \times \theta \to \mathbb{R}^m$ is continuously differentiable in both of its arguments.*

**Assumption 4.3.** *The set of learned parameters $\Theta$ is a compact convex set.*

Our primary goal is to find a controller which satisfies the following infinite dimensional constraints, when possible:

$$\underbrace{-\nabla B(x)[f(x) + g(x)\hat{\pi}(x, \theta)] - \gamma(B(x))}_{\Psi_1(x,\theta)} \leq 0 \quad \forall x \in \mathcal{X}_{\text{safe}}, \tag{4.2}$$

$$\underbrace{\nabla V(x)[f(x) + g(x)\hat{\pi}(x,\theta)] + \lambda V(x)}_{\Psi_2(x,\theta)} \leq 0 \quad \forall x \in \mathcal{X}_{\text{safe}}. \tag{4.3}$$

Here, the set $\mathcal{X}_{\text{safe}} \subset \mathcal{X} \subseteq \mathbb{R}^n$ is the safe-set defined by the 0-super-level set of $B$. In words, we want to train a controller $\hat{\pi}(\cdot,\theta)\colon \mathcal{X} \to \mathbb{R}^m$ which satisfies the safety and stabilization constraints that the chosen CBF and CLF impose on the real-world system. We make the following assumption:

**Assumption 4.4.** *The safe set $\mathcal{X}_{safe}$ is compact.*

Since it may not be possible to learn a controller which satisfies both sets of constraints simultaneously, our learning framework must be flexible enough to prioritize the safety objective over the stabilization objective when necessary. While we do not know the terms in $\Psi_1(x,\theta)$ and $\Psi_2(x,\theta)$ since the dynamics of the plant are unknown, these terms can be calculated for different values of $x \in \mathcal{X}_{\text{safe}}$ and $\theta \in \Theta$ if measurements of $\dot{V}$ and $\dot{B}$ are available when collecting data from the plant.

In order to enforce these constraints while minimizing control effort, we will solve optimizations of the form

$$\mathbf{P}_{(\rho_1,\rho_2)}\colon \quad \min_{\theta\in\Theta} \mathbb{E}_{x\sim X} L_{(\rho_1,\rho_2)}(x,\theta),$$

where

$$L_{(\rho_1,\rho_2)}(x,\theta) = \|\hat{\pi}(x,\theta)\|_2^2 + \rho_1 \left[\Psi_1(x,\theta)\right]^+ + \rho_2 \left[\Psi_2(x,\theta)\right]^+,$$

the hinge map $[\cdot]^+$ is defined by $[y]^+ = \max\{0, y\}$ for each $y \in \mathbb{R}$, and the probability distribution $X\colon \mathcal{X}_{\text{safe}} \to [0,1]$ is supported on $\mathcal{X}_{\text{safe}}$. Here, $X$ is understood to be the distribution of states visited when collecting samples from the real world plant during the learning process, and $\rho_1, \rho_2 \geq 0$ are penalty parameters to be chosen later.

**Remark 4.1.** *The requirement that $X$ is supported on all of $\mathcal{X}_{safe}$ is analogous to the persistency of excitation conditions found in the adaptive control literature [170], and ensures that the data is "rich enough" so that the correct controller is learned. Note that under this assumption the penalty terms $\mathbb{E}_{x\sim X}\rho_1 \left[\Psi_1(x,\theta)\right]^+$ and $\mathbb{E}_{x\sim X}\rho_2 \left[\Psi_2(x,\theta)\right]^+$ are positive if and only if the safety and stability constraints are violated, respectively, at some point $x \in \mathcal{X}_{safe}$. Thus this richness requirement guarantees that violations of the pointwise constraints are appropriately penalized by the optimization. The theoretical guarantees we provide below are algorithm agnostic, and seek to characterize the global optimizers of the problem.*

## 4.3 Theoretical Analysis

We now demonstrate that violations of the safety and stability constraints can be decreased to a pre-specified tolerance by scaling the penalty terms appropriately. For simplicity, we assume there exists at least one set of parameters which satisfies the safety constraint:

**Assumption 4.5.** *There exists $\theta^* \in \Theta$ such that for each $x \in \mathcal{X}_{safe}$ we have $\Psi_1(x,\theta^*) \leq 0$.*

Next, we build up some additional notation to simplify the statement of our theoretical results. First, define the maps $M_u, M_1, M_2 \colon \Theta \to \mathbb{R}_{\geq 0}$ by

$$M_u(\theta) = \mathbb{E}_{x \sim X} \|\hat{\pi}(x, \theta)\|_2^2,$$
$$M_1(\theta) = \mathbb{E}_{x \sim X} \left[\Psi_1(x, \theta)\right]^+,$$
$$M_2(\theta) = \mathbb{E}_{x \sim X} \left[\Psi_2(x, \theta)\right]^+.$$

For each chosen parameter $\theta \in \Theta$, $M_u(\theta)$ captures total energy exerted by the corresponding controller across the safe set, $M_1(\theta)$ is the extent to which the CBF constraint is violated, and $M_2(\theta)$ is the extent to which the CLF constraint is violated. Next, for each $\epsilon_1 \geq 0$ define

$$\Theta_{\epsilon_1} = \{\theta \in \Theta \colon M_1(\theta) \leq \epsilon_1\},$$

which is the set of parameters for which the total violation of the CBF constraint is less than $\epsilon_1$. We also define

$$\tilde{M}_2 = \min_{\theta \in \Theta_0} M_2(\theta), \tag{4.4}$$

which is the smallest extent to which the CLF constraint can be violated, subject to exact satisfaction of the CBF constraint, and is the ideal amount of violation of the CLF constraint that can be returned by our optimization problem. We then define for each $\epsilon_1, \epsilon_2 \geq 0$

$$\Theta_{\epsilon_1, \epsilon_2} = \{\theta \in \Theta_{\epsilon_1} \colon M_2(\theta) \leq \tilde{M}_2 + \epsilon_2\},$$

which is the set of parameters corresponding to learned controllers which violate the CBF and CLF constraints no more than $\epsilon_1 \geq 0$ and $\epsilon_2 \geq 0$ more than their ideal values.

We now present our first result, whose proof can be found in Appendix A.2:

**Theorem 4.1.** *There exist constants, $C_1, C_2, C_3 \geq 0$ such that if $\rho_1 \geq \frac{C_1 \rho_2 + C_2}{\epsilon_1}$ and $\rho_2 \geq \frac{C_3}{\epsilon_2}$ then each global optimizer $\theta^*$ of $\boldsymbol{P}_{(\rho_1, \rho_2)}$ satisfies $\theta^* \in \Theta_{\epsilon_1, \epsilon_2}$.*

The result indicates that if we choose $\rho_2 \gg 0$ and $\rho_1 \gg \rho_2$ our optimization correctly enforces safety over stability, satisfying the two constraints to the desired tolerances. Within the set of desired controllers specified by $\Theta_{\epsilon_1, \epsilon_2}$, the optimization is then left to reduce the amount of control effort required to achieve these objectives. However, driving both tolerances to zero requires taking $\rho_1, \rho_2 \to \infty$.

One practical approach for ensuring exact satisfaction of the safety constraint for a finite value of the multipliers is to add a small amount of extra conservativeness to the pointwise CBF constraint. Specifically, letting $\Psi_1^\delta(\theta, x) = \Psi_1(x, \theta) + \delta$ for some small parameter $\delta > 0$, one can replace $\Psi_1(x, \theta)$ with $\Psi_1^\delta(x, \theta)$ in the loss $L_{(\rho_1, \rho_2)}(x, \theta)$. Due to the continuity of the problem data, driving $\mathbb{E}_{x \sim X} \left[\Psi_1^\delta(\theta, x)\right]^+$ to be sufficiently small (which can be done with finite values of $\rho_1$) will ensure exact satisfaction of the original CBF constraint.

However, the attractive properties mentioned above only apply to the global minimizers of $\mathbf{P}_{(\rho_1, \rho_2)}$, which in general will be non-convex, meaning that in practice only local minimizers to the problem can be found using common incremental machine learning algorithms.

Thus, we seek conditions on the structure of the learned controller which ensure that the optimization problem is convex. Specifically, we analyze the case where the learned portion of the controller is of the form

$$\delta\pi(x,\theta) = \sum_{k=1}^{p} \theta_k \pi_k(x), \tag{4.5}$$

where $\{\pi_k\}_{k=1}^p$ is a set of features.

**Theorem 4.2.** *Suppose that the learned augmentation in* (4.1) *is of the form* (4.5)*, and that the set* $\{u_k\}_{k=1}^p$ *is linearly independent. Then* $\boldsymbol{P}_{(\rho_1,\rho_2)}$ *is strongly convex.*

We omit the proof of Theorem 4.2, as it closely follows the steps in the proof of Lemma 3.2 that can be found in Appendix A.1.

Many well-known bases such as radial basis functions [168] or polynomials can be used to recover any continuous function up to a desired degree of accuracy by including enough terms in the expansion. It is an important matter for future work to include these methods in our framework, as it would enable users to design networks for the learned controller which are guaranteed to be able to satisfy the CLF and CBF constraints to a desired degree of accuracy. However, function approximation schemes of the form (4.5) may require a prohibitive number of bases elements to ensure that the desired function is accurately reconstructed in high dimensions. Thus, in practice, more compact function approximators such as feed-forward neural networks must be used in high dimensions. Unfortunately, such networks generally lead to non-convexities in $\mathbf{P}_{(\rho_1,\rho_2)}$.

## 4.4 Reinforcement Learning Implementation

In practice, our method uses finite difference approximations to $\dot{B}$ and $\dot{V}$ to compute the terms in $\Psi_1$ and $\Psi_2$, and then solves the resulting approximations to $\mathbf{P}_{(\rho_1,\rho_2)}$ using standard model-free reinforcement learning algorithms.

Specifically, we will assume that during the learning process the learned controller is sampled every $\Delta t > 0$ seconds, and will let $t_k = k\Delta t$ for $k \in \mathbb{N}$ denote the sampling instances. When the control $\hat{\pi}(x(t_k),\theta)$ is applied over the interval $[t_k, t_{k+1}]$ we have

$$\Psi_1(x(t_k)),\theta) = \underbrace{-\frac{B(x(t_{k+1})) - B(x(t_k))}{\Delta t} - \gamma(B(x(t_k)))}_{=:\tilde{\Psi}_1(x,\theta)} + O(\Delta t^2),$$

$$\Psi_2(x(t_k)),\theta) = \underbrace{\frac{V(x(t_{k+1})) - V(x(t_k))}{\Delta t} + \lambda V(x(t_k))}_{=:\tilde{\Psi}_2(x,\theta)} + O(\Delta t^2).$$

Thus, for small $\Delta t > 0$ we approximate $L^{(\rho_1,\rho_2)}$ with

$$\tilde{L}_{(\rho_1,\rho_2)}(x,\theta) = \|\hat{\pi}(x,\theta)\|_2^2 + \rho_1 \left[\tilde{\Psi}_1(x,\theta)\right]^+ + \rho_2 \left[\tilde{\Psi}_2(x,\theta)\right]^+$$

and define the following reinforcement learning problem:

$$\tilde{\mathbf{P}}_{(\rho_1,\rho_2)}\colon \min_{\theta\in\Theta} E_{x_0\sim X}\left[\sum_{k=0}^{N} \tilde{L}_{(\rho_1,\rho_2)}(x_k,\theta)\right] \quad \text{s.t. } x_{k+1} = x_k + \int_{t_k}^{t_{k+1}} \left[f(x(t)) + g(x(t))\hat{\pi}(x_k,\theta)\right] dt \tag{4.6}$$

Here, $N \in \{1, 2, \dots\}$ is the length of the rollout for each experiment on the plant. Note that, as the equivalent problem formulated in Chapter 3, this problem is in standard form for reinforcement learning, which can be solved using any off-the-shelf algorithm.

## 4.5 Examples

### 4.5.1 Double Pendulum With Safety Constraint

We first apply the learning framework to the double pendulum in Figure 4.2 with two degrees of freedom $q = (\theta_1, \theta_2) \in \mathbb{R}^2$ and inputs $u = (\tau_1, \tau_2) \in \mathbb{R}^2$, where $\tau_i$ is a torque applied at the joints. The Lagrangian dynamics obey

$$M(q)\ddot{q} + \Gamma(q, \dot{q}) = Bu,$$

where $M(q)$ is the mass matrix and $\Gamma(q, \dot{q})$ collects the gravity and Coriolis terms. The overall state of the system is $x = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathbb{R}^4$.

The control objective is to stabilize the system to the origin, while ensuring that the $y$-position of the end-effector does not dip below the constraint depicted in Figure 4.2. In Figure 4.2 the origin corresponds to both arms pointing directly to the right. To guide the system towards the origin, the method from [9] is used to design a CLF of the form $V(x) = x^T P x$. We then design a CBF which ensures satisfaction of the safety constraint using the method of *exponential control barrier functions* (ECBFs) described in [144].

To set up the learning problem, we vary the dynamics parameters of the model (mass and length of arms) by 50 percent between the 'true' system dynamics and the nominal model used by the system designer. The learned controller is composed of a linear combination of 300 Gaussian radial basis functions distributed randomly throughout the state-space. We solve the reinforcement learning problem (3.13) with a rollout length of $N = 1$, penalty parameters $\rho_1 = 1000$ and $\rho_2 = 100$ and step-length of $\Delta t = 0.05$s. The Soft Actor Critic (SAC) algorithm from [78] is used to solve the problem. Figure 4.2 displays the performance of the learned controller after only 800 samples are collected, which corresponds to 40 seconds of data. The controller was tested from 20 initial conditions, maintaining safety and stability in each scenario.

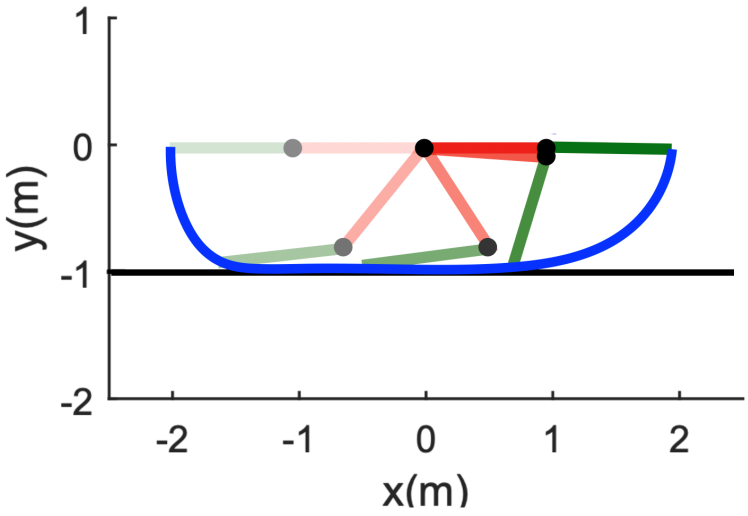



Figure 4.2: A trace of a trajectory for the double pendulum under the influence of the learned controller. The horizontal black line represents the safety constraint, while the blue curve traces the end-effector.

### 4.5.2 Safe Bipedal Locomotion on Stepping Stones

We will now apply the presented method to the Hybrid Zero Dynamics (HZD) framework in order to learn an efficient, stable and safe walking controller for a RABBIT bipedal robot walking on a discrete terrain of randomly spaced stepping stones 4.1. The robot is modelled as a hybrid system with impulse effects, as already presented in the previous chapter in (3.17).

The method of Hybrid Zero Dynamics (HZD) aims to drive the actuated states to zero thereby constraining the system to evolve on a lower dimensional zero dynamics manifold $\Psi = \{(\eta, z) \in \mathcal{T} \times \mathcal{Z} \colon \eta = 0\}$, which contains a stable walking gait for the model. As in [9], the system can be stabilized to this surface using feedback linearization to construct a CLF for the actuated coordinates. Following the method in [148], we also design a CBF which takes in the relative distance between the current and subsequent stepping stones and forces the robot to step down on the next stepping stone during each impact event. Both of these functions are only used to constrain the evolution of the continuous dynamics, but are constructed so as to maintain safe, stable walking for the full hybrid dynamics. Because of this, we can directly apply our framework to overcome model uncertainty in the continuous dynamics.

To set up the learning problem, model uncertainty is introduced by scaling the mass and inertia of each of the robot's links to be three times those of the nominal model. The learned policy takes the form of a neural network with two hidden layers of size $400 \times 300$, and *tanh* activation functions. The training data consists of rollouts of 2 consecutive walking steps with randomly perturbed initial conditions and desired step lengths $l_d$ sampled uniformly from

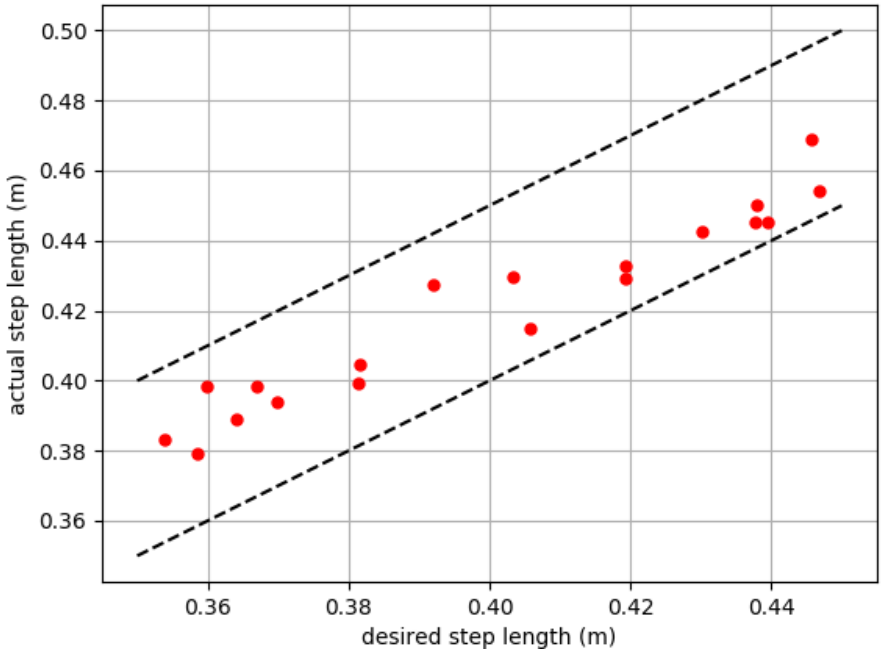



Figure 4.3: Plot of the desired step length vs actual step length achieved by the learned controller for the walking simulation. The black dashed lines indicate the necessary step length constraint required to successfully walk over stepping stones.

$\mathcal{L} := [0.35, 0.45]m$. We again use SAC to train the policy, with a time step of $\Delta t = 1/1000s$ for numerical simulations. The training process converges in about 200,000 time steps, corresponding to about 3 minutes and 20 seconds of data.

The trained policy is tested on 100 simulations of 10 walking steps each, with desired step lengths uniformly sampled from $\mathcal{L}$. The robot only has knowledge of the position of the next stepping stone. A simulation is considered as a failure if the robot fails to land on any of the desired stepping stones, or if it losses stability and falls. Out of the 100 simulations, 93 were successful using the learned controller, while only 26 simulations were successful with the nominal controller without the learning component. This ability of the learned controller to adapt to different required step lengths is clearly reflected in Figure 4.3.

## 4.6 Chapter Summary

In this chapter, we have presented a framework for learning a safe, stabilizing controller for a system with unknown dynamics using model-free policy optimization algorithms. This method extends the one presented in Chapter 3 to design policies that keep the state of the system within a set of safe states. Using a nominal dynamics model, the user specifies a candidate Control Lyapunov Function (CLF) around the desired operating point, and specifies

the desired safe-set using a Control Barrier Function (CBF). Using penalty methods from the optimization literature, we proposed a family of policy optimization problems which attempt to minimize control effort while satisfying the pointwise constraints used to specify the CLF and CBF. We showed that when the penalty coefficients are scaled correctly, the optimization prioritizes safety over stability, and stability over optimality. We then introduced how standard reinforcement learning algorithms can be applied to the problem, and validated the approach through simulation. We finally illustrated how the approach can be applied to a class of hybrid models commonly used in the dynamic walking literature, and used it to learn safe, stable walking behavior over a randomly spaced sequence of stepping stones.

# Chapter 5

# Lyapunov-based Cost Design for Robust and Efficient Robotic Reinforcement Learning

*This chapter is based on the paper titled "Lyapunov Design for Robust and Efficient Robotic Reinforcement Learning" [211], co-authored by Tyler Westenbroek, Ayush Agrawal, S. Shankar Sastry and Koushil Sreenath.*

One of the main goals of this dissertation is to design approaches that use structural

Figure 5.1: We learn precise stabilizing policies on hardware for the `Quanser` cartpole [161] (top) and the `Unitree A1` quadruped [195] (bottom) using only seconds and a few minutes of real-world data, respectively. A video of our experiments can be found here.

knowledge about the system to supervise data-driven methods. This should allow the user to embed desirable behaviors into the resulting policies. In this chapter, we introduce an approach to bias reinforcement learning policy optimization algorithms towards stabilizing solutions.

While most reinforcement learning researchers and practitioners spend countless hours finding a reward function that leads to the desired behavior, there is little research, to date, studying how the choice of reward function influences the learning outcomes. In this chapter, we propose a Lyapunov-theoretic principled method for reward shaping, which leads to more sample efficient and robust policy optimization problems.

## 5.1 Introduction

A key challenge in robotics is reasoning about the long-horizon behavior induced by a control policy. This is because important system properties such as stability are inherently long-horizon phenomena. In reinforcement learning, the *discount factor* implicitly controls how far into the future policy optimization algorithms plan when optimizing the objective specified by the user. Standard approaches to designing objective functions for robotic RL, such as penalizing the distance to a reference trajectory, inherently require a large discount factor to learn control policies which stabilize the system [156, 64]. Unfortunately, problems with large discount factors can be extremely difficult to solve, often requiring vast data sets and careful tuning of hyper-parameters [61]. As a number of recent success stories have demonstrated [117, 112, 152, 153, 122, 16], ever-increasing computational resources can be used to solve these problems in simulation and deploy the resulting controllers directly on the real-world system. However, because it is impractical to model every detail of complex hardware platforms, achieving the best performance will require learning from real-world data.

This chapter introduces a cost-shaping framework which enables users to reliably learn stabilizing control policies with small amounts of real-world data by solving problems with small discount factors. Our approach uses *Control Lyapunov Functions* (CLFs), a standard design tool from the control theory literature [11, 177, 7, 9]. CLFs are 'energy-like' functions for the system which reduce the search for a stabilizing controller to a myopic one-step criterion. In particular, any controller which decreases the energy of the CLF at each instance of time will stabilize the system. Thus, CLFs reduce the long-horizon objective of stabilizing the system to a simple one-step condition. When a CLF is available and the dynamics are known, constructive techniques from the control literature can be used to synthesize a stabilizing controller. However, when there is uncertainty in the dynamics, it is difficult to guarantee that a controller will always decrease the value of the CLF, or that we have even designed a true CLF for the system.

Our approach is to 1) design an approximate CLF for the real-world system using an approximate dynamics model and 2) modify the 'standard' choice of cost functions mentioned above by adding a term which incentivizes controllers which decrease the approximate CLF

over time. This technique effectively uses the approximate CLF as supervision for reinforcement learning, enabling the user to embed known system structures into the learning process while retaining the flexibility of RL to overcome unknown dynamics. Indeed, as our analysis demonstrates, when our approach is used reinforcement learning algorithms implicitly learn to 'correct' the approximate CLF provided by the user. When the candidate CLF is close to being a true CLF for the system (in a sense we make precise below), a stabilizing controller can be efficiently learned by solving a problem with a small discount factor. Moreover, the addition of the approximate CLF 'robustifies' the search for a stabilizing controller by ensuring that even highly suboptimal policies will stabilize the system. Finally, in situations where it is too difficult to design a nominal CLF by hand, we demonstrate how one can be learned using a simulation model and the standard style of RL objective discussed above. Specifically, we use the value function learned by the RL algorithm as an approximate CLF for the real-world system. Altogether, beyond accelerating and robustifying RL, our approach also expands the applicability of CLF-based design techniques.

We apply this technique to develop data-efficient fine-tuning strategies, wherein a nominal controller developed using a simulation model is refined with small amounts of real-world data. For the A1 experiment, the nominal controller is a model-based control architecture [50], and we hand-design a CLF using a highly simplified linearized reduced-order model for the system. Even though this model is very crude, we are nonetheless able to learn a precise tracking controller for this 18 DOF system with only 5 minutes of real-world data. For the cart-pole swing-up task we used the value function from a simulation-based RL problem as the candidate CLF for the real-world system, using the learning process described above. Our fine-tuning approach then learned a robust swing-up controller after observing only one 10 second trajectory from the real-world system.

### 5.1.1 Related Work

In this section, we outline how our approach departs from related work.

**Discount Factors, Sample Complexity and Reward Shaping:** It is well-understood that the discount factor has a significant effect on the size of the data set that RL algorithms need to achieve a desired level of performance. Specifically, it has been shown in numerous contexts [20, 173, 141, 157] that smaller discount factors lead to problems which can be solved more efficiently. This has led to a number of works which explicitly treat the discount factor as a parameter which can be used to control the complexity of the problem alongside reward shaping techniques [94, 154, 61, 189, 35, 143]. Compared to these works, our primary contribution is to demonstrate how CLFs can be combined with model-free algorithms to rapidly learn stabilizing controllers for robotic systems.

**Fine-tuning with Real World Data:** Recently, there has been much interest in using RL to fine-tune policies which have been pre-trained in simulation [176, 97, 96, 133]. These methods typically optimize the same cost function with a large discount factor in both simulation and on the real robot. In contrast, using our cost reshaping techniques, we solve a different problem with a smaller discount factor on hardware which can be solved

more efficiently. In Section 5.6, we show that our method outperforms typical fine-tuning approaches under moderate perturbations to the dynamics model.

**Learning with Control Lyapunov Functions:** A number of recent works have also tried to overcome the reality gap using data-driven methods to improve CLF-based controllers [186, 185, 210, 208, 27, 38]. While these methods work well when a true CLF for the real-world system is available, our method is more general as we can still efficiently learn stabilizing controllers when only an approximate CLF is available by modulating the discount factor used to optimize our cost.

## 5.2 Problem Setting

Throughout this chapter, we consider deterministic discrete-time systems of the form introduced in (2.3),

$$x_{k+1} = F(x_k, u_k),$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state at time $k$, $u_k \in \mathcal{U} \subseteq \mathcal{X}$ is the input applied to the system at that time, and $F \colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is the transition function for the system. We remind the readers that in this dissertation $\Pi$ denotes the space of all control polices $\pi \colon \mathcal{X} \to \mathcal{U}$ for the system. To ease exposition, for the theoretical analysis of this chapter we will again focus on the case where the goal is to stabilize the system to a single point, namely the origin. Through our examples we will demonstrate how our cost-shaping technique can be leveraged to achieve more complicated tasks.

### 5.2.1 Discrete-Time Control Lyapunov Functions

We introduce the discrete-time version of the definition of Control Lyapunov Function that we presented in Chapter 2. To avoid confusion with other value functions introduced in this chapter, we will denote the discrete-time CLF as $W$.

**Definition 5.1.** *We say that a positive definite function $W \colon \mathbb{R}^n \to \mathbb{R}$ is a* discrete-time Control Lyapunov Function (CLF) *for* (2.3) *if the following condition holds for each $x \in \mathcal{X} \backslash \{0\}$:*

$$\min_{u \in \mathcal{U}} W(F(x, u)) - W(x) < 0. \tag{5.1}$$

The condition (5.1) ensures that for each $x \in \mathcal{X}$ there exists a choice of input which decreases the 'energy' $W(x)$. Any policy which satisfies the one-step condition $W(F(x, \pi(x))) - W(x) < 0$ can be guaranteed to asymptotically stabilize the system [101]. Given a CLF for the system, model-based methods constructively synthesize a controller which satisfies this property using either closed-form equations [177] or by solving an online (convex) optimization problem [62, 9] to satisfy (5.1). However, when the dynamics are unknown it is difficult to ensure that we have synthesized a 'true' CLF for the system.

**Remark 5.1.** *(Designing Control Lyapunov Functions) While there is no general procedure for designing CLFs by hand for general nonlinear systems, there do exist constructive procedures for designing CLFs for many important classes of robotic systems, such as manipulator arms [7] and robotic walkers [9] using structural properties of the system. Moreover, in our examples we will investigate how a CLF can be learned from a simulation model and how very coarse CLF candidates can be used to accelerate learning a stabilizing controller.*

### 5.2.2 Stability of Dynamic Programming and Reinforcement Learning

Here, we build on the background on optimal control presented in Section 2.2.1. We investigate the conditions that optimal control problems that use the common class of cost functions introduced in (2.5) need to satisfy to lead to stabilizing solutions.

We recall from (2.8) that, given a discount factor $\gamma \in [0, 1]$, an optimal policy $\pi^*_\gamma$ will satisfy

$$\pi^*_\gamma(x) \in \arg\min_{u \in \mathcal{U}} \left[\gamma V^*_\gamma(F(x, u)) + \ell(x, u)\right], \quad \forall x \in \mathcal{X}.$$

We also remind the readers that in Section 2.2.1 we defined $\ell(x, u) = Q(x) + R(u)$ to be the running cost incurred when applying control $u \in U$ at state $x \in \mathcal{X}$, and $V^*_\gamma(x)$ is the optimal value function at $x \in \mathcal{X}$ (2.6). The running cost components $Q$ and $R$ are assumed to be positive definite.

Unfortunately, it is impractical to directly search over $\Pi$ to find a policy that minimizes the above condition. This necessitates the use of function approximation schemes (e.g. feed-forward neural networks) to instead represent a subset of policies $\hat{\Pi} \subset \Pi$ to search over. Indeed, modern RL approaches for robotics randomly sample the space of trajectories to optimize problems of the form:

$$\inf_{\pi \in \hat{\Pi}} \mathbb{E}_{x_0 \sim X_0}\left[V^\pi_\gamma(x_0)\right], \tag{5.2}$$

where $X_0$ is a distribution over initial conditions and $V^\pi_\gamma$ is the value function associated to $\pi$. While this approach enables these methods to optimize high-dimensional policies, they are data-hungry, can display high-variance and thus frequently return highly sub-optimal policies when data is limited. To better understand the effect that this has on the stability of learned policies, for each $\pi \in \hat{\Pi}$ and $\gamma \in [0, 1]$ define the *optimality gap*:

$$\epsilon^\pi_\gamma(x) = V^\pi_\gamma(x) - V^*_\gamma(x).$$

The temporal difference equation [20] dictates that for each $x \in \mathcal{X}$ the policy satisfies:

$$V^\pi_\gamma(x) = \gamma V^\pi_\gamma(F(x, \pi(x))) + \ell(x, \pi(x)). \tag{5.3}$$

From these equations we can obtain:

$$V_\gamma^\pi(F(x,\pi(x))) - V_\gamma^\pi(x) = \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)V_\gamma^\pi(x)\big) \tag{5.4}$$

$$= \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)[V_\gamma^*(x) + \epsilon_\gamma^\pi(x)]\big) \tag{5.5}$$

$$\leq \frac{1}{\gamma}\big(-Q(x) + (1-\gamma)[V_\gamma^*(x) + \epsilon_\gamma^\pi(x)]\big), \tag{5.6}$$

where we have first rearranged (5.3), then used $V_\gamma^\pi(x) = V_\gamma^*(x) + \epsilon_\gamma^\pi(x)$, and finally we have used $\ell(x,\pi(x)) \geq Q(x)$. Inequalities of this sort are the building block for proving the stability of suboptimal polices in the dynamic programming literature [64, 156].

**Remark 5.2.** *(Value Functions as CLFs) By inspecting the cost* (2.5) *we see that $V_\gamma^\pi$ is positive definite (since $Q$ is positive definite). Thus, if the right-hand side of* (5.6) *is negative for each $x \in \mathcal{X} \setminus \{0\}$, this inequality shows that $V_\gamma^\pi$ is a CLF for* (2.3)*, and that $\pi$ is an asymptotically stabilizing control policy. In other words, $V_\gamma^\pi$ is a CLF which is implicitly learned during the training process. Indeed, many RL algorithms directly learn an estimate of the value function, a fact which we later exploit to learn a CLF for the cart-pole swing up-task in Section 5.5 using the nominal simulation environment.*

Note that the right hand side of (5.6) will only be negative if $V_\gamma^*(x) + \epsilon_\gamma^\pi(x) < \frac{1}{1-\gamma}Q(x)$. Since from (2.5) we know that $V_\gamma^*(x) > Q(x)$ for each $x \in \mathcal{X}$, even the optimal policy (which has no optimality gap) will only be stabilizing if $\gamma$ is large enough. On the other hand, for a fixed $\gamma \in (0,1]$, this inequality also quantifies how sub-optimal a policy can be while maintaining stability. To make these observations more quantitative we make the following assumption:

**Assumption 5.1.** *For each $\gamma \in [0,1]$ there exists $C_\gamma \geq 1$ such that $V_\gamma^*(x) \leq C_\gamma Q(x)$ for each $x \in \mathcal{X}$.*

Growth conditions of this form are standard in the literature on the stability of approximate dynamic programming [124, 156, 64]. Note that, because the running cost $\ell$ is non-negative, we have $C_{\gamma'} \leq C_{\gamma''}$ if $\gamma' \leq \gamma''$. In particular, the constant $C_1$ upper-bounds the ratio between the one-step cost and the optimal undiscounted value function. When $C_1$ is smaller, the optimal undiscounted policy is more 'contractive' and approximate dynamic programming methods converge more rapidly to an optimal solution [124]. Thus, intuitively the constants $C_\gamma \geq 1$ will be smaller when the system is easier to stabilize. The following result is essentially a specialization of the main result from [64]:

**Theorem 5.1.** *Let Assumption 5.1 hold and let $\gamma \in [0,1]$ and $\pi \in \hat{\Pi}$ be fixed. Further assume that there exists $\delta > 0$ such that for each $x \in \mathcal{X}$ we have i) $\epsilon_\gamma^\pi(x) \leq \delta Q(x)$ and ii) $C_\gamma + \delta < \frac{1}{1-\gamma}$. Then, $\pi$ asymptotically stabilizes* (2.3)*.*

*Proof.* Combining conditions $i$) and $ii$) with equation (5.6) yields:

$$V_\gamma^\pi(F(x,\pi(x))) - V_\gamma^\pi(x) \leq \frac{1}{\gamma}\big(-1 + (1-\gamma)[C_\gamma + \delta]\big)Q(x). \qquad \square$$

Thus the RHS of the preceding equation will be negative-definite if $C_\gamma + \delta < \frac{1}{1-\gamma}$, which demonstrates the desired result.

**Remark 5.3.** *(Stability Properties of the Cost Function) In the following section we will derive an analogous result to Proposition 5.1 for the novel reshaped cost function we propose below. When comparing these results we will primarily focus on the effect of the constants $C_\gamma \geq 1$ (and the equivalent constants for the new setting). The $C_\gamma$ constants can be used to bound how large of a discount factor is need to stabilize the system. In particular, Proposition 5.1 implies that the optimal policy will stabilize the system for each $\gamma$ which satisfies $\gamma > 1 - \frac{1}{C_\gamma}$. The $C_\gamma$ constants also characterizes how 'robust' the cost function is to suboptimal policies. In particular, for a fixed discount factor, the policy will stabilize the system if $\delta < \frac{1}{1-\gamma} - C_\gamma$. Thus smaller values of the $C_\gamma$ constants permit more suboptimal policies.*

## 5.3 Lyapunov Design for Infinite Horizon Reinforcement Learning

Our method uses a positive definite candidate Control Lyapunov Function $W\colon \mathbb{R}^n \to \mathbb{R}$ for the nonlinear dynamics (2.3), and reshapes (2.5) to our proposed new long horizon cost $\tilde{V}_\gamma^\pi\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$:

$$\begin{aligned}\tilde{V}_\gamma^\pi(x_0) = &\sum_{k=0}^{\infty} \gamma^k \Big( [W\big(F(x_k,\pi(x_k))\big) - W(x_k)] + \ell(x_k,\pi(x_k)) \Big) \\ &\text{s.t. } x_{k+1} = F(x_k,\pi(x_k)).\end{aligned} \tag{5.7}$$

As we shall see below, our method works best when $W$ is in fact a discrete-time CLF for the system, but still provides benefits when it is only an 'approximate' discrete-time CLF for the system (in a sense we will make precise later). For each $\gamma \in [0,1]$ the new optimal value function is given by:

$$\tilde{V}_\gamma^*(x) = \inf_{\pi \in \Pi} \tilde{V}_\gamma^\pi(x). \tag{5.8}$$

The new cost (5.7) includes the amount that $W$ changes at each time step, and thus encourages choices of inputs which decrease $W$ over time. In this case, the Bellman equation [20] dictates:

$$\tilde{V}_\gamma^*(x) = \inf_{u \in \mathcal{U}} \big[\gamma \tilde{V}_\gamma^*(F(x,u)) + \Delta W(x,u) + \ell(x,u)\big], \quad \forall x \in \mathcal{X}, \tag{5.9}$$

where $\Delta W(x,u) := W(F(x,u)) - W(x)$. To gain some intuition for the approach let us consider the two extremes where $\gamma = 0$ and $\gamma = 1$. In the case where $\gamma = 1$, by inspection we see that $\tilde{V}_1^* = V_1^* - W$ solves the Bellman equation. Plugging in this solution demonstrates that any optimal policy $\tilde{\pi}_1^*$ must satisfy $\tilde{\pi}_1^*(x) \in \arg\min_{u\in\mathcal{U}}[V_1^*(F(x,u)) + \ell(x,u)]$. This is precisely the optimality condition for the original cost (2.5) when $\gamma = 1$, and thus the set of optimal policies for the two problems coincide. Thus, in this case, by embedding the CLF in the cost we are effectively using $W$ as a warm-start initial guess for the optimal value function. In the other extreme where $\gamma = 0$, from (5.9) we see that an optimal policy must satisfy $\tilde{\pi}_0^*(x) \in \arg\min_{u\in\mathcal{U}} \big[\Delta W(x,u) + \ell(x,u)\big]$. Thus, when $\gamma = 0$ the optimal policy attempts to greedily decrease the value of the candidate CLF and the one-step cost on the input. As we shall see below, when intermediate discount factors are used, optimal policies may instead decrease the value of $W$ over the course of several steps.

Using the new cost function (5.7), each policy must satisfy the new difference equation:

$$\tilde{V}_\gamma^\pi(x) = \gamma \tilde{V}_\gamma^\pi\big(F(x,\pi(x))\big) + W\big(F(x,\pi(x))\big) - W(x) + \ell(x,\pi(x)). \tag{5.10}$$

In our stability analysis, we will use the following composite function as a candidate CLF for (2.3):

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) = W(x) + \gamma \tilde{V}_\gamma^\pi(x). \tag{5.11}$$

We provide an interpretation of this curious candidate CLF in Remark 5.4 below, but first perform an initial analysis similar to the one presented in the previous section. Defining for each $\pi \in \hat{\Pi}$, $\gamma \in [0,1]$ and $x \in \mathcal{X}$ the new optimality gap:

$$\tilde{\epsilon}_\gamma^\pi(x) = \tilde{V}_\gamma^*(x) - \tilde{V}_\gamma^\pi(x), \tag{5.12}$$

and following steps analogous to those taken in (5.4)-(5.6), we can obtain the following:

$$\begin{aligned}
\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi\big(F(x,\pi(x))\big) - \tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) &= -\ell(x,\pi(x)) + (1-\gamma)\tilde{V}_\gamma^\pi(x) && (5.13)\\
&= -\ell(x,\pi(x)) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\epsilon}_\gamma^\pi(x)\big] && (5.14)\\
&\leq -Q(x) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\epsilon}_\gamma^\pi(x)\big]. && (5.15)
\end{aligned}$$

Similar to the analysis in the previous section, we will aim to understand when the right-hand side of (5.15) is negative, as this will characterize when $\pi$ stabilizes the system. One key difference between the inequalities (5.6) and (5.15) is that, while the original value function $V_\gamma^*$ is necessarily positive definite, $\tilde{V}_\gamma^*$ can actually take on negative values since the addition of the CLF term allows the new running cost in (5.7) to be negative. As we shall see, this forms the basis for the stability and robustness properties our cost formulation enjoys when $W$ is designed properly.

**Remark 5.4.** *(Learning Corrections to W) When the right hand side of* (5.15) *is negative for each $x \in \mathcal{X} \setminus \{0\}$, inequality* (5.15) *demonstrates that $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi$ is in fact a CLF for* (2.3) *and that $\pi$ stabilizes the system (see Theorem 5.2). We can think of $W$ as an 'initial guess'*

*for a CLF for the system, while $\gamma \tilde{V}_\gamma^\pi$ is a 'correction' to $W$ that is implicitly made by a learned policy $\pi$. Roughly speaking, the larger the discount factor, the larger this correction. Thus, the user can trade-off how much the learned policy is able to correct the candidate CLF $W$ against the additional complexity of solving a problem with a higher discount factor, depending on how 'good' they believe the CLF candidate to be.*

We first state a general stability result for suboptimal policies associated to the new cost, and then discuss how the choice of $W$ affects the stability of suboptimal control policies:

**Assumption 5.2.** *For each $\gamma \in [0, 1]$ there exists $\tilde{C}_\gamma \in \mathbb{R}$ such that $\tilde{V}_\gamma^*(x) \leq \tilde{C}_\gamma Q(x)$ for each $x \in \mathcal{X}$.*

Because the reshaped one-step cost $W(F(x, u)) - W(x) + \ell(x, u)$ can take on negative values, so can the $\tilde{C}_\gamma$ constants. Moreover, in this case it is possibe to have $\tilde{C}_{\gamma'} \geq \tilde{C}_{\gamma''}$ when $\gamma' \leq \gamma''$. This is because when larger discount factors are used, the optimal policy can benefit from decreasing $W$ further into the future. The following stability result is analogous to Proposition 5.1:

**Theorem 5.2.** *Let Assumption 5.2 hold and let $\gamma \in [0, 1]$ and $\pi \in \hat{\Pi}$ be fixed. Further assume that there exists $\tilde{\delta} > 0$ such that for each $x \in \mathcal{X}$ we have i) $\tilde{\epsilon}_\gamma^\pi(x) \leq \delta Q(x)$ and ii) $\tilde{C}_\gamma + \tilde{\delta} < \frac{1}{1-\gamma}$. Then, $\pi$ asymptotically stabilizes* (2.3)*.*

The proof is conceptually similar to the proof of Proposition 1; we delegate the proof to Appendix A.3 for brevity. Indeed, note that the conditions for stability under the new cost are essentially identical to those for the previous cost in Proposition 5.1.

As alluded to in Remark 5.3, we will primarily focus on comparing how large the constants $C_\gamma \geq 1$ and $\tilde{C}_\gamma \in \mathbb{R}$ are for the two problems, as they control the discount factor required to learn a stabilizing policy and also the 'robustness' of the cost to suboptimal controllers. We provide two characterizations which ensure that $\tilde{C}_\gamma < C_\gamma$. The first condition is taken from the model-predictive control literature [89, 70], where CLFs are used as terminal costs for finite-horizon prediction problems. Proof of the following result can be found in Appendix A.3:

**Lemma 5.1.** *Suppose that for each $x \in \mathcal{X}$ the following condition holds:*

$$\inf_{u \in U} W(F(x, u)) - W(x) + \ell(x, u) \leq 0. \tag{5.16}$$

*Then Assumption 5.2 is satisfied with constant $\tilde{C}_\gamma \leq 0$.*

The hypothesis of Lemma 5.1 implies that *i)* $W$ is a true CLF for the system and *ii)* $W$ dominates the running cost $\ell$, in the sense that $W$ can be decreased more rapidly than $\ell$ accumulates. Effectively, this condition implies that it is advantageous for polices to myopically decrease $W$ at each time step. Consequently, when this condition holds optimal

polcies associated to the reshaped costs (5.7) will stabilize the system for any choice of discount factor.

The following definition generalizes this condition to cases where $W$ may not be a true CLF for the system but can be decreased over several time-steps:

**Definition 5.2.** *We say that the candidate CLF $W$ $\bar{\gamma}$-dominates the running cost $\ell$ if for each discount factor $\gamma$ such that $\bar{\gamma} \leq \gamma \leq 1$ and $x \in \mathcal{X}$ we have $\tilde{V}_\gamma^*(x) \leq V_\gamma^*(x)$.*

The condition in (5.2) effectively provides a way of characterizing how 'close' $W$ is to being a true CLF for the real-world system. In particular, the larger $\bar{\gamma}$ the further into the future RL algorithms must look to see the benefits of decreasing $W$. Our previous discussion, which showed that $\tilde{V}_1^* = V_1^* - W$, demonstrates that every candidate CLF 1-dominates the cost. Moreover, clearly $W$ can only 0-dominate the original cost if it is a CLF for the system. While this condition is more difficult to verify for intermediate values of $\bar{\gamma}$, it provides qualitative insight into how even approximate CLFs for the system can still make it easier to obtain stabilizing controllers.

**Remark 5.5.** *(Robustness of reshaped cost) When the condition of Lemma 5.1 is satisfied we will have $\tilde{C}_\gamma \leq 0 < C_\gamma$, implying the new cost enjoys the desirable robustness properties discussed above. When $W$ satisfies the 'approximate CLF' condition in Definition* (5.2)*, it will only enjoy these benefits when the discount factor is large enough. We leave it as a matter for future work to provide quantitative estimates for the $\tilde{C}_\gamma$ constants in these regimes, and to provide sufficient conditions which ensure $W$ $\bar{\gamma}$-dominates the running cost.*

## 5.4 Connection to Stability of Model Predictive Control

Here, we provide a relationship between the cost-shaping approach for reinforcement learning problems that we proposed in the previous section and MPC terminal costs.

We briefly review stability results from the model predictive control literature, focusing our discussion on the benefits of using a CLF as the terminal cost. In their simplest form, MPC control schemes minimize a cost functional of the form

$$\inf_{\hat{\mathbf{u}} \in \mathcal{U}^N} J_{MPC}^N(x_k, \hat{\mathbf{u}}) = \sum_{k=0}^{N-1} \big(Q(\hat{x}_k) + R(\hat{u}_k)\big) + \hat{W}(\hat{x}_N)$$
$$\text{s.t. } \hat{x}_{k+1} = F(\hat{x}_k, \hat{u}_k), \quad \hat{x}_0 = x_k,$$

where $x_k$ is the the current state of the real-world system, $N \in \mathbb{N}$ is the prediction horizon, $\{\hat{x}_k\}_{k=0}^N$ and $\hat{\boldsymbol{u}} = \{\hat{u}_k\}_{k=0}^{N-1} \in \mathcal{U}^N$ are a predictive state trajectory and control sequence, $Q$ and $R$ are as in the previous section, and $\hat{W} \colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is the terminal cost which is assumed to be a proper function. The MPC controller then applies the first step of the resulting open

loop control and the process repeats, implicitly defining a control law $u_{MPC}(x)$. The MPC cost $J^N_{MPC}(x_k, \cdot)$ can be thought of as a finite-horizon approximation of the original cost (2.5) (except that it is defined over an open-loop sequence of control inputs instead of being a cost over policies).

Stability results from the MPC literature focus primarily on the effects of the prediction horizon $N$ and the choice of terminal cost $\hat{W}$. Under mild conditions, for any choice of terminal cost (including $\hat{W}(\cdot) \equiv 0$), the user can guarantee that the MPC scheme stabilizes the system on any desired operating region by making the prediction horizon $N$ sufficiently large [88, 70]. Thus, there is a clear connection between the explicit prediction horizon $N$ in MPC schemes and the discount factor $\gamma$, as both need to be sufficiently large if a stabilizing controller is to be obtained (since trajectory optimization problems with longer time horizons are generally more difficult to solve). Indeed, in [156] it was pointed out that the *implicit prediction horizon* $\frac{1}{1-\gamma}$, a factor which shows up in the stability conditions in Proposition 5.1, plays essentially the same role in stability analysis as $N$ for an MPC scheme with no terminal cost when the running cost is $\ell = Q + R$. Thus, much like the 'typical' policy optimization problems discussed in Section 5.2.2, MPC schemes with no terminal cost (or one which is chosen poorly) may require an excessively long prediction horizon to stabilize the system.

Fortunately, the MPC literature has a well-established technique for reducing the prediction horizon needed to stabilize the system: use an (approximate) CLF for the terminal cost $\hat{W}$ [90, 88, 70]. Indeed, roughly speaking, these results guarantee that for *any prediction horizon* $N \in \mathbb{N}$ the MPC scheme will be stabilizing if $\hat{W}$ is a valid CLF for the system. Extensive empirical evidence [89] and formal analysis [90] has demonstrated that well-designed CLF terminal costs reduce the prediction horizon needed to stabilize the system on a desired set and increase the robustness of the overall MPC control scheme [69]. Thus, in many ways our cost-reshaping approach can be seen as a way to obtain these benefits in the context of infinite horizon model-free reinforcement learning.

## 5.5 Overview of Experimental Results

We summarize the main results for each of our examples, and explain each of them in detail in Section 5.6. In every experiment we report, the soft actor-critic algorithm (SAC) [78] is used as the learning algorithm to optimize the various reward structures we investigate.

### 5.5.1 Velocity Tracking for A1 Quadruped

We apply our approach to train a neural network controller which augments and improves a nominal model-based controller [50] for a quadruped robot using real-world data. As illustrated by the pink curve in Fig. 5.2 (left), the nominal controller fails to accurately track desired velocities specified by the user. We design a CLF around the desired gait using a linearized reduced-order model for the system. We then collect rollouts of $10s$ on the robot

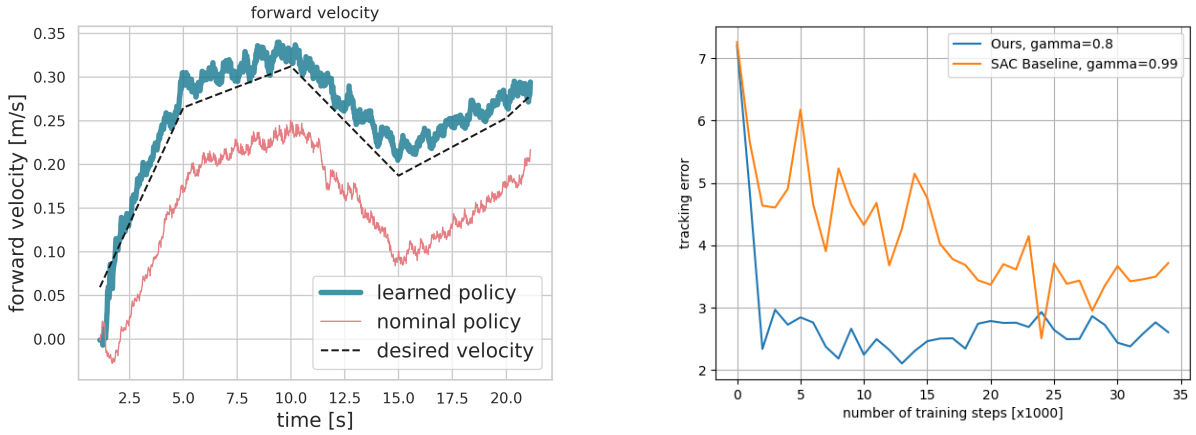



Figure 5.2: (Left) Plot illustrating improved velocity tracking of the learned policy (in dark green) compared to the nominal locomotion controller (in pink) to track a desired velocity profile (in dashed black line) using our proposed method on the `Unitree A1` robot hardware. (Right) Plot from the simulated benchmark study illustrating cumulative velocity tracking error (lower is better) over 10s rollouts at different stages of the training. In orange, we show the results of fine-tuning using SAC with a standard RL cost. In blue, we fine-tune using SAC with our reward reshaping method, with a candidate CLF designed on a nominal linearized model of the robot. In both cases, we plot the results using the discount factor that achieved the best performance.

hardware with randomly chosen desired velocity profiles, and solve an RL problem using our cost and a discount factor $\gamma = 0$. Our approach is able to learn a policy which significantly improves the tracking performance of the nominal controller within 5 minutes (30 episodes) of hardware data, as shown in Fig. 5.2 (left). A video of these results can be found in this link. Furthermore, in Fig. 5.2 (right) we benchmark our approach in simulation against an RL agent trained with a 'standard' cost which penalizes the squared error with respect to the desired velocity. As this figure demonstrates, our method is able to rapidly decrease the average tracking error in only around 2 thousand steps from the environment. In contrast, the benchmark approach is only able to reach this level of performance for the first time after around 24 thousand steps.

### 5.5.2 A1 Quadruped Walking with an Unknown Load

We attach an un-modeled load to the A1 quadruped, that is equivalent to one-third the mass of the robot. Fine-tuning on hardware the same base controller from the previous set-up where the CLF is designed to stabilize to the target gait, our approach is able to significantly decrease the tracking error to about one-third its nominal value with only one minute of data collected on the robot hardware as illustrated in Fig. 5.3. Additionally, in Section 5.6, we

run a simulated benchmark comparison and verify that our method clearly out-performs the 'standard' cost baseline for this task.

### 5.5.3 Fine-tuning a Learned Policy for cart-pole Swing-Up

We fine-tune a swing-up controller for the `Quanser` cart-pole system [161] using real-world data and an initial policy which was pre-trained in simulation but that does not translate well to the real system. Due to the underactuated nature of the system, synthesizing a CLF by hand is challenging. Thus, as alluded to previously, we use a 'typical' cost function of the form (2.5) and a discount factor of $\gamma = 0.999$ to learn a stabilizing neural network policy $\pi_\phi$ for a simulation model of the system. Given the discussion in Remark 5.2, we use the value function $V_\theta$ associated with the simulation-based policy as the candidate CLF ($W = V_\theta$) for our reward reshaping formulation (5.7). When improving the simulation-based policy $\pi_\phi$ with real-world data, we keep the parameters of this network fixed and learn an additional smaller policy $\pi_\psi$ (so that the overall control action is produced by $\pi_\phi + \pi_\psi$) using our proposed CLF-based cost formulation. We solve the reshaped problem with a discount factor $\gamma = 0$ and collect rollouts of $10s$ on hardware. Our CLF-based fine-tuning approach is able to successfully complete the swing-up task after collecting data from just one rollout. After collecting data from an additional rollout, the controller is reliable and robust enough to recover from several pushes. A video of these experiments can be found here, and more details and plots of the results are provided in Section 5.6. Furthermore, in that section we provide a simulation study comparing a standard fine-tuning approach to our method, showing that our approach is able to more rapidly learn a reliable swing-up policy than the baseline and also achieves a higher reward.

### 5.5.4 Fine-tuning a Bipedal Walking Controller in Simulation

We also apply our design methodology to fine-tune a model-based walking controller [9] for a bipedal robot with large amounts of dynamics uncertainty. Model uncertainty is introduced by doubling the mass of each link of the robot. The nominal controller fails to stabilize the gait and falls within a few steps. To apply our method, we design a CLF around the target gait as in [9] to be used in our reward formulation. As a benchmark comparison, we also train policies with a reward which penalizes the distance to the target motion (no CLF term), as is most commonly done in RL approaches for bipedal locomotion which use target gaits in the reward [122]. Our approach is able to significantly reduce the average tracking error per episode after only 40000 steps of the environment (corresponding to 40 seconds of data), while the baseline does not reach a similar level of performance even after 1.2 million steps, as illustrated in Fig. 5.7.

### 5.5.5 Inverted Pendulum with Input Constraints

Our final example demonstrates the utility of our method even when $W$ is a crude guess for a CLF for the system, through the use of moderate discount factors. We illustrate this for a simple inverted pendulum simulator by varying the magnitude of the input constraints for the system. We use the procedure from [9] to design a candidate CLF for the system. Like many CLF design techniques, this approach assumes there are no input constraints and encourages the pendulum to swing directly up. As the input constraints are tightened, $W$ becomes a poorer candidate CLF, as there is not enough actuation authority to decrease $W$ at each time step. Even in this case, in line with the discussion of Remark 5.5, if a proper discount factor is used, the addition of the candidate CLF in the reward enables our method to rapidly learn a stabilizing controller for each setting of the input bound. These results are presented in detail in the next section.

## 5.6 Examples and Comparison Studies

We now provide more details of the experimental results reported in Section 5.5 and also additional evaluations. While we have chosen to minimize costs in the main portion of the paper, as this is more consistent with the notation used in the literature on Lyapunov theory and the stability of dynamic programming, most RL algorithms take in rewards that are to be maximized. Thus, for the sake of consistency with practical implementations, in this section we report the reward functions used in our code, which are simply the costs from before with the sign flipped.

For training from hardware data, we used asynchronous off-policy updates, similar to the framework presented in [73]. In particular, we have two separate threads, with one running episodes on the hardware system with the latest available policy and adding the transition data to the replay buffer, and the other one sampling from this buffer and performing the actor and critic updates. We only synchronize the policy network weights at the beginning of each episode.

### 5.6.1 A1 Quadruped Results

To illustrate the efficacy of our approach, we run two sets of experiments with the A1 robot: 1) accurately tracking a target velocity when the gains $k_p$ and $k_d$ are not well tuned (Section 5.5); and 2) accurately tracking the height of the robot with an unknown load attached to it. Here we provide additional details of experiments related to these experiments. For both settings, we use the locomotion controller presented in [50, Section 3.2] as our nominal baseline controller. This controller uses a linearized rigid-body model to formulate a quadratic-program (QP)-based controller to track a desired body pose of the robot. Specifically, the following QP is solved to obtain the ground reaction forces $f$ for the feet in contact

with the ground:

$$\begin{aligned} \min_{f} \ & \|\mathbf{M}f - \tilde{g} - \ddot{q}_d\|_Q + \|f\|_R \\ \text{s.t. } & f_z \geq 0, \\ & -\mu f_z \leq f_x \leq \mu f_z, \\ & -\mu f_z \leq f_y \leq \mu f_z, \end{aligned} \tag{5.17}$$

where $\mathbf{M}$ is the inverse inertia matrix of the rigid body, $\tilde{g} := [0, 0, g, 0, 0, 0]$ denotes the acceleration due to gravity and $\ddot{q}_d \in \mathbb{R}^6$ are the desired pose accelerations of the robot's body. In particular, the desired accelerations are obtained using a PD controller,

$$\ddot{q}_d = -k_p(q - q_d) - k_d(\dot{q} - \dot{q}_d), \tag{5.18}$$

with $q \in \mathbb{R}^6$ denoting the robot's body pose.

Next, we provide further details for each set of experiments on the A1 robot.

### Velocity Tracking for A1 Quadruped

When the feedback gains $k_p, k_d \in \mathbb{R}^6$ are not well tuned, large tracking errors in the forward speed of the robot can persist as illustrated in Fig. 5.2 (left). To compensate for the increased tracking error, we learn a policy $\pi_\theta$ (MLP with two hidden layers of size $32 \times 32$) that outputs an additional acceleration term in (5.18), making the final desired acceleration $\ddot{q}_d = -k_p(q - q_d) - k_d(\dot{q} - \dot{q}_d) + \pi_\theta$. $\pi_\theta$ can therefore be viewed as a learned fine-tuning policy with respect to a model-based controller. The observations for the RL agent include the forward and lateral velocity, the roll and pitch orientation and the desired forward velocity of the robot. The actions include offsets to the desired forward and lateral accelerations.

The policy $\pi_\theta$ is learned directly on the robot hardware using a CLF $W$ designed for the nominal rigid body dynamics of the robot following the procedure described in [9]. For training, we use SAC [78] with the reward $r_k = \frac{(W(F(x_k,u_k)) - W(x_k))}{\Delta t_k} + \lambda\|u_k\|^2$. The CLF term in the reward allows us to use a discount factor $\gamma = 0$, which considerably reduces the complexity of the learning problem. Indeed, within only 5 minutes of data collected from the robot hardware, our method is able to significantly reduce the tracking error in the forward velocity compared to the nominal locomotion controller, as shown in Figure 5.2 (left).

### Height Tracking with an Unknown Load

In this experiment, we use the same base controller and an equivalent offset policy $\pi_\theta$ as in the previous set-up and attempt to track a target gait. The CLF is designed to stabilize to the target gait as in the previous experiment. Figure 5.3 plots the tracking error of the learned controller versus the nominal controller after only 1 minute of training data. As the figure demonstrates, our approach is able to significantly decrease the error to about one-third its nominal value with only a small amount of data.

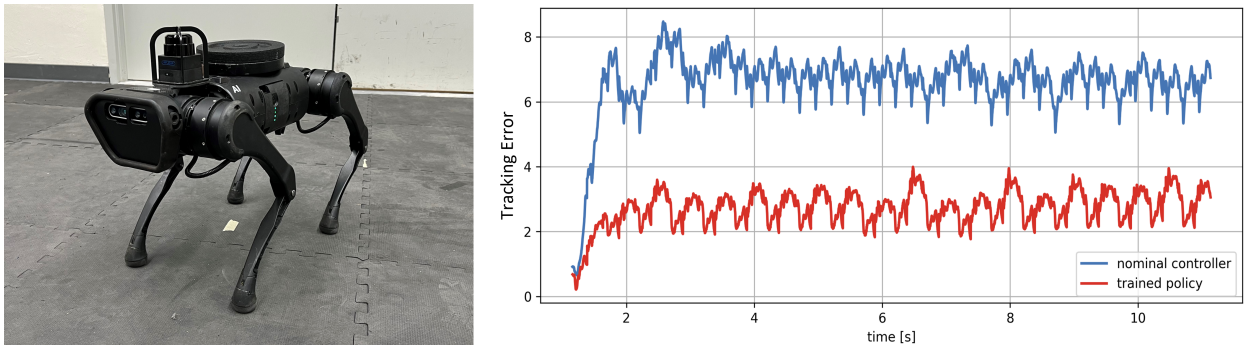



Figure 5.3: Comparison between nominal controller and learned policy after training on 60s of real-world data on the A1 robot with an added 10lb weight. The learned policy is able to significantly reduce the tracking error caused by the added weight.

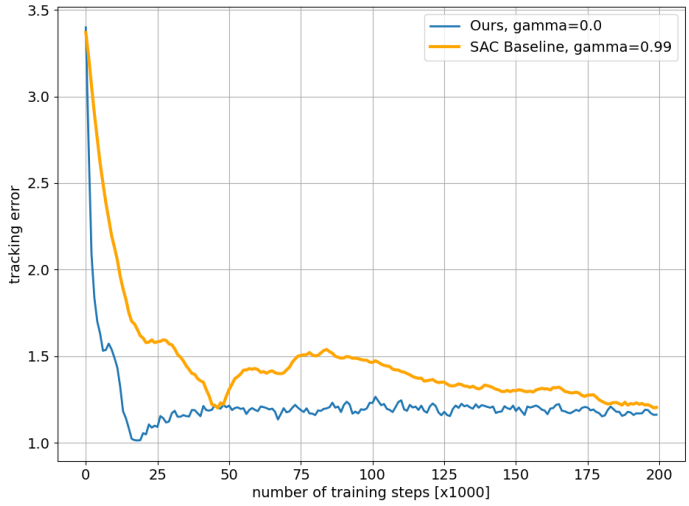



Figure 5.4: Cumulative gait tracking error (lower is better) over 10s rollouts at different stages of the simulated fine-tuning benchmark comparison of the A1 quadruped with an unknown load. In orange, we show the results of fine-tuning using SAC with a standard RL cost which penalizes the distance to the desired gait with a discount factor of $\gamma = 0.99$. In blue, we plot the performance of our cost reshaping method with SAC and a discount factor of $\gamma = 0$. For both cost formulations, we plot the discount factor that led to the best performance.

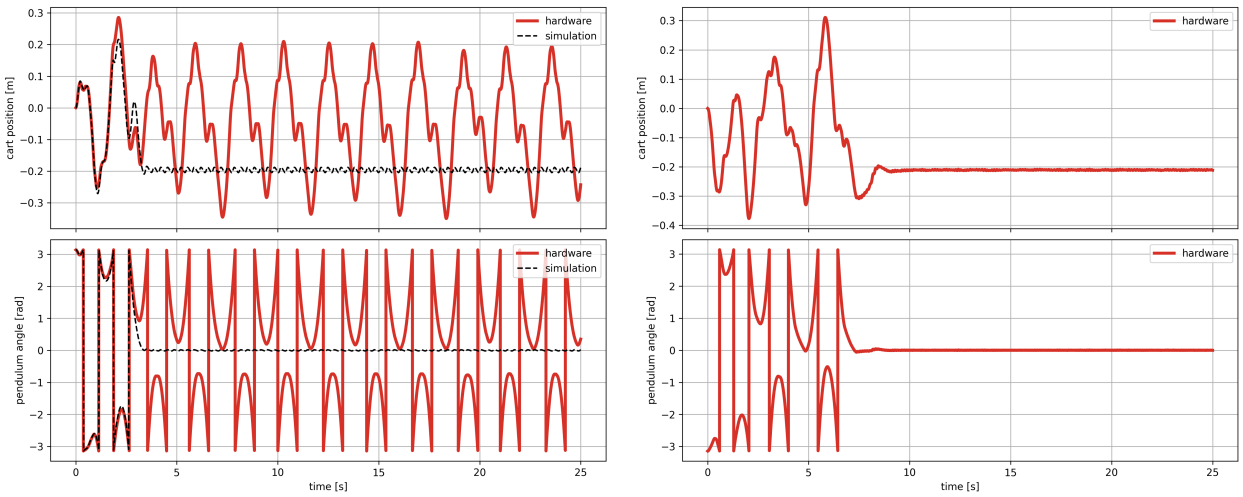



Figure 5.5: Experimental plots of the cart position and pendulum angle of the cart-pole system. (left) The policy trained only in simulation fails to bring the real cart-pole system to the upright position; (right) by fine-tuning the learned policy with 20$s$ of real-world data using our CLF-based reward function, we obtain a successful policy.

To verify that our method out-performs the baseline for this task, we run a simulated benchmark comparison similar to the A1 simulation study for velocity tracking that was presented in Section 5.5 of the thesis. For this case, we reproduce the unknown load hardware experiment in simulation by adding a 10lb weight to the robot. When testing our method, we again use SAC with the same reward formulation from the hardware experiments above. For the baseline reward, we penalize the distance to the target that we want to track. Figure 5.4 depicts the best results that we have been able to obtain for each cost formulation across different discount factors and training hyper-parameters. As Fig. 5.4 depicts, our approach quickly converges to a stable walking controller which closely tracks the references after only around 22 thousand steps of the environment. The baseline does not match this performance until it has had access to around 48 thousand steps, and takes much longer to consistently approach the performance of our method.

### 5.6.2 Cart-pole Results

We first provide plots and give additional details for the cart-pole experiments presented in Section 5.5. Then, we present a comparison of the performance of our approach with respect to a typical fine-tuning method on a simulator of the cart-pole system.

**Additional Details of the Cart-pole Hardware Fine-tuning Experiments**

For the cart-pole experiments presented in Section 5.5, we used a Quanser Linear Servo Base Unit with Inverted Pendulum [161], with a pendulum length of 60cm. The system has 4 states, $x = [p,\ \alpha,\ \dot{p},\ \dot{\alpha}] \in \mathbb{R}^4$, corresponding to the cart position $p$, the pendulum angle $\alpha$, and their respective velocities. The control input is the voltage applied to the motor that actuates the cart $u \in \mathbb{R}$.

We first train a SAC agent in simulation using a 'conventional' RL reward that penalizes the distance to the equilibrium, control effort, and includes a penalty if the cart goes off-bounds $r(x_k, u_k) = -0.1\ (5\alpha_k^2 + p_k^2 + 0.05u_k^2) - 5 \cdot 10^3 \cdot \mathbb{1}(|p_k| \geq 0.3)$. The observations of the RL agent are state measurements, the actions are direct voltage commands with limits set to $|u| < 10$V as specified by the manufacturer, and the simulation is run at 100Hz. In order to obtain a stabilizing swing-up policy with this traditional reward, a high discount factor is needed, so we use $\gamma = 0.999$. After around 15 thousand seconds of simulation data with a learning rate of $5 \cdot 10^{-4}$, the RL agent learns to consistently swing-up and balance the pendulum at the upright position in simulation. However, when deployed on the cart-pole hardware system, the policy from simulation fails to obtain successful swing-up behaviors due to the sim-2-real gap, as shown in the attached video.

To tackle these issues, we exploit the fact that SAC uses a feedforward neural network to approximate the discounted value function of the problem, and we use this approximate value function (after 18,600 seconds of data) as a CLF candidate to fine-tune the learned policy directly on hardware.

Thus, we learn on hardware a fine-tuning policy $u_\psi$ (MLP with 2 hidden layers of $64 \times 64$) whose actions are added to the ones of the policy trained on simulation $u_\phi$ (MLP with 2 hidden layers of $400 \times 300$). The episodes are 10 seconds long, and the policy is run at 500Hz, with each episode consisting of 5000 data points. The action space limits for this new policy are set to $|u_\psi| < 4$V but we still have a saturation of the total voltage $|u_\phi + u_\psi| < 10$V. The reward for this new policy is $\hat{r}(x_k, u_k) = \Delta V_\theta(x_k, u_k) - 0.1 \cdot (5\alpha_k^2 + p_k^2 + 0.05u_k^2)$, where $V_\theta$ is the value function network of the SAC agent that was trained in simulation. This allows us to set the discount factor $\gamma = 0$ for the offset policy learned on hardware and therefore greatly reduce the complexity of the learning problem. After only one episode of 10 seconds of real-world data we obtain a policy that manages to swing-up the pendulum to the upright position, and stabilizes it at the top. However, the behavior near the top is not smooth, and it fails for some different initial conditions. After training with another episode of 10 seconds of data, we obtain a policy that consistently manages to swing-up and balance the pendulum at the top, while the cart stays in-bounds. The plots in Fig. 5.5 (right) show the cart position and the pendulum angle when deploying the fine-tuned policy in the real Quanser cart-pole system. The plots in Fig. 5.5 (left) show the results when using the policy that has been only trained in simulation, and how its performance is very different when deployed in simulation vs in hardware. A video with the results of the cart-pole experiments can be found in this link, and a sequence of snapshots of a successful experiment that uses the fine-tuned policy can be found in Figure 5.1.

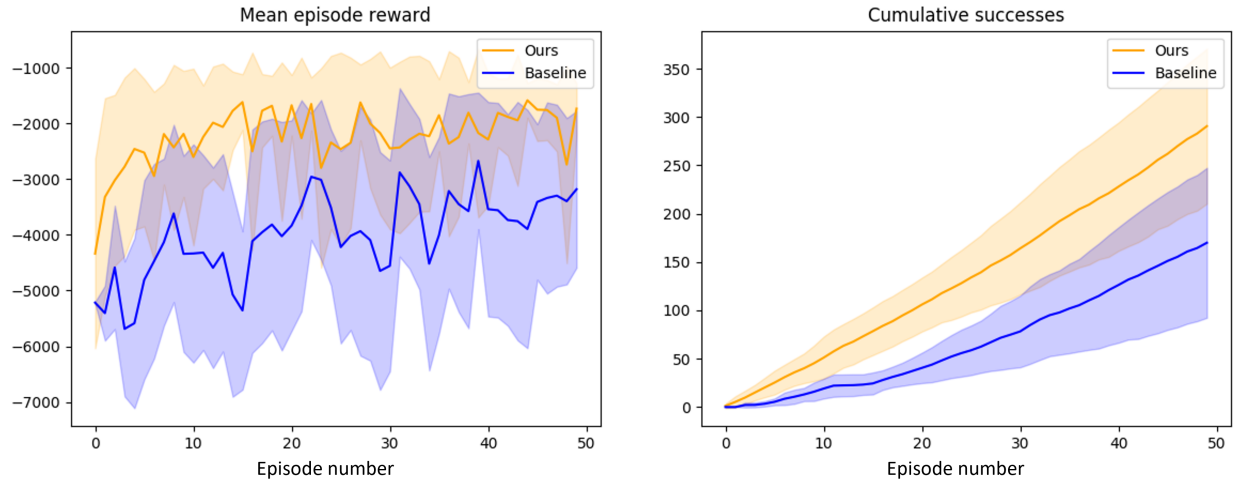



Figure 5.6: Comparison of the simulation results of fine-tuning a cart-pole swing-up policy after adding model mismatch. A policy trained on a nominal dynamics model of the cart-pole fails when deployed on the new dynamics. In blue, we show the results of continuing to train the agent with the original costs and discount factor. In orange, we fine-tune using our reshaping method with the pre-trained value function and a discount factor of $\gamma = 0$. For each episode of training on the new dynamics model, we compare the performance of both methods when running the cart-pole from 10 initial conditions: (on the left) the average original reward without the CLF term, and (on the right) the cumulative number of successful swing-ups. The plots show the mean and standard deviation of the results over 10 different training random seeds.

### Cart-pole Simulation Baseline Comparison with a Typical Fine-tuning Method

As explained at the beginning of the paper, previous work has shown that using hardware data to fine-tune a policy that has been pre-trained in simulation is a powerful approach to tackle the sim-2-real gap problem (e.g. [176, 97, 96, 133]). These methods typically take the RL agent trained in simulation and continue its learning process using hardware data, the original cost function and discount factor (see e.g. [176]). In contrast, our proposed approach stops the simulation training of $u_\phi$ and learns a smaller offset policy $u_\psi$ from hardware data using a separate learning process that has a different reward function $\hat{r}$ (with the CLF candidate being the learned value function in simulation) and a smaller discount factor (in this case $\gamma = 0$).

In Figure 5.6, we compare in simulation the results of using this standard fine-tuning approach with those obtained with our method. For both approaches, we first pre-train a policy $\pi_\phi$ and value function $V_\theta$ on a nominal set of dynamics using SAC and the reward $r(x_k, u_k) = -0.1\,(5\alpha_k^2 + p_k^2 + 0.05u_k^2) - 5 \cdot 10^3 \cdot \mathbb{1}(|p_k| \geq 0.3)$, and then perturb the parameters of the simulator to introduce model mismatch for the fine-tuning phase. Specifically, we

increase the weight and friction of the cart by 200%; and the mass, inertia and length of the pendulum by a 25%. After doing this, we randomly sample 10 initial conditions around the downright position ($-0.05m \leq p_0 \leq 0.05m$, $-\pi + 0.05rad \leq \alpha_0 \leq \pi - 0.05rad$, $-0.05m/s \leq \dot{p}_0 \leq 0.05m/s$, $-0.05rad/s \leq \dot{\alpha}_0 \leq 0.05rad/s$). We label a trial as success if within 10 seconds of simulation, the pendulum is stabilized in the set $-0.12rad < \alpha < 0.12rad$, $-0.3rad/s < \dot{\alpha} < 0.3rad/s$ and the cart never gets out of bounds ($|p| < 0.3$). The policy $u_\phi$ trained with data from the nominal dynamics model does not succeed for any of the 10 initial conditions due to the model mismatch. The baseline in Figure 5.6 is obtained by emptying the replay buffer and using data from the new environment to continue the training process of $u_\phi$ with the same reward $r(x_k, u_k)$. On the other hand, as with the hardware experiments, our method takes the learned value function $V_\theta$ from the nominal dynamics model and learns an offset policy $u_\psi$ using the modified reward $\hat{r}(x_k, u_k) = \Delta V_\theta(x_k, u_k) - 0.1 \cdot (5\alpha_k^2 + p_k^2 + 0.05u_k^2)$. In Figure 5.6, we plot for 10 training random seeds the average original reward $r(x_k, u_k)$ and the cumulative number of successes of the validation episodes ran from the initial conditions mentioned above. The x axis is the number of rollouts of fine-tuning data (each rollout consists of 10 seconds of data). As this figure clearly demonstrates, our approach is able to more rapidly learn a reliable swing-up controller than the baseline. Moreover, as the plot on the left displays, even though we are no longer optimizing for the original reward, by rapidly converging to a stabilizing controller our method still performs better on the original reward than the benchmark.

The above results show that our approach effectively serves to fine-tune policies when the dynamics of the system change. In fact, we have artificially added a severe model mismatch and shown that we can adapt to the new dynamics with a discount factor of 0. This is because the original value function is still a 'good' CLF candidate for the new system. However, if the change in the dynamics is drastic, or if the overall shape of the motion required to complete the task has to be greatly modified, then the value function from the original dynamics may not be a good CLF candidate, and our method might fail. We have observed that for the cart-pole example our method is very robust to variations in the parameters of the cart dynamics (in fact, in the above example we are multiplying both friction and mass of the cart by a factor of 3), but that if we drastically reduce the length and mass of the pendulum by a 50%, our method fails. We hypothesize that this might be related to the underactuated nature of the pendulum dynamics. An interesting direction for future work would therefore be to study under which conditions the original value function retains the CLF properties for a new set of dynamics.

### 5.6.3 Bipedal Walking Results

In this section, we provide further details on applying our design methodology to fine-tune a model-based walking controller for a bipedal robot. As mentioned in Section 5.5, we first design a CLF around the target gait using the nominal model as in [9] to be used in our reward formulation. As a benchmark comparison, we also train policies with a typical reward which penalizes the distance to the target motion. For both approaches we use the SAC algorithm

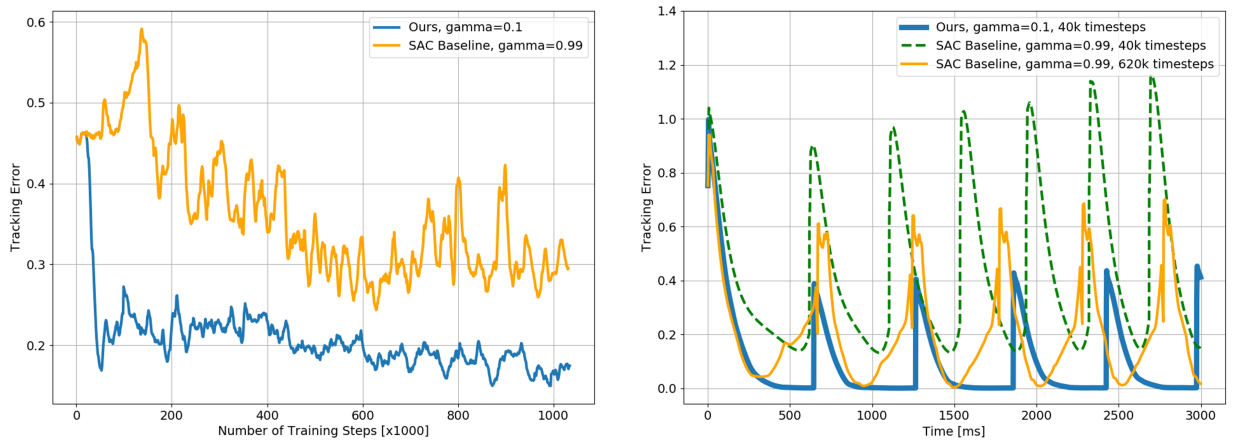



Figure 5.7: (Left) Average tracking error (lower is better) per episode at different stages of the training process when fine-tuning a model-based walking controller under model mismatch. In blue, using our CLF-based reward formulation and SAC, the robot learns a stable walking gait with only 40k steps (40 seconds) of training data. In orange, with a baseline that uses a typical reward penalizing the tracking error to the target gait, the training takes longer to converge and does not achieve the same performance. The results show the best performance for both method across different discount factors and training hyper-parameters. (Right) Comparison of the tracking error of roll-outs of different learned walking policies. In blue, a policy learned with 40k steps of the environment using our CLF-based reward. In dashed green, a policy learned using the baseline reward with 40k steps of the environment. In orange, a policy learned using the baseline reward with 620k steps of the environment (best baseline policy). The jumps in tracking error occur at the swing-leg impact times. The policy learned with our reward formulation clearly outperforms the baseline, even when the baseline has 15 times as much data.

to optimize the policy. We plot the best performance we have been able to obtain from each method by sweeping across different discount factors and algorithm hyper-parameters in Figure 5.7. In particular, the top of this Figure depicts snapshots of the stable walking controller our method obtains after only 40k steps of the environment, which corresponds to only 40 seconds of data given the 1kHz frequency of the controller. The bottom left depicts the average tracking error during the training process for both methods. Finally, the bottom-right plots the tracking error over a few representative rollouts. Note that the tracking error for both methods 'jumps' each time one of the feet impacts the ground. These jumps occur when the swing-foot impacts with the ground and are an unavoidable feature of the environment. Thus, in this context a stable walking controller needs to rapidly converge to the target motion over the course of the next step to maintain stability of the walking motion. As the learning curve demonstrates, our approach is able to significantly reduce the average tracking error per episode after only 40k steps of the environment, while the baseline

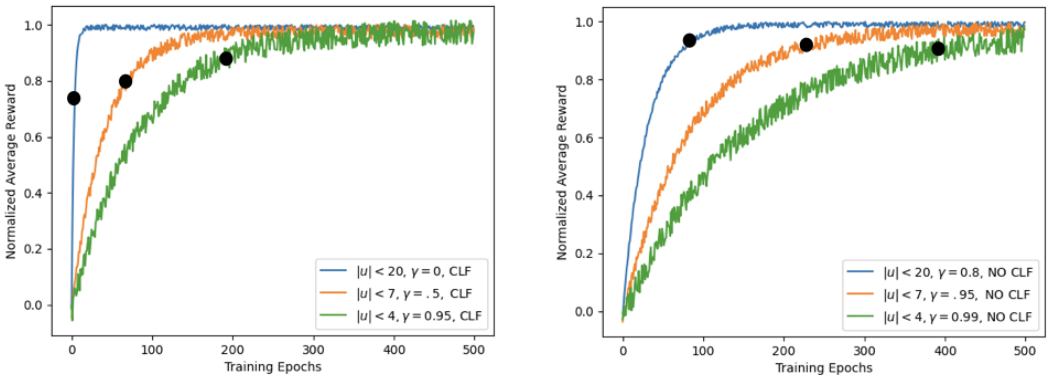



Figure 5.8: Learning curves for an inverted pendulum system under different input constraints. The curves plotted correspond to the smallest discount factors that led to stabilizing policies. On the left, the obtained learning curves use a CLF in the reward. On the right, the reward does not include the CLF term. The black dots denote the first stabilizing policy for each training. For each setting we plot the learning curve for the discount factor that achieved the best performance.

does not reach a similar level of performance even after 1.2 million steps. As the rollouts in the bottom-right demonstrate, our method learns a desirable tracking controller which smoothly decreases the tracking error between each impact event after only 40 thousand steps. In contrast, after 40 thousand steps the baseline controller diverges from the target motion, corresponding to a fall after only a few steps. After 620 thousand steps, the baseline controller is able to maintain the stability of the walking motion, yet the tracking performance is notably worse than our method at 40 thousand steps, despite having access to around 15 times as many samples.

### 5.6.4 Inverted Pendulum Results

The states of the system are $x = (\theta, \dot{\theta}) \in \mathbb{R}^2$, where $\theta$ is the angle of the arm from the vertical position, and the input $u \in \mathbb{R}$ is the torque applied to the joint. In each of the reinforcement learning experiments reported in Section 5.5 for this system we sample initial conditions over the range $-\pi \leq \theta \leq \pi$ and $-0.1 < \dot{\theta} < 0.1$.

We first train a stabilizing controller using a 'typical' cost function of the form $r_k = -\|x_k\|_2^2 - 0.1\|u_k\|_2^2$, and then train a controller using the reshaped cost

$$r_k = -\left[W(F(x_k, u_k)) - W(x_k)\right] - \|x_k\|_2^2 - 0.1\|u_k\|_2^2.$$

We use the soft actor critic (SAC) algorithm [77] and each training epoch consisted of 5 episodes with 100 simulation steps each, where each time step for the simulator is 0.1 seconds. For both forms of cost function, we sweep across different values of discount factors

(from $\gamma = 0$ to $\gamma = 0.95$ in increments of 0.05 and also tried $\gamma = 0.99$) to 1) determine which values of discount factors lead to stabilizing policies and 2) which discount factor allows the agent to learn a stabilizing controller most rapidly. To determine whether a given controller stabilizes the system we randomly sample 20 initial conditions and see if each trajectory reaches the set $\{x \in \mathbb{R}^n : \|x\|_2 < 0.05\}$ within 20 seconds of simulation. For each scenario, the smallest discount factor that lead to a stabilizing controller was also the discount factor that cause the agent to learn a stabilizing controller with the least amount of data.

Training curves for each of the critical values of the discount factor are depicted in Figure 5.8 for each of the cost formulations and input constraints. Each curve indicates the average reward per epoch across 10 different training runs and reports the best results for each scenario after an extensive hyper-parameter sweep. We normalize each training curve so that a reward of 0 indicates the average reward during the first epoch, while a reward of 1 is the largest average reward obtained across all epochs. On each of the training curves the black dot denotes the first training epoch at which a stabilizing controller was obtained.

As illustrated by the plots in Figure 5.8 (a), the addition of the CLF enables our method to more rapidly learn a stabilizing controller in each setting and consistently decreases the amount of data that is needed to learn a stabilizing controller, even when $W$ is not a global CLF for the system. However, the effects are more pronounced when the input constraints are less restrictive and $W$ is a better candidate CLF. For example, when $|u| < 20$ our approach is able to learn a stabilizing controller in 5 iterations, whereas it takes 92 iterations with the original cost (our approach takes $\sim 5.4\%$ as many samples). Meanwhile when $|u| < 4$ our approach takes 198 iterations while the original cost takes 389 iterations (our approach takes $\sim 51\%$ as many samples).Moreover, we observe that larger discount factors are required when $|u| \leq 7$ and $|u| \leq 4$, as $W$ becomes a poorer candidate CLF for these cases.

## 5.7 Chapter Summary

In this chapter, we have introduced a novel cost-shaping method to address the poor sample complexity issue that reinforcement learning algorithms face for real-world robotic applications. By incorporating a Control Lyapunov Function term into conventional cost formulations, our approach significantly reduces the number of samples required to learn a stabilizing controller, and ensures the stability of even highly suboptimal policies.

Theoretical results validate the use of smaller discount factors in our method, which are known to reduce sample complexity. Empirical evidence is provided through two hardware examples: a cart-pole and an A1 quadruped robot. In both cases, stabilizing controllers are learned with minimal fine-tuning data—seconds for the cart-pole and few minutes for the A1 quadruped. Furthermore, our simulation benchmark studies reveal that our proposed cost-shaping method outperforms standard cost designs, requiring orders of magnitude less data to obtain stabilizing policies.

# Chapter 6

# Constraining Distributional Shift with Nonlinear Control Self-Supervision

*This chapter is based on the paper titled "In-Distribution Barrier Functions: Self-Supervised Policy Filters that Avoid Out-of-Distribution States" [29], co-authored by Haruki Nishimura, Rowan McAllister, Koushil Sreenath and Adrien Gaidon.*

Previous chapters focused on incorporating structural knowledge about the system into reinforcement learning problems through the reward function. In this chapter, we show that even when we do not have any knowledge about the underlying dynamical structures of the system, we can still impose control-theoretical guidance on pure deep-learning based problems to achieve desired behaviors.

In particular, we show that CBFs—tools from the control theory literature whose goal is to guarantee long-term constraint satisfaction—can be incorporated as inductive biases in end-to-end visuomotor policy learning problems to make the resulting policies safe. We tackle the important problem of *distributional shift* of data-driven control policies: these policies can behave unexpectedly when far from the data distribution. This chapter builds a self-supervised framework to constrain the state from a real system from entering out-of-distribution regions.

## 6.1 Introduction

The modern advances in the representation learning literature have been an enabling factor for the recent surge of a wide variety of methods for robotic control directly from images or high-dimensional sensory observations [202, 57, 79, 118, 222, 197]. These approaches for visuomotor planning and control have the potential to solve challenging tasks in which the state of the system might not be directly observable, or even not possible to model analytically. While promising, the high-dimensionality of the problem make these methods susceptible to several open challenges. For example, the exploration requirements of rein-

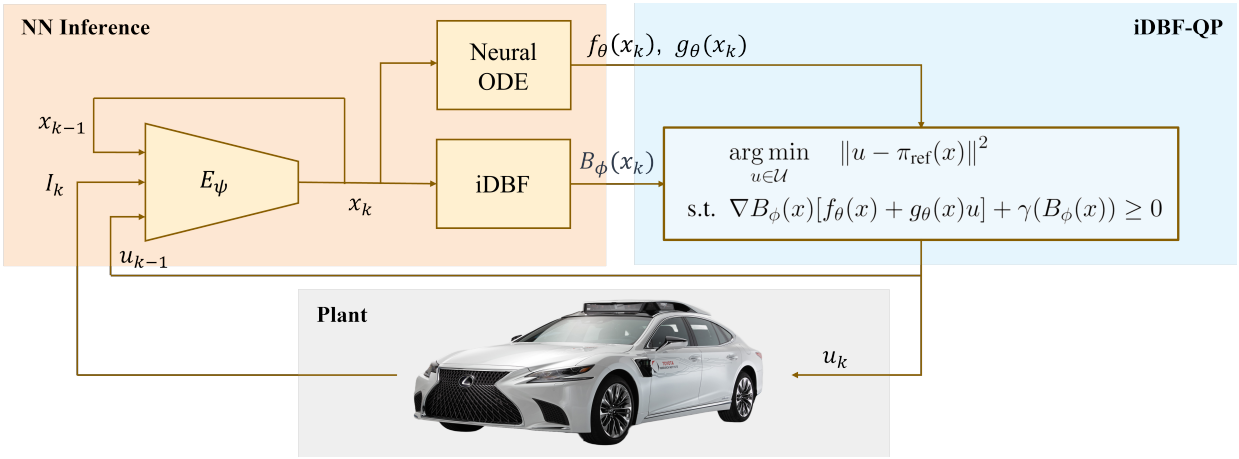



Figure 6.1: Our framework's inference diagram. At each timestep, based on the current observation $I_k$ and the previous latent state and action, the encoder network $E_\psi$ outputs a new latent state $x_k$. Then, the in-Distribution Barrier Function (iDBF) and dynamics networks give the values of $B_\phi(x_k)$, $f_\theta(x_k)$ and $g_\theta(x_k)$ which are passed to the iDBF-QP policy filter. The iDBF-QP takes a reference control input for the current timestep $\pi_{\text{ref}}(x_k)$ and returns the closest action that keeps the system in-distribution with respect to the offline-collected dataset of safe demonstrations. For both of the examples of Section 6.5, the total inference time (NN Inference + solving the iDBF-QP) of our framework is less than 5 milliseconds.

forcement learning (RL) algorithms are significantly exacerbated for these tasks, due to the high-dimensionality of the observations. This means that trying to learn safe control policies using RL often requires us to accept that abundant failures will occur during training. On the other hand, supervised learning approaches for control, such as behavioral cloning, would in principle seem less prone to exhibit unsafe behaviors. However, it is well known that simply because of their data-dependent nature, these methods are still susceptible to a key challenge named *distributional shift*: if the trajectories of the system divert from the training data distribution, the controller might take unexpected actions.

On the other hand, the control theory literature extensively covers the problem of long-horizon constraint satisfaction. In particular, Control Barrier Functions (CBFs, [6]) are a popular model-based tool used to restrict the trajectories of the system from entering undesirable regions of the state-space. One of the properties of CBFs that explain their recent popularity is that they decouple the problem of constraint-satisfaction from any performance objective. Specifically, if a CBF is available, then [6] showed that we can construct a minimally invasive safety filter that transforms into safety-preserving control actions any unsafe commands that an arbitrary reference policy could output.

The main question we want to address in this work naturally emerges from the previous discussion: can we take inspiration from CBFs to avoid out-of-distribution (OOD) states

when using data-driven controllers for visuomotor tasks? Even though CBFs are model-based tools that, as such, require knowledge of the state-space and dynamics of the system, the recent advances on learning latent state-space representations and associated dynamics models clearly set a path for linking data-driven visuomotor policy learning with the use of model-based control-theoretic tools such as CBFs.

**Contributions**: We present an end-to-end self-supervised approach for learning a task-agnostic policy filter which prevents the system from entering OOD states. We do not assume knowledge of the state-space or system dynamics. In addition, our framework only requires an offline-collected dataset of safe demonstrations (where the concept of safety is only linked to the demonstrator's subjectivity, as it is their responsibility to provide the dataset). We therefore do not require any unsafe demonstrations to learn a safe policy filter, in contrast to most other works tackling constrained policy learning. Furthermore, to the best of our knowledge, this is the first work that uses CBFs for constructing policy filters in learned latent state-spaces. This endows our approach with the flexibility of being applicable to systems with high-dimensional sensory observations, in contrast to most prior CBF-based methods. We present simulation experiments on two different visuomotor control tasks, which suggest that our framework, taking only raw RGB images as input, can learn to significantly reduce the distributional shift from safe demonstrations and, consequently, critically improve the safety of both systems.

### 6.1.1 Related Work

There exist some constructive procedures for synthesizing CBFs based on sum-of-squares programming [93, 132, 51, 201] or Hamilton-Jacobi reachability [39]. However, these methods require knowledge of the dynamics of the system and typically suffer from scalability issues for high-dimensional systems. More in line with this work, some recent results show that CBFs can be learned from data [95, 54, 160, 167, 125, 1, 91]. None of these works, however, consider systems with high-dimensional observations. Furthermore, the works [95, 54, 160] assume a priori knowledge of a control-invariant safe set, and focus on building a CBF for that particular set. The line of research of [167, 125] has the most similar problem setup to our work, as they also consider learning from safe demonstrations. Although, notably, the authors provide formal verification arguments for the learned CBFs, their methods are not applicable to high-dimensional observations, assume a nominal dynamics model is given, and use an algorithmic approach for the detection of the boundary of the dataset that does not scale to large datasets. Other recent approaches build signed distance functions from sensory measurements that are obtained from a LiDAR or stereo cameras [129, 181, 45]. However, these functions are not encouraged to satisfy any set invariance property.

Extensions of the CBF-based control filters to systems with dynamics or measurement-model uncertainty have also been recently proposed [147, 30, 188, 56, 55]. These works assume that a CBF is provided, and formulate uncertainty-robust optimization problems for the controller design. They can be considered complementary to our deterministic but

end-to-end approach. Future work should explore quantifying uncertainty estimates within our framework to robustify the learned policy filters.

Several existing approaches for OOD prevention learn density models of training data that can be then used to restrict the agent from taking low likelihood actions or moving towards unvisited states [136, 166, 218, 113]. Although some of these methods have been shown to be effective at offline RL settings that are specially susceptible to distributional shift, the learned density models have no notion of control invariance and, therefore, do not consider the problem of how to prevent distributional shift over a long time horizon. A notable exception is the work of [99] to constrain long-term distributional shift, in which a min-max Bellman backup operator is constructed so that Lyapunov-like functions arise as value functions of an offline RL problem. This work however does not consider the extension to visuomotor control tasks in learned latent spaces. Furthermore, for our approach we choose not to rely on a min-max backup operator to learn the certificate function and, instead, use the very suitable theory of CBFs to devise a self-supervised learning framework.

Finally, the work of [215] presents a framework to learn safe sets in a latent state-space for iterative control tasks. Compared to this work, our framework has the advantage that it is task-agnostic and does not require any interactions with the environment during training.

## 6.2 Problem Statement

We start by revising some basic concepts about Control Barrier Functions that were already introduced in Chapter 2. CBFs are particularly well-suited for continuous-time nonlinear control-affine systems of the form in (2.2)

$$\dot{x} = f(x) + g(x)u,$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ the control input.

We now remind the readers of the definition of CBF (which was already introduced in Definition 6.1), but phrase it in a convenient way for this chapter, putting each condition that needs to be satisfied for a function to be a CBF as a separate entry in the definition.

**Definition 6.1** (Control Barrier Function, [8])**.** *We say that a continuously differentiable function $B : \mathcal{X} \to \mathbb{R}$ is a* Control Barrier Function (CBF) *for system* (2.2) *with associated safe-set $\mathcal{X}_{safe} \subset \mathcal{X}$ if the following three conditions are satisfied:*

$$B(x) \geq 0 \quad \forall x \in \mathcal{X}_{safe}, \tag{6.1}$$

$$B(x) < 0 \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_{safe}, \tag{6.2}$$

$$\exists u \in \mathcal{U} \quad s.t. \quad \dot{B}(x,u) + \gamma(B(x)) \geq 0 \quad \forall x \in \mathcal{X}, \tag{6.3}$$

*where $\gamma : \mathbb{R} \to \mathbb{R}$ is an extended class $\mathcal{K}_\infty$ function.*

In the CBF literature, safety is considered as a set invariance problem, as presented in Definition 2.6. The existence of a CBF $B$ guarantees that for system (2.2) any Lipschitz continuous control policy $\pi$ satisfying

$$\pi(x) \in \{u \in \mathcal{U} : \underbrace{\nabla B(x)[f(x) + g(x)u]}_{=\dot{B}(x,u)} + \gamma\left(B(x)\right) \geq 0\} \tag{6.4}$$

will render the set $\mathcal{X}_{\text{safe}}$ forward invariant [8, Corollary 2].

For a given task-specific reference controller $\pi_{\text{ref}} : \mathcal{X} \to \mathcal{U}$ that might be safety-agnostic, the condition of (6.4) can be used to formulate an optimization problem that, when solved at every time-step, yields a minimally-invasive policy safety filter [6]:

$$\begin{aligned} \pi_{\text{CBF}}(x) = \underset{u \in \mathcal{U}}{\arg\min} \quad & \|u - \pi_{\text{ref}}(x)\|^2 \\ \text{s.t.} \quad & \nabla B(x)[f(x) + g(x)u] + \gamma(B(x)) \geq 0. \end{aligned} \tag{CBF-QP}$$

Assuming that the actuation constraints that define $\mathcal{U}$ are linear in $u$, this problem is a quadratic program (QP). This is a consequence of the dynamics of the system (2.2) being control-affine, and it practically means that the problem can be solved to a high precision very quickly (around $10^3$Hz). This is critical since the CBF-QP needs to be solved at the real-time control frequency.

The CBF-QP constitutes a very appealing approach for practitioners: it provides a task-agnostic minimally invasive filter that can wrap safety around any given policy $\pi_{\text{ref}}$, and therefore rewrite any unsafe control input that $\pi_{\text{ref}}$ could output at any time. However, designing a valid CBF is nontrivial. In fact, it is still an active research topic even when assuming perfect knowledge of the dynamics of the system [51, 39, 201]. The two main difficulties in the design of a CBF are the following: first, a control-invariant set $\mathcal{X}_{\text{safe}}$ must be obtained (which in general is different from the geometric constraint set that could be obtained, for instance, from a signed-distance field) and, second, a function that satisfies condition (6.3) must be found for that set. Furthermore, even after obtaining a CBF, solving the CBF-QP requires perfect state and dynamics knowledge.

With our framework, we take initial steps towards building a safe policy filter from high-dimensional observations. Specifically, we take inspiration from CBFs to design an end-to-end learning framework to constrain deep learning models to remain in-distribution of the training data. We take as input a dataset of high-dimensional observations of different safe demonstrations, and build a neural CBF-like function that encourages the system to always stay in-distribution with respect to the observations from the safe demonstrations. This, in turn, significantly improves the safety of the system during deployment.

More concretely, for a given dataset of $N$ safe trajectories $\mathbb{D} = \left\{(I_t^i, u_t^i)_{t=0}^{t=T_i}\right\}_{i=1}^{i=N}$ we tackle the problem of designing a policy filter that can be applied to any reference controller $\pi_{\text{ref}}$ to detect and override actions from $\pi_{\text{ref}}$ that lead to OOD states. We denote $I_t^i$ and $u_t^i$ the high-dimensional observation and control input, respectively, measured at time $t$ for the $i$th

trajectory. Furthermore, $T_i$ is the final time-step of trajectory $i$. The demonstrations in the dataset $\mathbb{D}$ might correspond to different tasks and they do not need to be optimal with respect to any objective. In fact, our only assumption is that the dataset only contains safe demonstrations (in the sense that these trajectories should not contain any states from which the system is deemed to fail, even if it has not failed yet), so that we can encourage long-term constraint satisfaction using CBFs.

## 6.3 In-Distribution Barrier Functions

In this section, we introduce a self-supervised approach for synthesizing neural CBF-like functions whose aim is to constrain the system to remain in-distribution with respect to an offline dataset of safe demonstrations. We call these functions *in-Distribution Barrier Functions* (iDBFs). We will for now assume that we have a parametric continuous-time control-affine model of the dynamics of the system in a state-space $\mathcal{X} \subset \mathbb{R}^n$

$$\dot{x} = f_\theta(x) + g_\theta(x)u, \tag{6.5}$$

and present the iDBF learning procedure for this system. Furthermore, for this section we assume that the dataset $\mathbb{D}$ of safe trajectories contains true state measurements, i.e., $\mathbb{D} = \left\{ (x_t^i, u_t^i)_{t=0}^{t=T_i} \right\}_{i=1}^{i=N}$, where $x_t^i$ and $u_t^i$ are the state and control input, respectively, measured at time $t$ for the $i$th trajectory. In Section 6.4, we will provide details on how to learn a dynamics model of this form in a latent state-space when we have a dataset containing high-dimensional sensory observations.

We parameterize an iDBF $B_\phi : \mathcal{X} \to \mathbb{R}$ as a neural network with parameters $\phi$, and construct an empirical loss function that encourages it to satisfy the three CBF conditions (6.1), (6.2) and (6.3) with respect to a set $\mathcal{X}_{\text{safe}}$ that is also implicitly learned through self-supervision. To design the loss function, we take inspiration from previous literature on learning CBFs [54, 160, 32]. Nevertheless, instead of assuming that the safe-set $\mathcal{X}_{\text{safe}}$ is given and that we can sample from it and from its unsafe complement $\mathcal{X}_{\text{unsafe}} \doteq \mathcal{X} \setminus \mathcal{X}_{\text{safe}}$, we build our loss function in a self-supervised manner just from the dataset of safe demonstrations. We accomplish this by leveraging ideas from contrastive learning [75, 151, 40, 206, 172]. In particular, as we explain in detail later, we build a contrastive distribution from which to sample candidate unsafe states, given that we do not have any unsafe demonstrations in our dataset. The loss function we propose for learning an iDBF takes the following form:

$$\begin{aligned}\mathcal{L}_{\text{iDBF}} = \frac{w_{\text{safe}}}{N_{\text{safe}}} \sum_{x_{\text{safe}}} \left[\epsilon_{\text{safe}} - B_\phi(x_{\text{safe}})\right]^+ + \frac{w_{\text{unsafe}}}{N_{\text{unsafe}}} \sum_{x_{\text{unsafe}}} \left[\epsilon_{\text{unsafe}} + B_\phi(x_{\text{unsafe}})\right]^+ + \\ \frac{w_{\text{ascent}}}{N_{\text{safe}}} \sum_{(x_{\text{safe}}, \pi_{\text{safe}})} \left[\epsilon_{\text{ascent}} - \left(\nabla B_\phi(x_{\text{safe}})[f_\theta(x_{\text{safe}}) + g_\theta(x_{\text{safe}})\pi_{\text{safe}}] + \gamma(B_\phi(x_{\text{safe}}))\right)\right]^+, \end{aligned} \tag{6.6}$$

where $[\cdot]^+ \coloneqq \max(0, \cdot)$; $(x_{\text{safe}}, \pi_{\text{safe}})$ are samples from the empirical distribution of the dataset $\mathbb{D}$; $x_{\text{unsafe}}$ are samples from a contrastive distribution that we will define soon; $w_{\text{safe}}$, $w_{\text{unsafe}}$ and $w_{\text{ascent}}$ are the weights of the different loss terms; and $\epsilon_{\text{safe}}$, $\epsilon_{\text{unsafe}}$ and $\epsilon_{\text{ascent}}$ are positive constants that serve to enforce strictly the inequalities and generalize outside of the training data.

The goal of the first two terms in the loss function is to learn an iDBF that has a positive value in states that belong to the data distribution of safe demonstrations, and negative everywhere else (meaning we are encouraging the satisfaction of conditions (6.1) and (6.2) of the definition of CBF). Note that this classification objective is very related to the notion of energy-based models (EBMs) —neural network density models that assign a low energy value to points close to the training data distribution and a high value to points that are far from it [83]. In fact, we took inspiration from the Noise Contrastive Estimation (NCE, [75]) training procedure of EBMs, in particular the InfoNCE loss [151], to design (6.6). Intuitively, these methods use a noise contrastive distribution to generate candidate examples where to increase the value of the energy of the EBM, while decreasing the energy at the training data points. We precisely want the opposite result for our problem: a high value of $B_\phi$ on the training data distribution, and a low value everywhere else. However, we have one additional requirement, which is that the iDBF should have a value of zero at the boundary, as set by conditions (6.1) and (6.2). This is the reason why we design the two first terms of the loss function using the $[\cdot]^+$ operator.

In the third term of the loss (6.6), note that we do not encourage the satisfaction of condition (6.3) over the entire state-space, but only over the dataset of safe demonstrations. However, in the definition of CBF, if condition (6.3) is only satisfied $\forall x \in \mathcal{X}_{\text{safe}}$ instead of $\forall x \in \mathcal{X}$, the CBF still guarantees the control-invariance of $\mathcal{X}_{\text{safe}}$. We are therefore using our empirical data distribution of safe demonstrations as a sampling distribution covering the set $\mathcal{X}_{\text{safe}}$, which we are also implicitly learning as the zero-superlevel set of $B_\phi$. Furthermore, compared to prior approaches that encourage the satisfaction of condition (6.3) for a single policy [54, 160], we instead use all pairs $(x_{\text{safe}}, \pi_{\text{safe}})$ present in the dataset $\mathbb{D}$ to compute this term of the loss. This way, we force the set of admissible control inputs (6.4) to be as large as our dataset allows, reducing the conservatism of the learned iDBF.

In order to generate the contrastive distribution from which to sample $x_{\text{unsafe}}$, as we ultimately want to learn the iDBF in a latent state-space in which it might not be intuitive how to construct a noise distribution, we take the following steps. 1) Based on the dataset of safe demonstrations $\mathbb{D}$, we train a neural behavioral cloning (BC) model that outputs a multimodal Gaussian distribution over actions conditioned on the state, with density $\pi_{\text{BC}}(u|x)$. 2) Then, during the training process of the iDBF, for each $x_{\text{safe}}$ state sampled from $\mathbb{D}$ we randomly take $N_{\text{candidate}}$ control inputs $u_{\text{candidate}}$ and evaluate their density value based on the BC model $\pi_{\text{BC}}(u_{\text{candidate}}|x_{\text{safe}})$. 3) If the value of the density falls below a threshold, then that control input is forward-propagated for one timestep using the dynamics model (6.5) to generate a sample $x_{\text{unsafe}}$. This way, we generate a contrastive data distribution by propagating actions that are unlikely present in the dataset of safe demonstrations. Furthermore, by only propagating these actions for one timestep, the contrastive distribution is close to

the training data, which is desirable for the learning process [76].

## 6.4 Learning iDBFs from High-Dimensional Observations

After introducing the training procedure for an iDBF when the state representation and dynamics model (6.5) are given, we now relax these assumptions and present an approach to learn a latent state-space representation and a continuous-time dynamics model of the form (6.5), suitable to be integrated in the same end-to-end learning framework. We therefore now consider precisely the problem setting described in Section 6.2, in which we only assume having access to a dataset containing observation-action pairs of safe demonstrations $\mathbb{D} = \left\{ (I_t^i, u_t^i)_{t=0}^{t=T_i} \right\}_{i=1}^{i=N}$.

We use an autoencoder architecture to obtain the latent state-space representation, and employ the training procedure of Neural Ordinary Differential Equations (Neural ODEs, [34]) to learn a dynamics model of the form (6.5) in the latent state-space. Note that by enforcing the continuous-time control-affine structure of the dynamics model, we ensure that the iDBF-QP policy filter (equivalent to the CBF-QP, see Figure 6.1) obtained with the learned iDBF and dynamics model will also be a quadratic program.

The inference procedure of our end-to-end learning framework is depicted in Figure 6.1. We use a recursive encoder network $E_\psi$ that takes the current measurement $I_k$, as well as the previous latent state $x_{k-1}$ and action $u_{k-1}$ to generate the new latent state $x_k$ at each timestep $k$. The decoder network $D_\xi$ generates a reconstructed observation $\hat{I}_k$ for each latent state $x_k$. The proposed loss function for the latent state-space representation and dynamics model penalizes both the prediction error of the dynamics model and the observation reconstruction error:

$$\mathcal{L}_{\text{dyn}} = \frac{1}{N_{\text{dyn}}(T_{\text{pred}}+1)} \sum_{j=1}^{N_{\text{dyn}}} \sum_{k=0}^{T_{\text{pred}}} \left[ w_{\text{state}} \left\| \tilde{x}_{t_j+k|t_j} - x_{t_j+k} \right\|^2 + w_{\text{rec}_1} \left\| \tilde{I}_{t_j+k|t_j} - I_{t_j+k} \right\|^2 + w_{\text{rec}_2} \left\| \hat{I}_{t_j+k} - I_{t_j+k} \right\|^2 \right]. \tag{6.7}$$

Here, $x_{t_j+k} = E_\psi(I_{t_j+k}, x_{t_j+k-1}, u_{t_j+k-1})$ is the latent state at timestep $t_j + k$. $\tilde{x}_{t_j+k|t_j}$ denotes the latent state prediction obtained by forward-propagating the dynamics model (6.5) to timestep $t_j+k$ starting from the state $x_{t_j}$ and using zero-order hold on the sequence of control inputs $(u_{t_j}, u_{t_j+1}, ..., u_{t_j+k-1})$. Additionally, $\tilde{I}_{t_j+k|t_j} \coloneqq D_\xi(\tilde{x}_{t_j+k|t_j})$ is the reconstructed observation from the dynamics prediction for timestep $t_j + k$. Finally, $\hat{I}_{t_j+k} \coloneqq D_\xi(x_{t_j+k})$ is the encoded-decoded observation at timestep $t_j + k$.

Note that we use a multiple-shooting error for the loss (6.7), as the prediction horizon $T_{\text{pred}}$ does not need to coincide with the length of the trajectories in the dataset $\mathbb{D}$. In particular, the loss (6.7) is computed by sampling a batch of trajectories from $\mathbb{D}$ and then splitting them into $N_{\text{dyn}}$ portions of length $T_{\text{pred}}$. The initial timestep of each portion $j = 1, ..., N_{\text{dyn}}$

is denoted as $t_j$. The first two terms in the loss function are then penalizing the state and reconstruction error of the multistep predictions of the dynamics model from each initial state $x_{t_j}$. The last term in the loss function penalizes the reconstruction error of the autoencoder directly, without using the dynamics model. The recent results of [15] show that multiple shooting loss functions lead to more accurate predictions compared to single-step prediction losses, and to better conditioned learning problems compared to single-shooting propagation losses.

An iDBF can be learned together with the autoencoder and dynamics model by optimizing jointly the losses (6.6) and (6.7). For the iDBF loss, each $x_{\text{safe}}$ is obtained by encoding the observations sampled from the dataset $\mathbb{D}$, and $x_{\text{unsafe}}$ is obtained by forward propagating the actions that have a low probability according to the pretrained BC model, as explained at the end of last section.

Once the iDBF $B_\phi$; dynamics model $f_\theta$ and $g_\theta$; and encoder $E_\psi$ networks are trained, we can construct a policy filter —which we call iDBF-QP in Figure 6.1— in an equivalent manner to the CBF-QP that was introduced in Section 6.2.

**Remark 6.1.** *It is important to note that our iDBF training procedure encourages the satisfaction of the CBF conditions* (6.1)*,* (6.2) *and* (6.3) *only at a discrete set of training points (which has measure zero). Because of this, we do not have control invariance guarantees for any particular set, and solving the iDBF-QP does not theoretically assure that the system will remain in-distribution. Although obtaining rigorous theoretical guarantees should be a priority for future work, the empirical results of Section 6.5 show that our framework takes a promising first step towards building effective policy filters from raw high-dimensional observations.*

## 6.5 Examples

In this section, we present the empirical evaluation of our framework on two different simulation environments: a toy example of a robot navigation task using top-down images of the scene, and an autonomous driving scenario with egocentric image observations. For both cases, given a safety-agnostic reference controller $\pi_{\text{ref}}$, we use our iDBF-QP at each timestep with the latest image measurement to find the closest control input to $\pi_{\text{ref}}$ among those that prevent the system from entering OOD states (see Figure 6.1). For each environment, we train the iDBF, autoencoder and dynamics model using a dataset containing $64 \times 64$ RGB images of offline-collected trajectories.

**Robot Navigation with Top-Down View Images:** In this example, a circular robot with radius of 1 meter navigates inside of a $10 \times 10$ meter room that has a square-shaped $4 \times 4$ meter static obstacle in the middle, as shown in Figure 6.2 (left). The underlying dynamics of the robot are those of a 2D single integrator, with two control inputs corresponding to the $x$ and $y$ velocity commands, although we do not assume having access to that knowledge. Instead, we only have a dataset of image-action pairs corresponding to 5000 trajectories of

Table 6.1: Evaluation of the collision rate and cumulative filter intervention (a measure of how intrusive the filter is with respect to the reference controller) for the top-down view robotic navigation example (over 20 simulations of 5-seconds each with random initial and goal states) and for the egocentric view autonomous driving example (over 20 simulations of 50-seconds each with random initial heading angles). For the BC and ensemble filters, we provide results for 3 different threshold values: $(p_{\text{low}}, p_{\text{mid}}, p_{\text{high}}) = (0.32, 0.35, 0.38)$ for the navigation example, and $(0.2, 0.5, 0.8)$ for driving; and $(\delta_{\text{low}}, \delta_{\text{mid}}, \delta_{\text{high}}) = (0.0005, 0.001, 0.002)$ for both examples.

| | | $\boldsymbol{\pi_{\text{ref}}}$ | **Ours** | **BC Filter** | | | **Ensemble Filter** | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $p_{\text{low}}$ | $p_{\text{mid}}$ | $p_{\text{high}}$ | $\delta_{\text{low}}$ | $\delta_{\text{mid}}$ | $\delta_{\text{high}}$ |
| **Top-Down Navigation** | Collision Rate (%) | $\mathbf{46.72 \pm 8.36}$ | $\underline{\mathbf{0.28 \pm 0.27}}$ | $\mathbf{35.60 \pm 7.20}$ | $\mathbf{13.86 \pm 4.96}$ | $\mathbf{2.48 \pm 1.57}$ | $\mathbf{43.92 \pm 7.90}$ | $\mathbf{43.82 \pm 7.41}$ | $\mathbf{42.88 \pm 7.41}$ |
| | Cumulative Intervention | $0.0 \pm 0.0$ | $109.2 \pm 20.1$ | $85.6 \pm 6.8$ | $146.4 \pm 9.6$ | $189.1 \pm 11.6$ | $150.2 \pm 19.3$ | $94.5 \pm 16.3$ | $52.7 \pm 11.5$ |
| **Egocentric Driving** | Collision Rate (%) | $\mathbf{81.00 \pm 0.23}$ | $\underline{\mathbf{1.56 \pm 1.20}}$ | $\mathbf{21.94 \pm 1.85}$ | $\mathbf{14.44 \pm 2.69}$ | $\mathbf{8.78 \pm 1.86}$ | $\mathbf{78.74 \pm 0.20}$ | $\mathbf{78.60 \pm 1.63}$ | $\mathbf{81.50 \pm 0.27}$ |
| | Cumulative Intervention | $0.0 \pm 0.0$ | $278.1 \pm 32.6$ | $713.8 \pm 1.4$ | $726.7 \pm 2.6$ | $750.8 \pm 6.9$ | $28.7 \pm 2.1$ | $42.8 \pm 3.6$ | $208.9 \pm 5.7$ |

100 points each (corresponding to 2 seconds since the time-step is 0.02s). These trajectories satisfy two requirements: 1) the robot should never collide against the obstacle, and 2) the center of the robot should never leave the room limits. The trajectories are collected applying random actions at each time-step, and we check both conditions before adding a trajectory to the dataset. We use our framework to train an autoencoder with latent state-space of dimension 3, a dynamics model, and an iDBF. The reference policy $\pi_{\text{ref}}$ simply applies a velocity in the direction of a goal-point, with magnitude proportional to the distance. In Figure 6.2, we show the results of applying our iDBF-QP when the goal state (marked with an $\times$) is outside of the room limits and at the other side of the obstacle. Even though the reference controller is trying to take the shortest path, which would go through the obstacle, the iDBF-QP prevents the robot from first, colliding with the obstacle, and second, from having its center exit the room limits.

**Autonomous Driving with Egocentric View Images:** We use the environment provided by [98], which is based on the Bullet physics simulator and the Panda3d graphics engine [66] to obtain egocentric RGB image measurements. The car navigates in a corridor which has four 90-degree turns to form a square-shaped center-line. One of such turns is shown in the snapshots of Figure 6.3. The car has two control inputs: the desired forward velocity and the steering angle. Given the high-order dynamics of the simulator, we collect data manually to make sure no trajectories included in the dataset are deemed to collide with any of the walls. We split the collected data into 450 trajectories of 100 points each (5 seconds since the timestep is 0.05s). This makes for a much sparser and less diverse (since it is collected by a human) dataset compared to the previous example. During deployment, we use a reference controller $\pi_{\text{ref}}$ that simply drives the car forward at a constant speed of $3.5m/s$. Our iDBF-QP framework of Figure 6.1, taking the latest egocentric RGB measurement as

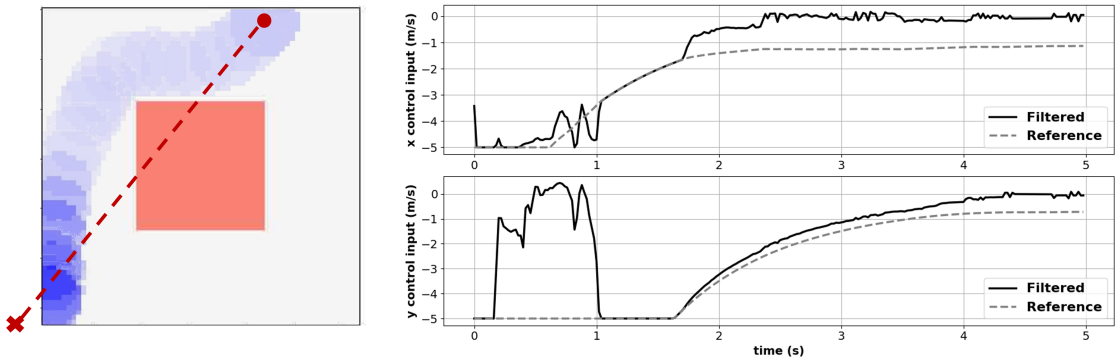



Figure 6.2: Example result using our proposed policy filter for a robot top-down visual navigation task. The reference controller simply tries to bring the robot (blue circle) to a goal state (denoted with $\times$). Our proposed filter, by keeping the system in-distribution, prevents the robot from colliding against the obstacle (orange square) and keeps its center-point inside the limits of the image. A video with several demonstrations of our approach for this task can be found in this link.

input, is very effective at preventing the car from colliding against the walls, as shown in Table 6.1. Figure 6.3 contains snapshots of our iDBF-QP forcing the car to take a turn as it approaches a corner, even though the reference command is to drive forward.

Using these simulation environments we also aim to compare our proposed approach with other techniques for avoiding distributional shift. Other works that consider this problem use data density models to constrain the learned policies [166, 136, 218], or use uncertainty estimation schemes, such as ensemble models, to avoid taking actions that lead to highly uncertain states [42]. We build our baselines upon a conditional BC density model of the training data and an ensemble of latent state-space dynamics models:

**BC Density Filter Baseline:** As explained in Section 6.3, we train a BC multi-modal Gaussian model that is used to generate the contrastive training distribution for the iDBF. For any state, the BC model outputs a probability distribution over actions, with density function $\pi_{\text{BC}}(u|x)$. We train this BC model using privileged true-state information of the system, and use its density values to build a filter that serves as an apples-to-apples baseline comparison to our approach. Specifically, the baseline also takes the reference controller $\pi_{\text{ref}}$ and, at every timestep, it finds the closest control action to $\pi_{\text{ref}}(x)$ that satisfies $\pi_{\text{BC}}(u|x) \geq p$, out of 200 randomly sampled actions. If no control action satisfying that condition is found, the reference control input is applied without filtering. Given the clear dependence on the threshold value $p$, we implement this baseline for several values of $p$ and show the results in Table 6.1 for three representative cases $p_{\text{low}}$, $p_{\text{mid}}$ and $p_{\text{high}}$.

**Ensemble Variance Filter Baseline:** We also train an ensemble of independent latent state-space dynamic models ($f_\theta$ and $g_\theta$), keeping the rest of the framework introduced in Section 6.4 unchanged. During deployment, at every timestep we look for the closest control

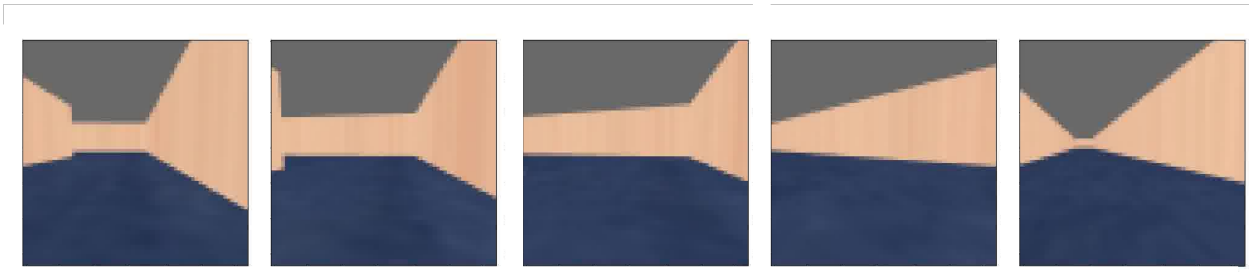


Figure 6.3: Snapshots of egocentric view images of a driving simulation when the car is approaching a corner. The reference controller just commands the car to drive straight, but our iDBF-QP policy filter forces a left turn as the car approaches the corner. Therefore, our filter prevents a collision as a result of staying in-distribution with respect to the safe training data. A video with several demonstrations of our approach for this task can be found in this link.

action to $\pi_{\text{ref}}(x)$ that keeps the variance $\sigma^2_{\text{ens}}(x,u)$ of the predicted dynamics $f_\theta(x) + g_\theta(x)u$ under a threshold $\delta$. As in the previous baseline, we also look over 200 randomly sampled actions at each timestep, and different threshold levels $\delta_{\text{low}}$, $\delta_{\text{mid}}$ and $\delta_{\text{high}}$. Again, if no control action satisfying the threshold condition is found, the reference control input is applied without filtering.

In Table 6.1, we provide a summary of the comparison results for both environments. We use the collision rate as a proxy for distributional shift, since the training data only includes collision-free trajectories. The collision rate for the robot navigation example is computed as the fraction of time that the robot spends either in collision with the obstacle or having its center-point outside of the room limits. For the driving scenario, the collision rate is the fraction of time that the robot is in collision with any of the walls. For both examples, our method drastically reduces the collision rate compared to using the reference (unfiltered) controller. Furthermore, we achieve the lowest collision rates when compared to the baselines. From the baselines, only the BC density filter (with a very restrictive threshold $p_{\text{high}}$) manages to achieve small collision rates, at the cost of a very high cumulative filter intervention rate. The filter intervention rate is computed for both examples as $\sum_t \|u_t - \pi_{\text{ref}}(x_t)\|^2$, where each control input dimension is normalized between $-1$ and 1.

## 6.6 Chapter Summary

In this chapter, we have taken first-steps towards merging control-theoretic CBFs with practical robotic tasks that involve high-dimensional perception modules. While existing methods based on CBFs require a known low-dimensional state representation, our proposed approach is directly applicable to systems that rely solely on high-dimensional visual observations by

learning in a latent state-space. We considered a realistic problem setting in which no unsafe demonstrations are available, and proposed a self-supervised learning approach to learn a policy filter that effectively restricts the system from diverging towards OOD states.

By learning this filter in a latent state-space, our framework should be flexible-enough to be applicable to a wide variety of visuomotor tasks, and should be compatible with the use of large-scale pretrained representation learning models. In this chapter, we have demonstrated that our method is effective for two different visuomotor control tasks in simulation environments, including both top-down and egocentric view settings.

# Part II

# Leveraging Data to Safely Bridge the Reality Gap

# Chapter 7

# Reinforcement Learning for Feedback Linearization Controllers under Model Uncertainty

*This chapter is based on the paper titled "Improving Input-Output Linearizing Controllers for Bipedal Robots via Reinforcement Learning" [28], co-authored by Mathias Wulfman, Ayush Agrawal, Tyler Westenbroek, S. Shankar Sastry, Claire J. Tomlin and Koushil Sreenath.*

In this second part of the thesis, we tackle the challenges that model-based controllers face due to model uncertainty. In particular, we propose a set of methods that use data to bridge the gap between an approximate mathematical model of the system and the real world.

This chapter specifically focuses on using reinforcement learning to compensate the effects of model mismatch when trying to feedback-linearize a nonlinear system. Since feedback linearization controllers are very commonly used for bipedal walking robots through the framework of Hybrid Zero Dynamics, we focus this chapter on this particular application.

## 7.1 Introduction

### 7.1.1 Motivation

Research on humanoid walking robots is gaining in popularity due to the robots' medical applications as exoskeletons for people with physical disabilities and their usage in dangerous disaster and rescue missions. Model-based controllers have traditionally been applied to obtain stable walking controllers but, in general, they heavily rely on having perfect model knowledge and unlimited torque capacity. In this chapter we take a data-driven approach to address these two topics of current research interest which still constitute challenges in bipedal robot control: uncertainty in the dynamics and input saturation.

### 7.1.2 Related work

Input-output linearization is a nonlinear control technique that can be used to get the outputs of a nonlinear system to track desired reference trajectories in a simple manner. By introducing an appropriate state transformation, this control technique permits rendering the input-output dynamics linear. Afterward, linear systems control theory can be used to track the desired outputs. However, input-output linearization requires precise knowledge of the system's dynamics, which directly conflicts with the fact that actual systems' dynamics might have nonlinearities that can be extremely challenging to model precisely. Several efforts have been made to address this issue, using different methods including robust and adaptive control techniques [145, 171, 47, 170] or, more recently, data-driven learning methods [186, 209]. This chapter will take the later approach to address this challenge, specifically combining reinforcement learning (RL) and the Hybrid Zero Dynamics (HZD) method for getting bipedal robots to walk.

The high nonlinearity, underactuation and hybrid nature of bipedal robotic systems pose additional problems that need to be addressed. The virtual constraints and HZD methods [71, 214, 212, 140] provide a systematic approach to designing asymptotically stable walking controllers if there is full model knowledge. These methods have been very successful in dealing with the challenging dynamics of legged robots, being able to achieve fast enough convergence to guarantee stability over several walking steps. By the HZD method, a set of output functions is chosen such that, when they are driven to zero, a time-invariant lower-dimensional zero dynamics manifold is created. Stable periodic orbits designed on this lower-dimensional manifold are also stable orbits for the full system under application of, for instance, input-output linearizing [179], or Control Lyapunov Function (CLF)-based controllers [9]. The later is based on solving online quadratic programs, whereas the former approach does not rely on running any kind of online optimization. The CLF-based method has also been successful in taking into account torque saturation [65], but it assumes perfect model knowledge too. In fact, taking input saturation into account is of major importance and not doing it is one of the main disadvantages of input-output linearization controllers that is often overlooked.

In this work, we build on the formulation proposed in [209] wherein policy optimization algorithms from the RL literature are used to overcome large amounts of model uncertainty and learn linearizing controllers for uncertain robotic systems. Specifically, we extend the framework introduced in [209] to the class of hybrid dynamical systems typically used to model bipedal robots using the HZD framework. Unlike the systems considered in [209], here we must explicitly account for the effects of underactuation when designing the desired output trajectories for the system to ensure that it remains stable. Additionally, we demonstrate that a stable walking controller can be learned even when input constraints are added to the system. By focusing on learning a stabilizing controller for a single task (walking), we are able to train our controller using significantly less data than was used in [209], where it was trained to track all possible desired output signals.

### 7.1.3 Contributions

The contributions of this chapter thus are:

- We extend the work in [209] to the case of hybrid, underactuated bipedal robots with input constraints.
- We directly address the challenge of dealing with a statically unstable underactuated system, designing a new training strategy that uses a finite-time convergence feedback controller to track desired walking trajectories.
- We perform Poincaré analysis to claim local exponential stability of our proposed RL-enhanced input-output linearization controller in the presence of torque saturation.

## 7.2 Feedback Linearization

We now give a brief background on an important class of controllers for nonlinear systems: those that use feedback control to linearize the dynamics. The advantage of using this method is very clear as, after cancelling out the nonlinearities of the dynamics, linear systems theory can be used to describe the behavior of the resulting linear system. In this chapter, we consider nonlinear control-affine systems of the form in (2.2), but explicitely also define a set of outputs:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ y &= h(x), \end{aligned} \tag{7.1}$$

with $x \in \mathcal{X} \subset \mathbb{R}^n$ being the system state, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ the control input and $y \in \mathbb{R}^m$ the outputs of the system, assuming there are the same number of outputs as inputs. We make the standard assumption that the vector fields $f$, $g$ and $h$ are Lipschitz continuous. Then, if the vector relative degree of the outputs is $r$, we have

$$y^{(r)} = L_f^r h(x) + L_g L_f^{r-1} h(x) u, \tag{7.2}$$

where the functions $L_f^r h$ and $L_g L_f^{r-1} h$ are known as $r^{th}$ order Lie derivatives. More information about high-order Lie derivatives can be found in [169]. Here, $y^{(r)}$ is the vector of $r^{th}$ derivatives of each output in $y$, and (7.2) indicates that the inputs $u$ first appear at the $r^{th}$ derivative of these outputs, and not for lower derivatives. If $L_g L_f^{r-1} h \neq \mathbf{0}$ and nonsingular $\forall\ x \in \mathcal{X}$, with $\mathcal{X} \subset \mathbb{R}^n$ being a compact subset containing the origin, then we can use a feedback control law $\pi_{\text{FL}}$ which renders the input-output dynamics of the system linear

$$\pi_{\text{FL}}(x, \mu) = \pi_{ff}(x) + \left(L_g L_f^{r-1} h(x)\right)^{-1} \mu, \tag{7.3}$$

where $\pi_{ff}$ is the feedforward term

$$\pi_{ff}(x) = -\left(L_g L_f^{r-1} h(x)\right)^{-1} L_f^r h(x) \tag{7.4}$$

and $\mu$ is the auxiliary input.

Using this control law yields the input-output linearized system $y^{(r)} = \mu$, and we can define a state transformation $\Phi : x \to (\eta, z)$, with

$$\eta = [h(x)^\top, L_f h(x)^\top, ..., L_f^{r-1} h(x)^\top]^\top \tag{7.5}$$

and $z \in \mathcal{Z}$, where $\mathcal{Z} = \{x \in \mathcal{X} |\ \eta \equiv 0\}$ is the zero-dynamics manifold. The closed-loop dynamics of the system can then be represented as a linear time invariant system on $\eta \in \mathcal{T}$ and the zero-dynamics on $z \in \mathcal{Z}$:

$$\begin{cases} \dot{\eta} = F\eta + G\mu, \\ \dot{z} = \kappa(\eta, z), \end{cases} \tag{7.6}$$

where

$$F = \begin{bmatrix} 0 & I_m & . & . & 0 \\ 0 & 0 & I_m & . & 0 \\ . & . & . & & \\ 0 & . & . & . & I_m \\ 0 & . & . & . & 0 \end{bmatrix} \text{ and } G = \begin{bmatrix} 0 \\ . \\ . \\ 0 \\ I_m \end{bmatrix}, \tag{7.7}$$

with $F \in \mathbb{R}^{mr \times mr}$ and $G \in \mathbb{R}^{mr \times m}$.

So, independently of the choice of the additional input $\mu$, the control law of (7.3) input-output linearizes the dynamics of the system (7.1). The additional input $\mu$ is typically set as a feedback of the transverse dynamics $\eta$.

## 7.3 Effects of Uncertainty on the Feedback Linearization

In this section, we study the case in which there is a mismatch between the model and the actual plant dynamics and derive the effects of this uncertainty on the input-output linearization of the plant. Now, plant and model are represented by:

| (Unknown) Plant Dynamics | (Known) Model Dynamics |
|---|---|
| $\begin{cases} \dot{x} = f(x) + g(x)u, \\ y = h(x), \end{cases}$ (7.8) | $\begin{cases} \dot{x} = \tilde{f}(x) + \tilde{g}(x)u, \\ y = h(x). \end{cases}$ (7.9) |

We assume: 1) the vector fields $f$, $g$, $\tilde{f}$, $\tilde{g}$ are Lipschitz continuous and 2) the vector relative degrees of the model and plant dynamics are the same $(r)$. These are the standard assumptions that have been made in most of the literature [146, 209, 186, 187] to tackle the mismatch terms analytically.

The feedback linearization control law (7.3) computed based on the nominal model $(\tilde{f}, \tilde{g})$ has the following form

$$\tilde{\pi}_{\mathrm{FL}}(x, \mu) = \tilde{\pi}_{ff}(x) + \left(L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x)\right)^{-1} \mu, \tag{7.10}$$

with a feedforward term

$$\tilde{\pi}_{ff}(x) := -\left(L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x)\right)^{-1} L_{\tilde{f}}^{r} h(x). \tag{7.11}$$

Using the control law of (7.10) in (7.2) results in the input-output dynamics of the true plant

$$y^{(r)} = \mu + \Delta_1(x) + \Delta_2(x)\mu, \tag{7.12}$$

with the uncertainty terms

$$\begin{aligned} \Delta_1(x) :&= L_f^r h(x) - L_g L_f^{r-1} h(x) \left(L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x)\right)^{-1} L_{\tilde{f}}^{r} h(x), \\ \Delta_2(x) :&= L_g L_f^{r-1} h(x) \left(L_{\tilde{g}} L_{\tilde{f}}^{r-1} h(x)\right)^{-1} - I_m. \end{aligned} \tag{7.13}$$

The dynamics of $\eta$ from (7.6) now become:

$$\dot{\eta} = (F\eta + G\Delta_1(\eta, z)) + G\left(I_m + \Delta_2(\eta, z)\right)\mu. \tag{7.14}$$

Note that this equation is the same as (7.6) if the uncertainty terms are zero. Therefore, (7.6) can be considered as a nominal model for the true input-output linearized dynamics (7.14).

## 7.4 Reinforcement Learning for Uncertainty Compensation

In this section we will use RL to define an additive additional input whose goal is to cancel out the uncertainty terms present in the input-output linearized dynamics (7.14), and therefore get the transverse dynamics to behave like (7.6), as done in [209].

If instead of (7.10) we now apply the input

$$\hat{\pi}_\theta(x, \mu) = \tilde{\pi}_{\mathrm{FL}}(x, \mu) + \pi_\theta(x, \mu), \tag{7.15}$$

with $\tilde{\pi}_{\mathrm{FL}}$ as defined in (7.10) and with

$$\pi_\theta(x, \mu) := \alpha_\theta(x)\mu + \beta_\theta(x), \tag{7.16}$$

to (7.2), we obtain

$$y^{(r)} = \mu + \left(\Delta_1(x) + L_g L_f^{r-1} h(x)\beta_\theta(x)\right) + \left(\Delta_2(x) + L_g L_f^{r-1} h(x)\alpha_\theta(x)\right)\mu, \tag{7.17}$$

where $\theta \in \Theta \subset \mathbb{R}^N$ are parameters of a neural network that are to be learned.

If we manage to obtain $y^{(r)} = \mu$ then we will have input-output linearized the dynamics of the true plant, so we can now clearly see the goal of the RL agent for this approach: design policies $\alpha_\theta$ and $\beta_\theta$ such that $y^{(r)}$ is as close as possible to $\mu$. Thus, the time-wise loss function can be defined as

$$l_{IO,\theta} := ||y^{(r)} - \mu||^2 \ . \tag{7.18}$$

## 7.5 Implementation for Bipedal Walking

Now, we will formulate our problem as a canonical RL problem [183] for the specific case of bipedal walking. In this chapter, we model the bipedal walking robot as a hybrid dynamical system of the form

$$\mathcal{H} = \begin{cases} \dot{x} = f(x) + g(x)u, & x \notin \mathcal{S}, \\ x^+ = \Delta(x^-), & x^- \in \mathcal{S}, \end{cases} \tag{7.19}$$

where $\mathcal{S}$ is the switching surface (when the swing foot impacts the ground).

For the specific case of walking and having outputs of relative degree $r = 2$, meaning that $\eta = [y, \dot{y}]$, a typical choice for the additional control input $\mu$ is the nonlinear feedback law

$$\mu(\eta) = \tfrac{1}{\epsilon^2}\psi_a(y, \epsilon\dot{y}), \quad \text{with} \quad \begin{cases} \psi_a(y, \epsilon\dot{y}) = -sign(\epsilon\dot{y})|\epsilon\dot{y}|^a - sign(\phi_a(y, \epsilon\dot{y}))|\phi_a(y, \epsilon\dot{y})|^{\frac{a}{2-a}}, \\ \phi_a(y, \epsilon\dot{y}) = y + \dfrac{1}{2-a} sign(\epsilon\dot{y})|\epsilon\dot{y}|^{2-a}. \end{cases} \tag{7.20}$$

This $\mu$ ensures finite time convergence to the zero-dynamics manifold $Z$ and $\epsilon$ controls the rate of convergence [213]. Then, using outputs of relative degree $r = 2$, we can rewrite the time-wise loss function in (7.18) as

$$l_{IO,\theta}(x) := ||\ddot{y} - \mu(x)||^2 \ . \tag{7.21}$$

For the term $\ddot{y}$ present in the loss function, we use a finite difference approximation of the second derivative of the outputs of the plant.

Even though only $\alpha_\theta$ and $\beta_\theta$ are learned, for the sake of simplicity let $\hat{\pi}_\theta : x \mapsto \hat{\pi}_\theta(x)$ be our policy taking the current state $x$ and returning the control action $u$ as defined in (7.15), and let the reward for a given state $x$ be $R(x, \hat{\pi}_\theta) = -l_{IO,\theta}(x) + R_e(x)$, where $R_e(x)$ is a penalty value if the state $x$ is associated with a fallen robot configuration or a bonus value

otherwise. Then, we can define the learning problem as

$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{x_0 \sim X_0, w \sim \mathcal{N}(0,\sigma^2)} \int_0^T R(x(\tau), \hat{\pi}_\theta(\tau)) d\tau, \\ \text{s.t.} \quad & \dot{x} = f(x) + g(x)(\hat{\pi}_\theta(x) + w_t), \\ & u_{min} \leq \pi_\theta(x) \leq u_{max}, \end{aligned} \tag{7.22}$$

where $X_0$ is the initial state distribution, $T > 0$ is the duration of the episode, $w$ is an additive zero-mean noise term and $u_{min}$ and $u_{max}$ are the torque limits. An episode ends when the robot completes an entire step or when it falls. Discrete-time approximations of this problem can be solved using standard on-policy and off-policy RL algorithms. Note that our proposed controller (7.15) with the chosen loss (7.18) and the inclusion of input constraints in the optimization (7.22) addresses the classical challenges of input-output linearization: model uncertainty and input constraints. For simplicity, we will call *original IO controller* the one of (7.10) and *RL-enhanced IO controller* the one of (7.15).

## 7.6 Numerical Validation

### 7.6.1 System Description

For the simulations, we use a model of the five-link planar robot RABBIT [37], wherein the stance phase is parametrized by a suitable set of coordinates (Fig. 3.1). RABBIT is a 7 Degrees-of-Freedom (DOF) underactuated system with 4 actuated DOF, with the actuators being located at the four joints (the two hip joints and the two knee joints). The dynamics of this 14-dimensional system are extremely coupled and nonlinear.

### 7.6.2 Reference Trajectory Generation

In order to generate a reference trajectory offline, we use the Fast Robot Optimization and Simulation Toolkit (FROST) [81]. The four actuated DOF ($q_1$, $q_2$, $q_3$ and $q_4$) are virtually constrained to be Bézier Polynomials of the stance leg angle $\theta = q_5 + q_1 + 0.5q_3$, which is monotonically increasing during a walking step. This way, the trajectory that has been generated is time-invariant, which makes the controlled system more robust to uncertainties [213]. Taking the difference between the actual four actuated joint angles and the desired ones (coming from the reference trajectory) as output functions $y$, the system is input-output linearizable with vector relative degree two. Consequently, we can use the *RL-enhanced IO controller* presented in the previous section.

We train our controller using a Deep Deterministic Policy Gradient Algorithm (DDPG) [175]. DDPG is used to tune the parameters of the actor and critic feedforward neural networks. They each have two hidden layers of widths 400 and 300 and ReLU activation functions. The actor neural network maps 14 observations, which are the states of the robot

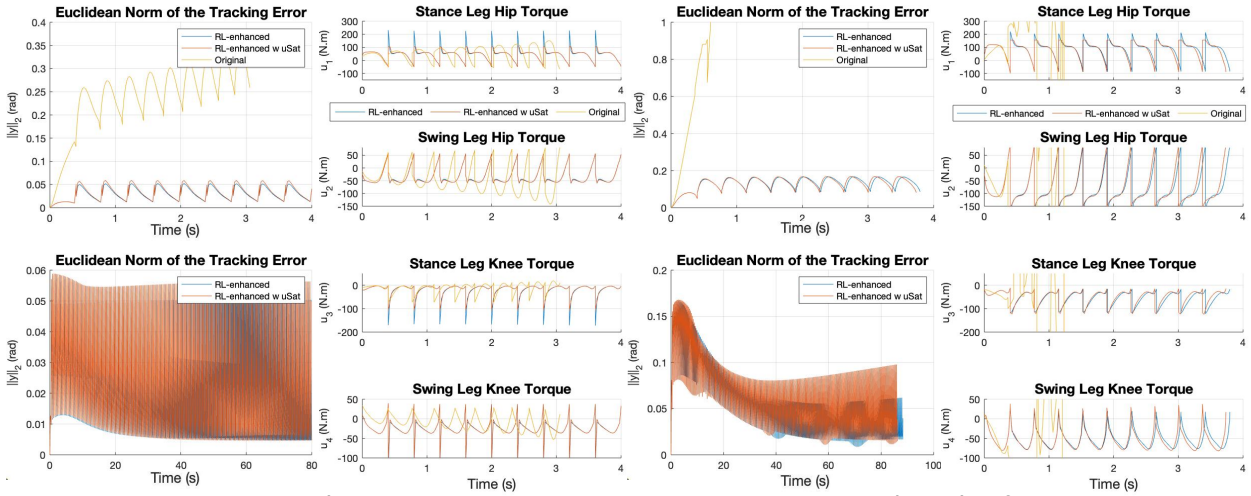



Figure 7.1: Euclidean norm of the tracking error (for 10 and 200 steps) and joint torques (for 10 steps), for the *original IO controller* (yellow), the *RL-enhanced IO controller* (blue), and the *RL-enhanced IO controller* with torque saturation (red). Torque saturation for the *RL-enhanced IO controller* is set at 105 $Nm$ when $scale = 1.5$, and at 155 $Nm$ when $scale = 3$. There is no torque saturation for the *original IO controller*.

—we are assuming a fully observable environment— to 20 outputs, corresponding to the $4 \times 4$ $\alpha_\theta$ and the $4 \times 1$ $\beta_\theta$.

### 7.6.3 Model-Plant Mismatch and Torque Saturation Simulation Results

We introduce model uncertainty by scaling all the masses and inertia values of the plant's links by some factor (*scale*) with respect to the known model. After about twenty minutes of training when the *scale* is 1.5 and about an hour when the *scale* is 3, we obtain the results shown in Fig. 7.1, in which we compare the tracking error and the joint torques when using (i) the *original IO controller*, (ii) the *RL-enhanced IO controller* without torque saturation and (iii) the *RL-enhanced IO controller* when there is torque saturation. For these results we did not need to include torque saturation in the training process, and Fig. 7.1 shows that the *RL-enhanced IO controller* still performs well in the presence of input constraints if they are not too severe. The beneficial effects of including torque saturation constraints during training will be discussed later.

It can be observed that the *RL-enhanced IO controller* with and without saturation is able to stabilize the system indefinitely each time, whereas the *original IO controller* accumulates

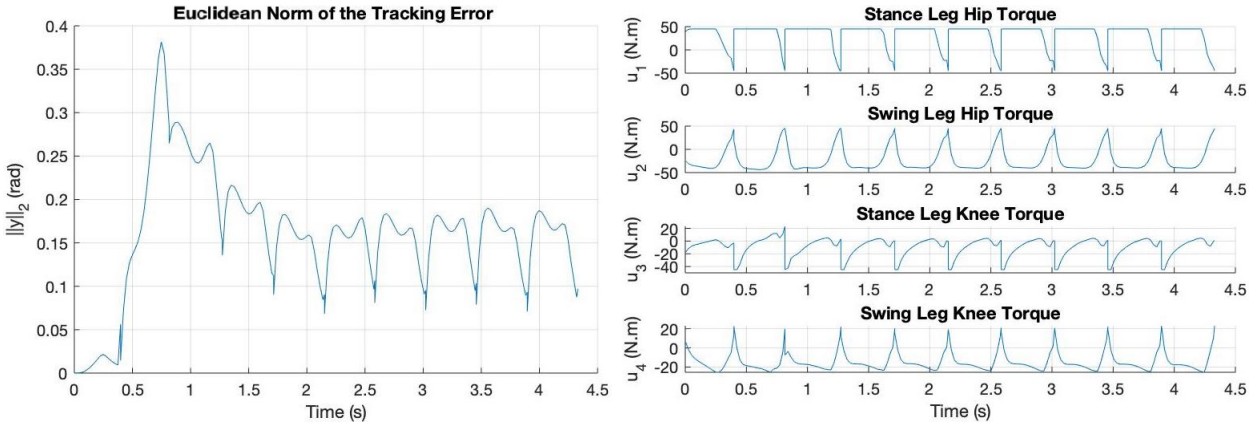



Figure 7.2: *RL-enhanced IO controller* with torque saturation at 45 $Nm$ and $scale = 1$. Euclidean norm of the tracking error (left) and joint torques (right) for a simulation of 10 walking steps. The *original IO controller* fails after one step and is not shown in this figure.

error on the outputs and the robot falls after some steps. Moreover, the *RL-enhanced IO controller* achieves this without increasing the magnitude of the torques when compared with the *original IO controller*.

The stability of the periodic gait obtained using the *RL-enhanced IO controller* can also be studied by the method of Poincaré. We consider the post-impact double stance surface, $S_{DS}$, as a Poincaré section, and define the Poincaré map $P : S_{DS} \to S_{DS}$. We can numerically calculate the eigenvalues of the linearization of the Poincaré map about the obtained periodic gait, which results in a dominant eigenvalue of magnitude 0.67 for $scale = 1.5$ and no torque saturation, 0.78 for $scale = 1.5$ with torque saturation, 0.76 for $scale = 3$ and no torque saturation and 0.83 for $scale = 3$ with torque saturation. The magnitude of the dominant eigenvalue being always less than one means that the designed controllers achieve local exponential stability [213].

Next, we study the case of having no mismatch between the plant and the model dynamics but, instead, having heavy input constraints in the torques, which make the *original IO controller* fail. By training while taking into account the torque saturation, we obtain a *RL-enhanced IO controller* that achieves stable walking under the presence of severe input constraints, as shown in Fig. 7.2.

### 7.6.4 Tracking Untrained Trajectories

Depicted in Fig. 7.3 are the tracking errors and torques produced by the *RL-enhanced IO controller* for a *scale* of 3 when it is trying to follow periodic orbits it was not trained on. These trajectories differ from the one used for the training (*trajectory 1*) in the maximum hip height during a step. As it can be seen in the left part of Fig. 7.3, *trajectory 2* and *trajectory 1* are relatively similar, whereas *trajectory 3* constitutes a noticeably different

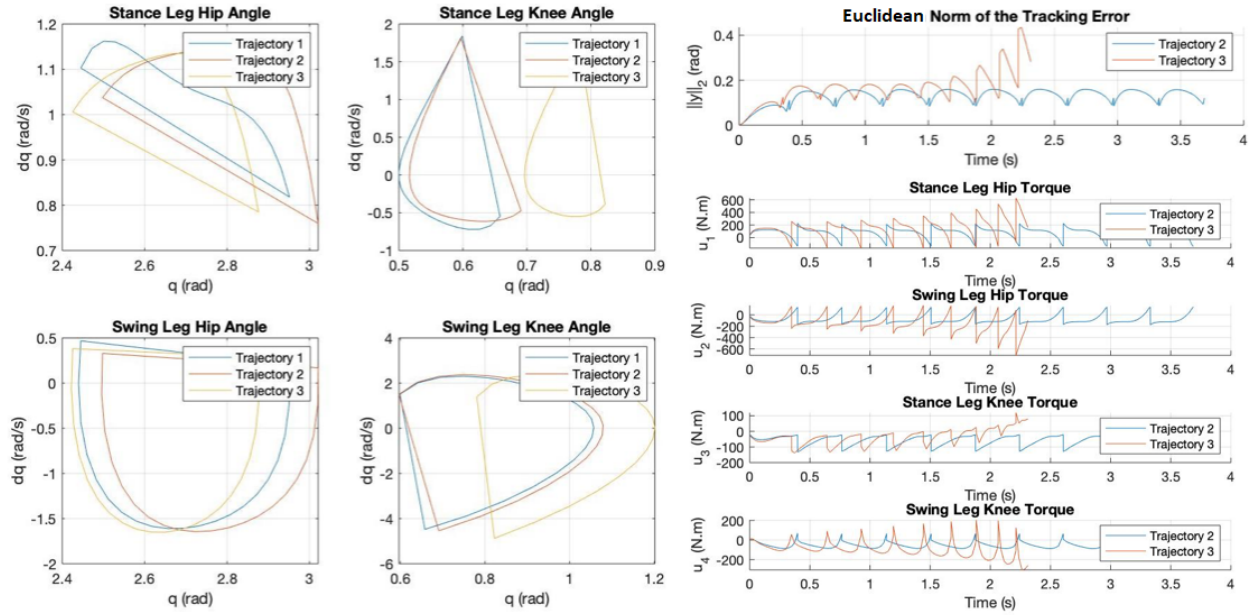



Figure 7.3: Left: Phase portrait of the periodic orbits. Right: Euclidean norm of the tracking error and joint torques for a simulation of 10 steps on untrained trajectories.

walking gait. From the figures, we can see that the *RL-enhanced IO controller* performs better when tested in *trajectory 2* than in *trajectory 3*. Actually, it will be able to stably track *trajectory 2* for an indefinitely long horizon and not *trajectory 3*. This was expected, since the more different the trajectory is, the farther the state of the robot will be from the distribution of states the DDPG agent has been trained on. Also, the output functions we have defined depend on the Bézier coefficients of the reference trajectory, and so the actual input-output linearizing controller is different for each trajectory. Still, thanks to training the DDPG agent on a stochastic distribution of initial states, we get enough exploration to achieve good tracking performance on untrained trajectories as long as they are not too different from the one the agent was trained on.

## 7.7 Chapter Summary

In this chapter, we presented a framework for improving an input-output linearizing controller for a bipedal robot when uncertainty in the dynamics and input constraints are present. We demonstrated the effectiveness of this approach by testing the learned controller on the hybrid, nonlinear and underactuated five-link walker RABBIT.

# Chapter 8

# Reinforcement Learning for CLF and CBF-based Controllers under Model Uncertainty

*This chapter is based on the paper titled "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions" [38], co-authored by Jason Choi, Claire J. Tomlin and Koushil Sreenath.*

In this chapter, we address the challenges that another important class of nonlinear controllers face due to model uncertainty: Control Barrier Function and Control Lyapunov Function-based controllers. In particular, we again use data collected from the real system to overcome the adverse effects of the reality gap through a reinforcement learning framework. We proposed several methods to mitigate these effects, and validate them in numerical simulations of a RABBIT 3.1 bipedal robot.

## 8.1 Introduction

In this chapter, we propose an approach that benefits from the recent successes of learning-based control in highly uncertain dynamical systems, such as in [87, 119], yet is also able to reason about safety in a formal way. We seek to combine the benefits of these data-driven approaches with the benefits of classical model-based control methods which have theoretical guarantees on stability and safety. Towards this end, we use Control Lyapunov Function and Control Barrier Function-based controllers designed on nominal systems that are then trained through reinforcement learning (RL) to work on systems with uncertainty.

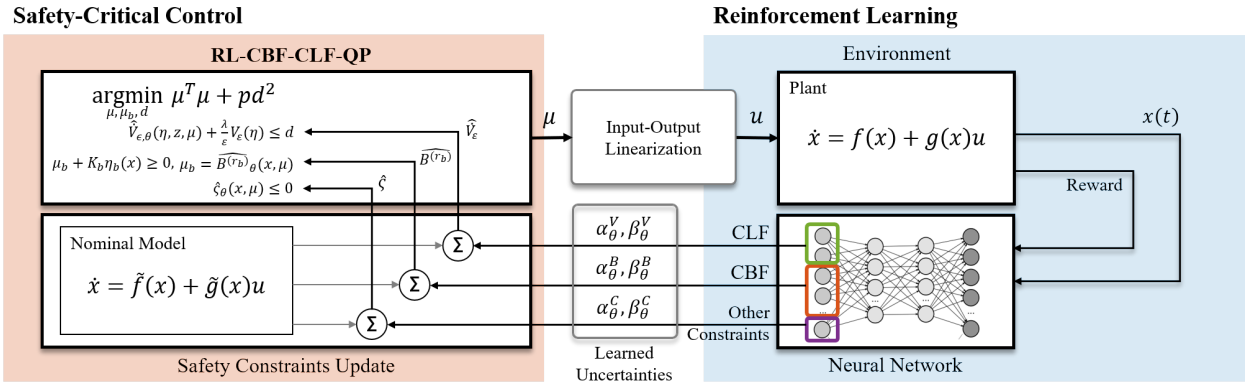



Figure 8.1: Our method (RL-CBF-CLF-QP): We propose a control barrier function (CBF) and control Lyapunov function (CLF) based parametrized quadratic program, where the parameter $\theta$ corresponds to weights of a neural network that estimates the uncertainty in the CLF and CBF dynamics through reinforcement learning. An RL agent is used to learn uncertainties in the CLF, CBF and other constraint dynamics. The quadratic program uses learned uncertainties in combination with safety and stability constraints from a nominal model to solve for the control input point-wise in time.

### 8.1.1 Related Work

In the field of controls, Control Lyapunov Function (CLF)-based and Control Barrier Function (CBF)-based control methods have been shown to be successful for safety-critical control. The seminal works [65, 7] have shown that CLF-based quadratic programs (CLF-QP) with constraints can be solved online in order to perform locomotion and manipulation tasks. In [6], CBFs are incorporated with the CLF-QP, namely CBF-CLF-QP, to handle safety constraints effectively in real time.

These CLF-based and CBF-based methods heavily rely on accurate knowledge of the system model. When the model is uncertain, we must consider adaptive or robust versions. In [145], an $L_1$ adaptive controller is incorporated with the CLF-QP in order to adapt to model uncertainty, and is shown to work effectively for bipedal walking. In [147], a robust version of the CBF-CLF-QP is proposed, that solves the quadratic program for the worst case effect of model uncertainty. While these methods can address the adverse effects of model uncertainty to some degree, they may often fail to account for the correct magnitudes of adaptation and uncertainty.

Recently, several methods addressing the issue of model uncertainty in the control problem using a data-driven approach have been proposed. The work [209] proposes an RL-based method to learn the model uncertainty compensation for input-output linearization control. In [28] the former method is extended to underactuated bipedal walking on a flat terrain. Each of the methods presented in [186, 187] addresses how to learn the uncertainty in CLF

and CBF constraints respectively, using empirical risk minimization. Our methodologies most closely align with these works in that we are also using learning methods to reduce model uncertainty explicitly in input-output linearization, CLF, and CBF-based control. However, the main novelty in our approach is that we have devised a unified RL-based framework for learning model uncertainty in CLF, CBF, and other dynamic control-affine constraints altogether in a single learning process. In addition to the aforementioned papers, there are also a few approaches [12, 19, 60] that learn model uncertainty through probabilistic models such as Gaussian Processes. Although these approaches allow for an insightful analysis of the learned model or policy, they can scale poorly with state dimension.

### 8.1.2 Contributions

In this chapter, we present a novel RL-based framework which combines two key components: 1) an RL agent which learns model uncertainty in multiple general dynamic constraints including CLF and CBF constraints through training, and 2) a quadratic program that solves for the control that satisfies the safety constraints under the learned model uncertainty. We name this framework *Reinforcement Learning-based Control Barrier Function and Control Lyapunov Function Quadratic Program (RL-CBF-CLF-QP)*. After training, the RL-CBF-CLF-QP can be executed online with fast computation. The overall diagram of our framework is presented in Fig. 8.1. Here is the summary of the contribution of this chapter:

1. We present an RL framework that learns model uncertainty for CLF, CBF and other control-affine dynamic constraints in a single learning process.

2. We generalize our method to high relative-degree outputs and Control Barrier Functions.

3. Our method can learn the uncertainty in the dynamics of parameterized CBFs that are not only state-dependent but also dependent on other parameters.

4. We numerically validate our method on an underactuated nonlinear hybrid system: a bipedal robot walking on stepping stones with significant model uncertainty.

## 8.2 Control Lyapunov Functions and Control Barrier Functions under Feedback Linearization

We now briefly present how Control Lyapunov Functions (Definition 2.5) and Control Barrier Functions (Definition 2.7) can be incorporated in the feedback linearization framework introduced in Section 7.2 to efficiently design control laws.

As in the precious chapter, we consider control-affine nonlinear systems of the form in (7.1)

$$\begin{aligned}\dot{x} &= f(x) + g(x)u,\\ y &= h(x),\end{aligned}$$

where $x \in \mathbb{R}^n$ is the system state, $u \in \mathbb{R}^m$ the control input and $y \in \mathbb{R}^m$ the output of the system. Then, if the vector relative degree of the outputs is $r$, we can apply a control input which renders the input-output dynamics of the system linear:

$$\pi_{\mathrm{FL}}(x, \mu) = \pi_{ff}(x) + \left(L_g L_f^{r-1} h(x)\right)^{-1} \mu, \tag{8.1}$$

where $\pi_{ff}$ is the feedforward term

$$\pi_{ff}(x) = -\left(L_g L_f^{r-1} h(x)\right)^{-1} L_f^r h(x) \tag{8.2}$$

and $\mu$ is the *auxiliary input*.

As explained in Section 7.2, using this control law yields the input-output linearized system $y^{(r)} = \mu$, and we can define a state transformation $\Phi : x \to (\eta, z)$, with

$$\eta = [h(x)^\top, L_f h(x)^\top, ..., L_f^{r-1} h(x)^\top]^\top \tag{8.3}$$

and $z \in \mathcal{Z}$, where $\mathcal{Z} = \{x \in \mathcal{X} |\ \eta \equiv 0\}$ is the zero-dynamics manifold. The closed-loop dynamics of the system can then be represented as a linear time invariant system on $\eta \in \mathcal{T}$ and the zero-dynamics on $z \in \mathcal{Z}$:

$$\begin{cases}\dot{\eta} = F\eta + G\mu,\\ \dot{z} = \kappa(\eta, z),\end{cases} \tag{8.4}$$

where

$$F = \begin{bmatrix} 0 & I_m & . & . & 0 \\ 0 & 0 & I_m & . & 0 \\ . & . & . & & \\ 0 & . & . & . & I_m \\ 0 & . & . & . & 0 \end{bmatrix} \text{ and } G = \begin{bmatrix} 0 \\ . \\ . \\ 0 \\ I_m \end{bmatrix}, \tag{8.5}$$

with $F \in \mathbb{R}^{mr \times mr}$ and $G \in \mathbb{R}^{mr \times m}$.

### 8.2.1 Control Lyapunov Function Based Quadratic Programs

We now introduce the use of Control Lyapunov Functions under feedback linearization control schemes.

In [9] a control method that guarantees exponential stability of the transverse dynamics $\eta$ with a rapid enough convergence rate is presented. It introduces the concept of a *rapidly exponentially stabilizing control Lyapunov function (RES-CLF)*. Specifically, a one-parameter

family of continuously differentiable functions $V_\varepsilon : \mathbb{R}^{mr} \to \mathbb{R}$ is said to be an RES-CLF for system (7.1) if $\exists\ \gamma,\ c_1,\ c_2 > 0$ such that $\forall\ 0 < \varepsilon < 1$ and $\forall\ \eta \in \mathbb{R}^{mr}$, the following holds:

$$c_1\|\eta\|^2 \leq V_\varepsilon(\eta) \leq \frac{c_2}{\varepsilon^2}\|\eta\|^2, \tag{8.6}$$

$$\dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon}V_\varepsilon(\eta) \leq 0. \tag{8.7}$$

If we define an auxiliary control input $\mu$ that makes $\eta$ exponentially stable, of the form

$$\mu = \left[-\frac{1}{\varepsilon^r}K_r,\ ...,\ -\frac{1}{\varepsilon^2}K_2,\ -\frac{1}{\varepsilon}K_1\right]\eta = K\eta, \tag{8.8}$$

where $K \in \mathbb{R}^{m\times mr}$, then we can choose a quadratic CLF candidate $V_\varepsilon(\eta) = \eta^T P_\varepsilon \eta$, where $P_\varepsilon$ is the solution of the Lyapunov equation $A^T P_\varepsilon + P_\varepsilon A = -Q$, with $A$ being the closed-loop dynamics matrix $A = F + GK$ and Q any symmetric positive-definite matrix. Defining $\bar{f} = F\eta$, $\bar{g} = G$, we can write the derivative of the RES-CLF as:

$$\dot{V}_\varepsilon(\eta, \mu) = L_{\bar{f}}V_\varepsilon(\eta) + L_{\bar{g}}V_\varepsilon(\eta)\mu, \tag{8.9}$$

with

$$L_{\bar{f}}V_\varepsilon(\eta) = \eta^T\left(F^T P_\varepsilon + P_\varepsilon F\right)\eta, \quad L_{\bar{g}}V_\varepsilon(\eta) = 2\eta^T P_\varepsilon G. \tag{8.10}$$

We can then define for every time step an optimization problem in which condition (8.7) becomes a linear constraint on the auxiliary input $\mu$. The objective function can be set to minimize the norm of the control inputs, in which case the optimization problem is a quadratic program (QP):

---

**FL-CLF-QP**:

$$\mu_{\text{CLF}}(x) = \underset{\mu}{\text{argmin}}\ \ \|\mu\|_2^2 \tag{8.11}$$

$$\text{s.t.}\quad \dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon}V_\varepsilon(\eta) \leq 0 \tag{CLF}$$

---

This problem constitutes an alternative to the CLF-QP presented in (2.20), but based on a feedback linearization control scheme.

### 8.2.2 Control Barrier Function and Control Lyapunov Function Based Quadratic Programs

We next present the integration of Control Barrier Functions into feedback linearization control schemes for systems with high relative degree. In [144] the concept of an Exponential Control Barrier Function (ECBF) is defined. Specifically, a function $B : \mathbb{R}^m \to \mathbb{R}$ is an ECBF of relative degree $r_b$ for the system (7.1) if there exists $K_b \in \mathbb{R}^{1\times r_b}$ such that

$$\sup_u \left[L_f^{r_b}B(x) + L_g L_f^{r_b-1}B(x)u + K_b\eta_b(x)\right] \geq 0 \tag{8.12}$$

for $\forall x \in \{x \in \mathbb{R}^n |\ B(x) \geq 0\}$ with

$$\eta_b(x) = \begin{bmatrix} B(x) \\ \dot{B}(x) \\ \ddot{B}(x) \\ \vdots \\ B^{(r_b-1)}(x) \end{bmatrix} = \begin{bmatrix} B(x) \\ L_f B(x) \\ L_f^2 B(x) \\ \vdots \\ L_f^{r_b-1} B(x) \end{bmatrix}, \tag{8.13}$$

that guarantees $B(x_0) \geq 0 \implies B(x(t)) \geq 0,\ \forall t \geq 0$.

We can then choose a *virtual input* $\mu_b$ that input-output linearizes the ECBF dynamics:

$$B^{(r_b)}(x, \mu) = L_f^{r_b} B(x) + L_g L_f^{r_b-1} B(x) \pi_{\text{FL}}(x, \mu) =: \mu_b, \tag{8.14}$$

with $\pi_{\text{FL}}$ defined in (8.1). We refer readers to [144] for more details. The condition in (8.12) then translates to choosing a $\mu_b$ such that

$$\mu_b + K_b \eta_b \geq 0, \tag{8.15}$$

which is added to the following QP, where safety is prioritized over stability by relaxing the CLF constraint:

---

**FL-CBF-CLF-QP**:

$$\begin{aligned} \mu_{\text{CBF-CLF}}(x) = \underset{\mu,\ \mu_b,\ d}{\arg\min}\ & \|\mu\|_2^2 + p\ d^2 && (8.16) \\ \text{s.t.}\quad & \dot{V}_\varepsilon(\eta, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq d && \text{(CLF)} \\ & \mu_b + K_b\ \eta_b \geq 0 && \text{(CBF)} \\ & \mu_b = B^{(r_b)}(x, \mu) \\ & A_c(x)\mu + b_c(x) \leq 0 && \text{(Constraints)} \end{aligned}$$

---

Formulating a QP allows us to incorporate additional control-affine constraints (last line in (8.16)). These could be input saturation constraints or other state-dependent constraints such as contact-force constraints.

## 8.3 Reinforcement Learning for CLF-QP Based Controllers under Model Mismatch

In this section, we address the issue of having a mismatch between the model and the plant dynamics when the true plant vector fields $f,\ g$ are not precisely known. We instead have

an approximate nominal dynamics model of the form (2.25). Specifically, between this and the next sections we analytically examine the effects of model uncertainty on the dynamics of the CLF, CBF and other control-affine dynamic constraints. For each of these cases we will define the goal of the RL agent and the policy to be learned.

### 8.3.1 Reinforcement Learning for CLF-QP Based Controllers: First Approach

The first approach to address model uncertainty in CLF-based controllers is a direct application of the method presented in the Chapter 7. By correctly input-output linearizing the unknown plant, there will be no uncertainty in the FL-CLF-QP of (8.11) since $\dot{V}_\varepsilon$ only depends on the matrices $F$ and $G$ of the input-output linearized dynamics of the system, which do not contain any uncertainty terms.

Thus, after following the procedure described in Chapter 7, the FL-CLF-QP of (8.11) can be solved point-wise in time to get $\mu$ and the final control input used is obtained by plugging $\mu$ in (7.15).

### 8.3.2 Reinforcement Learning for CLF-QP Based Controllers: Second Approach

In the second approach, we do not directly correct the uncertain terms of the transverse dynamics (7.14) as we did in the first approach. Instead, we directly analyze the impact of these mismatch terms on the dynamics of the CLF.

For this approach, we assume that the CLF designed for the nominal model's transverse dynamics is also a CLF for the true plant's transverse dynamics (7.14).

In the presence of uncertainty, $\dot{V}_\varepsilon$ becomes

$$\dot{V}_\varepsilon(\eta, z, \mu) = L_{\bar{f}} V_\varepsilon(\eta, z) + L_{\bar{g}} V_\varepsilon(\eta, z)\mu, \tag{8.17}$$

where

$$\begin{aligned} L_{\bar{f}} V_\varepsilon(\eta, z) &= L_{\tilde{\bar{f}}} V_\varepsilon(\eta) + \underbrace{2\eta^\intercal P_\varepsilon G \Delta_1(\eta, z)}_{=:\, \Delta_1^v(\eta, z)}, \\ L_{\bar{g}} V_\varepsilon(\eta, z) &= L_{\tilde{\bar{g}}} V_\varepsilon(\eta) + \underbrace{2\eta^\intercal P_\varepsilon G \Delta_2(\eta, z)}_{=:\, \Delta_2^v(\eta, z)}. \end{aligned} \tag{8.18}$$

Here, $\tilde{\bar{f}}$ and $\tilde{\bar{g}}$ are the nominal model input-output linearized dynamics: namely, $\widetilde{\dot{V}}_\varepsilon(\eta, \mu) = L_{\tilde{\bar{f}}} V_\varepsilon(\eta) + L_{\tilde{\bar{g}}} V_\varepsilon(\eta)\mu$. Therefore, under uncertainty:

$$\dot{V}_\varepsilon(\eta, z, \mu) = \widetilde{\dot{V}}_\varepsilon(\eta, \mu) + \Delta_1^v(\eta, z) + \Delta_2^v(\eta, z)\mu. \tag{8.19}$$

In this second approach we use RL to estimate the uncertainty terms in $\dot{V}_\varepsilon$: $\Delta_1^v$ and $\Delta_2^v$. For this purpose, we construct an estimate

$$\widehat{\dot{V}}_{\varepsilon,\theta}(\eta, z, \mu) = \widetilde{\dot{V}}_\varepsilon(\eta, \mu) + \beta_\theta^V(\eta, z) + \alpha_\theta^V(\eta, z)\mu, \tag{8.20}$$

where $\theta \in \Theta \subset \mathbb{R}^N$ are neural network parameters to be learned. The goal of RL is then obvious: learn a policy $\{\alpha_\theta^V,\ \beta_\theta^V\}$ such that $\widehat{\dot{V}}_{\varepsilon,\theta}$ is as close as possible to $\dot{V}_\varepsilon$. Any reward function that penalizes the absolute value of the difference between the two terms can be used. More details on the specific RL implementation are discussed in Section 8.5.

**Remark 8.1.** *For convenience, it is assumed here that $\alpha_\theta^V$, $\beta_\theta^V$ share the same network parameters $\theta$, but this does not need to be the case. In this chapter, we will assume that all the policy functions to be learned are sharing the same parameters.*

The estimate $\hat{\dot{V}}_{\varepsilon,\theta}$ in (8.20) is then used as our best guess of $\dot{V}_\varepsilon$ for the optimization problem:

---

**RL-CLF-QP**:

$$\mu_\theta^*(x) = \arg\min_{\mu} \|\mu\|_2^2 \tag{8.21}$$

$$\text{s.t.} \quad \widehat{\dot{V}}_{\varepsilon,\theta}(\eta, z, \mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \leq 0 \tag{RL-CLF}$$

---

**Remark 8.2.** *In this chapter, we have illustrated the case in which the CLF is applied to the input-output linearized dynamics. The reason why we use a CLF on the input-output linearized dynamics instead of the full dynamics is that in this way we have a systematic way of computing a CLF candidate, whereas on the original nonlinear system this process could be challenging. However, this approach is not confined to the input-output linearization structure and is also applicable to any general nonlinear control-affine system.*

## 8.4 Reinforcement Learning for CBF-CLF-QP Based Controllers under Model Mismatch

Having studied how to compensate for the effects of model uncertainty on CLF-based min-norm controllers, we will now extend our framework to the safety-critical FL-CBF-CLF-QP by following a similar approach.

### 8.4.1 Reinforcement Learning for CBFs

In the presence of uncertainty, (8.14) becomes

$$\widetilde{B^{(r_b)}}(x, \mu) = L_{\tilde{f}}^{r_b} B(x) + L_{\tilde{g}} L_{\tilde{f}}^{r_b - 1} B(x) \tilde{\pi}_{\mathrm{FL}}(x, \mu), \tag{8.22}$$

and the actual CBF's $r_b^{th}$ derivative can be written as:

$$B^{(r_b)}(x,\mu) = \widetilde{B^{(r_b)}}(x,\mu) + \Delta_1^b(x) + \Delta_2^b(x)\mu, \tag{8.23}$$

where $\Delta_1^b$ and $\Delta_2^b$ are the uncertain terms that arise from the model-plant mismatch. We omit analytic expressions of $\Delta_1^b, \Delta_2^b$ for conciseness, but they can be derived similarly to (7.13).

**Remark 8.3.** *When the state of the system can be represented as $x = [q, \dot{q}]^T$, as in most robotic systems, even for high relative degree CBFs model uncertainty only affects the $r_b^{th}$ time derivative of $B$, since $B^{(r_b)}$ is the only term that depends on the plant dynamics through the vector fields $f$ and $g$.*

Next, we present how to estimate the uncertainty terms for the CBF and for other dynamic constraints using RL. The approach presented in Section 8.3.1 cannot be used here since the CBF functions depend on the full dynamics of the system, and not the transverse dynamics.

We build an estimator of $B^{(r_b)}$:

$$\widehat{B^{(r_b)}}_\theta(x,\mu) = \widetilde{B^{(r_b)}}(x,\mu) + \beta_\theta^B(x) + \alpha_\theta^B(x)\mu, \tag{8.24}$$

and the goal of RL is to learn a policy $\alpha_\theta^B$, $\beta_\theta^B$ such that $\widehat{B^{(r_b)}}_\theta$ is as close as possible to $B^{(r_b)}$.

In order to integrate everything in a new QP we define the new virtual input of the CBF dynamics as

$$\mu_b \coloneqq \widehat{B^{(r_b)}}_\theta. \tag{8.25}$$

In cases where the CBF also depends on a set of parameters $\psi \in \mathbb{R}^q$ , then we need to define the CBF as $B : \mathbb{R}^{n\times q} \to \mathbb{R}$. The neural-network policy will now need to take $\psi$ as additional inputs $\alpha_\theta^B : \mathbb{R}^{n\times q} \to \mathbb{R}^m$, $\beta_\theta^B : \mathbb{R}^{n\times q} \to \mathbb{R}$ and the proposed estimate of the $r_b^{th}$ time derivative of $B$ becomes:

$$\widehat{B^{(r_b)}}_\theta(x,\mu,\psi) = \widetilde{B^{(r_b)}}(x,\mu,\psi) + \beta_\theta^B(x,\psi) + \alpha_\theta^B(x,\psi)\mu. \tag{8.26}$$

### 8.4.2 Reinforcement Learning for Additional Control-Affine Dynamic Constraints

Now we study the effects of uncertainty on other linear constraints that depend on the dynamics of the system:

$$\underbrace{A_c(x,f,g)\mu + b_c(x,f,g)}_{=:\,\zeta(x,\mu)} \leq 0. \tag{8.27}$$

In the presence of model mismatch we have

$$\begin{aligned} b_c(x,f,g) &= b_c(x,\tilde{f},\tilde{g}) + \Delta_1^c(x), \\ A_c(x,f,g) &= A_c(x,\tilde{f},\tilde{g}) + \Delta_2^c(x), \end{aligned} \tag{8.28}$$

where $\Delta_1^c$ and $\Delta_2^c$ represent the uncertainty terms. We can then define the nominal constraint

$$\tilde{\zeta}(x,\mu) = b_c(x,\tilde{f},\tilde{g}) + A_c(x,\tilde{f},\tilde{g})\mu. \tag{8.29}$$

And the real value of the constraint can be expressed as

$$\zeta(x,\mu) = \tilde{\zeta}(x,\mu) + \Delta_1^c(x) + \Delta_2^c(x)\mu. \tag{8.30}$$

We can build an estimator of the form

$$\hat{\zeta}_\theta(x,\mu) = \tilde{\zeta}(x,\mu) + \beta_\theta^C(x) + \alpha_\theta^C(x)\mu, \tag{8.31}$$

with a learned policy $\alpha_\theta^C$, $\beta_\theta^C$. The goal of the RL agent is again in this case to make the estimator $\hat{\zeta}_\theta$ as close as possible to $\zeta$. Expanding $\tilde{\zeta}$ we can rewrite the estimator as

$$\hat{\zeta}_\theta(x,\mu) = \underbrace{\Big(b_c(x,\tilde{f},\tilde{g}) + \beta_\theta^C(x)\Big)}_{=:\, b_\theta^c(x)} + \underbrace{\Big(A_c(x,\tilde{f},\tilde{g}) + \alpha_\theta^C(x)\Big)}_{=:\, A_\theta^c(x)}\mu. \tag{8.32}$$

So far, we have explained our method of constructing an estimator of a single $B^{(r_b)}$ and a single $\zeta(x,\mu)$. This can be applied to $n_b$ multiple CBFs and $n_c$ multiple control-affine constraints. The final optimization problem, which includes all the learned estimates of the uncertain terms is:

---

**RL-CBF-CLF-QP**:

$$\begin{aligned}
\mu_\theta^*(x) = \ & \underset{\mu,\ \mu_b,\ d}{\arg\min} \ \|\mu\|_2^2 + p\ d^2 && (8.33)\\
\text{s.t.} \quad & \widehat{\dot{V}}_{\varepsilon,\theta}(\eta,z,\mu) + \frac{\lambda}{\varepsilon} V_\varepsilon(\eta) \le d && \text{(RL-CLF)}\\
\text{for } i = 1\cdots n_b \ & \mu_{b,i} + K_{b,i}\ \eta_{b,i} \ge 0 && \text{(RL-CBF)}\\
& \mu_{b,i} = \widehat{B^{(r_b)}}_{i,\theta}(x,\mu) &&\\
\text{for } j = 1\cdots n_c \ & A_{j,\theta}^c(x)\mu + b_{j,\theta}^c(x) \le 0 && \text{(RL-Constraints)}
\end{aligned}$$

---

## 8.5 Reinforcement Learning-based Framework

In this section, we present a unified RL framework that can learn the uncertainty terms in the CLF, CBF, and other dynamic constraints by building the terms specified in the earlier sections as $\alpha_\theta^V, \alpha_\theta^B, \alpha_\theta^C, \beta_\theta^V, \beta_\theta^B, \beta_\theta^C$.

A diagram of this framework is illustrated in Fig. 8.1. The RL agent learns a policy, which is a combination of uncertainty terms in CLF, CBF and other dynamic constraints. These terms are then added to the QP constraints derived from the nominal model, resulting in the estimates of the true plant constraints. Using these estimates, the RL-CBF-CLF-QP optimization problem, in which model uncertainty is addressed, is solved point-wise in time to obtain the control input.

The reward function of the learning problem is designed such that it minimizes each of the estimation errors. Thus, the time-wise loss functions are defined as:

$$\begin{aligned} l_{V,\theta} &:= ||\dot{V}_\varepsilon - \widehat{\dot{V}}_{\varepsilon,\theta}(x,\mu)||^2 \\ l_{B,\theta} &:= ||B^{(r_b)} - \widehat{B^{(r_b)}}_\theta(x,\mu)||^2 \\ l_{C,\theta} &:= ||\zeta - \hat{\zeta}_\theta(x,\mu)||^2 \end{aligned} \tag{8.34}$$

It is important to note that the true plant's dynamics information is not used for computing the values of these loss functions. We use explicit expressions for $V_\varepsilon$, $B$ and $\zeta$ and compute the time-derivatives $\dot{V}_\varepsilon$, $B^{(r_b)}$ using numerical differentiation. For the CBF, it is important to note that regardless of the value of $r_b$ we only need to do numerical differentiation once, as follows from Remark 8.3.

A canonical RL problem can be formulated, with the reward for a given state $x$ defined as the weighted sum of the negative loss functions in (8.34), in addition to a user-specific failure-case penalty $-l_e : \mathbb{R}^n \to \mathbb{R}$:

$$R(x,\theta) = -w_v l_{V,\theta} - \sum_{i=1}^{n_b} w_{b,i} l_{B_i,\theta} - \sum_{j=1}^{n_c} w_{c,j} l_{C_j,\theta} - l_e(x). \tag{8.35}$$

The learning problem is then defined as:

$$\begin{aligned} &\max_\theta \ \mathbb{E}_{x_0 \sim X_0, w \sim \mathcal{N}(0,\sigma^2)} \int_0^T R\left(x(\tau),\theta\right) d\tau, \\ &\text{s.t. } \dot{x} = f(x) + g(x)\tilde{\pi}_{\mathrm{FL}}(x, \mu^*_\theta(x) + \omega), \end{aligned} \tag{8.36}$$

where $\mu^*_\theta(x)$ is the solution of (8.33), $X_0$ is the initial state distribution, and $w \sim \mathcal{N}(0,\sigma^2)$ is white noise added to encourage exploration. A discretized version of this problem can be solved using conventional RL algorithms.

**Remark 8.4.** *While running training experiments or simulations, it is assumed that the robot operates under the true plant dynamics. We will later show in Section 8.7 that the trained policy works well even when the true plant in the evaluation differs from the plant of the training environment.*

## 8.6 Application to Bipedal Robots

The goal of this section is to validate that the RL-CBF-CLF-QP framework enables safety-critical control when model uncertainty is present. We test our method on RABBIT [37], a planar five-link bipedal robot, walking on a discrete terrain of stepping stones with one step preview.

### 8.6.1 Simulation Settings

We run two simulation scenarios with our method and offer comparisons with the previous methods. The first simulation consists of RABBIT simply walking on a flat terrain. We evaluate the CLF based methods in Section 8.3 in this scenario. This is to verify only the stabilizing capacity of our proposed method under model uncertainty. In the second simulation, we put the robot on a discrete terrain of randomly spaced stepping stones (Fig. 8.4). The robot's task here is to always place the foot on the next stepping stone, while managing the stability and not violating the contact-force constraint. The full RL-CBF-CLF-QP is tested in this simulation scenario.

The main model uncertainty in both demonstrations is introduced by scaling all mass and inertia parameters of each link by a constant scale factor = 2, i.e. the nominal model's mass and inertia terms are half of those of the actual plant.

A single periodic walking gait trajectory is generated offline by the Fast Robot Optimization and Simulation Toolkit (FROST) [81]. The output function $h(x)$ is defined as the difference between the actuated joint angles and the desired trajectory's joint angles from the obtained periodic orbit. The gait's nominal step length is 0.35m. Finally, a torque saturation of 200Nm is applied to the control inputs of all simulations, including training and evaluation.

### 8.6.2 Reinforcement Learning Settings

We train our agent using a standard Deep Deterministic Policy Gradient Algorithm (DDPG) [175]. The input for the actor neural network is 14 observations, which is RABBIT's full state $x$, in addition to the CBF parameter $\psi = l_{min,k}$ corresponding to the minimum step length of the $k$th stepping stone (Fig. 8.4) in the second simulation. We use two CBFs $B_1$ and $B_2$ to constrain the position of the swing foot so that it lands on the stepping stone, as shown in Fig. 8.4. We use two dynamic constraints $C_1$ and $C_2$ which correspond to the unilateral normal force and friction cone constraints respectively. The output dimension is 25, corresponding to the $4\times1$ $\alpha_\theta^V, \alpha_\theta^{B_1}, \alpha_\theta^{B_2}, \alpha_\theta^{C_1}, \alpha_\theta^{C_2}$ and the $1\times1$ $\beta_\theta^V, \beta_\theta^{B_1}, \beta_\theta^{B_2}, \beta_\theta^{C_1}, \beta_\theta^{C_2}$.

Both actor and critic neural networks have two hidden layers of widths 400 and 300. This agent is trained on the simulation of ten walking steps per episode, and a discrete time step $T_s = 0.01$sec is used. The failure cases are determined by the robot's pose. Training on six multiple cores of Intel(R) Core(TM) i5-9400F CPU (2.90GHz) without the use of GPU

took about 34 seconds per episode. The final agent in use is obtained after 110, 79 and 133 episodes for IO-RL + FL-CLF-QP, RL-CLF-QP and RL-CBF-CLF-QP respectively.

## 8.7 Numerical Validation

During the evaluation, the robot is tested not only on the uncertainty that is introduced in the training, but in addition to it, two other kinds of uncertainty are also introduced. First, the robot's motor dynamics that restricts the rate of change of joint torques is applied in every evaluation. The time constant of motors used in the simulation is 0.004 seconds. Second, the robot is also tested on an alternative kind of uncertainty, which consists of an added weight to the torso of the robot, instead of scaling the links masses and inertias. This weight can represent the robot carrying a payload, and it is deliberately introduced to evaluate the trained policy's robustness to an unfamiliar kind of uncertainty that it was not trained on.

### 8.7.1 Simulation 1: Bipedal Walking on Flat Ground

For the first simulation, we evaluate the two RL approaches for CLF explained in Section 8.3, and compare them with the standard L1 Adaptive CLF-QP method of [145], which guarantees the CLF to be bounded to a small value under model uncertainty if using a sufficiently large adaptation gain.

As illustrated in Fig. 8.2, both of the proposed methods manage to get RABBIT to stably walk for multiple steps, while the L1 Adaptive CLF-QP controller leads to failure. The original nominal FL-CLF-QP, although not shown in the figure, also fails under this scaled model uncertainty. Note that all three methods do not have friction constraints in the QP and could potentially violate them. In particular, the RL-CLF-QP method succeeds in satisfying the friction constraint ($|F_T/F_N| \leq k_f = 0.8$) for all steps, the IO-RL + CLF-QP exceeds the limit in the first two steps, and the L1-CLF-QP violates it for multiple steps. Therefore, IO-RL + CLF-QP needs the inclusion of friction constraints in the QP.

Displayed in Fig. 8.3 is the plot of tracking error and contact force ratio of the three controllers when, instead of the mass-inertia-scaling, an additional torso weight of 32kg (100% of the robot mass) is introduced. It is notable that both the RL-CLF-QP and IO-RL + CLF-QP manage to adapt to this uncertainty, which has not been faced during the training. Furthermore, the RL-CLF-QP manages to stabilize the walking gait with an additional torso weight of up to 72kg (225% of robot mass). On the other hand, IO-RL + CLF-QP manages to adapt to additional weights up to 53kg (166% of robot mass).

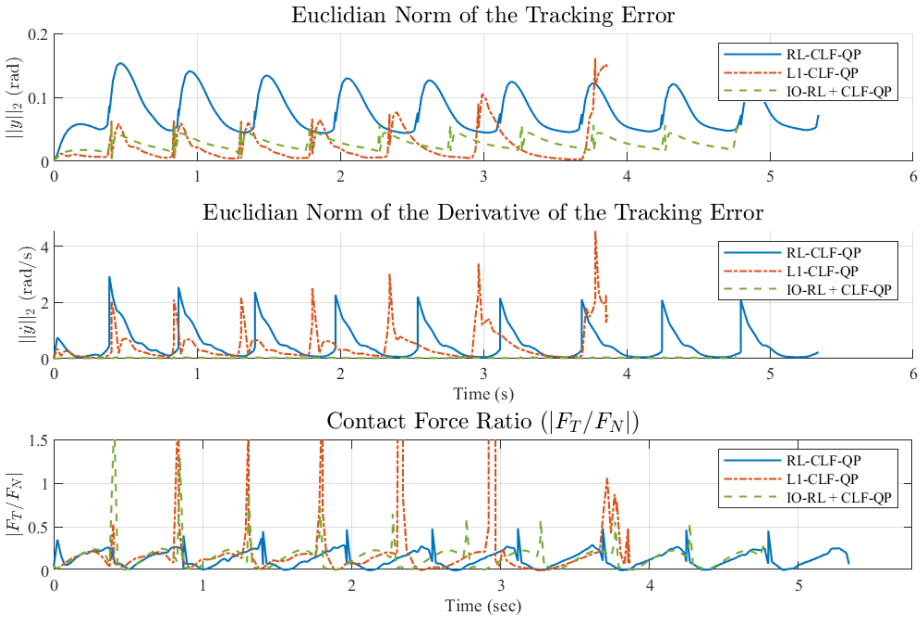



Figure 8.2: Tracking error (top), its derivative (middle), and tangential-normal contact force ratio (bottom) of IO-RL + CLF-QP (Sec. 8.3.1), RL-CLF-QP (Sec. 8.3.2), and L1-CLF-QP [145] controllers, simulated for ten walking steps, where the robot's mass and inertia values are scaled by a factor of 2. Both IO-RL + CLF-QP and RL-CLF-QP maintain the stability while L1-CLF-QP fails. Only the RL-CLF-QP satisfies the friction cone constraint $|F_T/F_N| \leq k_f = 0.8$.

### 8.7.2 Simulation 2: Bipedal Walking on Stepping Stones with One Step Preview

We now evaluate the full RL-CBF-CLF-QP method with the safety-critical constraint of walking on stepping stones and the inclusion of friction constraints, which are dependent on the dynamics. In this simulation scenario, for each step the robot faces a random placement of a stepping stone. Therefore, when the swing foot hits the ground at the end of the step, we want the step length to be within a specific range:

$$l_{min,k} \leq l_k \leq l_{max,k}, \tag{8.37}$$

where k indicates the step index. Two position-constraints-based second order ECBFs parameterized by $l_{min,k}$, $l_{max,k}$ that are a sufficient condition for (8.37) are devised by [147]. Basically these constraints imply that the swing foot position ($F$ in Fig. 8.4) needs to stay within the grey area. Note that $l_{min,k}$, $l_{max,k}$ change for every step.

We also include contact force constraints in the RL-CBF-CLF-QP as control-affine dynamic constraints, following the procedure of Subsection 8.4.2. These are important since

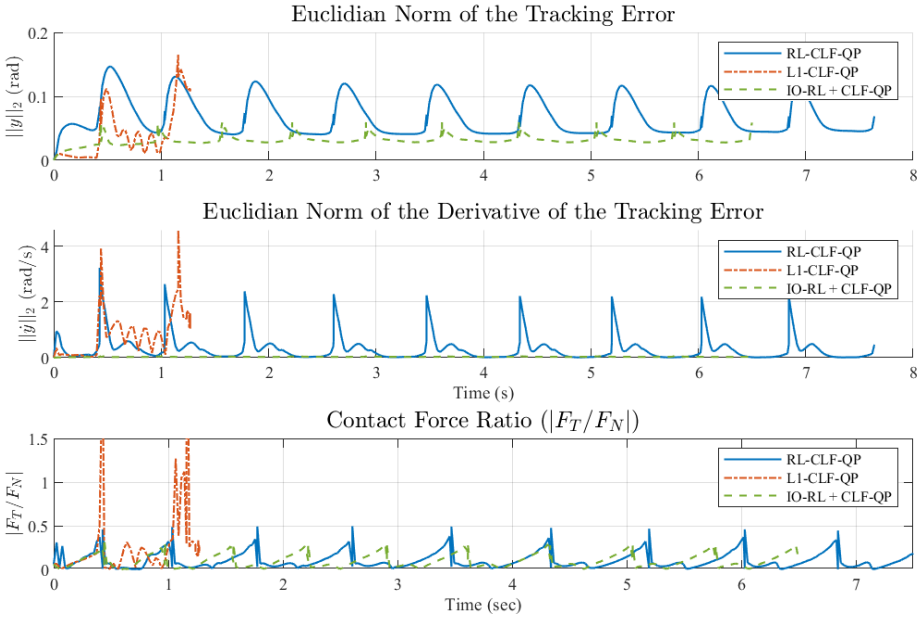



Figure 8.3: Tracking error (top), its derivative (middle), and contact force ratio (bottom) of the three CLF-based controllers, simulated for ten walking steps with the additional torso weight 32kg (this amounts to the weight of RABBIT, i.e. 100% additional weight).

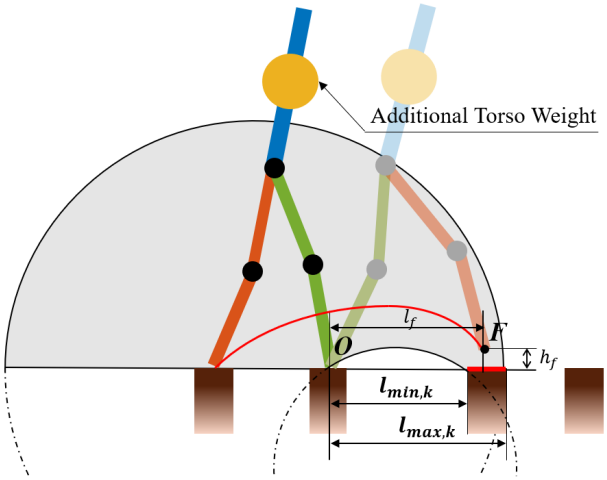



Figure 8.4: Safety Constraint: In order to guarantee the swing foot lands on the stepping stone, we use two CBFs to ensure the swing foot position $F$ is within the grey area during the entire walking step.

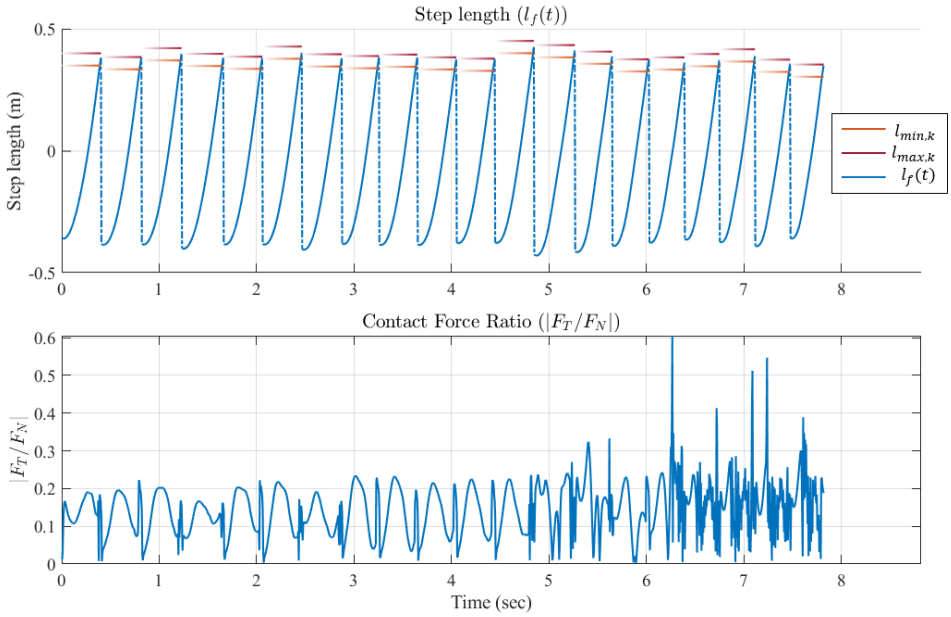



Figure 8.5: Results of the simulation of 20 steps of walking on stepping stones, where the robot's mass and inertia values are scaled by a factor of 2. (Top) History of swing foot position $l_f$ for each step, with the stepping stone constraints $l_{min}$, $l_{max}$. (Bottom) History of tangential-normal contact force ratio that satisfies to stay below $|F_T/F_N| \leq k_f = 0.8$.

the original FL-CBF-CLF-QP violates the friction cone and the unilateral normal force constraints repeatedly.

The robot is trained to walk on randomly spaced stepping stones, of which $l_{min}$ is sampled from a normal distribution $\mathcal{N}(0.35\text{m}, 0.02\text{m})$, truncated at $2.5\sigma$. $l_{max}$ is set as $l_{min} + 0.05\text{m}$.

Fig. 8.5 shows the result of the evaluation, where the robot walks on 20 randomly spaced stepping stones. We can check that the foot placement is always on the stepping stones. Also, it is verified that the contact force never exceeds the friction limit. Note that the sample distribution of $l_{min}$ here is same as during training.

Whereas our RL-CBF-CLF-QP method performs well, we have also tested the nominal model-based FL-CBF-CLF-QP method on this simulation for comparison. The FL-CBF-CLF-QP is also solved together with the friction constraints. However, it violates the step length safety constraints after an average of 5.6±4.64 steps. This value is obtained from 10 random executions of 20 steps simulation.

Finally, for the case of having an additional torso weight applied to the original unscaled plant, RL-CBF-CLF-QP still manages to stay within the safety and friction constraints when the weight is in the range of [43kg, 72kg] (134-225% of robot mass).

### 8.7.3 Discussion of Results

We have demonstrated that our method can compensate well for the trained model uncertainty and that it shows some robustness to the introduction of additional uncertainty during evaluation. It is important to note that our method is not restricted to mass and inertia scaling uncertainties, rather they have been used as illustrative examples for this chapter. We have additionally tested our framework for other uncertainties: a simplified model of joint friction (assuming that joint friction reduces motor power by a 15%, value taken from [37]), joint damping (up to 1 $(rad/s^2)/(rad/s)$) and bending of links (up to 5% of their length) obtaining successful results.

However, a primary drawback of our approach is that we need the designed nominal controller to not rapidly fail on the uncertain system before RL can learn the uncertainty. This may not always be possible depending on the level of uncertainty. Following this same reasoning, for high levels of uncertainty the CLF designed for the nominal model may not be a CLF for the true plant, in which case our assumption would not hold and the method would fail. There is ongoing research on designing CLFs for systems with uncertain dynamics [165, 191] that could be used to solve this issue, since our method is not restricted to any specific CLF.

An illustration of the aforementioned limitation is that we have also tested our framework for mass-inertia uncertainty scales of 0.7 and 0.5. For the case of scale=0.7, our framework produces a stabilizing controller that respects safety and friction constraints for indefinitely long periods of walking, whereas the nominal model-based controller fails after just one step. In contrast, for the scale of 0.5, the nominal controller fails after just 0.06 seconds, which makes the training a lot more challenging and our framework fails.

Another limitation is that the measurements of $\dot{V}_\varepsilon$ and $B^{(r_b)}$ obtained from numerical differentiation could be noisy in experiments.

Finally, it must be noted that feasibility of a CBF-CLF-QP with additional constraints, such as friction, is not guaranteed in general. However, using the trained RL-CBF-CLF-QP model, we observe that the feasibility drastically improves compared to the nominal FL-CBF-CLF-QP.

## 8.8 Chapter Summary

In this chapter, the issue of model uncertainty in safety-critical control was addressed with a data-driven approach. For this purpose, we used the structure of a feedback linearization controller based on a nominal model along with a Control Barrier Function and Control Lyapunov Function based Quadratic Program (CBF-CLF-QP). Specifically, we have proposed a novel reinforcement learning framework which learns the model uncertainty present in the CBF and CLF constraints, as well as other control-affine dynamic constraints in the quadratic program. The trained policy was combined with the nominal model-based CBF-CLF-QP, resulting in the *Reinforcement Learning-based CBF-CLF-QP (RL-CBF-CLF-QP)*,

which addresses the problem of model uncertainty in the safety constraints. The performance of the proposed method was validated by testing it on an underactuated nonlinear bipedal robot walking on randomly spaced stepping stones with one step preview, obtaining stable and safe walking under model uncertainty.

# Chapter 9

# Probabilistic Stabilizing Control under Uncertainty

*This chapter is based on the paper titled "Gaussian Process-based Min-norm Stabilizing Controller for Control-Affine Systems with Uncertain Input Effects and Dynamics" [27], co-authored by Jason J. Choi, Bike Zhang, Claire J. Tomlin and Koushil Sreenath.*

The previous two chapters presented approaches to bridge the reality gap using deep learning models for different model-based controllers. However, the proposed models lack a measure of the confidence of the prediction—there are no guarantees that the outputs of the neural network policies will be anywhere close to the true desired outcomes. This motivates the final three technical chapters of this dissertation, which use Bayesian nonparametric models to infer the effects of model uncertainty from data. The main advantage of these methods is that we can obtain an estimate of how confident the model is of its prediction, which allows to obtain high-probability guarantees about the desired controller properties (such as stability and safety) holding on the real system.

In particular, this chapter presents a method to design a min-norm Control Lyapunov Function (CLF)-based stabilizing controller for a control-affine system with uncertain dynamics using Gaussian Process (GP) regression. In order to estimate both state and input-dependent model uncertainty, we propose a novel compound kernel that captures the control-affine nature of the problem. Furthermore, by the use of GP Upper Confidence Bound analysis, we provide probabilistic bounds of the regression error, leading to the formulation of a CLF-based stability chance constraint which can be incorporated in a min-norm optimization problem. We show that this resulting optimization problem is convex, and we call it "Gaussian Process-based Control Lyapunov Function Second-Order Cone Program" (GP-CLF-SOCP). The data-collection process and the training of the GP regression model are carried out in an episodic learning fashion. We validate the proposed algorithm and controller in numerical simulations of an inverted pendulum and a kinematic bicycle model, resulting in stable trajectories which are very similar to the ones obtained if we actually knew the true plant dynamics.

## 9.1 Introduction

Model-based controllers have a problem inherent to their nature: model uncertainty. In this chapter, we directly address this issue for the case of Lyapunov-based stabilizing controllers for nonlinear control-affine systems by using Gaussian Process (GP) regression to estimate the adverse effects of model uncertainty.

Control Lyapunov Functions (CLFs) [11, 177] have been widely used in recent years for nonlinear model-based stabilizing control of robotic systems [65, 146, 164]. Typically, the robot is stabilized by enforcing the CLF to decay to zero with a constraint in an optimization problem [7]. However, CLF-based optimization methods heavily rely on the assumption that the model used for the controller design accurately represents the true plant's dynamics. If there is model-plant mismatch, convergence guarantees are often lost. Past research has directly addressed this issue by using both robust [146] and adaptive [145] control theory. More recently, various kinds of data-driven methods that use neural networks have been introduced [186, 38, 210]. Although these are demonstrated to be effective in practice, it is often difficult to verify the reliability of the neural network predictions.

For this chapter, we are more interested in another class of data-driven approaches to tackle this problem, which use GP regression to allow for the analysis of the confidence of the prediction. The method of applying GPs to the CLF constraint was first introduced for closed-loop systems in [18]. Then, similar approaches have also been proposed for the construction of stability and safety constraints to be incorporated in min-norm controllers [191, 58, 36, 223].

However, all of these papers make an important assumption that might restrict their applicability, which is that the considered model uncertainty is unaffected by the control input. In contrast, for many controlled systems, uncertain input effects[1] are prevalent, e.g., in a mechanical system, uncertainty in the inertia matrix directly induces uncertain input effects. In the work presented in [104], a similar problem is addressed for the case of Control Barrier Function-based safety constraints [8] by the use of a Matrix-Variate GP regression. However, it does not provide a regression confidence analysis and results in an optimization problem that is not always convex. Finally, all the aforementioned GP-based approaches apply GP regression directly to the dynamics vector fields, which scale poorly with the system dimension.

In this chapter, we develop solutions to overcome the presented limitations of the previous GP-based methods. First, we provide a formal way to deal with input-dependent model uncertainty of control-affine systems by proposing a specific GP kernel structure suitable for this problem. Since we apply GP regression to a scalar uncertainty term in the CLF constraint directly, compared to learning the uncertainty terms in the dynamics, we can reduce the computation of the regression significantly while still capturing many realistic forms of uncertainty. A similar kernel structure was used in [192] to learn the uncertainty terms in the autonomous and control vector fields separately for a single-input system.

[1] Uncertainty in the control vector field $g(x)$ in (2.2).

Here, we generalize the kernel to an arbitrary input dimension and derive expressions for the posterior GP of a combined input-dependent uncertainty term whose mean and variance are linear and quadratic in the input, respectively. By doing so, we can formulate a Second-Order Cone Program (SOCP) which incorporates a chance constraint that takes into account the confidence of the GP model and provides the exponential stabilizability of the system. We call it *Gaussian Process-based Control Lyapunov Function Second-Order Cone Program* (GP-CLF-SOCP). Formulation of the SOCP is crucial in that it can be solved quickly enough for real-time applications due to its convexity. Finally, since the inference time of GP regression is directly determined by the size of the training data, we maximize data efficiency by the use of an algorithm that iteratively collects data and improves the GP regression model in an episodic learning fashion.

## 9.2 Problem Setting

Throughout this chapter we consider nonlinear control-affine systems of the form in

$$\dot{x} = f(x) + g(x)u,$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state of the system and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. The vector fields $f\colon \mathcal{X} \to \mathbb{R}^n$ and $g\colon \mathcal{X} \to \mathbb{R}^{n\times m}$ are assumed to be locally Lipschitz continuous and $f(0) = 0$.

The main objective of this chapter is the construction of a locally stabilizing controller for such a system even when its dynamics are uncertain. A system is called *stabilizable* when it is asymptotically controllable to the origin with a feedback control law $\pi\colon \mathcal{X} \to \mathcal{U}$ that is continuous except possibly at the origin.

For this purpose, we utilize the concept of Control Lyapunov Function that was introduced in Chapter 2, in Definition 2.5. As we discussed in that chapter, if the dynamics of the true system (2.2) were perfectly known, then we could obtain an exponentially stabilizing control law by solving point-wise the CLF-QP of (2.20).

### 9.2.1 Effects of Model Uncertainty on CLF-based Controllers

The main problem concerned in this chapter is how to reformulate the min-norm stabilizing controller CLF-QP defined in (2.20) in the presence of model uncertainty, using Gaussian Process regression to estimate the uncertain terms.

First, we provide some necessary settings and assumptions for our problem formulation. Let's assume that we have a *nominal model* of the form in (2.25)

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u,$$

where $\tilde{f} : \mathcal{X} \to \mathbb{R}^n$, $\tilde{g} : \mathcal{X} \to \mathbb{R}^{n\times m}$ are Lipschitz continuous vector fields and $\tilde{f}(0) = 0$. We assume that we have a locally exponentially stabilizing CLF $V$ for the nominal model, and

that the plant is also locally exponentially stabilizable with the same $V$. Note that the region of exponential stabilizablity around the origin can be sufficiently small ($\Omega_{c_{exp}}$ in (2.19)). Also, the assumption can be relaxed to asymptotic stabilizability if the user is concerned with enforcing condition (2.18) instead of (2.19). In general, $\tilde{f}$ and $\tilde{g}$ would be different from the true plant vector fields $f$ and $g$ because the nominal model is imperfect. The assumption implies, however, that they share some similarity through the stabilizing property of the same function $V$. Finally, we also assume that we have access to measurements of state and control input at every sampling time $\Delta t$.

Our main objective is to construct the exponential CLF constraint (2.20b) for the true plant when we only know the model dynamics $\tilde{f}$ and $\tilde{g}$. Since $\dot{V}(x,u) = L_f V(x) + L_g V(x)u$ depends on the dynamics of the plant, the estimate based on the nominal model $\dot{\tilde{V}}(x,u) = L_{\tilde{f}} V(x) + L_{\tilde{g}} V(x)u$, will differ from its true value. We define $\Delta_V : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ as the difference between these:

$$\Delta_V(x,u) := \dot{V}(x,u) - \dot{\tilde{V}}(x,u). \tag{9.1}$$

Then, the exponential CLF constraint for the true plant (2.20b) becomes

$$L_{\tilde{f}} V(x) + L_{\tilde{g}} V(x)u + \Delta_V(x,u) + \lambda V(x) \leq 0. \tag{9.2}$$

Therefore, verifying the exponential CLF constraint for the true plant amounts to a problem of learning the mismatch term $\Delta_V(x,u)$ correctly and then enforcing (9.2). We can learn this function from the past data by formulating a supervised learning problem. Specifically, we will use GP regression, a method that was introduced in Chapter 2. Note that learning $\Delta_V$ is advantageous rather than learning the full dynamics of the system (as is typically done in the model-based reinforcement learning literature) since $\Delta_V$ is a scalar function. Indeed, this function condenses the stability-relevant model uncertainty into a scalar.

**Remark 9.1.** *In* (9.1)*, if we express* $\dot{V}$ *and* $\dot{\tilde{V}}$ *with their respective Lie derivatives, we get*

$$\Delta_V(x,u) = \underbrace{(L_f V(x) - L_{\tilde{f}} V(x))}_{=:L_{\Delta f} V(x)} + \underbrace{(L_g V(x) - L_{\tilde{g}} V(x))}_{=:L_{\Delta g} V(x)} u. \tag{9.3}$$

*Note that we do not have access to* $L_{\Delta f} V(x)$ *and* $L_{\Delta g} V(x)$ *in this equation since we are unaware of* $f$ *and* $g$*. It is tempting to learn each of these terms separately with supervised learning. However, we can only measure* $\Delta_V(x,u)$*, which makes this approach intractable. Nevertheless, we can exploit the fact that the mismatch term* $\Delta_V(x,u)$ *is control-affine.*

## 9.3 Gaussian Process Regression for Affine Target Functions

### 9.3.1 High Probability Bounds of the GP Prediction

As explained in Chapter 2, a Gaussian Process is a powerful probabilistic model to fit an unknown function $h : \bar{\mathcal{X}} \to \mathbb{R}$ from data. Assuming that the prior mean function $q$ is selected

to be zero, given a covariance or kernel function $k : \bar{\mathcal{X}} \times \bar{\mathcal{X}} \to \mathbb{R}$, the prediciton of the value of the unknown function at an unseen query point $x_*$ given a dataset of measurements $\mathbb{D}_N$ is given by (2.26) and (2.27):

$$\mu(x_*|\mathbb{D}_N) = \mathbf{z}^T(K + \sigma_n^2 I)^{-1}K_*^T,$$

$$\sigma^2(x_*|\mathbb{D}_N) = k\left(x_*, x_*\right) - K_*(K + \sigma_n^2 I)^{-1}K_*^T,$$

Any positive definite kernel function[2] can be a valid covariance function $k$ [207]. Such a kernel $k(x, x')$ can be used to generate a set of functions that satisfy a specific property, namely a "reproducing" property: the inner product between such a function $h$ and the kernel $k(\cdot, x)$ should reproduce $h$, i.e., $\langle h(\cdot), k(\cdot, x)\rangle = h(x),\ \forall x \in \bar{\mathcal{X}}$. Such a set of functions is called a Reproducing Kernel Hilbert Space (RKHS, [207]), a specific class of function space, and is denoted as $\mathcal{H}_k(\bar{\mathcal{X}})$. The RKHS norm $\|h\|_k := \sqrt{\langle h, h\rangle}$, which will be used in Lemma 9.1, is a measure of the smoothness of $h$ with respect to the kernel function[3]. Note that an appropriate inner product in the above expressions would be determined by the specific choice of the associated reproducing kernel $k$.

Note that there exist various choices of kernel functions and many of them depend on some hyperparameters which determine the kernel's characteristics. Depending on the choice of kernel and hyperparameters, the result of the regression varies, and the problem of choosing the best kernel and its hyperparameters is known as the "training" process of the GP regression [216]. In this chapter, we use marginal likelihood maximization, which is one of the most common training methods.

After training, one would like to study how close the GP model approximates the target function. In order to do this, we use the GP Upper Confidence Bound (UCB) analysis [180], specifically, the following lemma.

**Lemma 9.1.** *[180, Thm. 6] Assume that the noise sequence $\{\epsilon_j\}_{j=1}^{\infty}$ is zero-mean and uniformly bounded by $\sigma_n$. Let the target function $h : \bar{\mathcal{X}} \to \mathbb{R}$, with bounded domain $\bar{\mathcal{X}}$, be a member of $\mathcal{H}_k(\bar{\mathcal{X}})$ associated with a bounded kernel $k$, with its RKHS norm bounded by $\eta$. Then, with probability of at least $1 - \delta$, the following holds for all $x \in \bar{\mathcal{X}}$ and $N \geq 1$*

$$\mathbb{P}\left\{|\mu(x|\mathbb{D}_N) - h(x)| \leq \beta\sigma(x|\mathbb{D}_N), \forall x \in \bar{\mathcal{X}},\ \forall N \geq 1\right\} \geq 1 - \delta,$$

*where*

$$\beta = \left(2\eta^2 + 300\gamma_{N+1}\ln^3((N+1)/\delta)\right)^{0.5}.$$

Here, $\gamma_{N+1}$ is the maximum information gain after getting $N + 1$ data points, and $\mu$, $\sigma^2$ are the mean and variance of the posterior GP given by (2.26) and (2.27).

*Proof.* See [180, Thm. 6]. □

[2] $k$ is a positive definite kernel if its associated kernel matrix $K(x_1, x_2)$, whose $(i^{th}, j^{th})$ element is defined as $k(x_i, x_j)$, is positive semi-definite for any distinct points $x_1, x_2 \in \mathcal{X}$.

[3] The measure of smoothness is defined as $\|h(x) - h(x')\|_2 \leq \|h\|_k \|k(x, \cdot) - k(x', \cdot)\|_k\ \forall x, x' \in \bar{\mathcal{X}}$

In this lemma, the assumption about the boundedness of $\|h\|_k$ implicitly requires a "low complexity" of the target function [41]. The bound $\eta$ is usually unknown a priori, but a trial-and-error approach to find its value suffices in practice [180]. $\gamma_{N+1}$ quantifies the reduction of uncertainty about $h$ in terms of entropy. It has a sublinear dependency on $N$ for many commonly used kernels and it can be efficiently approximated up to a constant [180].

### 9.3.2 GP Regression for Affine Target Functions

In this section, we use GP regression to learn the mismatch term $\Delta_V(x,u)$ (9.1) from data. From (9.3), we know that $\Delta_V(x,u)$ is affine in $u$. If we use an arbitrary kernel, we cannot exploit this information in the GP regression. Therefore, our first objective is to construct an appropriate kernel that captures the control-affine structure of $\Delta_V$ in the regression. In order to do this, we introduce the general formulation of this problem in this section. Consider $p$ functions, $h_i : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \cdots, p$, and define

$$h_c(x,y) \coloneqq [h_1(x)\ h_2(x)\ \cdots\ h_p(x)] \cdot y, \tag{9.4}$$

where $y \in \mathcal{Y} \subset \mathbb{R}^p$. Our objective is to estimate the function $h_c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ which is affine in $y$ by using GP regression, given its measurements $z_j = h_c(x_j, y_j) + \epsilon_j$ for $j = 1, \cdots, N$.

The underlying structure of $h_c(x,y)$ tells us that it contains information about $p$ random functions $\{h_i(x)\}_{i=1}^p$ condensed to a single scalar value by a dot product with $y$. Therefore, it is natural to consider $p$ underlying kernels and their composition. We propose to use the following kernel structure.

For $i = 1, \ldots, p$, consider covariance functions $k_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

**Definition 9.1** (Affine Dot Product Compound Kernel)**.** *Define* $\mathbf{k}$ *given by*

$$\mathbf{k}\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) \coloneqq y^T Diag([k_1(x,x'), \cdots, k_p(x,x')])y', \tag{9.5}$$

*as the* Affine Dot Product *(ADP) compound kernel of* $p$ *individual kernels* $k_1(x,x')$, $\cdots$, $k_p(x,x')$.

Note that for a fixed $(x,x')$, the ADP compound kernel resembles the well-known dot product kernel, defined as $k(y,y') = y^T y'$ [216].

**Lemma 9.2.** *If* $k_1, \cdots, k_p$ *are positive definite kernels, the ADP compound kernel* $\mathbf{k}$ *is also positive definite. Furthermore, if* $k_1, \cdots, k_p$ *are bounded kernels,* $\mathbf{k}$ *is also bounded.*

*Proof.* Consider the Gram matrix of $\mathbf{k}$, $K_c \in \mathbb{R}^{N \times N}$ for $\{(x_j, y_j)\}_{j=1}^N$. Let $K_i$ be the Gram matrix of $k_i$ for $\{x_j\}_{j=1}^N$. Define $Y \coloneqq [y_1\ y_2\ \cdots\ y_N] \in \mathbb{R}^{p \times N}$, and let $\mathbf{y_i}^T$ be the $i$-th row of $Y$. Then,

$$K_c = \sum_{i=1}^{p} \left(\mathbf{y_i}\ \mathbf{y_i}^T\right) \circ K_i,$$

where $\circ$ indicates the Hadamard product [84]. By the Schur Product Theorem [84], if the $k_i$ are positive definite kernels, then since each $\mathbf{y_i}\mathbf{y_i}^T$ and $K_i$ are positive semidefinite, $K_c$ is a positive semidefinite matrix. Therefore, $\mathbf{k}$ is a positive definite kernel by definition. Also, if the $k_i$ are bounded kernels, each $K_i$ is bounded so $K_c$ is also bounded. Therefore, $\mathbf{k}$ is a bounded kernel. □

By Lemma 9.2, since $\mathbf{k}$ is positive definite, it is a valid covariance function. Consider a set of functions $\mathcal{H}_{\mathbf{k}}(\mathcal{X} \times \mathcal{Y}) := \{h_c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \mid \exists h_i \in \mathcal{H}_{k_i} \text{ for } i = 1, \cdots, p, \text{ s.t. } h_c(x, y) = [h_1(x), \cdots, h_p(x)] \cdot y\}$ where each $\mathcal{H}_{k_i}$ is the RKHS whose reproducing kernel is $k_i$. Then, the following holds:

**Theorem 9.1.** *$\mathcal{H}_{\mathbf{k}}(\mathcal{X} \times \mathcal{Y})$ is an RKHS whose reproducing kernel is $\mathbf{k}$ in Definition 9.1.*

*Proof.* Define the inner product of $\mathcal{H}_{\mathbf{k}}$ to be

$$\langle h_c, h_c' \rangle_c := \sum_{i=1}^{p} \langle h_i, h_i' \rangle_i,$$

for $\forall h_c, h_c' \in \mathcal{H}_{\mathbf{k}}$ where $\{h_i\}_{i=1}^p$ and $\{h_i'\}_{i=1}^p$ are sets of functions whose $i$-th elements are from $\mathcal{H}_{k_i}$ that satisfy $h_c(x, y) = [h_1(x), \cdots, h_p(x)] \cdot y$ and $h_c'(x, y) = [h_1'(x), \cdots, h_p'(x)] \cdot y$, respectively. Such sets of functions should exist by definition of $\mathcal{H}_{\mathbf{k}}$. $\langle h_i, h_i' \rangle_i$ is the inner product of $\mathcal{H}_{k_i}$. It is trivial that this definition satisfies the axioms of the inner product. Then,

$$\begin{aligned}\left\langle h_c(\cdot, \blacksquare), \mathbf{k}\left(\begin{bmatrix}\cdot \\ \blacksquare\end{bmatrix}, \begin{bmatrix} x \\ y \end{bmatrix}\right)\right\rangle_c &= \sum_{i=1}^{p} y_i \langle h_i(\cdot), k_i(\cdot, x) \rangle_i \\ &= \sum_{i=1}^{p} y_i h_i(x) = h_c(x, y).\end{aligned}$$

The first equality holds because of Definition 9.1 and the definition of $\langle \cdot, \cdot \rangle_c$. The second equality holds because of the reproducing property of each $k_i(\cdot, \cdot)$. □

Theorem 9.1 allows us to apply the UCB result from Lemma 9.1 to $h_c(x, y)$ with some additional conditions which will be specified in Section 9.4. Fitting an affine function $h_c(x, y)$ can now be treated in the same way as any other kind of general GP regression, but with a specific choice of covariance function given by (9.5).

One caveat of this regression is that depending on the distribution of the inputs $y_j$ in the data, this problem can be underdetermined. For instance, when every $y_j$ is a constant vector, there are infinitely many choices of valid $h_i(x)$ that give the same estimation error. Nevertheless, under our GP regression structure, this evidence of underdetermination is implicitly captured by larger values of the variance of the posterior. In practice, it is preferable to avoid such underdetermination since we want to reduce the uncertainty of the GP posterior. Therefore, we need to carefully collect the training data to make sure we capture rich

enough information about the target function. In Section 9.5 and, more in depth in Chapter 10 of this thesis, we propose several methods for this purpose. In the system identification literature, this is related to the property of persistency of excitation [198].

Finally, the main benefit of exploiting the affine structure in the kernel is revealed in the expressions for the posterior distribution's mean and variance. This is the main difference in how we use the ADP kernel compared to [192], where a similar kernel is proposed for a special case $p = 2$. Let $X \in \mathbb{R}^{n\times N}$, $Y \in \mathbb{R}^{p\times N}$ be matrices whose column vectors are the inputs $x_j$ and $y_j$ of the collected data, respectively, and let $\mathbf{z} \in \mathbb{R}^N$ be the vector containing the output measurements $z_j$. Then, plugging them and the ADP compound kernel into (2.26) and (2.27) gives the following expressions for the mean and variance of the posterior query point $(x_*, y_*)$:

$$\mu(x_*, y_*|\mathbb{D}_N) = \underbrace{\mathbf{z}^T(K_c + \sigma_n^2 I)^{-1}K_{*Y}^T}_{=:\, m(x_*|\mathbb{D}_N)^T} y_*, \tag{9.6}$$

$$\sigma^2(x_*, y_*|\mathbb{D}_N) = y_*^T \underbrace{\big(K_{**} - K_{*Y}(K_c + \sigma_n^2 I)^{-1}K_{*Y}^T\big)}_{=:\, \Sigma(x_*|\mathbb{D}_N)} y_*, \tag{9.7}$$

where $K_c \in \mathbb{R}^{N\times N}$ is the Gram matrix of $\mathbf{k}$ for the training data inputs $(X, Y)$, $K_{**} = Diag\,([k_1(x_*, x_*), \ldots, k_{m+1}(x_*, x_*)]) \in \mathbb{R}^{p\times p}$, and $K_{*Y} \in \mathbb{R}^{p\times N}$ is given by

$$K_{*Y} = \begin{bmatrix} K_{1*} \\ \vdots \\ K_{p*} \end{bmatrix} \circ Y, \quad K_{i*} = [k_i(x_*, x_1), \cdots, k_i(x_*, x_N)],$$

Here, $\circ$ denotes the element-wise product. Readers can observe that (9.6) and (9.7) are affine and quadratic in $y_*$, respectively. These structures are critical when formulating the uncertainty-aware CLF chance constraint as a second-order cone constraint in the next section.

## 9.4 Uncertainty-Aware Min-norm Stabilizing Controller

### 9.4.1 Probabilistic Bounds on the CLF Derivative

We have already presented all the necessary tools to verify the probabilistic bounds on the mismatch term $\Delta_V(x, u)$ in (9.3). Indeed, learning $\Delta_V$ corresponds to the GP regression problem defined by (9.6), (9.7), in which the target function $h_c$ is $\Delta_V$, $x$ is the state, $y = [1, u^T]^T$, $p = m + 1$, $h_1$ is $L_{\Delta f}V$, and $h_{i+1}$ is $L_{\Delta g}V$'s $i$-th element for $i = 1, \cdots, m$.

**Assumption 9.1.** *Consider bounded reproducing kernels $k_i$ for $i = 1, \cdots, m+1$. We assume that $L_{\Delta f}V$ is a member of $\mathcal{H}_{k_1}$ and each $i$-th element of $L_{\Delta g}V$ is a member of $\mathcal{H}_{k_{i+1}}$ for $i = 1, \cdots, m$, respectively. We assume that their RKHS norms are bounded.*

**Lemma 9.3.** *Under Assumption 9.1 and with a compact set of admissible control inputs $\mathcal{U}$, $\Delta_V$ is a member of $\mathcal{H}_{\mathbf{k}}$, the RKHS created by the ADP compound kernel of $\{k_i\}_{i=1}^{i=m+1}$. Moreover, its RKHS norm is bounded, namely $\|\Delta_V\|_{\mathbf{k}} \leq \eta$.*

*Proof.* The proof follows from Thm. 9.1 and the definition of the inner product for $\mathcal{H}_{\mathbf{k}}$ in the proof of Thm. 9.1. □

**Assumption 9.2.** *We have access to measurements $z_i = \dot{V}(x_i, u_i) - (L_{\tilde{f}}V(x) + L_{\tilde{g}}V(x)u_i) + \epsilon_i$, and the noise term $\epsilon_i$ is zero-mean and uniformly bounded by $\sigma_n$.*

With Assumptions 9.1, 9.2 and Lemma 9.3, we can now apply Lemma 9.1 to our regression problem.

**Theorem 9.2.** *Let Assumptions 9.1 and 9.2 hold. Let $\beta := \left(2\eta^2 + 300\gamma_{N+1}\ln^3((N+1)/\delta)\right)^{0.5}$, with $N$ the number of data points, and $\gamma_{N+1}$ as defined in Lemma 9.1. Let $\mu_V(x, u|\mathbb{D}_N)$ and $\sigma_V^2(x, u|\mathbb{D}_N)$ be the mean and variance of the posterior for $\Delta_V$ using the ADP compound kernel, at a query point $(x_*, u_*)$ as obtained from (9.6) and (9.7), with $y = [1, u]$. Let $\mathcal{X}$ and $\mathcal{U}$ be bounded sets. Then, the following holds:*

$$\mathbb{P}\big\{\, |\mu_V(x, u|\mathbb{D}_N) - \Delta_V(x_*, u_*)| \leq \beta\sigma_V(x, u|\mathbb{D}_N), \forall x \in \mathcal{X},\ \forall u \in \mathcal{U},\ \forall N \geq 1\big\}. \tag{9.8}$$

*Proof.* Proof follows from Lemmas 9.1 and 9.3. □

The error in the estimation of the mismatch term $\Delta_V$ is now bounded for some confidence level. From (9.8) we can easily derive the bounds on the true derivative of the CLF for a probability of at least $1 - \delta$:

$$\tilde{\dot{V}}(x, u) + \mu_V(x, u|\mathbb{D}_N) - \beta\sigma_V(x, u|\mathbb{D}_N) \leq \dot{V}(x, u) \leq \tilde{\dot{V}}(x, u) + \mu_V(x, u|\mathbb{D}_N) + \beta\sigma_V(x, u|\mathbb{D}_N). \tag{9.9}$$

### 9.4.2 GP-Based CLF Second-Order Cone Program

Taking the upper bound of (9.9), we can enforce the exponential CLF constraint of (2.20b) with a probability of at least $1 - \delta$, and incorporate the resulting chance constraint into a min-norm optimization problem that defines a feedback control law $\pi_{\text{GP-CLF}}\colon \mathcal{X} \to \mathcal{U}$ pointwise:

---

**GP-CLF-SOCP**:

$$\begin{aligned} \pi_{\text{GP-CLF}}(x) = \quad & \underset{u\in\mathcal{U},\ d\in\mathbb{R}}{\arg\min} \quad \|u\|_2^2 + p\ d^2 \\ \text{s.t.} \quad & \tilde{\dot{V}}(x, u) + \mu_V(x, u|\mathbb{D}_N) + \beta\sigma_V(x, u|\mathbb{D}_N) + \lambda V(x) \leq d. \end{aligned} \tag{9.10}$$

---

**Remark 9.2.** *The stability constraint is relaxed in order to guarantee the feasibility of the problem. If the initial state $x_0$ is outside the CLF maximum sublevel set for exponential stability $\Omega_{c_{exp}}$, we cannot guarantee exponential convergence and neither can the controller which uses the true plant dynamics. However, even for this case, we still do guarantee that the approximation error of the CLF derivative is bounded as given by* (9.9) *with probability* $1 - \delta$.

Note that this optimization problem does not require knowledge about the true plant dynamics. The fact that $\mu_V(x, u|\mathbb{D}_N)$ and $\sigma_V^2(x, u|\mathbb{D}_N)$ are affine and quadratic in $u$, respectively, is crucial for the following main result of the chapter:

**Theorem 9.3.** *Using the proposed ADP compound kernel from Definition 9.1, the uncertainty-aware optimization problem* (9.10) *is convex, meaning that its global minimum can be reliably recovered. Specifically, it is a Second-Order Cone Program (SOCP).*

*Proof.* The standard form for an SOCP consists of a linear objective function subject to one or more second-order cone inequality constraints and/or linear equality constraints.

Since we assume that the bounds on the control input $u \in \mathcal{U}$ are linear, these are directly a special case of second-order cone constraints.

Let's first transform the quadratic objective function into a second-order cone constraint and a linear objective. Let the objective function be $J(u, d) \coloneqq \|u\|_2^2 + p\ d^2$. Note that by taking $\phi = [u^T, d]^T$ we can express the objective as $J(\phi) = \phi^T Q \phi$. Next, by setting $z \coloneqq L\phi$, where $L$ is the matrix square-root of $Q$, we can rewrite $J(z) = \|z\|_2^2$. Note that minimizing $J$ gives the same result as minimizing $\bar{J}(z) \coloneqq \|z\|_2$. Now we can move the objective function $J'$ into a second-order cone constraint by taking the epigraph form $\|z\|_2 \leq t$ and minimizing the new linear objective function $\bar{\bar{J}}(t) \coloneqq t$.

The next step is to prove that the CLF chance constraint is a second-order cone constraint. Note that $\dot{\tilde{V}}(x, u) = L_{\tilde{f}} V(x) + L_{\tilde{g}} V(x) u$, and $\mu_V(x, u|\mathbb{D}_N) = m_V(x|\mathbb{D}_N)^T [1, u^T]^T$ from (9.6) are both control-affine. Note that $\sigma_V(x, u|\mathbb{D}_N) = \sqrt{[1, u^T] \Sigma_V(x|\mathbb{D}_N) [1, u^T]^T}$ from (9.7) can be rewritten as $\sigma_V(x, u|\mathbb{D}_N) = \|M(x)u + n(x)\|_2$, although we omit the expressions of $M$ and $n$ for conciseness. Therefore, the CLF chance constraint is a second-order cone constraint, and the resulting optimization problem is an SOCP with two second-order cone constraints corresponding to the original objective function and the CLF chance constraint (plus the input bounds). SOCPs are inherently convex. □

## 9.5 Data Collection

In this section, we introduce an algorithm that efficiently collects measurements of $\Delta_V$ for the GP regression. This data should contain rich enough information about $\Delta_V$, especially about its dependency on $u$ as discussed in Section 9.3, and since our goal is to obtain a locally stabilizing controller, it is preferable to exclude the data from outside the region of attraction (RoA) of the origin for efficiency. To this end, we propose an algorithm that

iteratively collects new data and trains a new GP model in an episodic learning fashion. The algorithm uses the level sets of the CLF as "guides" for expanding the training region by exploiting Lemma 2.1. In addition, we use the idea of greedy search in the Bayesian Optimization literature [180] to actively explore the most uncertain area of the training region. Our algorithm is based on the active learning algorithm of [19], although while [19] focuses on guaranteeing safety online, our objective is to maximize the efficiency of the offline data collection.

### 9.5.1 Discrete-Time Measurements

First consider how to obtain inputs $(x_j, u_j)$ and labels $(z_j)$—measurements of $\Delta_V(x_j, u_j)$—of the training data. Let $x(t)$ and $u(t)$ be the state and control input measurements at time $t$ and $x(t + \Delta t)$ be the state measurement at the next timestep. We can use these values to create input-label pairs with $\mathcal{O}(\Delta t^2)$ approximation error:

$$x_j = \frac{x(t + \Delta t) + x(t)}{2}, \quad u_j = u(t),$$
$$z_j = \frac{V(x(t + \Delta t)) - V(x(t))}{\Delta t} - \tilde{\dot{V}}(x_j, u_j).$$

Note that $u_j$ is the control input during the interval $[t, t+\Delta t)$, and $z_j$ is the difference between the value of $\dot{V}(x_j, u_j)$ obtained from numerical differentiation and the nominal model-based estimate.

### 9.5.2 Estimation of the Region of Attraction

Next, we introduce a new certificate with the learned uncertainty for a conservative estimation of the RoA of the origin. Notice that the condition for inclusion in the RoA provided by Lemma 2.1, is only valid when there is no model-plant mismatch. Thus, we have to incorporate the learned uncertainty terms from Section 9.3 as we do when we formulate the GP-CLF-SOCP in Section 9.4.

**Theorem 9.4.** *Taking the GP posterior distribution from the training data* $\mathbb{D}_N = \{(x_j, u_j, z_j)\}_{j=1}^N$*, and* $\beta$ *from* (9.8)*, if there exists a* $c > 0$ *such that for all* $x \in \Omega_c := \{x \in \mathcal{X} : V(x) \leq c\}$ *it holds that*

$$\inf_{u \in \mathcal{U}} \ \tilde{\dot{V}}(x, u) + \mu_V(x, u|\mathbb{D}_N) + \beta \sigma_V(x, u|\mathbb{D}_N) < 0, \tag{9.11}$$

*then* $\Omega_c$ *is in the RoA with probability at least* $(1 - \delta)$*.*

*Proof.* Proof follows from Lemma 2.1 and Theorem 9.2. □

Notice that this certificate is "conservative" in the sense that it takes the worst-case bound of the effect of the uncertainty term, based on the collected data. Therefore, if we

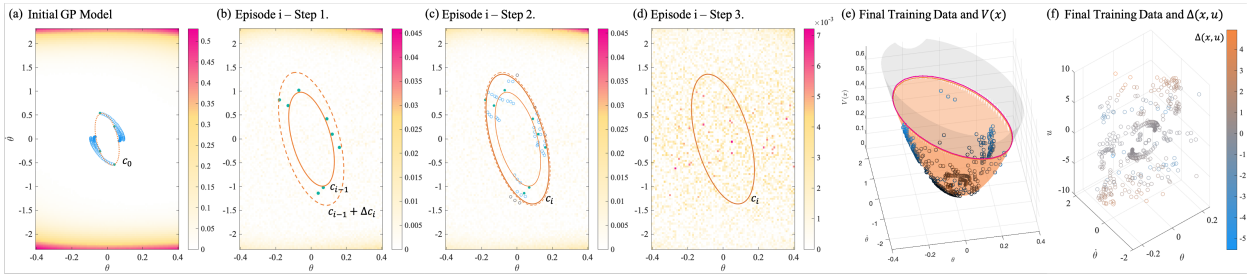



Figure 9.1: **(a–d): Visualization of the episodic learning data collection algorithm running on the inverted pendulum example:** Color map represents the maximum variance of the posterior GP, $\max_{u \in U} \sigma_V(x, u)$. Orange curves: level curves of the CLF. Green points: initial states for the rollouts, Blue points: trajectory points added to the training data. Grey points: trajectory points excluded from the training data since they are outside $\Omega_{c_i}$. **(a) Initial GP Model:** Trajectories sampled from the initial level set $\Omega_{c_0}$ by running the CLF-QP are collected to create an initial GP model. **(b) Episode $i$-Step 1:** $N_e$ initial states and initial control inputs in $(\Omega_{(c_{i-1}+\Delta c_i)} \setminus \Omega_{c_{i-1}}) \times \mathcal{U}$ are determined where $\sigma_V$ are maximal. **(c) Episode $i$-Step 2:** Simulations are run from such initial points and the resulting trajectories are saved. At the same time, $c_i$ is determined by evaluating (9.11) for the sampled trajectories. **(d) Episode $i$-Step 3:** Finally, the $i$-th GP model is updated. Note the reduction in the variance. (Total episodes $= 7$, $i = 3$ for (b), (c), (d).) **(e, f): Distribution of the final training data** plotted in the $x$–$V(x)$ space (blue points) and plotted in $x$–$u$ space, respectively. (e) Level curve in color magenta is the $\Omega_{c_{max}}$ (maximum level set contained in the RoA) for the true plant. The value of CLF is plotted in grey and the orange region is the region verified as RoA through the data collection algorithm. (f) The color indicates the value of $z_i$, the measurement of $\Delta_V(x_i, u_i)$. The number of data points is 425.

collect more data and improve our GP model to have less uncertainty, then the conservatism will reduce and we will be able to obtain a bigger subset of the RoA. This is the central principle of the algorithm.

### 9.5.3 Algorithm Overview

Finally, we give an overview of the proposed algorithm.

#### Initial GP Model

We start by considering a level set $\Omega_{c_0}$ which is small-enough to be a subset of the RoA (Fig 9.1.a). Such $c_0 > 0$ always exists due to our assumption that $V$ is a locally valid CLF. We collect an initial batch of training data $\mathbb{D}_{N_0}$ from a set of trajectories whose initial states

are randomly sampled from $\Omega_{c_0}$, and train an initial GP regression model. Here, we use the nominal model-based CLF-QP from (2.20) as our stabilizing controller.

**Episodic Learning**

The main loop of our algorithm consists of a series of episodes, and each $i$-th episode is mainly composed of three steps. **1)** In the first step (Fig. 9.1.b), we obtain a set of $N_e$ points from $(\Omega_{(c_{i-1}+\Delta c_i)} \setminus \Omega_{c_{i-1}}) \times \mathcal{U}$ at which the variance of the posterior of the current GP model is maximal. $\Delta c_i$ is the parameter that determines the size of the new exploration region. **2)** Next (Fig 9.1.c), we run short rollouts by taking each point from Step 1 as our initial state and initial control input. During the rollouts, we also evaluate the stabilizability condition (9.11) at each timestep. Note that such evaluation is a feasibility problem which is also an SOCP since (9.11) is a second-order cone constraint. After the rollouts, we expand the level of $V$ (we determine $c_i$) up to a point for which (9.11) becomes infeasible. **3)** Finally (Fig 9.1.d), we add the data obtained from the trajectories within $\Omega_{c_i}$ to our data set, and train the next GP regression model.

**Remark 9.3.** *In Step 2 of an episode, we check condition* (9.11) *only for finite sampled states in* $\Omega_{c_i} \setminus \Omega_{c_{i-1}}$*, whereas Theorem 9.4 requires* (9.11) *to be satisfied at every state in* $\Omega_{c_i}$*. Notice that brute-force verification for the whole region of* $\Omega_{c_i}$ *will scale poorly with state dimension. Even though we do not have the rigorous guarantee of Theorem 9.4 with this algorithm, the error in the estimated* $c_{max}$ *does not affect the probabilistic guarantee of the resulting GP-CLF-SOCP controller. In practice, we observe that we can well approximate* $c_{max}$ *such that* $\Omega_{c_{max}}$ *is contained in the true RoA (See Fig. 9.1(e)).*

## 9.6 Examples

### 9.6.1 Two-dimensional System: Inverted Pendulum

Consider a control-affine two-dimensional inverted pendulum as the one in [19], with parameters of the plant $m_{\text{plant}} = 2\text{kg}$, $l = 1\text{m}$ and for the model, $m_{\text{model}} = 1\text{kg}$, $l = 1\text{m}$, which results in model uncertainty in both $f$ and $g$ in (2.2).

A CLF-QP controller (2.20) based on the nominal model is designed to stabilize the pendulum to the upright position. In order to illustrate the effects of model uncertainty, we compare it with the CLF-QP controller based on the true plant dynamics. The difference between the two controllers (Fig. 9.2) is due to the effects of model uncertainty. Specifically, in this case the model uncertainty makes the system converge more slowly.

Fig. 9.1 depicts the data collection algorithm and the resulting training data for the GP model. The results of deploying the GP-CLF-SOCP controller, with a confidence level of $1-\delta = 0.95$, are presented in Fig. 9.2 in blue lines. Note that the results are very similar to those from the CLF-QP based on the true plant dynamics, which means that the GP-CLF-SOCP successfully captures the correct effects of model uncertainty. Also, the computation

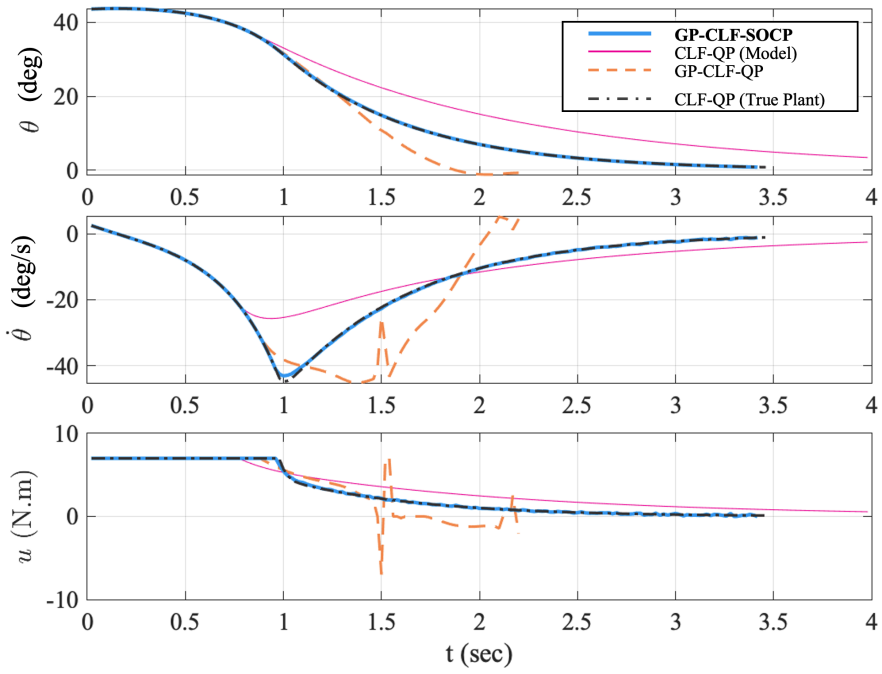



Figure 9.2: Simulation results of applying the GP-CLF-SOCP to the inverted pendulum example, with a model-plant mismatch of $m_{\text{plant}}$=2kg, $m_{\text{model}}$=1kg, compared to the nominal-model-based CLF-QP, and to the GP-CLF-QP that does not consider the uncertainty affected by $u$. Results of the CLF-QP based on the true plant are also provided to show that the GP-CLF-SOCP learns the correct exponential CLF constraint.

time of the GP-CLF-SOCP, including the GP inference time, is $9.1 \pm 2.2$ms (max: 25.7ms) on a laptop with a 10th-gen Intel Core i7 and 32GB RAM.

In order to benchmark the GP-CLF-SOCP, we compare its performance with the one obtained if we only learn the uncertainty in $f$, as done in previous works [58, 223]. For this, we design a GP-based Control Lyapunov Function Quadratic Program (GP-CLF-QP) that only learns the uncertainty $L_{\Delta f}V$ in (9.3). The results of this controller are also shown in Fig. 9.2.

### 9.6.2 System with Multiple Control Inputs: Kinematic Bicycle

Next, in order to show that our method can be successfully applied to systems with higher state dimension and multiple control inputs, we apply it to track a reference trajectory using a kinematic bicycle model. The state is defined as $x = [p_x, p_y, v, \theta, \gamma]^T$ ($p_x, p_y$: position coordinates, $v$: speed, $\theta$: heading angle, $\gamma$: tangent of the steering angle). The dynamics of the system are given as

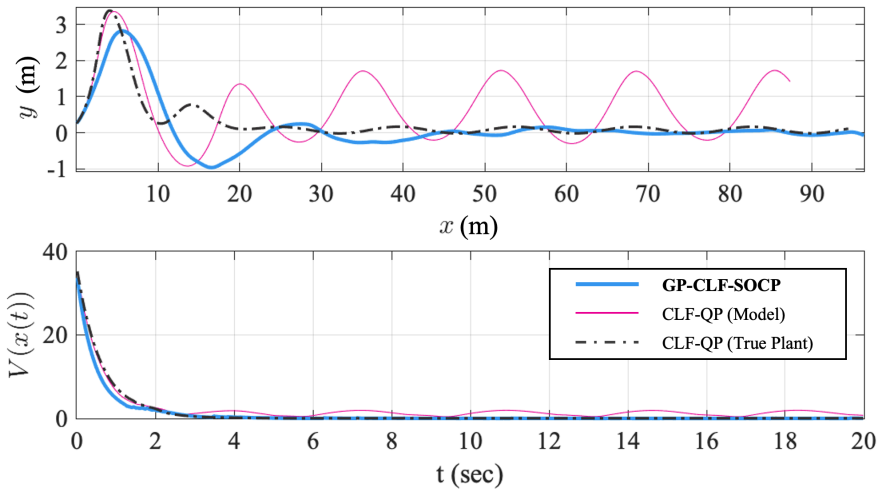



Figure 9.3: Trajectories in $x-y$ plane (Top) and histories of $V(x(t))$ (Bottom) of the kinematic vehicle under artificial drift and friction to illustrate the applicability of the GP-CLF-SOCP to multi-input systems. Comparison between GP-CLF-SOCP, CLF-QP(Model), and CLF-QP(Plant). The sampling time is set as 20ms, and the comptutation time of the GP-CLF-SOCP per timestep is $10.3 \pm 1.9$ms (max: 20.4ms). Number of training data points for GP-CLF-SOCP: 961.

$$\dot{x} = f(x) + g(x)u, \quad f(x) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ -f_\mu \\ v\gamma \\ 0 \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_v & 0 \\ 0 & 0 \\ 0 & b_\gamma \end{bmatrix}, \tag{9.12}$$

where $u \in \mathbb{R}^2$, and $f_\mu$, $b_v$, $b_\gamma$ are constants that emulate friction and skid effects. For the nominal model, we assume no such effects ($f_\mu = 0$, $b_v = b_\gamma = 1$) and for the plant, we use $f_\mu = 1$, $b_v = 1.5$, $b_\gamma = 0.75$. The objective is to stabilize to a constant-velocity trajectory along the x-axis; $v(t) = 5$, $p_y(t) = \theta(t) = \gamma(t) = 0$. The initial state is set as $x_0 = [0, 0.25, 2, 0.25, 0.25]^T$.

Fig. 9.3 shows the simulation results of the GP-CLF-SOCP and those of the CLF-QP based on the nominal model and the true plant. Here, we use a polynomial CLF [163], which is verified to be locally stabilizing for the nominal model. While the nominal model-based CLF-QP oscillates around the reference trajectory, the GP-CLF-SOCP successfully converges to the reference trajectory.

## 9.7 Chapter Summary

In this chapter, we introduced a method to design a stabilizing controller for control-affine systems with both state and input-dependent model uncertainty using GP regression. For

this purpose, we have proposed the novel ADP compound kernel, which captures the control-affine nature of the problem. This permits the formulation of the so-called GP-CLF-SOCP, which is solved online to obtain an exponentially stabilizing controller with probabilistic guarantees. After testing it on the numerical simulation of two different systems, we obtain a clear improvement with respect to the nominal model-based CLF-QP and we are able to closely match the performance of the true plant-based controller.

# Chapter 10

# Recursively Feasible Probabilistic Safe Control under Uncertainty

*This chapter is based on the article titled "Recursively Feasible Probabilistic Safe Online Learning with Control Barrier Functions" [31][1], co-authored by Jason J. Choi, Wonsuhk Jung, Bike Zhang, Claire J. Tomlin and Koushil Sreenath.*

In this chapter, we first introduce a reality-gap-aware reformulation of CBF-based safety-critical controllers using Gaussian Process (GP) regression to bridge the gap between an approximate mathematical model and the real system. Our proposed robust controller takes into account the GP prediction uncertainty and outputs control actions that guarantee safety with high-probability when feasible.

Compared to previous work, we study the conditions under which the resulting robust safety-critical controller is feasible. This feasibility analysis results in a set of richness conditions that the available information about the system should satisfy to guarantee that a safe control action can be found at all times. Then, we use these conditions to devise an event-triggered online data collection strategy that ensures the recursive feasibility of the learned safety-critical controller. Our proposed methodology endows the system with the ability to reason at all times about whether the current information at its disposal is enough to ensure safety or if new measurements are required. This, in turn, allows us to provide formal results of forward invariance of a safe set with high probability, even in a priori unexplored regions.

[1]This article is the journal version of the conference paper "Pointwise Feasibility of Gaussian Process-based Safety-Critical Control under Model Uncertainty" [30].

# 10.1 Introduction

## 10.1.1 Motivation

In many real-world control systems, such as aircrafts, industrial robots or autonomous vehicles, in order to prevent system failure and catastrophic events, it is crucial to ensure that the system always stays within a set of safe states. Mathematical models of the system's dynamics can often be useful to design controllers that can enforce such safety constraints. However, since designing accurate models for complex real systems is not easy, imperfect models are typically used in practice, and the guarantees of the designed controllers can be lost when these model imperfections are not addressed appropriately.

On the other hand, modern data-driven control techniques have emerged as promising tools for solving complex control tasks. Nevertheless, in the absence of interpretable model-based knowledge, such methods usually fall short of theoretical guarantees. Moreover, data-driven approaches typically require collecting enough real-world data to accurately characterize the system. This presents a dilemma for safety-critical systems: in order to collect data, we need to deploy the system, but without having previously secured sufficient data to confidently deploy the system in a safe manner, we cannot dare to do so.

In this chapter, we present an approach to address this dilemma and guarantee the safety of systems with uncertain dynamics. Our methodology lies at the intersection of model-based and data-driven control techniques. An imperfect dynamics model is complemented by the information gathered from data collected safely online from the real system, which allows us to ensure safety without having a perfect model of the system nor offline data.

To intuitively illustrate the working principle of our approach, consider an adaptive cruise control problem where an ego vehicle must maintain a safe distance from the car in front. The key idea behind our method is to make sure that at all times the ego car has enough information about its dynamics reacting to a safe control action (e.g., braking), derived from prior knowledge or data. If the braking effect is well understood and safety is not compromised, the ego vehicle is allowed to follow the driver's commands. However, if the braking uncertainty reaches a critical level, our method commands the ego vehicle to brake, allowing it to measure the braking effect and improve confidence. This critical level constitutes the maximum tolerable uncertainty before it becomes impossible to assure with high confidence that the car will be safe after pressing the brake. This way, if the front car gets closer, the effect of braking is always sufficiently well characterized so that the ego vehicle is ready to prevent a collision. Thus, our method fundamentally focuses on determining if the combination of prior model knowledge and collected data can maintain low uncertainty in a safe control direction or if new information is needed instead.

## 10.1.2 Related Work

In the model-based control literature, various approaches exist for the design of controllers satisfying safety constraints, including Control Barrier Functions (CBFs, [8]), Hamilton-

Jacobi Reachability [13], and Model Predictive Control [200] to name a few. In this chapter, we focus on the CBF-based implementations of safe controllers for nonlinear systems. The main advantages of using CBFs for safety-critical control are twofold. First, the zero-superlevel set of a CBF, which is control invariant, explicitly verifies the state domain where safety is guaranteed. Second, while guaranteeing set invariance requires long horizon reasoning, CBFs condense this problem into a simple single time-step condition that should be satisfied at each time, similar to Lyapunov-based methods for stability. This single time-step constraint on the control input derived from a CBF guarantees that the system does not exit the boundary of the zero-superlevel set and thus, remains safe.

Importantly, such safety constraints based on CBFs depend on the dynamics of the system. This means that when the dynamics are uncertain, the usage of an incorrect model in the CBF-based controller might lead to violation of safety. By applying the techniques of adaptive control, this issue can be tackled with an online adaptation of the control law to capture the effects of the uncertainty [145, 184, 130, 21]. However, these approaches usually assume the uncertainties have some restrictive structure, which is hard to verify a priori. Robust control approaches instead consider the worst-case effects of model uncertainty [147, 39], or use the notion of input(disturbance)-to-state safety [107, 4, 111]. Nonetheless, with these methods, an inaccurate characterization of the disturbance bounds used for the robust design can lead to the violation of safety when the estimated bounds are too optimistic, or can lead to impractical conservative behaviors when the bounds are unnecessarily large.

These limitations allude to the core motivation of using modern data-driven techniques to address the effects of model mismatch: learn and adapt to the uncertainties with minimal structural assumptions, and learn the correct magnitude of the robustness bounds. Extensive recent research has in fact empirically proved the validity of data-driven control methods for this purpose. Many of these works use neural networks to learn the mismatch terms [186, 187, 210, 38]. Although these approaches are demonstrated to be practical and effective, it is often difficult to verify the accuracy of the neural network predictions.

Other works, like this work, use non-parametric regression methods, most notably Gaussian Process (GP) regression, that provide a probabilistic guarantee of the prediction quality under mild assumptions. However, many of these works make the important simplifying assumption that the uncertainty in the system dynamics is not affected by actuation [18, 19, 60, 191, 58, 36, 43]. In contrast, for many controlled systems, uncertain input effects, or actuation uncertainty, is very common. For instance, uncertainty in the inertia matrix of a mechanical system directly induces uncertain input effects.

Recent work in [27, 188, 68, 56, 24] has sought to overcome this limitation. However, most of these methods require access to high-coverage data that completely characterize the dynamics of the system. Usually, collecting these data would require exciting the system in many control directions, which might compromise safety in real-world experiments. From our perspective, guaranteeing safety for uncertain systems by using only data that can be safely collected remains an open problem.

### 10.1.3 Contributions

Our work uses Gaussian Process regression to learn the effects of an uncertain dynamics model on the CBF-based safety constraint. Then, a second-order cone program (SOCP)-based controller is proposed that gives a probabilistic safety guarantee when the program is feasible.

The data quality of the GP model significantly impacts the GP prediction accuracy and feasibility of the SOCP controller, consequently affecting the probabilistic safety guarantee. In this chapter, we first establish necessary and sufficient conditions for the feasibility of the SOCP controller, for a given fixed GP model and dataset. This analysis provides a one-directional linkage between data quality and safety, that is, a theoretical check of safety given the dataset. However, if the SOCP controller navigates to state-space regions with insufficient data, feasibility can be lost, leading to safety violations. Existing frameworks in [56, 24] face precisely this issue.

In contrast to these works, in this chapter we also include an event-triggered online data collection mechanism that ensures the recursive feasibility of the SOCP controller. By achieving this, we also fill in the linkage between data quality and safety in the opposite direction: we now use the safety check to inform the data collection. Thus, the two links acting together—evaluating safety to judge whether and how to improve the data, and using the data to make predictions with the GP model and guarantee safety—constitute a systematic online learning-based safety framework for uncertain systems.

Our event-triggered online data collection algorithm ensures at all times the availability of a control input direction that can render the system safe with high probability. If the available prior knowledge from the model and past data is sufficient to characterize such a backup direction, our proposed method simply acts as a safety filter applied to a performance-driven control law. However, whenever the uncertainty in the safe control direction reaches a critical level, our algorithm takes a safe exploration action that improves the knowledge of the system's response to such control inputs. Unlike the strategy in [193], which aims to improve overall accuracy of the GP prediction for a feedback linearization-based controller, our event-triggered data collection focuses on exciting safe control directions and reducing uncertainty specifically in such directions.

Finally, we prove local Lipschitz continuity of the probabilistic safety-critical controller and give formal arguments about the existence and uniqueness of closed-loop executions of the system under our proposed safe online learning algorithm. In turn, this allows us to provide the main theoretical result of this chapter (Theorem 10.5), establishing safety in terms of *set invariance with high probability*, even in regions where there is no prior data and the model knowledge is limited. To our knowledge, this is the first work in the area of CBFs applied to systems with uncertain dynamics that collects data online and provides recursive feasibility guarantees of the CBF-based safe controller.

## 10.2 Problem Statement

Throughout the chapter we again consider control-affine nonlinear systems of the form in (2.2):

$$\dot{x} = f(x) + g(x)u,$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control input. In this chapter we assume that there are no control limits. As noted in Chapter 2, many important classes of real-world systems, such as those with Lagrangian dynamics, can be represented in this form. We assume that $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous. We will call system (2.2) the *true plant*. The problem addressed in this chapter is how to guarantee the safety of the true plant (2.2) when its dynamics $f$ and $g$ are unknown, while trying to accomplish a desired task. Our proposed method will tackle this problem using real-time data and an approximate *nominal model* of the system's dynamics, with $\tilde{f} : \mathcal{X} \to \mathbb{R}^n$ and $\tilde{g} : \mathcal{X} \to \mathbb{R}^{n\times m}$, as in (2.25):

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u.$$

To achieve this safety objective, we use the model-based tool of Control Barrier Functions (from Definition 2.7), through the safety with result introduced in Lemma 2.2.

As we showed in Chapter 2, given a safety-agnostic reference controller $\pi_{\text{ref}} : \mathcal{X} \to \mathbb{R}^m$, the condition in (2.22) can be used to formulate a minimally-invasive safety-filter [8]:

---

**CBF-QP**:

$$\pi_{\text{CBF}}(x) = \quad \underset{u\in\mathbb{R}^m}{\arg\min} \quad \|u - \pi_{\text{ref}}(x)\|_2^2 \tag{10.1a}$$

$$\text{s.t.} \quad L_f B(x) + L_g B(x)u + \gamma(B(x)) \geq 0, \tag{10.1b}$$

---

which is a quadratic program (QP) if the input bounds are linear. This problem is solved pointwise in time to obtain a safety-critical control law $\pi_{\text{CBF}} : \mathcal{X} \to \mathcal{U}$ that only deviates from the reference controller $\pi_{\text{ref}}$ when safety is compromised. However, note that this optimization problem requires perfect knowledge of the dynamics of the system, since the Lie derivatives of $B$ appear in the constraint. Note that this constraint is affine in the control input.

In [219, Thm. 8], it is shown that if $\pi_{\text{ref}}$ and $\gamma$ are Lipschitz continuous functions, $B$ has a Lipschitz continuous gradient and if it satisfies the relative degree one condition in $\mathcal{X}$, i.e., $L_g B(x) \neq 0 \ \forall x \in \mathcal{X}$; then the CBF-QP of (2.23) yields a locally Lipschitz control policy, therefore guaranteeing the forward invariance of $\mathcal{X}_{\text{safe}}$ by Lemma 2.2.

### 10.2.1 Safety under Model Uncertainty

The problem that we want to address in this chapter of the dissertation is how to guarantee that a system with uncertain dynamics (2.2) remains safe with respect to a set $\mathcal{X}_{\text{safe}}$ while

trying to accomplish a safety-agnostic task (as defined by a reference control policy $\pi_{\text{ref}} : \mathcal{X} \to \mathbb{R}^m$). The dynamics of (2.2) are uncertain and only a nominal model (2.25) is available. Moreover, we do not assume having access to any dataset containing previous trajectories of the true plant. Instead, the system must autonomously reason about what data it needs to collect online in order to stay safe with a high probability.

**Problem 10.1.** *For a given safe set $\mathcal{X}_{safe} = \{x \in \mathcal{X} : B(x) \geq 0\}$ and nominal dynamics model* (2.25)*, design a data collection strategy and a data-driven control law $\bar{\pi} : \mathcal{X} \to \mathbb{R}^m$ that together render the set $\mathcal{X}_{safe}$ forward invariant for system* (2.2) *with a high probability, i.e.,*

$$\mathbb{P}\{\ \forall x_0 \in C,\ x(0) = x_0 \implies x(t) \in C, \forall t \in [0, \tau_{max})\ \} \geq 1 - \delta,$$

where $\delta$ is a user-defined risk tolerance and $\tau_{max}$ is the maximum time of existence and uniqueness of the solution $x(t)$ of (2.2) under the control law $\bar{\pi}$.

**Assumption 10.1.** *We assume we have access to a function $B : \mathcal{X} \to \mathbb{R}$ that is a valid CBF for the true plant* (2.2)*, with zero-superlevel set $\mathcal{X}_{safe} = \{x \in \mathcal{X} : B(x) \geq 0\}$.*

Designing good CBFs (in the sense of not overly conservative) for uncertain systems is however non-trivial, and in fact it is an active research topic [54, 160, 91, 125, 95]. Note that our contribution is tangential to such line of research, since even when a valid CBF is available, obtaining a control policy that can guarantee safety of an uncertain system is still an open problem. In fact, Assumption 10.1 is also present in the prior works that most closely align with our research [187, 188, 38, 27, 56, 68]. For our simulations, we use the nominal model to design the CBF. This is known to be a reasonable procedure for feedback linearizable systems whose relative degree is known, due to the inherent robustness properties of CBFs [219, 107].

In practice, Assumption 10.1 guarantees that there exists a control policy that keeps the true plant (2.2) safe. However, since the true dynamics of the system are unknown, without further knowledge it is impossible to verify whether a control input $u$ satisfies the CBF constraint (10.1b).

Note that the CBF constraint (10.1b) for the true plant can be expressed as

$$L_{\tilde{f}}B(x) + L_{\tilde{g}}B(x)u + \Delta_B(x, u) + \gamma(B(x)) \geq 0, \tag{10.2}$$

where $L_{\tilde{f}}B$ and $L_{\tilde{g}}B$ are the Lie derivatives of $B$ computed using the nominal dynamics model (2.25), and the uncertain term $\Delta_B$ is defined for each $x \in \mathcal{X}$, $u \in \mathbb{R}^m$ as

$$\Delta_B(x, u) := (L_f B - L_{\tilde{f}} B)(x) + (L_g B - L_{\tilde{g}} B)(x)u. \tag{10.3}$$

We now present a method to estimate the function $\Delta_B$ using data from the true plant and Gaussian Process (GP) regression. This follows closely the approach presented in Chapter 9 of this thesis. By doing so, it is possible to formulate a probabilistic version of the optimization problem (10.1) that takes into account the current best estimate of the term $\Delta_B$

and the estimation uncertainty. Note that learning $\Delta_B$ is advantageous rather than learning the full dynamics of the system (as is typically done in the model-based reinforcement learning literature) since $\Delta_B$ is a scalar function. Indeed, this function condenses all the safety-relevant model uncertainty into a scalar. Moreover, in order to retain the convexity of the CBF constraint we exploit the control-affine structure of $\Delta_B$ during learning, as in Section 9.3. We now briefly revisit this learning procedure.

## 10.3 Gaussian Process Regression to Estimate the Safety-Critical Uncertainty

We now use Gaussian Process Regression with the ADP compound kernel introduced in Definition 9.1 to estimate the uncertainty $\Delta_B$ taking into account its control-affine structure.

We first introduce some useful notation. We can rewrite (10.3) as

$$\Delta_B(x,u) = \Phi_B(x) \cdot \begin{bmatrix} 1 \\ u \end{bmatrix}, \tag{10.4}$$

where

$$\Phi_B(x) := \begin{bmatrix} L_f B(x) - L_{\tilde{f}} B(x), & L_g B(x) - L_{\tilde{g}} B(x) \end{bmatrix}. \tag{10.5}$$

We can then define a GP prediction model for $\Delta_B$, with domain $\bar{\mathcal{X}} := \mathcal{X} \times \mathbb{R}^{m+1}$, where $\mathbb{R}^{m+1}$ is the space of $y := [1, u^T]^T$.

Looking at Definition 9.1 and letting the target function $h(x,y)$ be $\Delta_B(x,u)$ with $y = [1,u^T]^T$, we can clearly see that by using the ADP compound kernel the prediction of $\Delta_B(x_*,u_*)$ at a query point $(x_*,u_*)$ has a mean function (9.6) that is affine in the control input $u_*$ and a variance (9.7) that is quadratic in $u_*$. This is crucial for the construction of the convex optimization-based safety filter that will be introduced in the next section. We will denote the mean and variance of the prediction of $\Delta_B$ at a query point $(x_*,u_*)$ as $\mu_B(x_*,u_*|\mathbb{D}_N)$ and $\sigma_B^2(x_*,u_*|\mathbb{D}_N)$, respectively. Here, $\mathbb{D}_N$ is the dataset containing $N$ noisy measurements of $\Delta_B$, as explained in Lemma 10.1.

### 10.3.1 Probability Bounds of the GP Prediction

We now revisit [180, Thm. 6] to give a probabilistic bound on the deviation of the true value of $\Delta_B$ from its mean prediction function $\mu_B$. In order to provide guarantees about the behavior of an unknown function at any arbitrary point in its domain that may not belong to the discrete set of available data points, [180, Thm. 6] requires some assumptions. In particular, equivalently to the theoretical result in Lemma 9.1 of the previous chapter, the target function $\Delta_B$ is required to belong to the Reproducing Kernel Hilbert Space (RKHS, [207]) $\mathcal{H}_k(\bar{\mathcal{X}})$ of the chosen kernel, and have a bounded RKHS norm $\|\cdot\|_k$.

**Lemma 10.1.** *[180, Thm. 6] Consider $m+1$ bounded kernels $k_i$, for $i = 1,\dots,(m+1)$. Assume that the $i$th element of $\Phi_B$ is a member of $\mathcal{H}_{k_i}$ with bounded RKHS norm, for $i =$*

$1, \ldots, (m+1)$. *Moreover, assume that we have access to a dataset* $\mathbb{D}_N = \{(x_j, u_j),\ \Delta_B(x_j, u_j) + \epsilon_j\}_{j=1}^N$ *of* $N$ *noisy measurements, and that* $\epsilon_j$ *is zero-mean and uniformly bounded by* $\sigma_n > 0$. *Let* $\beta := \left(2\eta^2 + 300\kappa_{N+1}\ln^3((N+1)/\delta)\right)^{0.5}$, *with* $\eta$ *the bound of* $\|\Delta_B\|_{\mathbf{k}}$, $\kappa_{N+1}$ *the maximum information gain after getting* $N+1$ *data points, and* $\delta \in (0,1)$. *Let* $\mu_B$ *and* $\sigma_B^2$ *be the mean* (9.6) *and variance* (9.7) *of the GP regression for* $\Delta_B$, *using the ADP compound kernel* $\mathbf{k}$ *of* $k_1, \ldots, k_{m+1}$, *at a query point* $(x_*, u_*)$, *where* $x_*$ *and* $u_*$ *are elements of bounded sets* $\mathcal{X} \subset \mathbb{R}^n$ *and* $\mathcal{U} \subset \mathbb{R}^m$, *respectively. Then, the following holds:*

$$\mathbb{P}\Bigg\{ \left|\mu_B(x_*, u_*|\mathbb{D}_N) - \Delta_B(x_*, u_*)\right| \leq \beta\sigma_B(x_*, u_*|\mathbb{D}_N), \\ \forall N \geq 1,\ \forall x_* \in \mathcal{X},\ \forall u_* \in \mathcal{U} \Bigg\} \geq 1 - \delta. \quad (10.6)$$

## 10.4 Probabilistic Safety Filter

We now make use of the probability bound given by Lemma 10.1 to build an uncertainty-aware CBF chance constraint that can be incorporated in a minimally invasive probabilistic safety filter. Let us take the lower bound of (10.6) and note that $\dot{B}(x,u) = \tilde{\dot{B}}(x,u) + \Delta_B(x,u)$. Then, as a result of Lemma 10.1, the following inequality holds with a compound probability (for all $x \in \mathcal{X}$, $u \in \mathcal{U}$ and $N \geq 1$) of at least $1-\delta$:

$$\dot{B}(x,u) \geq \tilde{\dot{B}}(x,u) + \mu_B(x,u|\mathbb{D}_N) - \beta\sigma_B(x,u|\mathbb{D}_N), \quad (10.7)$$

where $\tilde{\dot{B}}(x,u) = L_{\tilde{f}}B(x) + L_{\tilde{g}}B(x)u$ is the CBF derivative computed using the nominal dynamics model of (2.25).

Inequality (10.7) gives a worst-case high-probability bound for the CBF derivative of the true plant (2.2). An important observation is that the right-hand side of (10.7) can be evaluated without having explicit knowledge of the dynamics of the true plant. For each state and control input, the standard deviation $\sigma_B$ of the GP prediction determines the tightness (and, therefore, the conservativeness) of the bound.

We use this lower bound of the CBF derivative to construct a probabilistically robust CBF chance constraint that can be evaluated without explicit knowledge of the dynamics of the true plant, and we incorporate it in a chance-constrained reformulation of the CBF-QP (10.1) safety filter:

---

**GP-CBF-SOCP**:

$$\pi_{\text{GP-CBF}}(x) = \quad \underset{u \in \mathbb{R}^m}{\arg\min}\ \|u - \pi_{\text{ref}}(x)\|_2^2 \quad (10.8a)$$

$$\text{s.t.} \quad \tilde{\dot{B}}(x,u) + \mu_B(x,u|\mathbb{D}_N) - \beta\sigma_B(x,u|\mathbb{D}_N) + \gamma(B(x)) \geq 0. \quad (10.8b)$$

---

This problem is solved at each timestep in real-time to obtain a safety-filtered control law $\pi_{\text{GP-CBF}} : \mathcal{X} \to \mathbb{R}^m$ that only deviates from the reference $\pi_{\text{ref}}$ when safety is compromised for the desired probability bound of $1 - \delta$.

The linear and quadratic structures of the expressions for the mean (9.6) and variance (9.7), respectively, of the GP prediction of $\Delta_B$ when using the ADP compound kernel, lead to this problem being a Second Order Cone Program (SOCP). This is equivalent to what was shown for CLFs in Theorem 9.3 of the previous chapter. Therefore, by exploiting the control-affine structure of the system during the GP regression, we obtain a convex optimization problem that can be solved at high frequency rates when using modern solvers.

**Theorem 10.1.** *For an unknown control-affine system* (2.2) *with associated CBF $B$, let $\mu_B$ and $\sigma_B^2$ be the mean and variance functions of the GP prediction of $\Delta_B$ using the ADP compound kernel from Definition 9.1. Then, the probabilistic safety filter of* (10.8) *is convex. Specifically, it is a Second-Order Cone Program (SOCP).*

*Proof.* In this proof, we rewrite the GP-CBF-SOCP in the standard form for SOCPs. The resulting form will be useful for the analysis in the following sections of the chapter.

The proof follows the steps of Theorem 9.3 replacing the Control Lyapunov Function chance constraint with the probabilistic CBF constraint of (10.8b), and with the different objective of minimizing the distance to the reference controller $\pi_{\text{ref}}$.

The standard form for an SOCP consists of a linear objective function subject to one or more second-order cone inequality constraints and/or linear equality constraints. We first transform the quadratic objective function into a second-order cone constraint and a linear objective. Let the objective be $J(u) \coloneqq \|u - \pi_{\text{ref}}(x)\|_2^2$. Note that for a particular state $x \in \mathcal{X}$, minimizing $J$ over $u$ gives the same result as minimizing $\bar{J}(u) := \|u - \pi_{\text{ref}}(x)\|_2$ over $u$. Now we can move the objective function $\bar{J}$ into a second-order cone constraint by taking the epigraph form $\|u - \pi_{\text{ref}}(x)\|_2 \leq t$ and minimizing the new linear objective function $\bar{\bar{J}}(t) \coloneqq t$.

Next, we prove that the CBF chance constraint (10.8b) is a second-order cone constraint. Note that $\tilde{\dot{B}}(x, u) = L_{\tilde{f}} B(x) + L_{\tilde{g}} B(x) u$ is control-affine. Furthermore, since we use the ADP compound kernel, we can exploit the structure of the posterior in (9.6) and (9.7) to express

$$\mu_B(x, u|\mathbb{D}_N) = m_B(x|\mathbb{D}_N)^T [1 \; u^T]^T, \tag{10.9}$$

$$\sigma_B^2(x, u|\mathbb{D}_N) = [1 \;\; u^T] \Sigma_B(x|\mathbb{D}_N) [1 \; u^T]^T. \tag{10.10}$$

Since $k$ is a valid kernel and we assume the measurement noise variance is strictly positive ($\sigma_n^2 > 0$), $\Sigma_B(x|\mathbb{D}_N)$ is positive definite for any state $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$. We can therefore write

$$\sigma_B(x, u|\mathbb{D}_N) = \left\| \Sigma_B^{1/2}(x|\mathbb{D}_N) \begin{bmatrix} 1 \\ u \end{bmatrix} \right\|_2, \tag{10.11}$$

where $\Sigma_B^{1/2}(\cdot) \in \mathbb{R}^{(m+1)\times(m+1)}$ is the matrix square root of $\Sigma_B(\cdot)$.

We now define the following quantities, where numerical subscripts denote elements of vectors or matrices:

$$\widehat{L_f B}(x|\mathbb{D}_N) := L_{\tilde{f}} B(x) + m_B(x|\mathbb{D}_N)_{[1]} \in \mathbb{R}, \tag{10.12}$$

$$\widehat{L_g B}(x|\mathbb{D}_N) := L_{\tilde{g}} B(x) + m_B(x|\mathbb{D}_N)^T_{[2:(m+1)]} \in \mathbb{R}^{1\times m}, \tag{10.13}$$

$$\Sigma^{1/2}_{L_f B}(x|\mathbb{D}_N) := \Sigma^{1/2}_B(x|\mathbb{D}_N)_{[1:(m+1)],[1]} \in \mathbb{R}^{m+1}, \tag{10.14}$$

$$\Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N) := \Sigma^{1/2}_B(x|\mathbb{D}_N)_{[1:(m+1)],[2:(m+1)]} \in \mathbb{R}^{(m+1)\times m}. \tag{10.15}$$

Note that $\widehat{L_f B}(x|\mathbb{D}_N)$ and $\widehat{L_g B}(x|\mathbb{D}_N)$ are the mean predictions of the true plant's $L_f B(x)$ and $L_g B(x)$, respectively, at a point $x \in \mathcal{X}$. Furthermore, $\Sigma^{1/2}_{L_f B}(x|\mathbb{D}_N)$ and $\Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N)$ correspond to the components of the uncertainty matrix $\Sigma^{1/2}_B(x|\mathbb{D}_N)$ appearing in expression (10.11) depending on whether they multiply the control input or not.

After introducing these quantities, we can now express (10.8b) in the standard form for second-order cone constraints:

$$\beta \left\| \Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N) u + \Sigma^{1/2}_{L_f B}(x|\mathbb{D}_N) \right\|_2 \leq \widehat{L_g B}(x|\mathbb{D}_N) u + \left( \widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)) \right).$$

The GP-CBF-SOCP can therefore be rewritten in the standard form for SOCPs as:

---

**GP-CBF-SOCP (Standard Form)**:

$$\pi_{\text{GP-CBF}}(x) = \underset{(u,t)\in\mathbb{R}^{m+1}}{\arg\min}\ t \qquad \text{s.t.} \tag{10.16a}$$

$$\|u - \pi_{\text{ref}}(x)\|_2 \leq t, \tag{10.16b}$$

$$\beta \left\| \Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N) u + \Sigma^{1/2}_{L_f B}(x|\mathbb{D}_N) \right\|_2 \leq \widehat{L_g B}(x|\mathbb{D}_N) u + \widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)). \tag{10.16c}$$

---

□

## 10.5 Analysis of Pointwise Feasibility

The GP-CBF-SOCP, if feasible, is guaranteed to provide a control input that satisfies the true CBF constraint (10.1b) with high probability. However, since the GP-CBF-SOCP of (10.8) needs to be robust to the prediction uncertainty (through the term involving $\sigma_B$), and the CBF chance constraint is not relaxed, the problem will be infeasible when the uncertainty ($\sigma_B$) is dominant in the CBF chance constraint (10.8b).

**Remark 10.1.** *Note that unlike QPs, SOCPs with even only a single hard constraint can be infeasible. In fact, the GP-CBF-SOCP becomes infeasible when the prediction uncertainty eclipses the discovery of a control input that can guarantee the system's safety, as follows*

*from* (10.8b). *This is in contrast to the uncertainty-free case, where the CBF-QP* (10.1) *is guaranteed to always be feasible by the definition of CBF. It is therefore essential to study under which conditions the GP-CBF-SOCP becomes infeasible, as safety could be compromised in those cases.*

### 10.5.1 Necessary Condition for Pointwise Feasibility

The first feasibility result we present is a necessary condition for pointwise feasibility of the GP-CBF-SOCP.

**Lemma 10.2** (Necessary condition for pointwise feasibility of the GP-CBF-SOCP)**.** *If for a given dataset $\mathbb{D}_N$, the GP-CBF-SOCP* (10.8) *is feasible at a point $x \in \mathcal{X}$, then it must hold that*

$$\begin{bmatrix} \widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)) \\ \widehat{L_g B}(x|\mathbb{D}_N)^T \end{bmatrix}^T \Sigma_B(x|\mathbb{D}_N)^{-1} \begin{bmatrix} \widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)) \\ \widehat{L_g B}(x|\mathbb{D}_N)^T \end{bmatrix} \geq \beta^2. \tag{10.17}$$

*Proof.* See Appendix A.4. □

To provide insight into this condition, for a given dataset $\mathbb{D}_N$ and at a particular point $x \in \mathcal{X}$, note that the left-hand side of (10.17) encodes a trade-off between the uncertainty matrix $\Sigma_B(x|\mathbb{D}_N)$ and the mean prediction of the terms of the CBF constraint (as in the vector $[\widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)),\ \widehat{L_g B}(x|\mathbb{D}_N)]$). In fact, the left-hand side of (10.17) can be expressed as a sum of products, including the control-independent components (the mean prediction $\widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x))$ and the upper left block of $\Sigma_B(x|\mathbb{D}_N)$), and the control-dependent components (the mean prediction $\widehat{L_g B}(x|\mathbb{D}_N)$ and the lower right block of $\Sigma_B(x|\mathbb{D}_N)$).

The term $\widehat{L_g B}(x|\mathbb{D}_N)$ reflects the mean prediction of how a control input $u$ can influence the change of the value of the CBF $B(x)$. Speaking informally, the dynamics of the CBF $B(x)$ are *controllable* at a particular point $x$ when $L_g B(x)$ is a non-zero vector, and $\widehat{L_g B}(x|\mathbb{D}_N)$ is our mean prediction of $L_g B(x)$. In this case, the control-dependent components of (10.17) reveal that the necessary condition for pointwise feasibility is more easily satisfied if the value of $\widehat{L_g B}(x|\mathbb{D}_N)$ is dominant over the lower-right block of $\Sigma_B(x|\mathbb{D}_N)$ (which represents the growth of the prediction uncertainty with respect to $u$). In fact, this tradeoff between $\widehat{L_g B}(x|\mathbb{D}_N)$ and the lower-right block of $\Sigma_B(x|\mathbb{D}_N)$ constitutes by itself a sufficient condition for pointwise feasibility, as will be explained next.

### 10.5.2 Sufficient Condition for Pointwise Feasibility

Connecting the previous discussion with the term $\Sigma^{1/2}_{L_g B}$ from (10.15) note that the lower-right block of the uncertainty matrix $\Sigma_B(x|\mathbb{D}_N)$, can be expressed as

$$\Sigma_{L_g B}(x|\mathbb{D}_N) := \Sigma_B(x|\mathbb{D}_N)_{[2:(m+1)],[2:(m+1)]} = \Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N)^T\ \Sigma^{1/2}_{L_g B}(x|\mathbb{D}_N) \in \mathbb{R}^{m\times m}.$$

We now state the sufficient condition for pointwise feasibility of the GP-CBF-SOCP, which will be the foundation for the algorithm we present in the next section. As explained earlier, at a particular point $x \in \mathcal{X}$, this condition encodes a tradeoff between two terms: the mean prediction, $\widehat{L_g B}(x|\mathbb{D}_N)$, of the true plant's safest control direction $L_g B(x)$, and the matrix $\Sigma_{L_g B}(x|\mathbb{D}_N)$ which informs about the uncertainty growth in each control input direction. This tradeoff is embodied in a symmetric matrix $\mathcal{F}(x|\mathbb{D}_N) \in \mathbb{R}^{m\times m}$, which we call the *feasibility tradeoff matrix*:

$$\mathcal{F}(x|\mathbb{D}_N) \coloneqq \beta^2 \Sigma_{L_g B}(x|\mathbb{D}_N) - \widehat{L_g B}(x|\mathbb{D}_N)^T \widehat{L_g B}(x|\mathbb{D}_N). \tag{10.18}$$

**Lemma 10.3** (Sufficient condition for pointwise feasibility of the GP-CBF-SOCP)**.** *Given a dataset $\mathbb{D}_N$, for a point $x \in \mathcal{X}$ let $\lambda_\dagger(x|\mathbb{D}_N)$ be the minimum eigenvalue of the feasibility tradeoff matrix $\mathcal{F}(x|\mathbb{D}_N)$ defined in* (10.18)*, and $e_\dagger(x|\mathbb{D}_N)$ be its associated unit eigenvector. If $\lambda_\dagger(x|\mathbb{D}_N) < 0$, the GP-CBF-SOCP* (10.8) *is feasible at $x$, and there exists a constant $\alpha_{min} \geq 0$ such that for any $\alpha > \alpha_{min}$,*

$$\pi_{safe}(x) = \alpha \ sgn\big(\widehat{L_g B}(x|\mathbb{D}_N) e_\dagger(x|\mathbb{D}_N)\big) \ e_\dagger(x|\mathbb{D}_N) \tag{10.19}$$

*is a feasible solution of* (10.8) *at $x$.*

*Proof.* See Appendix A.4. □

Intuitively, Lemma 10.3 states that if at the current state $x$, there exists a control direction along which the *controllability* of the CBF is dominant over the rate of growth of the prediction uncertainty, then the problem is feasible. Furthermore, it provides an expression for such control input direction $\pi_{\text{safe}}(x)$ (10.19) in closed from.

Note that, with this condition, a single scalar value ($\lambda_\dagger$) being negative guarantees the feasibility of the GP-CBF-SOCP. This can be easily checked online before solving the problem. Furthermore, for a particular state $x \in \mathcal{X}$, the value of $\lambda_\dagger(x|\mathbb{D}_N)$ can be clearly associated with a notion of *richness* of the dataset $\mathbb{D}_N$ for safety purposes—if it is negative, then there exists at least one control input direction which we are certain that keeps the system safe with high probability. This condition serves as the foundation for the safe online learning methodology that we present in Section 10.6.

### 10.5.3 Necessary and Sufficient Condition for Pointwise Feasibility

Lastly, we state the necessary and sufficient condition for pointwise feasibility of the GP-CBF-SOCP. This condition combines and generalizes Lemmas 10.2 and 10.3.

**Theorem 10.2** (Necessary and sufficient condition for pointwise feasibility of the GP-CBF–SOCP)**.** *Given a dataset $\mathbb{D}_N$, for a point $x \in \mathcal{X}$ let $\lambda_\dagger(x|\mathbb{D}_N)$ be the minimum eigenvalue of the feasibility tradeoff matrix $\mathcal{F}(x|\mathbb{D}_N)$ defined in* (10.18)*. Then, the GP-CBF-SOCP* (10.8)

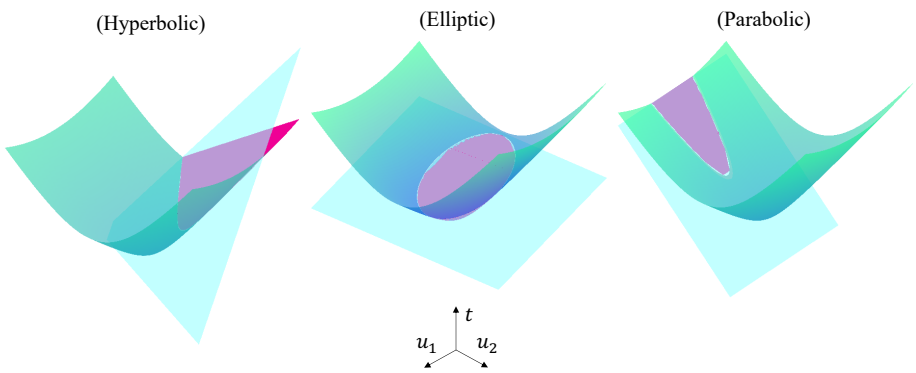



Figure 10.1: Visualization of the feasibility conditions of Theorem 10.2. The green surface is the hyperboloid $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$, the blue hyperplane is $\widehat{L_gB}(x|\mathbb{D}_N)u + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = t$, and the pink region indicates the feasible set. Case 1 corresponds to a hyperbolic intersection, Case 2 to an elliptical intersection and Case 3 to a parabolic intersection. Note that for Cases 2 and 3, if (10.20) and (10.21) are not satisfied, respectively, the feasible set is empty.

*is feasible at $x$ if and only if condition* (10.17) *is satisfied and one of the following cases holds:*
***1:*** $\lambda_\dagger(x|\mathbb{D}_N) < 0$*;*
***2:*** $\lambda_\dagger(x|\mathbb{D}_N) > 0$*, and*

$$\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) - \widehat{L_gB}(x|\mathbb{D}_N)\mathcal{F}(x|\mathbb{D}_N)^{-1}\big[\beta^2 \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) - \widehat{L_gB}(x|\mathbb{D}_N)^T \big(\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x))\big)\big] \geq 0; \quad (10.20)$$

***3:*** $\lambda_\dagger(x|\mathbb{D}_N) = 0$*, and*

$$\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) - \widehat{L_gB}(x|\mathbb{D}_N)\Sigma_{L_gB}(x|\mathbb{D}_N)^{-1}\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) > 0. \quad (10.21)$$

Case 1 matches the sufficient condition of Lemma 10.3, and it corresponds to the feasible set being hyperbolic. Cases 2) and 3) correspond to elliptic and parabolic feasible sets, respectively. This geometric interpretation of the feasibility analysis is further explained in Figure 10.1.

*Proof.* See Appendix A.4. □

Under our hypotheses, Theorem 10.2 provides tight conditions that the available data $\mathbb{D}_N$ should satisfy in order to obtain probabilistic safety guarantees for systems with actuation uncertainty.

## 10.6 Probabilistic Safe Online Learning

### 10.6.1 Proposed Safe Online Learning Strategy

In this section, we present a safe online learning algorithm that guarantees safety of the true plant (2.2) with high probability even when no prior data is available, using only the nominal dynamics model of (2.25) and the online stream of data collected by the system as its state trajectory evolves with time, constructing a dataset $\mathbb{D}_N$ online. Our proposed safe learning strategy is designed with the goal of ensuring the recursive feasibility of the GP-CBF-SOCP. This will be accomplished by guaranteeing that the sufficient condition for pointwise feasibility of Lemma 10.3 always holds. By doing so, we ensure that there always exists a backup control direction $\pi_{\text{safe}}$ (10.19) that can guarantee safety with a high probability.

**Remark 10.2.** *Note that Lemma 10.3 is only a sufficient condition for feasibility of the SOCP* (10.8)*, and that the problem could be feasible at $x \in \mathcal{X}$ even when the condition $\lambda_\dagger(x|\mathbb{D}_N) < 0$ of Lemma 10.3 does not hold. The necessary and sufficient feasibility condition is given in Theorem 10.2. However, if $\lambda_\dagger(x|\mathbb{D}_N) < 0$ does not hold, it means that there does not exist any control input direction at the current state that can serve as a backup safety direction, and the problem* (10.8) *is only feasible at $x$ in this case if the CBF condition can be guaranteed with $u \to 0$. We believe that this situation is not desirable since the system might later on move towards states where the true CBF constraint cannot be satisfied unless a control input is applied, in which case the problem would become infeasible.*

Note that the matrix $\Sigma_{L_gB}(x|\mathbb{D}_N)$ appearing in Lemma 10.3 characterizes the growth of the uncertainty $\sigma_B^2$ in each control direction. If in the neighborhood of a state $x \in \mathcal{X}$, all of the data points $(x_j, u_j)$ in the dataset $\mathbb{D}_N$ have control inputs $u_j$ coming from a performance-driven control law like $\pi_{\text{ref}}$, then the uncertainty growth in the unknown safe control direction $L_gB(x)$ can be potentially high, for instance if it is significantly different from the performance control direction, because of the resulting structure of $\Sigma_{L_gB}(x|\mathbb{D}_N)$. In this case, the condition $\lambda_\dagger(x|\mathbb{D}_N) < 0$ of Lemma 10.3 may not be satisfied and infeasibility would occur if the probabilistic CBF constraint of (10.8b) cannot be met when $u \to 0$ (see Remark 10.2). This situation could happen in our case if the system is directly controlled by the GP-CBF-SOCP (10.8) in regions where the CBF constraint is not active, since in that case the collected data points would have control inputs along the direction of the reference control policy $\pi_{\text{ref}}$. The problem is that later on, if the system approaches the safe set boundary and the CBF constraint becomes active (meaning that a safety control action is needed), $\lambda_\dagger(x|\mathbb{D}_N) < 0$ may not hold and the SOCP controller may become infeasible, as it would not be able to find any control input direction along which the controllability of the CBF is dominant over the growth of the uncertainty. As will be explained in the following, the crux of our safe learning algorithm is to make sure we never end up in this situation. We accomplish this by applying control inputs (and adding those points to the dataset) in

**Algorithm 10.1:** Safe Online Learning

Initialize $t = 0$, $x(0) = x_0$. Get $N(0)$, $\mathbb{D}_{N(0)}$.
**while** $t < T_{\max}$ **do**
  $x \leftarrow x(t)$
  $\lambda_\dagger \leftarrow \text{getLambdaDagger}\big(x, \mathbb{D}_{N(t)}\big)$
  **if** $\lambda_\dagger < -\varepsilon$ **then**
    $u \leftarrow \pi_{\text{GP-CBF}}(x)$ from the SOCP (10.8)
  **else**
    $u \leftarrow \pi_{\text{safe}}(x)$ from (10.19)
  **end**
  **if** $(\lambda_\dagger \geq -\varepsilon)$ **or** $(t \bmod \tau = 0)$ **then**
    Measure $z_B = \Delta_B(x, u) + \epsilon_{N(t)}$
    $\mathbb{D}_{N(t)} \leftarrow \mathbb{D}_{N(t)} \cup \{(x, u), z_B\}$
    $N(t) \leftarrow N(t) + 1$
  **end**
**end**

the safety backup direction in a *precautious* event-triggered fashion before the uncertainty growth in that direction becomes dominant.

Algorithm 10.1 shows a concrete implementation of our safe online learning framework. We propose using the GP-CBF-SOCP of (10.8) as the control law for system (2.2) whenever the value of $\lambda_\dagger$ lies under a threshold $-\varepsilon < 0$, which is a negative constant close to 0. However, if the value of $\lambda_\dagger$ reaches $-\varepsilon$, we propose taking a control input along $\pi_{\text{safe}}$ and adding the resulting measurement to the GP dataset, in order to reduce the uncertainty along the direction of $\pi_{\text{safe}}$ and consequently decrease the value of $\lambda_\dagger$ to below $-\varepsilon$ for the following time steps. Nonetheless, since apart from guaranteeing safety we also want the reference controller $\pi_{\text{ref}}$ to accomplish its objective without being too conservative, in addition to the event-triggered updates when $\lambda_\dagger$ reaches $-\varepsilon$ we propose collecting time-triggered measurements to update the dataset, with triggering period $\tau$. Thus, Algorithm 10.1 constructs a time-varying dataset $\mathbb{D}_{N(t)}$ and implicitly defines a closed-loop control law.

**Remark 10.3.** *Using Algorithm 10.1, the number of data points $N$ grows with time. Since each data point is added individually, rank-one updates to the kernel matrix inverse can be computed in $O(N^2)$. However, $N$ can still quickly become large-enough to compromise real-time computation. A practical and effective strategy to only select the most useful data points and therefore reduce this computational complexity is presented in Chapter 11 of this dissertation.*

### 10.6.2 Theoretical Analysis

In this section, we provide theoretical results about the effectiveness of Algorithm 10.1 in guaranteeing safety of the unknown system (2.2) with respect to the safe set $\mathcal{X}_{\text{safe}}$. We start by showing that with Algorithm 10.1 we can keep $\lambda_\dagger < 0$ for the full trajectory under some assumptions.

**Assumption 10.2.** *We assume that we have an initial dataset $\mathbb{D}_{N(0)}$ (which can be an empty set) such that at the initial state $x_0 \in \mathcal{X}$ and initial time $t = 0$, we have $\lambda_\dagger(x_0|\mathbb{D}_{N(0)}) < 0$.*

**Assumption 10.3.** *We assume that the CBF $B$ satisfies the relative degree one condition in $\mathcal{X}$, i.e., $L_g B(x) \neq 0 \ \forall x \in \mathcal{X}$. Furthermore, we assume that for any $x_0 \in \mathcal{X}$, for the trajectory $x(t)$ generated by running Algorithm 10.1, with $\mathbb{D}_{N(t)}$ being the dataset at time $t$, we have $\widehat{L_g B}(x(t)|\mathbb{D}_{N(t)}) \neq 0 \ \forall t$.*

**Assumption 10.4.** *Running Algorithm 10.1 from any $x_0 \in \mathcal{X}$, let $\{t_\kappa\}_{\kappa\in\mathbb{N}}$ be the sequence of times at which $\lambda_\dagger(x(t)|\mathbb{D}_{N(t)}) \geq -\varepsilon$. We assume that for every $\kappa$ we have $e_\dagger\big(x(t_\kappa)|\mathbb{D}_{N(t_\kappa)}\big)^T \widehat{L_g B}\big(x(t_\kappa)|\mathbb{D}_{N(t_\kappa)+1}\big) \neq 0$.*

Assumption 10.2 requires that at the initial state we have a backup safety direction $\pi_{\text{safe}}$ available. This can be achieved through a good nominal model (2.25) or an initial small set of points $\mathbb{D}_{N(0)}$ in the neighborhood of $x_0$. In the first part of Assumption 10.3, we require a relative degree 1 of the CBF $B$. This was already required to guarantee Lipschitz continuity of the solutions of the original CBF-QP (10.1), as explained in Section 10.2. The second part of Assumption 10.3 is needed to make sure that we do not lose the relative degree of the mean predicition of the CBF condition, and Assumption 10.4 makes sure that during an event-triggered update of the dataset, the new safety direction does not completely cancel the previous one. Both of these assumptions are in accordance with the high probability statement of Lemma 10.1.

**Lemma 10.4.** *Under Assumptions 10.2, 10.3 and 10.4, for all $x_0 \in \mathcal{X}$, let $x(t)$ be the trajectory generated by running Algorithm 10.1 for system (2.2). Let $\mathbb{D}_{N(t)}$ be the time-varying dataset generated during the execution of Algorithm 10.1. If the trajectory $x(t)$ exists and is unique during some time interval $t \in [0, \tau_{max})$, then it holds that $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)}\big) < 0$ for all $t \in [0, \tau_{max})$.*

*Proof.* See Appendix A.4. □

Lemma 10.3 previously demonstrated that $\lambda_\dagger(x|\mathbb{D}_N) < 0$ is a sufficient condition for pointwise feasibility of the GP-CBF-SOCP at a point $x \in \mathcal{X}$ using a dataset $\mathbb{D}_N$. Now, Lemma 10.4 ensures that $\lambda_\dagger(x(t)|\mathbb{D}_{N(t)}) < 0$ always holds along each trajectory $x(t)$ and dataset $\mathbb{D}_{N(t)}$ obtained by running the safe learning algorithm. Therefore, we have established recursive feasibility of the GP-CBF-SOCP when using the proposed safe learning strategy, as formalized in the following statement.

**Theorem 10.3** (Recursive feasibility of the GP-CBF-SOCP)**.** *Under Assumptions 10.2, 10.3 and 10.4, for all $x_0 \in \mathcal{X}$ let $x(t)$ be the trajectory generated by running Algorithm 10.1 for system* (2.2)*. Let $\mathbb{D}_{N(t)}$ be the time-varying dataset generated during the execution of Algorithm 10.1. If the trajectory $x(t)$ exists and is unique during some time interval $t \in [0, \tau_{max})$, then the probabilistic safety constraint* (10.8b) *is feasible at all times $t \in [0, \tau_{max})$ for the trajectory $x(t)$ and dataset $\mathbb{D}_{N(t)}$.*

*Proof.* This is a direct consequence of Lemmas 10.3 and 10.4. □

Next, we prove that the trajectory $x(t)$ generated by running Algorithm 10.1 locally exists and is unique. Note that the policy that Algorithm 10.1 defines is a switched control law, since new data points are added at discrete time instances. We start by showing that for a fixed dataset $\mathbb{D}_N$, the solution of the GP-CBF-SOCP is locally Lipschitz continuous under some assumptions:

**Assumption 10.5.** *We assume that the Lie derivatives of $B$ computed using the nominal model $L_{\tilde{f}}B(x)$, $L_{\tilde{g}}B(x)$, as well as the function $\gamma(B(x))$, the reference policy $\pi_{ref}(x)$ and the GP prediction functions for any fixed dataset $\mu_B(x,u)$, $\sigma_B(x,u)$ are twice continuously differentiable in $x$, for all $x \in \mathcal{X}$.*

Note that the GP prediction functions are twice continuously differentiable in $x$ when the components $k_1, \ldots, k_{m+1}$ of the ADP compound kernel (9.5) use many typical kernels, for instance the squared exponential kernel.

**Lemma 10.5** (Lipschitz continuity of solutions of the GP-CBF-SOCP)**.** *Under Assumption 10.5, for a point $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$ such that $\lambda_\dagger(x|\mathbb{D}_N) < 0$ holds, the solution of the GP-CBF-SOCP* (10.8) *is locally Lipschitz continuous around $x$.*

*Proof.* See Appendix A.4. □

**Remark 10.4.** *To the best of our knowledge, Lemma 10.5 is the first result concerning Lipschitz continuity of SOCP-based controllers using CBFs or, equivalently, Control Lyapunov Functions (CLFs) for a general control input dimension. Very recently, several SOCP-based frameworks have been developed for robust data-driven safety-critical control using CBFs and CLFs [56, 188, 55, 25, 68], and verifying the local Lipschitz continuity of the SOCP solution serves to guarantee local existence and uniqueness of trajectories of the closed-loop dynamics.*

Using Lemmas 10.4 and 10.5, we now establish local existence and uniqueness of the closed-loop solutions of system (2.2) under the switched control law defined by Algorithm 10.1.

**Theorem 10.4** (Local existence and uniqueness of executions of the safe learning algorithm)**.** *Under Assumptions 10.2, 10.3, 10.4 and 10.5, there exists a $\tau_{max} > 0$ such that for any $x_0 \in \mathcal{X}$ a unique solution $x(t)$ of* (2.2) *under the control law defined by Algorithm 10.1 exists for all $t \in [0, \tau_{max})$.*

*Proof.* See Appendix A.4. □

Previously, Theorem 10.3 gave conditions under which the probabilistic constraint (10.8b) is recursively feasible when using the control law defined by Algorithm 10.1. This means that, with high probability, the true CBF constraint (10.1b) can be satisfied at every timestep, as follows from Lemma 10.1. This fact can now be combined with the local existence and uniqueness result of Theorem 10.4 to establish forward-invariance of a safe-set $\mathcal{X}_{\text{safe}}$ with high probability, as was originally formulated in Problem 10.1.

**Theorem 10.5.** *Under Assumptions 10.1, 10.2, 10.3, 10.4 and 10.5, the control law defined by Algorithm 10.1 applied to the true plant* (2.2) *renders the set* $\mathcal{X}_{safe} = \{x \in \mathcal{X} : B(x) \geq 0\}$ *forward invariant with a probability of at least* $1 - \delta$.

*Proof.* Let the control law defined by Algorithm 10.1 be denoted as $\bar{\pi}(x)$. For all $x_0 \in \mathcal{X}$, the solution $x(t)$ of (2.2) under $\bar{\pi}(x)$ satisfies $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)}\big) < 0$, $\forall t \in [0, \tau_{max})$ with $\tau_{max} > 0$ from Theorem 10.4 and Lemma 10.4. Here, $\mathbb{D}_{N(t)}$ is the time-varying dataset generated by Algorithm 10.1. Moreover, from Lemma 10.3 and Theorem 10.3, this means that the GP-CBF-SOCP is feasible $\forall t \in [0, \tau_{max})$. Furthermore, note that $\bar{\pi}$ is the solution of (10.8), except at times when $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)}\big) \geq -\varepsilon$ in which case it takes the value of $\pi_{\text{safe}}(x(t))$. However, since even at those times $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)}\big) < 0$, $\pi_{\text{safe}}(x(t))$ is also a feasible solution of (10.8). Therefore, under $\bar{\pi}$, the constraint (10.8b) is satisfied for all $t \in [0, \tau_{max})$. This fact, together with the probabilistic bound on the true plant CBF derivative $\dot{B}$ (10.7) that arises from Lemma 10.1, leads to:

$$\mathbb{P}\big\{\ \dot{B}\big(x(t), \bar{\pi}(x(t))\big) + \gamma\big(B(x(t))\big) \geq 0, \forall x_0 \in \mathcal{X},\ \forall t \in [0, \tau_{max})\ \big\} \geq 1 - \delta. \tag{10.22}$$

Noting that the trajectory $x(t)$ is a continuous function of time that exists and is unique for all $t \in [0, \tau_{max})$ (from Theorem 10.4), we can now use Assumption 10.1 and the bound of (10.22) to obtain

$$\mathbb{P}\{\ \forall x_0 \in C,\ x(0) = x_0 \implies x(t) \in C, \forall t \in [0, \tau_{max})\ \} \geq 1 - \delta. \tag{10.23}$$

This is precisely the expression that appears in Problem 10.1, and it means that the trajectories $x(t)$ will not leave the set $\mathcal{X}_{\text{safe}} = \{x \in \mathcal{X} : B(x) \geq 0\}$ for all $x_0 \in \mathcal{X}_{\text{safe}}$ with a probability of at least $1 - \delta$, completing the proof. □

Theorem 10.5 establishes the forward invariance of $\mathcal{X}_{\text{safe}}$ with a probability of at least $1 - \delta$. This is possible because of the fact that Lemma 10.1 is not a pointwise result on the deviation of the GP prediction at a particular point, but instead a probability bound on the combination of all of the possible deviations (for all $N$, $x$ and $u$). The note [120] provides an insightful discussion of this topic.

**Remark 10.5.** *Note that the proposed framework can be easily extended to the problem of safe stabilization by adding a relaxed probabilistic CLF constraint to the SOCP* (10.8)*, as done in [30]. The entire theoretical analysis about feasibility and safety would directly follow as long as Assumption 10.5 is adapted to include the CLF-related terms.*

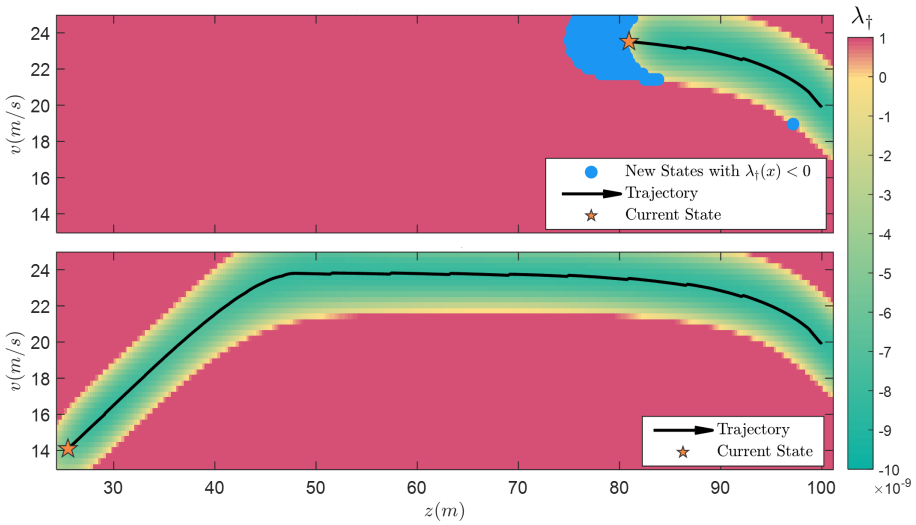



Figure 10.2: Color map of $\lambda_\dagger$ in the state-space of the adaptive cruise control system $x = [v, z]^T$ when running Algorithm 10.1 with no prior data. The region in which $\lambda_\dagger < 0$ is expanded online as Algorithm 10.1 collects new measurements. Top: snapshot when $\lambda_\dagger$ hits the threshold $-\varepsilon$, Algorithm 10.1 collects a measurement along $\pi_{\text{safe}}$ which expands the region where $\lambda_\dagger < 0$ (in blue). Bottom: result at the end of the trajectory.

## 10.7 Examples

In this section, we test our framework on the following two examples in numerical simulation. The first example of an adaptive cruise control system highlights how the feasibility of the controller improves from the data collected online through Algorithm 10.1. The second example of a kinematic vehicle system demonstrates the applicability of our framework to multi-input systems.

### 10.7.1 Adaptive Cruise Control

We apply our proposed framework to a numerical model of an adaptive cruise control system

$$\dot{x} = f(x) + g(x)u, \quad f(x) = \begin{bmatrix} -F_r(v)/m \\ v_0 - v \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}, \tag{10.24}$$

where $x = [v, z]^T \in \mathbb{R}^2$ is the system state, with $v$ being the ego car's velocity and $z$ the distance between the ego car and the car in front of it; $u \in \mathbb{R}$ is the ego car's wheel force; $v_0$ is the constant velocity of the front car (14 m/s); $m$ is the mass of the ego car; and $F_r(v) = f_0 + f_1 v + f_2 v^2$ is the rolling resistance force on the ego car. We introduce uncertainty in the mass and the rolling resistance.

A CLF is designed with the objective of stabilizing a desired speed of $v_d = 24$ m/s, and a CBF enforces a safe distance of $z \geq 1.8v$ with respect to the front vehicle. We specifically

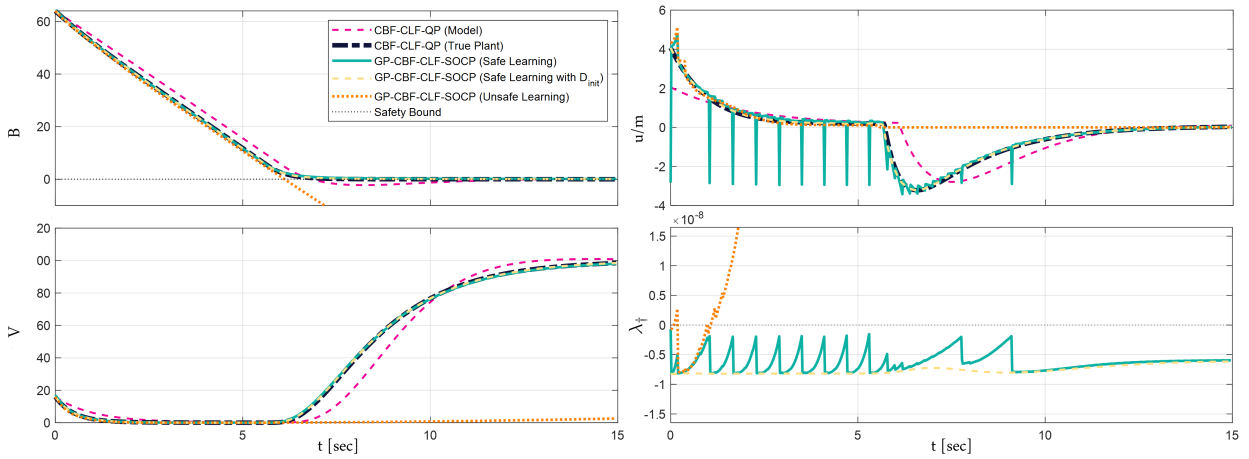



Figure 10.3: Simulation results of an adaptive cruise control system under model uncertainty, when controlled using different strategies: Algorithm 10.1 with no prior data (green); Algorithm 10.1 with an informative prior dataset (yellow); the GP-CBF-SOCP with no prior data using time-triggered updates online (orange); the CBF-QP using the uncertain dynamics (pink); and the oracle true-plant-based CBF-QP (black). Even when no prior data is available, Algorithm 10.1 keeps the system safe ($B > 0$) by collecting measurements in the safety direction (negative $u$) when $\lambda_\dagger$ approaches 0. Using the GP-CBF-SOCP with time-triggered data collection or the uncertain CBF-QP the system becomes unsafe, as shown in the $B$ plot.

use $V(x) = (v - v_d)^2$ and $B(x) = z - 1.8v$. Following Remark 10.5, the CLF-based stability constraint is added as a soft constraint to the SOCP controller, replacing the reference control input $\pi_{\text{ref}}$. Therefore, the CBF acts as a hard safety constraint which filters a performance-driven control policy based on the CLF (whose objective is to stabilize the car to the desired speed $v_d$).

Figure 10.2 shows a state-space color map of the value of $\lambda_\dagger$ at two different stages of the trajectory generated running Algorithm 10.1 for the adaptive cruise control system with no prior data from $x_0 = [20, 100]^T$. The top plot represents an intermediate state, in which the system is still trying to reach the desired speed of 24 m/s since the safety constraint (10.8b) is not active yet (the car in front is still far). Even though Algorithm 10.1 is collecting measurements in a time-triggered fashion using the SOCP (10.8) controller, the state gets close to the boundary of $\lambda_\dagger = 0$ frequently since the performance-driven control input obtained from the SOCP (10.8) when the safety constraint is not active is very different from $\pi_{\text{safe}}$. One such case is visualized in the top figure. However, Algorithm 10.1 detects that $\lambda_\dagger$ is getting close to zero and an event-triggered measurement in the direction of $\pi_{\text{safe}}$ is taken, which expands the region where $\lambda_\dagger < 0$. The bottom plot shows the color map of $\lambda_\dagger$ at the end of the process, with the final dataset. We can see that the safety constraint was active for a portion of the trajectory (when the ego vehicle approached the front one), but

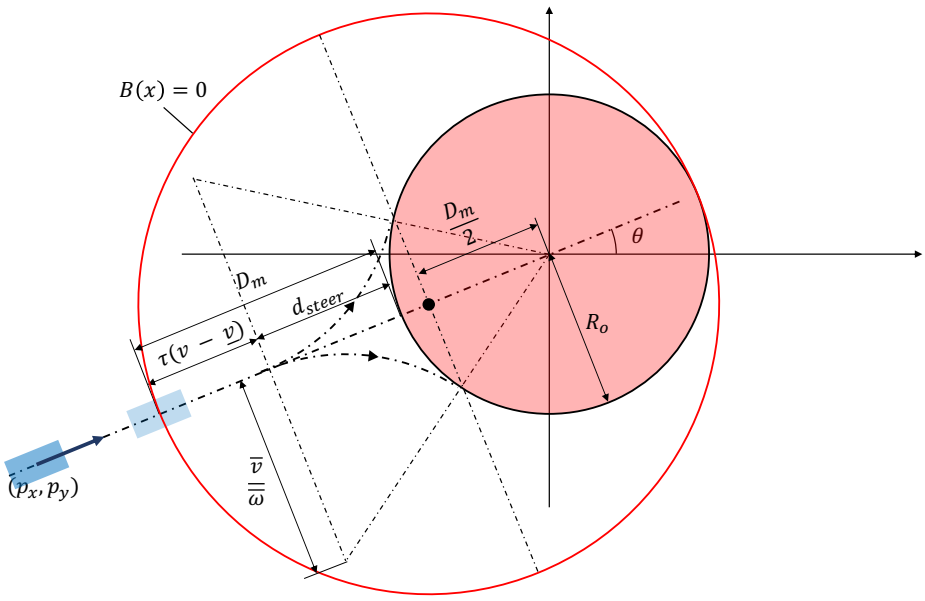



Figure 10.4: Illustration of the zero-level set of the CBF for the kinematic vehicle example. $D_m$ is the safety distance, which is computed by adding the minimum distance for the vehicle to steer with a maximal yaw rate without colliding with the obstacle $d_{steer}$ and a velocity-dependent distance margin $\tau(v - \underline{v})$.

the system stayed safe by virtue of using Algorithm 10.1 to keep a direction $\pi_{\text{safe}}$ available.

Figure 10.3 shows that while a nominal CBF-QP (in pink) fails to keep the system safe under model uncertainty, Algorithm 10.1 with no prior data (in green) always manages to keep $B > 0$ and $\lambda_\dagger < 0$. The same algorithm without the measurements along $\pi_{\text{safe}}$, triggered by the event $\lambda_\dagger \geq -\varepsilon$, fails (orange), since when the safety constraint (10.8b) becomes active, $\lambda_\dagger$ soon gets positive and the SOCP (10.8) becomes infeasible.

From another perspective, Figure 10.3 shows the importance of having a good nominal model or a prior database that properly characterizes a safe control direction. As shown in yellow, with such prior information Algorithm 10.1 keeps the system safe without having to take any measurements along $\pi_{\text{safe}}$. If no prior data is given, the control law is purely learned online, which leads to $\lambda_\dagger$ getting close to zero several times in the trajectory, and steps in the direction of $\pi_{\text{safe}}$ (negative $u$) are needed in order to prevent $\lambda_\dagger$ from actually reaching zero. This clearly damages the desired performance, as the car would be braking from time to time, nevertheless, this is required in order to be certain about how the system reacts to pressing the brake. Therefore, the proposed event-triggered design allows Algorithm 10.1 to automatically reason about whether the available information is enough to preserve safety or the collection of a new data point along $\pi_{\text{safe}}$ is required instead. Note that our algorithm is also useful for cases in which a large dataset is available a priori since safety is secured even when the system is brought to out-of-distribution regions by collecting new measurements.

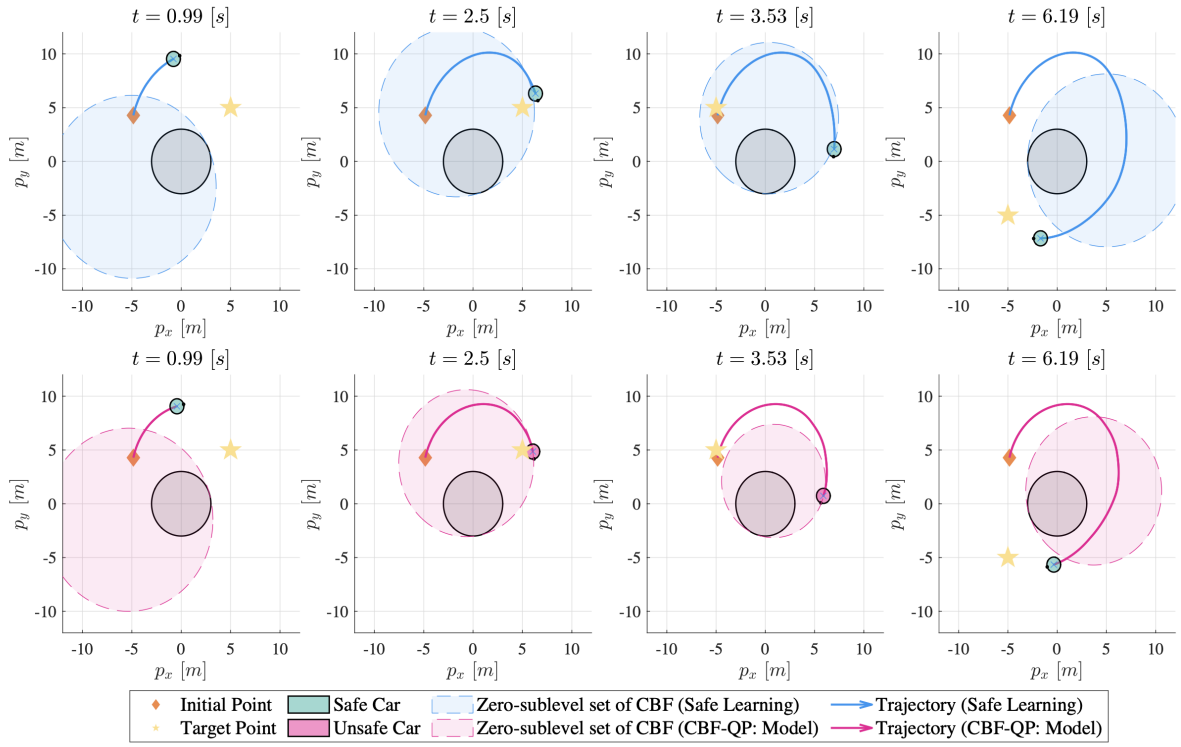



Figure 10.5: Snapshots that show the evolution over time of a 4-dimensional kinematic vehicle system under model uncertainty, when controlled using different methods: Algorithm 10.1 with no prior data (top row, blue); the CBF-QP based on the nominal model (bottom row, pink). Starting at the initial state $x_0$ (orange diamond), the vehicle pursues the target (yellow star), while trying to avoid a collision with the obstacle (grey circle). Each curved line indicates the trajectory of the vehicle's position up until the time when each snapshot is taken (the position at the snapshot time is represented by a green or red circle). Note that the circle is colored red when the vehicle violates the safety constraint (i.e., $B(x) < 0$). The large color-filled circle with a dashed border represents the set of unsafe states (zero-sublevel set of the CBF). To watch the full video of the vehicle running under each control algorithm, please visit this link.

### 10.7.2 Kinematic Vehicle

Next, in order to gauge our framework's applicability to systems with higher state dimensions and multiple control inputs, we apply our method to a four-dimensional kinematic vehicle system.

In this system, the state vector is denoted as $x = [p_x, p_y, \theta, v]^T \in \mathbb{R}^4$ which consists of the vehicle's position $(p_x, p_y)$, heading angle $\theta$, and longitudinal velocity $v$; the control input is denoted as $u = [w, a]^T \in \mathbb{R}^2$ which includes the vehicle's yaw rate $w \in [-\bar{w}, \bar{w}]$ and the

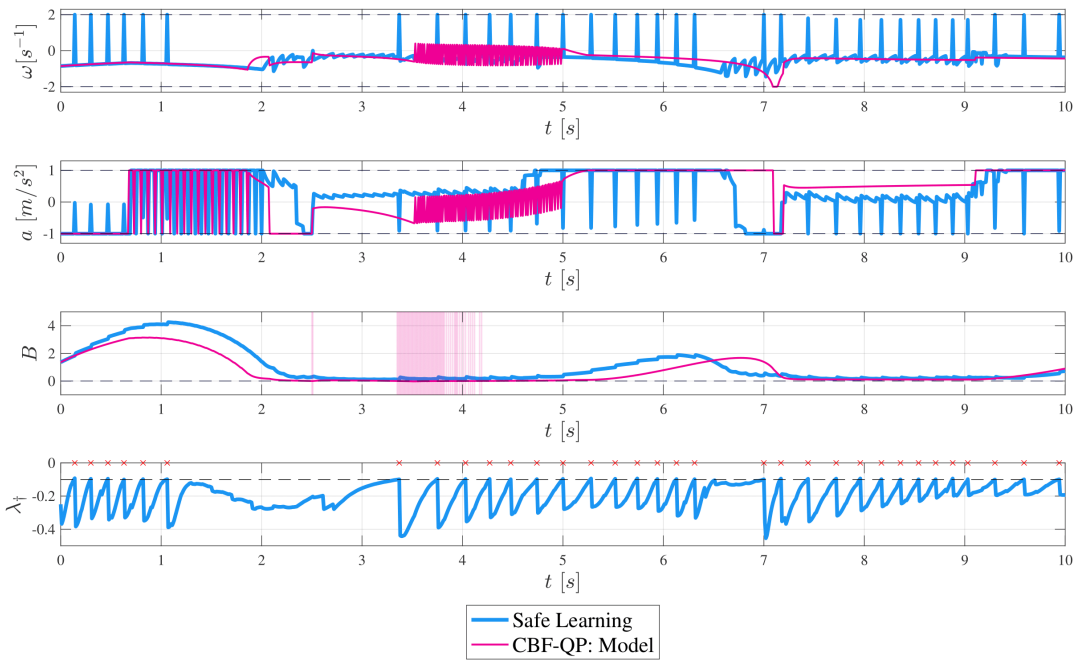



Figure 10.6: Simulation results of 4-dimensional kinematic vehicle system under model uncertainty, when using two strategies introduced in Figure 10.5 with the identical color notation. The four plots illustrate the yaw rate, the acceleration control inputs, the CBF values, and $\lambda_\dagger$ in time respectively. The dotted lines denote the input bounds, the zero-level of the CBF $B(x) = 0$; and the threshold $-\epsilon$ in Algorithm 10.1. The red bars in the third plot represent the time stamps when the nominal CBF-QP violates safety. In contrast, Algorithm 10.1 ensures $B(x) > 0$ at all times. The red cross points in the last plot indicate the time stamps when $\lambda_\dagger$ hits $-\epsilon$ and the safe exploration is executed according to Algorithm 10.1.

longitudinal acceleration $a \in [-\bar{a}, \bar{a}]$. We use the values $\bar{a} = 1$ and $\bar{w} = 2$. The dynamics of the system are modeled as

$$f(x) = \begin{bmatrix} k_v v \cos\theta \\ k_v v \sin\theta \\ 0 \\ -\mu v + s_e h(p_x, p_y) \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ k_w & 0 \\ 0 & k_a \end{bmatrix}, \tag{10.25}$$

where $k_v$, $k_w$, $k_a$ are coefficients that capture the skid, the term $\mu v$ represents the drag, and $s_e h(p_x, p_y)$ accounts for the effect of the slope of the terrain. We assume that the nominal model does not address such effects (i.e., $k_v = k_w = k_a = 1$, $\mu = s_e = 0$), while the uncertainty imposed on the true system is induced by $k_v = 2$, $k_w = 1.5$, $k_a = 1$, $\mu = 0.5$, $s_e = 0.5$, $h(p_x, p_y) = (p_x^2 + p_y^2)^{0.1}$. Note that unstructured uncertainties are imposed through

terms like $h(p_x, p_y)$, which can be arbitrary functions, unlike the previous example that only imposes parametric uncertainties.

As illustrated in Figure 10.5, the objective of the control is to reach the target points alternating in time while not colliding into a static circular obstacle of radius $R_o = 3$ centered at the origin. The reference controller $\pi_{\text{ref}}$ has two objectives: 1) it pursues a target point that alternates among a given set of points $S_T = \{(5,5),(5,-5),(-5,-5),(-5,5)\}$ every $p = 2.5$ seconds, and 2) it stabilizes the vehicle's velocity to $v_d$ while assuring that it is always bounded in $[\underline{v}, \bar{v}]$. We use the values $\underline{v} = 1$, $\bar{v} = 5$, and $v_d = 3$. All units are in the metric system.

The CBF we use is

$$B(x) = \sqrt{\left(p_x + \frac{D_m}{2}\cos\theta\right)^2 + \left(p_y + \frac{D_m}{2}\sin\theta\right)^2} - \left(R_o + \frac{D_m}{2}\right), \tag{10.26}$$

where

$$D_m = \tau(v - \underline{v}) + d_{steer}; \quad d_{steer} = R_o\sqrt{1 + \frac{2\bar{v}}{R_o\bar{w}}} - R_o. \tag{10.27}$$

This CBF adds a safety margin $D_m$ to the obstacle in the direction of the vehicle's heading angle, based on its minimum velocity and maximum steering rate as shown in Figure 10.4. We can analytically check that the zero-superlevel set of the CBF is control invariant and that the CBF constraint is always feasible under the input bounds.

Figures 10.5 and 10.6 illustrate that while the CBF-QP based on the nominal model (in pink) escapes the zero-superlevel set of the CBF, Algorithm 10.1 (in blue) without any prior data always keeps the vehicle inside. This result not only demonstrates the validity of the proposed strategy when applied to a multi-input system but also alludes to the intuition behind our strategy: when $\lambda_\dagger$ hits $-\epsilon$, the vehicle steers away from the obstacle and decelerates more in order to improve the certainty of its safe control direction.

## 10.8 Chapter Summary

In this chapter, we have introduced a Control Barrier Function-based approach for the safe control of uncertain systems. Our results show that it is possible to guarantee the invariance of a safe set for an unknown system with high probability, by combining any available approximate model knowledge with sufficient data collected from the real system. We achieve this by first introducing a safety-critical optimization-based controller that, by formulation, is probabilistically robust to the prediction uncertainty of the unknown system's dynamics. However, this optimization problem only produces a safe control action when the available information about the system (prior model knowledge and data) is sufficiently rich, as our feasibility analysis shows. As a means to fulfill this feasibility requirement, we later presented a formal method that, by collecting data online when required, is able to guarantee the recursive feasibility of the controller and therefore preserve the unknown system's safety with high probability. Algorithm 10.1 presents a simple embodiment of this idea; however,

we believe that future work should not be restricted to this particular implementation, since the most important contribution of this chapter is a principled reasoning procedure for conducting safe exploration when using data-driven control schemes.

# Chapter 11

# Online Data Selection for Scalable Probabilistic Safe Control under Uncertainty

*This chapter is based on the preprint titled "Scalable Stability and Safety Filters for Uncertain Robotic Systems through Constraint-Guided Online Data Selection", writen in collaboration with Jason J. Choi, Wonsuhk Jung, Bike Zhang, Claire J. Tomlin and Koushil Sreenath.*

In the final technical chapter of this dissertation, we showcase how Control Lyapunov Functions and Control Barrier Functions can be used to inform which data points are most instrumental to certify stability and safety of a control system, respectively. This analysis constitutes an example of how model-based certificate functions can serve to add meaning to individual data points and, thus, inform data-driven control approaches.

Note that the Gaussian Process-based safety filters introduced in the previous two chapters suffer from scalability issues, as the GP inference time scales poorly with the number of data points. This hinders their applicability to real robotic systems, which require large datasets. Building on the feasibility results presented in the previous chapter, we now introduce a scalable data selection strategy that can be run online and significantly reduces the computation time of these filters. This data selection strategy consciously selects at each state those data points that best characterize how to stay safe or stable.

## 11.1 Introduction

### 11.1.1 Motivation

Learning-based control methods have experienced a surge in popularity, enabling sophisticated control policies to be derived from large amounts of data. However, to ensure their successful implementation in real-world systems, it is crucial to verify that these policies

satisfy certain desired properties, such as stability and safety. This challenge has led to significant research effort aimed at understanding the theoretical foundations of data-driven control methods.

The prevailing method to certify the properties of learning-based control policies is *a posteriori* verification, i.e., analyzing the properties of these policies after they have been synthesized. A significant body of recent work on neural network verification follows this approach [127]. If the neural policy of interest does not pass the certification test, the designer typically modifies the learning setup and re-iterates this process until a satisfying policy is found. While promising, these approaches are in general not able to point towards the underlying cause of failure, which could be, for instance, linked to the inadequacy of the training dataset.

An alternative to a posteriori certification is the use of model-based safety filters, which impose constraints on policy outputs to ensure reliable control of the system. These filters leverage certificate functions, such as Control Barrier Functions [8], to separate the problem of safety certification from performance objectives. Although these filters are dependent on the underlying model, recent approaches have demonstrated the feasibility of designing them using data gathered from real systems [187, 38]. By combining these techniques with non-parametric learning methods, such as Gaussian Process regression, it is possible to obtain provable guarantees on the behavior of the actual system [30, 56, 188].

Nevertheless, the success of data-driven safety filters, like any learning-based control approach, is heavily reliant on the quality of the available training data [30]. Deploying these filters without sufficient data for certification may lead to failure, much like attempting to verify a posteriori policies that were trained on unsuitable datasets.

We, however, hypothesize that by decoupling verification from performance, certificate functions should be able to offer a powerful means to identify the most valuable data points for achieving certification objectives. Moreover, the non-parametric learning methods often employed by these filters can analytically capture the impact of each data point on the resulting filter. This raises the motivating question of this chapter: *can we harness these strengths to select the most appropriate data for certification purposes?*

This is especially relevant for real-world robotic systems, as large datasets are typically needed to fully characterize their high-dimensional dynamics, and the computational complexity of non-parametric learning methods scales poorly with the number of data points. Investigating which data points are most critical for certifying the desired properties of a control policy is crucial to properly address this scalability challenge.

### 11.1.2 Contributions

In this chapter, we present an approach to efficiently determine the most relevant data points for designing data-driven safety filters for real-world systems. Specifically, we provide a tractable method for identifying the data that is most instrumental in achieving robust certification, thereby enhancing the effectiveness and reliability of learning-based control policies in practice.

Utilizing the proposed approach, we showcase the applicability of Gaussian Process-based safety filters to high-dimensional and real robotic systems handling large datasets, overcoming the scalability constraints that previously limited the use of such filters to simple toy systems.

### 11.1.3 Related Work

Control Barrier Functions (CBFs, [8]) and Control Lyapunov Functions (CLFs, [11]) are model-based certificate functions that can be used to design policy filters to enforce safety and stability, respectively, of a controlled system. While initially conceived for systems with perfectly known dynamics, early results showed how to extend these filters to robust [149, 92, 147, 39] and adaptive [145, 184, 130] control settings.

Moreover, the integration of these certificate filters with data-driven methods has become increasingly popular for systems with uncertain or unknown dynamics. Several studies employ neural networks to learn the model mismatch terms [186, 187, 38]. Despite their practicality and effectiveness, verifying the accuracy of the neural network predictions can be challenging.

Alternative approaches, upon which our work builds, use non-parametric regression techniques for this purpose [27, 30, 188, 56, 24, 68]. Most notably, Gaussian Process (GP) regression models provide a probabilistic assurance of prediction quality under mild assumptions [180, 114]. This fact can be exploited to formulate robust data-driven safety filters able to keep uncertain systems safe with high probability [56, 30, 31]. However, these filters can suffer from infeasibilities when the available data does not fully capture the information needed for certification purposes [30]. Furthermore, the inference time complexity of exact GP regression scales poorly with the number of data points.

The GP research community has a rich history in developing methods to improve the computational complexity of GP inference, commonly referred to as Sparse GP regression [128, 162]. The work in [100] uses one of this methods (random features approximation) to speed up GP inference for data-driven safety filters. Additionally, existing approaches that quantify the importance of data for system identification mostly focus on optimizing information-theoretic metrics, such as the information gain, when developing exploration strategies [110, 159, 105, 5, 159, 44, 108]. However these general purpose methods lack awareness of any control objective.

Instead of aiming to obtain an approximate global GP regression model, the method we introduce in this chapter utilizes certificate functions to select a small set of data points that are useful for certification online at each state. This way, we aim to overcome the computational challenges of GP regression by exploiting the most relevant available information.

However, obtaining the best subset of data constitutes a combinatorial optimization problem that would be more computationally demanding than performing exact GP inference. For this reason, we instead present a control-informed efficient approximate data selection method that effectively serves to reduce the inference time of data-driven safety filters. This enables the deployment of these filters on real robotic systems.

The authors of [115, 116] propose a method to evaluate the importance of data for maintaining the stability of data-driven closed-loop systems. As such, they study the connection between data and the performance of a particular given policy. Additionally, they introduce a greedy data selection strategy for GP inference based on an importance measure they propose. However, these selection strategies are still too computationally expensive to be run online. Our work instead tackles the problem of robust control design, studying online which data is most relevant to achieve a desired certification property in the resulting data-driven control policy. Furthermore, our approach characterizes the relationship between data and safety in the control input space, emphasizing the richness of each data point for the specific certification objective, rather than relying on data density measures. This is a similar objective to the one of [26], where an algorithm to select the most useful data points for successfully performing multiple control tasks is presented. However, this method also suffers from scalability issues that prevent it from being applicable to real robotic systems.

## 11.2 Certifying Filters for Uncertain Systems

### 11.2.1 Uncertain Dynamics and Certifying Filter

In this chapter, we again study a control-affine system of the form in (2.2):

$$\dot{x} = f(x) + g(x)u,$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ represents the state, and $u \in \mathbb{R}^m$ denotes the control input. As explained throughout this dissertation, this form is suitable for representing various robotic systems, including those with Lagrangian dynamics. We assume that both $f$ and $g$ are locally Lipschitz continuous, and without loss of generality, we consider $f(0) = 0$ so that $x = 0$ is an equilibrium point. Throughout the chapter, we will refer to the system described in equation (2.2) as the *true plant*.

We address the challenge of ensuring critical system constraints for the true plant (2.2), such as safety and stability, when its dynamics $f$ and $g$ are unknown, while trying to accomplish a desired task. As in the previous chapter, we assume that a controller for the desired task has already been designed and is provided as a *reference controller* $\pi_{\text{ref}} : \mathcal{X} \to \mathbb{R}^m$. In the absence of the reference controller, we can consider $\pi_{\text{ref}}(x) \equiv 0$. This controller is often unaware of the system's constraints that are vital for preventing catastrophic failure, which we refer to as *system-critical constraints*. Common examples of these constraints include safety constraints, which can be expressed as constraints in the system's state space, and stability constraints that maintain the system's stability around a desired equilibrium point.

We aim to design a *certifying filter* that operates between the reference controller $\pi_{\text{ref}}$ and the true plant, ensuring the control applied to the true plant is filtered to satisfy the relevant *system-critical constraint*. When the reference controller $\pi_{\text{ref}}$ adheres to the constraint, the certifying filter simply passes $\pi_{\text{ref}}(x)$ to the true plant. However, if $\pi_{\text{ref}}$ violates the constraint, the filter overrides it with a safe control signal to prevent system failure. This filtering

structure is known by various names, most notably as a *safety filter* [199], and generalizes the CBF-QP and CLF-QP control structures introduced in Chapter 2.

As in most of this dissertation, we assume access to an approximate *nominal model* of the true plant's dynamics, represented by $\tilde{f} : \mathcal{X} \to \mathbb{R}^n$ and $\tilde{g} : \mathcal{X} \to \mathbb{R}^{n \times m}$ (2.25):

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u.$$

This nominal model typically represents the designer's best estimate of the true plant.

### 11.2.2 Certificate Function-based Control Design

In this chapter, we introduce the concept of *certificate functions* [53], which are also known by various names, such as safety indexes in [126] or energy functions in [204]. These functions generalize the notions of CBFs and CLFs that were presented in Definitions 2.7 and 2.5, respectively. Indeed, CBFs and CLFs are certificate functions ensuring the satisfaction of safety and stability constraints in a system. Using the concept of a certificate function allows us to present our results in a unified manner, independently of whether the desired system-critical constraint to be enforced is derived from a CBF or a CLF.

Informally, a certificate function is a scalar function of the state, and its value and gradient can be used to establish a sufficient condition for a control input $u$ to satisfy the desired system-critical constraint. This condition can then be employed as a *certifying constraint* in the certifying filter for the control input. If $\pi_{\text{ref}}(x)$ fails to meet the constraint, it is overridden with an appropriate control input $u$ that satisfies the constraint.

**Definition 11.1.** *A function $C : \mathcal{X} \to \mathbb{R}$ is a certificate function for the true plant (2.2) with an extended class $\mathcal{K}_\infty$ function $\gamma : \mathbb{R} \to \mathbb{R}$ (called comparison function) if*

1. *for all $x \in \mathcal{X}$, there exists a $u \in \mathbb{R}^m$ such that*

$$\dot{C}(x, u) + \gamma(C(x)) \geq 0, \tag{11.1}$$

*where $\dot{C}(x, u)$ is the Lie derivative of $C$ for the true plant (2.2), that is,*

$$\dot{C}(x, u) = \underbrace{\nabla C(x) \cdot f(x)}_{L_f C(x)} + \underbrace{\nabla C(x) \cdot g(x)}_{L_g C(x)} u, \tag{11.2}$$

2. *and if $u(t)$ satisfying (11.1) for all $t \geq 0$ is a sufficient condition for $x(t)$ satisfying the desired system-critical constraint for all $t \geq 0$.*

Both Control Barrier Functions (CBFs, Defintion 2.7) and Control Lyapunov Functions (CLFs, Definition 2.5) satisfy the aforementioned definition of certificate functions. Note that in order to align with the inequality form in (11.1), we need to negate the CLF. This adjustment ensures that both CBFs and CLFs can be used within the same framework to satisfy the desired system-critical constraints. The system-critical constraint for the CBF is

that the trajectory stays inside the zero-superlevel set of $C$ indefinitely, i.e., $x(t) \in \mathcal{X}_{\text{safe}} := \{x \in \mathcal{X} \mid C(x) \geq 0\}$ for all $t \geq 0$ [8]. The system-critical constraint for the CLF is that the trajectory is asymptotically stable to the equilibrium $x = 0$ [9].

Given a reference controller $\pi_{\text{ref}} : \mathcal{X} \to \mathbb{R}^m$, the condition in (11.1) can be used to formulate a minimally-invasive certifying filter[8]:

---

**Certificate Function-based Quadratic Program (CF-QP)**:

$$\pi_{\text{CF}}(x) = \underset{u \in \mathbb{R}^m}{\arg\min} \quad \|u - \pi_{\text{ref}}(x)\|_2^2 \tag{11.3a}$$

$$\text{s.t.} \quad L_f C(x) + L_g C(x) u + \gamma(C(x)) \geq 0. \tag{11.3b}$$

---

It is important to note that the constraint (11.3b) is affine in $u$, which means that the optimization problem is a quadratic program (QP). This problem is solved pointwise in time to obtain a filtered control law $\pi_{\text{CF}} : \mathcal{X} \to \mathbb{R}^m$ that only deviates from the reference controller $\pi_{\text{ref}}$ when the condition (11.3b) is violated. We will refer to (11.3b) as the *true certifying constraint* and (11.3) as the *oracle CF-QP*. When specifically using CBFs or CLFs in place of $C$, we may refer to (11.3) as the oracle CBF-QP and CLF-QP, respectively.

This optimization requires perfect knowledge of the system dynamics, which is not available since the Lie derivatives of $C$ appear in the constraint. Instead, we can use the nominal model and replace $L_f C(x)$ and $L_g C(x)$ with $L_{\tilde{f}} C(x)$ and $L_{\tilde{g}} C(x)$ respectively, the Lie derivatives of $C$ with respect to the nominal model. We call this a *nominal model-based CF-QP*.

The primary assumption we make in this chapter is that we have access to the certificate function $C$ that is valid for the true plant. This assumption ensures that a control policy exists to keep the true plant (2.2) in compliance with the system-critical constraint. However, even when a valid certificate function is available, obtaining such a control policy is not straightforward due to the lack of direct access to $f$ and $g$ in the true certifying constraint (11.3b). Due to the mismatch between the true plant dynamics and the nominal model, the nominal model-based CF-QP also does not provide any guarantee that the system-critical constraint will be met under the filtered control input. To examine this, the true certifying constraint in (11.3b) is expressed using the nominal model as follows:

$$\underbrace{L_{\tilde{f}} C(x) + L_{\tilde{g}} C(x) u}_{\dot{\tilde{C}}(x,u)} + \Delta_C(x, u) + \gamma(C(x)) \geq 0, \tag{11.4}$$

which reveals the *model uncertainty term* $\Delta_C$ affecting the constraint, defined for each $x \in \mathcal{X}$, $u \in \mathbb{R}^m$ as

$$\Delta_C(x, u) := (L_f C - L_{\tilde{f}} C)(x) + (L_g C - L_{\tilde{g}} C)(x) u = \begin{bmatrix} L_{\Delta f} C(x) & L_{\Delta g} C(x) \end{bmatrix} \begin{bmatrix} 1 \\ u \end{bmatrix}. \tag{11.5}$$

Note that like the original constraint (11.3b), $\Delta_C$ is also affine in the control input $u$. When the certificate function is a CLF, we use the term $\Delta_V$ (9.3), as in Chapter 9. When it is a

CBF, we use $\Delta_B$ (10.3) as in Chapter 10. It is clear now that we can use exactly the same Gaussian Process regression procedure to estimate $\Delta_C$ as was presented in Chapters 9 and 10 for $\Delta_V$ and $\Delta_B$, respectively.

**Remark 11.1.** *Discovering valid certificate functions for uncertain systems is far from trivial and is, in fact, an active area of research [54, 160, 91, 125, 95, 205]. Our contribution runs parallel to this line of research, and in fact, our work complements these efforts, as only when the design of the certificate function and the design of the certifying filter are combined, can the certifying filter for uncertain systems be effectively implemented. In our work, we employ the nominal model to find CBFs and CLFs to be used as certificate functions. Thus, this procedure implicitly assumes that the nominal model is sufficiently accurate in its approximation of the true plant to enable the identification of a valid CBF or CLF. The assumption made in this chapter is also present in prior works that most closely align with our research [187, 188, 38, 56, 68]. This approach is considered reasonable for feedback linearizable systems with known relative degree, owing to the inherent robustness properties of CBFs and CLFs [219, 107]. Indeed, the practice of using first-principle nominal models for designing CBFs is widely adopted for numerous complex robotics systems [217, 103, 139].*

### 11.2.3 Gaussian Process Regression for the Certifying Constraint Model Mismatch

In order to perform Gaussian Process regression on the model mismatch function $\Delta_C$ (11.5), we use the Affine Dot Product compound kernel introduced in Definition 9.1 to exploit its control-affine structure:

$$\mathbf{k}((x,u),(x',u')) := [1\ u^\top]\mathrm{Diag}(k_f(x,x'),k_{g_1}(x,x'),\cdots,k_{g_m}(x,x'))\begin{bmatrix}1\\u'\end{bmatrix}. \tag{11.6}$$

A dataset of noisy measurements of $\Delta_C$ is given as $\mathbb{D}_N := \{(x_j,u_j),\ \Delta_C(x_j,u_j)+\epsilon_j\}_{j=1}^N$, where $\epsilon_j \sim \mathcal{N}(0,\sigma_n^2)$ is white measurement noise, with $\sigma_n > 0$. Using the ADP compound kernel, the mean and variance of the GP posterior of $\Delta_C$ at a query point $(x_*,u_*)$ can be written as

$$\mu_C(x_*,u_*|\mathbb{D}_N) = \underbrace{\mathbf{z}^\top(K_{\mathbb{D}_N}+\sigma_n^2 I)^{-1}K_{*U}^\top}_{=:\ m_C(x_*|\mathbb{D}_N)}\begin{bmatrix}1\\u_*\end{bmatrix}, \tag{11.7}$$

$$\sigma_C^2(x_*,u_*|\mathbb{D}_N) = \begin{bmatrix}1\ \ u_*^\top\end{bmatrix}\underbrace{\left(K_{**}-K_{*U}(K_{\mathbb{D}_N}+\sigma_n^2 I)^{-1}K_{*U}^T\right)}_{=:\ \Sigma_C(x_*|\mathbb{D}_N)}\begin{bmatrix}1\\u_*\end{bmatrix}, \tag{11.8}$$

where $K_{\mathbb{D}_N} \in \mathbb{R}^{N\times N}$ is the Gram matrix of $\mathbf{k}$ for the training data inputs $(X,U)$, $K_{**} = \mathrm{Diag}\left(k_f(x_*,x_*),\cdots,k_{g_m}(x_*,x_*)\right) \in \mathbb{R}^{(m+1)\times(m+1)}$, and $K_{*U} \in \mathbb{R}^{(m+1)\times N}$ is given by

$$K_{*U} = \begin{bmatrix} k_f(x_*, x_1) & \cdots & k_f(x_*, x_N) \\ k_{g_1}(x_*, x_1) & \cdots & k_{g_1}(x_*, x_N) \\ & \vdots & \\ k_{g_m}(x_*, x_1) & \cdots & k_{g_m}(x_*, x_N) \end{bmatrix} \circ \begin{bmatrix} \mathbf{1}^{1\times N} \\ U_N \end{bmatrix},$$

where $\circ$ indicates the element-wise product, and $U_N := [u_1 \ \cdots \ u_N] \in \mathbb{R}^{m\times N}$.

One of the most significant advantages of using GP regression is that it generates predictions of the target function value in the form of a probability distribution, rather than deterministically, based on (11.7) and (11.8). This allows for the computation of a probabilistic bound on the true value of $\Delta_C(x_*, u_*)$ using $\mu_C(x_*, u_*|\mathbb{D}_N)$ and $\sigma_C(x_*, u_*|\mathbb{D}_N)$:

**Assumption 11.1.** *For a given $\delta \in (0, 1)$, there exists a constant $\beta > 0$ such that*

$$\mathbb{P}\left\{ \left| \mu_C(x_*, u_*|\mathbb{D}_N) - \Delta_C(x_*, u_*) \right| \leq \beta\sigma_C(x_*, u_*|\mathbb{D}_N) \right\} \geq 1 - \delta, \tag{11.9}$$

*for all $x_* \in \mathcal{X},\ u_* \in \mathbb{R}^m$.*

Numerous existing works, such as Lemmas 10.1 and 9.1 of the previous chapters have conducted theoretical analyses to determine the conditions under which Assumption 11.1 holds and to identify the values of $\beta$. The term $\mu_C(x_*, u_*|\mathbb{D}_N) + \beta\sigma_C(x_*, u_*|\mathbb{D}_N)$ is referred to as the GP upper confidence bound (GP-UCB). Similarly, $\mu_C(x_*, u_*|\mathbb{D}_N) - \beta\sigma_C(x_*, u_*|\mathbb{D}_N)$ is the lower confidence bound of $\Delta_C(x_*, u_*)$. Identifying $\beta$ is not within the focus of this dissertation, and we direct interested readers to the relevant literature for further details [180, 41, 60, 114, 59]. In some cases, additional or different assumptions on $\Delta_C$, $\bar{\mathcal{X}}$, and $\mathbb{D}_N$ might be required, such as in Lemmas 10.1 and 9.1.

### 11.2.4 Second-order Cone Program-based Certifying Filters

With the bound provided in (11.9), we can now present a data-driven certifying filter that offers a high probability guarantee of satisfying (11.3b) based on the learned GP model of $\Delta_C$. By employing the lower bound of $\Delta_C(x, u)$, we construct a *certifying chance constraint* that can be evaluated without explicit knowledge of the true plant's dynamics:

$$L_{\tilde{f}}C(x) + L_{\tilde{g}}C(x)u + \mu_C(x, u|\mathbb{D}_N) - \beta\sigma_C(x, u|\mathbb{D}_N) + \gamma(C(x)) \geq 0. \tag{11.10}$$

If the constraint (11.10) is satisfied, from Assumption 11.1, we have a guarantee that the true certifying constraint in (11.3b) is satisfied with a probability of at least $1 - \delta$.

Note that from the affine structure of the mean expression in (11.7), we get

$$\mu_C(x, u|\mathbb{D}_N) = m_c(x|\mathbb{D}_N) \begin{bmatrix} 1 \\ u \end{bmatrix} = \begin{bmatrix} \widehat{L_{\Delta f}C}(x) & \widehat{L_{\Delta g}C}(x) \end{bmatrix} \begin{bmatrix} 1 \\ u \end{bmatrix},$$

where

$$\widehat{L_{\Delta f}C}(x) := m_c(x|\mathbb{D}_N)_{[1]}, \quad \widehat{L_{\Delta g}C}(x) := m_c(x|\mathbb{D}_N)_{[2:(m+1)]}.$$

We define

$$\widehat{L_f C}(x|\mathbb{D}_N) := L_{\tilde{f}}C(x) + \widehat{L_{\Delta f}C}(x) \in \mathbb{R},$$
$$\widehat{L_g C}(x|\mathbb{D}_N) := L_{\tilde{g}}C(x) + \widehat{L_{\Delta g}C}(x) \in \mathbb{R}^{1\times m}.$$

Using these expressions, (11.10) can be represented as

$$\beta\sigma_C(x,u|\mathbb{D}_N) \leq \begin{bmatrix}\widehat{L_f C}(x|\mathbb{D}_N)+\gamma(C(x)) & \widehat{L_g C}(x|\mathbb{D}_N)\end{bmatrix}\begin{bmatrix}1\\ u\end{bmatrix}. \tag{11.11}$$

From the quadratic structure of the variance expression in (11.8) and $\Sigma_C(x|\mathbb{D}_N)$ being positive definite (due to the noise measurement variance being strictly positive), we can conclude that (11.11) is a second-order cone constraint.

This constraint is then incorporated into a chance-constrained reformulation of the CF-QP certifying filter:

---

**GP-CF-SOCP**:

$$\begin{aligned}\pi_{\text{GP-CF}}(x) = \quad & \underset{u\in\mathbb{R}^m}{\arg\min}\ \|u-\pi_{\text{ref}}(x)\|_2^2 \quad \text{s.t.} \\ & L_{\tilde{f}}C(x) + L_{\tilde{g}}C(x)u + \mu_C(x,u|\mathbb{D}_N) - \beta\sigma_C(x,u|\mathbb{D}_N) + \gamma(C(x)) \geq 0,\end{aligned} \tag{11.12}$$

---

wherein by leveraging the control-affine structure of the system during the GP regression, we obtain a convex optimization problem, which is a second-order cone program (SOCP) that can be solved efficiently at high-frequency rates using modern solvers. This was proved in Theorems 9.3 and 10.1 of the last two chapters. Indeed, when specifically using a CBF or a CLF in place of $C$, we refer to (11.12) as GP-CBF-SOCP or GP-CLF-SOCP, respectively, as it would correspond to the data-driven stability and safety filters presented in Chapters 9 and 10.

Note that the feasibility analysis of Chapter 10 is directly applicable to this optimization program. If the dataset utilized to conduct the GP regression is not informative-enough, the problem might become infeasible. Furthermore, the guarantee that the true certifying constraint will be satisfied with high probability exists only when the SOCP filter in (11.12) is feasible.

During the deployment of the SOCP filter on real-world systems, a practical strategy to address cases when infeasibility occurs is to use a backup control input computed by the following second-order cone program:

$$\pi_{\text{GP-CF}}(x) = \underset{u\in\mathcal{U}}{\arg\min}\left(\beta\sigma_C(x,u|\mathbb{D}_N) - \widehat{L_g C}(x|\mathbb{D}_N)u\right). \tag{11.13}$$

This selects a control input within the input bound that minimizes the violation of the constraint (11.11).

The main challenge in executing (11.12) online lies not in solving the optimization problem, but rather in the computationally demanding evaluation of $\sigma_C$ when the size of the dataset is large. The time complexity of the matrix inverse in (11.8), $(K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1}$, is $\mathcal{O}(N^3)$, while the remaining matrix multiplication involved in evaluating $\Sigma_C(x_*|\mathbb{D}_N)$ has a time complexity of $\mathcal{O}(N^2)$. Although the matrix inversion can be performed offline, when the dataset is large, the $\mathcal{O}(N^2)$ complexity still remains challenging. This issue primarily motivates the development of Sparse GP literature [128] and the online smart data selection algorithm proposed in this chapter, which can reduce the computational complexity to a linear dependence on $N$.

### 11.2.5 Running Example: 2D Polynomial System (Polysys)

We now introduce a simple running example system, referred to as Polysys, which is utilized throughout the chapter. It is important to note that this low-dimensional toy example is not intended to showcase the computational advantage of our method which will be presented in Section 11.4. Instead, its purpose is to provide a walk-through of the inner workings of our approach for the readers. To achieve this, we have access to the true plant dynamics, allowing us to compare our method with the ideal oracle certifying filter. Moreover, the dataset constructed for the data-driven certifying filters is not meant to represent a realistic dataset. Instead, it is a simplistic dataset designed for easy comprehension by the readers.

The dynamics of the system, whose vector fields are polynomial functions of the state $x$, are given by:

$$\dot{x} = \begin{bmatrix} f_1^T v \\ f_2^T v \end{bmatrix} + \begin{bmatrix} 1 + g_{11}^T v & g_{12}^T v \\ g_{21}^T v & 1 + g_{22}^T v \end{bmatrix} u, \tag{11.14}$$

where $x = [x_1 \quad x_2]^T$ is the state, $u = [u_1 \quad u_2]^T$ is the control input, $v = [x_1 \;\; x_2 \;\; x_1^2 \;\; x_1x_2 \;\; x_2^2 \;\; x_1^3 \;\; x_1^2x_2 \;\; x_1x_2^2 \;\; x_2^3] \in \mathbb{R}^9$ is a vector that aggregates the monomials of the state, and $\{f_1, f_2, g_{11}, g_{12}, g_{21}, g_{22}\} \in \mathbb{R}^9$ are randomly generated coefficient vectors. We introduce the parametric model uncertainty on the true plant by perturbing the coefficient vectors from the nominal model.

In this example, we aim to design a control policy that stabilizes the system to the zero equilibrium point. To achieve this by using the certifying filter, we design the CLF $V(x) = x^T P x$ as the certificate function, where $P$ is the solution of the Algebraic Riccati Equation for the linearized system of the nominal model (11.14) around $x = 0$. In this example, we set $\pi_{\text{ref}}(x) \equiv 0$ since we do not have any other explicit tasks to achieve. As shown in Figure 11.1, we can see that the oracle CLF-QP (orange) is able to stabilize the state to the equilibrium, confirming that the CLF is a valid certificate function for the true plant. However, due to the model uncertainty we introduce to the true plant, the nominal model-based CLF-QP (green) fails to stabilize the system.

We next show the application of the GP-CLF-SOCP in (11.12) to this example. We first construct the dataset $\mathbb{D}_N$ in order to apply GP regression to $\Delta_C$. We partition the subspace of the state space $[-2, 2] \times [-2, 2]$ into the coarse state grid of size $(10, 10)$. At every vertex $x_j$ of the state grid, we apply the randomly sampled control input $u_j$ to simulate the system (11.14) for a sampling time $\Delta t$ and collect a single data point $(\bar{x}_j = (x_j, u_j), \bar{z}_j)$. We account for the numerical differentiation error in obtaining $\bar{z}_j$ as measurement noise. In addition to the data from the coarse grid, we also incorporate some densely populated data points centered at a few selected state and action pairs, $(x_a, u_a)$. Around each of these points, a dense state-control grid is created by gridding up $[x_{a,1}-\delta, x_{a,1}+\delta] \times [x_{a,2}-\delta, x_{a,2}+\delta] \times [u_{a,1}-\delta, u_{a,1}+\delta] \times [u_{a,2}-\delta, u_{a,2}+\delta]$ in (2, 2, 2, 2) grid, where we set $\delta = 0.1$. This results in a total of 81 data points collected at each $(x_a, u_a)$. In the subsequent sections describing the Polysys example, we refer to the data points generated from a single dense grid as *a data cluster*. Combined together, we get in total $N = 361$ data points, visualized in Figure 11.2.

We use GP regression to fit $\Delta_C$ from the dataset presented above, using the ADP compound kernel with isotropic squared exponential kernels as components. Then, we apply the GP-CLF-SOCP of (11.12) to control the system. As shown in Figure 11.1, the GP-CLF-SOCP using the full dataset (black dashed line) is able to stabilize the system to the origin despite the uncertainty in the true plant dynamics.

## 11.3 Constraint-Guided Online Data Selection

In this section, we present the core contribution of our chapter: a smart online data selection algorithm that improves the time complexity of GP inference for the GP-CF-SOCP from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

Using the entire dataset to evaluate $\sigma_C^2(x_*, u_*|\mathbb{D}_N)$ would yield minimal uncertainty for any query point $x_*, u_*$, as we would utilize all available information, but at the cost of high computational demands. One way to alleviate this computational burden is to construct a model that approximates the exact GP inference offline, which targets learning the function itself to generalize to every possible unseen query point at runtime. However, our approach, similar to many existing Sparse GP methods, is based on the idea that it is not necessary to reduce the uncertainty globally [190, 196, 67]. Instead, we aim to reduce the uncertainty for specific input classes relevant to our problem. The former approach, known as *induction*, aims to regress the function with high quality across the entire input space. In contrast, our approach, which is called *transduction*, focuses on learning only for specific test points that we care about [162]. Revisiting the learning objective in our problem, we seek to find $\pi_{\text{GP-CF}}(x)$ such that the certifying chance constraint (10.8b) is feasible. Therefore, our data selection algorithm is designed to efficiently achieve this goal.

To facilitate the presentation of our algorithm, we first introduce some simplified notations and preliminaries that will be used in this section. We also present a sufficient condition for the feasibility of GP-CF-SOCP, from which we derive the main control input direction

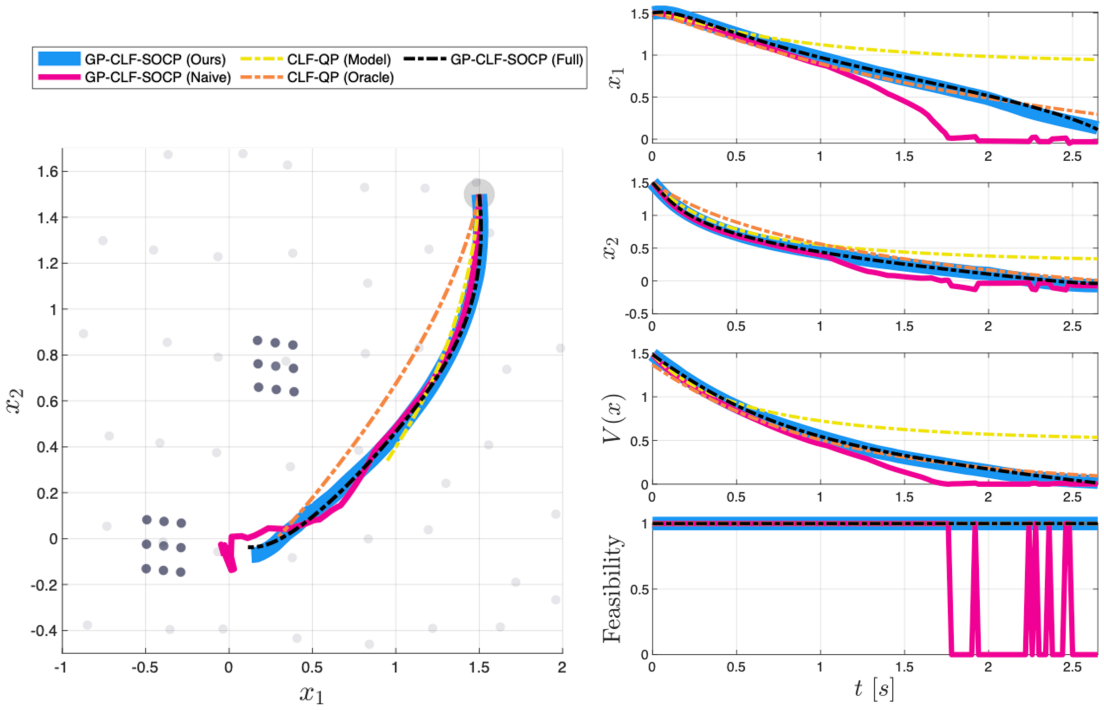



Figure 11.1: The simulation result of the Polysys example under various controllers: the nominal model-based CLF-QP (yellow dotted line), the oracle CLF-QP (orange dotted line), the GP-CLF-SOCP using full data (black dotted line), the GP-CLF-SOCP using naive data selection (magenta) discussed in Sec. 11.3.3, the GP-CLF-SOCP using our main data selection algorithm (blue) discussed in Sec. 11.3.4, both using the same number of online data, $M = 40$. The left plot illustrates the trajectory's progression in the state space for 2.6 seconds, with an initial state of $x_0 = [1.5\ 1.5]^T$. The four subplots on the right show the state $x_1$, $x_2$, the CLF values, and the feasibility of the QP and SOCP in time, respectively. While the naive approach often faces infeasibility and fails to stabilize the system close to the origin, our approach effectively selects an online dataset that secures the feasibility of the SOCP throughout the simulation.

we want to characterize. This control input direction is the foundation upon which we apply the concept of transduction in our data selection algorithm.

### 11.3.1 Preliminaries for the Data Selection Algorithm

#### Simplified Notations for Kernels

We introduce simplified notations as below:

$$\mathbf{k}_{ij} := \mathbf{k}\left((x_i, u_i), (x_j, u_j)\right),$$

$$\mathbf{k}_{**}(x, u) := \mathbf{k}\left((x, u), (x, u)\right),$$
$$\mathbf{k}_{*i}(x, u) := \mathbf{k}\left((x, u), (x_i, u_i)\right),$$

and $\mathbf{k}_i := \mathbf{k}_{ii}$, where $(x_i, u_i)$ is an input point in $\mathbb{D}_N$. We also consider the compound kernel that captures only the control vector field-relevant part:

$$\mathbf{k}^u((x, u), (x', u')) := u^\top \text{Diag}(k_{g_1}(x, x') \cdots, k_{g_m}(x, x')])u'. \tag{11.15}$$

Note that, from (11.6) and (11.15), $\mathbf{k}((x, u), (x', u')) = k_f(x, x') + \mathbf{k}^u\left((x, u), (x', u')\right)$. Similarly, we define

$$\mathbf{k}^u_{**}(x, u) := \mathbf{k}^u\left((x, u), (x, u)\right),$$
$$\mathbf{k}^u_{*i}(x, u) := \mathbf{k}^u\left((x, u), (x_i, u_i)\right).$$

Using simplified notations, we can express for any target point $(x_*, u_*)$

$$\begin{bmatrix}1 & u_*^\top\end{bmatrix} K_{*U} = \left[\mathbf{k}_{*1}(x_*, u_*) \ \cdots \ \mathbf{k}_{*N}(x_*, u_*)\right],$$

and (11.8) becomes

$$\sigma_C^2(x_*, u_* | \mathbb{D}_N) = \mathbf{k}_{**}(x_*, u_*) - \left[\mathbf{k}_{*1} \ \cdots \ \mathbf{k}_{*N}\right] (K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1} \begin{bmatrix}\mathbf{k}_{*1} \\ \vdots \\ \mathbf{k}_{*N},\end{bmatrix}$$

with $(x_*, u_*)$ dropped in $\mathbf{k}_{*i}$ for simplicity. Note that the first term on the right-hand side is contributed from the GP prior, and the choice of the data only affects the second term.

#### Sufficient Condition for Pointwise Feasibility of GP-CF-SOCP

The expression of the certifying chance constraint in (11.11) highlights the tradeoff required to evaluate its feasibility, which lies between the prediction uncertainty of the GP regression on the left-hand side and the mean-estimate of the true certifying constraint on the right-hand side. This structure is useful for verifying the following sufficient condition for the feasibility of (11.11).

**Lemma 11.1.** *Given a dataset $\mathbb{D}_N$, for a point $x \in \mathcal{X}$, If there exists a constant $\alpha > 0$ such that the following inequality holds, then the GP-CF-SOCP in* (11.12) *is feasible:*

$$\beta\, \sigma_C\Big(x, \alpha\widehat{L_gC}(x|\mathbb{D}_N)\big|\mathbb{D}_N\Big) < \alpha \left\|\widehat{L_gC}(x|\mathbb{D}_N)\right\|^2. \tag{11.16}$$

*The feasible control input can be found by taking $u = \alpha'\widehat{L_gC}(x|\mathbb{D}_N)^\top$ with sufficiently large $\alpha' > 0$.*

*Proof.* See Appendix A.5.1. □

The main implication of the above lemma is that the feasibility of (11.12) can be assessed by examining the size of the prediction uncertainty, $\sigma_C$, in just one control input direction, specifically the direction of the mean-based estimate of $L_gC(x)$, denoted as $\widehat{L_gC}(x|\mathbb{D}_N)$. This direction is particularly important because according to what the mean prediction of the GP tells, it is the control input direction in which we can most effectively regulate the value of $C(x)$. If the prediction uncertainty is sufficiently small in this direction, by taking the control input in this direction with large enough magnitude, we can ensure (11.12) to be feasible.

### 11.3.2 Data Selection Objective

We seek to design an online data selection algorithm, that selects a subset of data from the entire dataset, $\mathbb{D}_M(x) \subset \mathbb{D}_N$, with $M \ll N$, at every sampling time at the current state $x$. Once $M$ online data points are determined, the GP-CF-SOCP in (11.12) is solved with the online dataset $\mathbb{D}_M(x)$ in place of $\mathbb{D}_N$, to determine the filtered control input $\pi_{\text{GP-CF}}(x)$ which will be applied to the system next. Among the data points in the full dataset, we want to select a limited number of points that are most helpful in characterizing the control direction that secures the feasibility of the certifying chance constraint in (11.10).

We attempt to achieve this by utilizing the result of Lemma 11.1, trying to make sure that condition (11.16) is met with the limited $M$ data points we are allowed to use. Adopting the approach of transduction, the goal of the data selection is set to reduce the uncertainty in the direction of $\widehat{L_gC}(x|\mathbb{D}_M)$, i.e., select $\mathbb{D}_M(x)$ which best reduces $\sigma_C\Big(x, \alpha\widehat{L_gC}(x|\mathbb{D}_M)|\mathbb{D}_M\Big)$ for sufficiently large $\alpha$. However, we do not know how large $\alpha$ is sufficient to render (11.16) feasible prior to selecting $\mathbb{D}_M(x)$ and actually solving the SOCP. Therefore, we eliminate the dependency on the magnitude of $\alpha$ by considering the following problem:

$$\arg\min_{\mathbb{D}_M(x)} \left[\lim_{\alpha\to\infty} \frac{1}{\alpha}\sigma_C\Big(x, \alpha\widehat{L_gC}(x|\mathbb{D}_M)|\mathbb{D}_M\Big)\right]. \tag{11.17}$$

Note that we drop the dependency of $\mathbb{D}_M$ on $x$ whenever it is obvious, for notational simplicity. From the expression of the variance in (11.16), we can derive the following lemma that transforms the objective function above into a form without the appearance of $\alpha$:

**Lemma 11.2.** *The optimization problem* (11.17) *can be equivalently expressed as*

$$\arg\max_{\mathbb{D}_M \subset \mathbb{D}_N} \mathcal{J}_{\mathbb{D}_M}(x, \widehat{L_g C}(x|\mathbb{D}_M)), \tag{11.18}$$

*where*

$$\mathcal{J}_{\mathbb{D}_M}(x,u) := [\mathbf{k}^u_{*1}(x,u) \ \cdots \ \mathbf{k}^u_{*N}(x,u)] \, (K_{\mathbb{D}_M} + \sigma_n^2 I)^{-1} \begin{bmatrix} \mathbf{k}^u_{*1}(x,u) \\ \vdots \\ \mathbf{k}^u_{*M}(x,u) \end{bmatrix}, \tag{11.19}$$

*which is the second order term in the control input $u$ of the posterior variance $\sigma^2_C(x,u|\mathbb{D}_M)$.*

*Proof.* See Appendix A.5.2. □

Thus, we will consider $\mathcal{J}_{\mathbb{D}_M}(x, \widehat{L_g C}(x|\mathbb{D}_M))$ as the *objective function of the data selection algorithm.*

**Remark 11.2.** *Since we do not have access to $\widehat{L_g C}(x|\mathbb{D}_M)$ prior to determining $\mathbb{D}_M$, we can replace $\widehat{L_g C}(x|\mathbb{D}_M)$ in $\mathcal{J}_{\mathbb{D}_M}$ with $\widehat{L_g C}(x|\mathbb{D}_N)$, where $\mathbb{D}_N$ is the entire dataset. Note that $\widehat{L_g C}(x|\mathbb{D}_N)$ only requires the computation of $\mu_C(x,u|\mathbb{D}_N)$ but not $\sigma_C(x,u|\mathbb{D}_N)$. Since $\mathbf{z}^\top (K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1}$ in* (11.7) *can be precomputed offline, the time complexity of evaluating $\widehat{L_g C}(x|\mathbb{D}_M)$ online is $\mathcal{O}(N)$. When $N$ is very large, it may be impractical or computationally infeasible to evaluate $\mathbf{z}^\top (K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1}$ offline since it requires to compute the inverse of the matrix. In such cases, an effective approximation for $\widehat{L_g C}(x|\mathbb{D}_M)$ can still be achieved by using $\widehat{L_g C}(x|\mathbb{D}'_M)$, where $\mathbb{D}'_M$ represents the dataset selected online at the previous time step.*

### 11.3.3 Naive Data Selection Approach

Before we proceed to present the main algorithm of the chapter, let's take a moment to build a better understanding of the data points we wish to include in $\mathbb{D}_M(x)$. To facilitate this discussion and simplify our thought process, consider a scenario where all data points in $\mathbb{D}_N$ are not correlated with one another, meaning that $\mathbf{k}_{ij} = \mathbf{k}\left((x_i,u_i),(x_j,u_j)\right) = 0$ for all $i \neq j$. Additionally, let's for now consider that there is no noise in the data, so $\sigma_n = 0$. In this simplified case, $K_{\mathbb{D}_N} + \sigma_n^2 I = \mathrm{Diag}(\mathbf{k}_1, \cdots, \mathbf{k}_N)$, and from (11.19) it holds that

$$\begin{aligned}
\mathcal{J}_{\mathbb{D}_M}(x,u) &= [\mathbf{k}^u_{*1}(x,u) \ \cdots \ \mathbf{k}^u_{*N}(x,u)] \mathrm{Diag}\left(\frac{1}{\mathbf{k}_1}, \cdots, \frac{1}{\mathbf{k}_M}\right) \begin{bmatrix} \mathbf{k}^u_{*1}(x,u) \\ \vdots \\ \mathbf{k}^u_{*M}(x,u) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\mathbf{k}^u_{*1}(x,u)}{\sqrt{\mathbf{k}_1}} & \cdots & \frac{\mathbf{k}^u_{*M}(x,u)}{\sqrt{\mathbf{k}_M}} \end{bmatrix} \begin{bmatrix} \frac{\mathbf{k}^u_{*1}(x,u)}{\sqrt{\mathbf{k}_1}} \\ \vdots \\ \frac{\mathbf{k}^u_{*1}(x,u)}{\sqrt{\mathbf{k}_1}} \end{bmatrix} = \sum_{i=1}^{M} \left( \frac{\mathbf{k}^u_{*i}(x,u)}{\sqrt{\mathbf{k}_i}} \right)^2.
\end{aligned}$$

Thus, if we define

$$n_i(x,u) := \frac{|\mathbf{k}^u_{*i}(x,u)|}{\sqrt{\mathbf{k}_i}} = \frac{|\mathbf{k}^u\left((x,u),(x_i,u_i)\right)|}{\sqrt{\mathbf{k}\left((x_i,u_i),(x_i,u_i)\right)}}, \tag{11.20}$$

we get

$$\mathcal{J}_{\mathbb{D}_M}\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right) = \sum_{i=1}^{M} n_i^2\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right). \tag{11.21}$$

Therefore, we can optimize $\mathcal{J}_{\mathbb{D}_M}\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right)$ simply by selecting $M$ points from $\mathbb{D}_N$ that exhibit maximum values of $n_i\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right)$. The time complexity of finding such points is $\mathcal{O}(N)$, which can be achieved using efficient algorithms, such as a quick selection.

Equation (11.21) highlights that $n_i\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right)$ is the measure of the relevance of the data point $(x_i, u_i)$ to the feasible direction of the certifying chance constraint. Here, we offer a concise explanation of the geometric interpretation of this measure.

For kernels used in GP regression, note that the kernel value of two inputs, $k(x,x')$ can be interpreted as an inner product between the feature vectors of $x$ and $x'$, i.e. $k(x,x') = \phi(x)\cdot\phi(x')$ [216]. For the ADP kernel in Definition 9.1, denoting the feature vectors for individual kernels as $\phi_f,\ \phi_{g_1},\cdots,\phi_{g_m}$, we can express the ADP kernel's feature vector as $\boldsymbol{\phi}(x,u) := [\phi_f(x)\ \phi_{g_1}(x)\ \cdots\ \phi_{g_m}(x)]\begin{bmatrix}1\\ u\end{bmatrix}$. Consequently, we get

$$n_i\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right) = \lim_{\alpha\to\infty} \frac{\boldsymbol{\phi}(x, \alpha\widehat{L_g C}(x|\mathbb{D}_M))\cdot\boldsymbol{\phi}(x_i,u_i)}{\alpha\sqrt{\boldsymbol{\phi}(x_i,u_i)\cdot\boldsymbol{\phi}(x_i,u_i)}},$$

from (11.20), where we get rid of the autonomous vector field relevant part from the numerator in (11.20) by taking the limit of $\alpha\to\infty$. Thus, $n_i\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right)$ captures how well the data point is aligned in the feature space of the ADP kernel with the feasible input direction.

In summary, the naive approach, which selects $M$ points with maximum values of $n_i\left(x, \widehat{L_g C}(x|\mathbb{D}_M)\right)$ from the dataset $\mathbb{D}_N$, optimally achieves the objective in (11.17) under the ideal conditions of an uncorrelated dataset and absence of measurement noise. However, these assumptions do not accurately represent the characteristics of real-world datasets. In practice, data from actual systems often have a high correlation because sampled data points from trajectories are sequential and share similar properties due to their close proximity in time and space.

We use the Polysys example to highlight the failure of the naive approach in handling datasets that deviate from ideal conditions, particularly those containing self-correlated data points. Our demonstration reveals that the naive approach may choose an unsuitable $\mathbb{D}_M$, rendering the SOCP filter infeasible, for realistic settings. This limitation motivates the development of a more advanced data selection algorithm, which we present in the next section.

***Running Example–Polysys (Cont'd):*** As described in Section 11.2.5, the dataset created for the Polysys example contains highly correlated data points, including data clusters. Figure 11.2 (a) illustrates a failure case of the naive algorithm. In the first row of Figure 11.2 (a), we visualize the selected data points $\mathbb{D}_M(x)$ at a query state $x$ under various values of $M$. The second row represents the prediction uncertainty $\beta\sigma_C(x,u)$ in control-input space as an ellipse, and $\widehat{L_gC}(x|\mathbb{D}_M)$ as a dashed magenta line, thereby illustrating the competitive relationship between the left-hand side (ellipse) and the right-hand side (magenta line) of the certifying chance constraint (11.11).

Since the naive approach greedily selects the points that maximize $n_i\Big(x, \widehat{L_gC}(x|\mathbb{D}_M)\Big)$ without considering the correlation between them, the selected data points are sourced from the data cluster that is close to the query state. The effect of using such highly self-correlated data points as $\mathbb{D}_M$ is shown in the second row of the figure. It demonstrates that even after increasing the size of $M$ from 40 to 60, the uncertainty ellipse barely reduces its size, leading to the infeasibility of the SOCP. Clearly, selecting such concentrated data points does not provide additional information, which intuitively illustrates why the naive approach can fail.

### 11.3.4 Proposed Online Data Selection Algorithm

Selecting the data points in the dataset $\mathbb{D}_N$ that maximize our objective function $\mathcal{J}_{\mathbb{D}_M}(x, \widehat{L_gC}(x|\mathbb{D}_M))$ when the dataset is self-correlated is in fact a combinatorial optimization problem which is NP-hard [109]. This is because finding the optimal subset selection requires choosing those points that maximize the correlation with respect to the target point, but minimize the self-correlation between them, as (11.19) suggests. Therefore, directly optimizing for the objective function online is intractable. The result presented next, which is the main assertion of our chapter, allows us to indirectly find a good candidate $\mathbb{D}_M$ by maximizing a lower bound of the objective function.

**Theorem 11.1.** *For a given dataset $\mathbb{D}_N$ with $N \geq 2$, assume that there exists a constant $\epsilon \in [0,1)$ that satisfies*

$$\mathbf{k}_{ij}^2 < \epsilon^2 \mathbf{k}_i \mathbf{k}_j, \tag{11.22}$$

*for all $i, j = 1, \cdots, N$ and $i \neq j$, and*

$$\sigma_n^2 \leq \frac{\epsilon^2 (N-1) \min_i \mathbf{k}_i}{1-\epsilon}. \tag{11.23}$$

*Then, $\mathcal{J}_{\mathbb{D}_M}(x,u)$ is lower bounded by the inequality below*

$$\mathcal{J}_{\mathbb{D}_M}(x,u) \geq \frac{1-\epsilon}{1+\epsilon(N-2)} \sum_{i=1}^{N} n_i^2(x,u). \tag{11.24}$$

*Note that the equality is satisfied when $\epsilon = 0$.*

*Proof.* See Appendix A.5.3. □

The condition (11.22) requires the dataset to exhibit no more than a weak correlation, while condition (11.23) necessitates that the noise variance remains comparatively small with respect to the correlation threshold $\epsilon$. It is worth noting that the latter condition becomes less stringent as the value of $N$ increases. Under these conditions, Theorem 11.1 concludes that the lower bound of the objective function can be maximized by, again, selecting $M$ points with maximum values of $n_i\Big(x, \widehat{L_g C}(x|\mathbb{D}_M)\Big)$.

Choosing the value of the correlation threshold $\epsilon$ allows users to strike a balance between the *contribution* of the term $\sum_{i=1}^{M} n_i^2(x,u)$ and the *adverse impact* of self-correlation on the objective function $\mathcal{J}_{\mathbb{D}_M}(x,u)$. With a value of $\epsilon = 1$, the data selection is identical to the naive approach. However, the right-hand side of (11.24) being zero indicates that the information gained from the selected data points can be significantly compromised by their self-correlations, potentially resulting in no contribution to the objective function at all. Conversely, $\epsilon = 0$ prohibits users from using data points with even the slightest correlation, which is impractical in real-world scenarios.

Ideally, we should find the optimal value of $\epsilon$ that offers the best trade-off. However, determining the optimal $\epsilon$ is an NP-hard problem, as it shares the same problem complexity as maximizing the data selection objective $\mathcal{J}_{\mathbb{D}_M}$ directly. A practical and effective strategy is to leverage prior knowledge of the full dataset to identify an acceptable $\epsilon$ value, for instance, by evaluating the histogram of $\frac{\mathbf{k}_{ij}^2}{\mathbf{k}_i \mathbf{k}_j}$ for the dataset and selecting an $\epsilon$ that corresponds to a reasonable quantile of data satisfying (11.22).

Leveraging the result of Theorem 11.1, we aim to maximize the lower bound as a proxy for the original objective function, thereby rendering the problem more tractable. The essence of our main algorithm is to condition the dataset to satisfy the assumption in (11.22), ensuring that Theorem 11.1 holds, and then identify the data points for which $\sum_{i=1}^{M} n_i^2(x,u)$ is maximized.

We achieve this through a two-fold algorithm. First, during the offline phase, we compute a ready-to-use binary matrix $\mathcal{B} \in \mathbb{R}^{N \times N}$, with elements defined as follows:

$$\mathcal{B}_{ij} = \begin{cases} 1 & \text{if } \mathbf{k}_{ij}^2 < \epsilon^2 \mathbf{k}_i \mathbf{k}_j \\ 0 & \text{otherwise} \end{cases} \tag{11.25}$$

The matrix $\mathcal{B}$ can be efficiently constructed by applying an $\epsilon$-threshold to the matrix

$$\mathrm{Diag}\left(\frac{1}{\sqrt{\mathbf{k}_1}}, \cdots, \frac{1}{\sqrt{\mathbf{k}_M}}\right)^T K_{\mathbb{D}_N} \mathrm{Diag}\left(\frac{1}{\sqrt{\mathbf{k}_1}}, \cdots, \frac{1}{\sqrt{\mathbf{k}_M}}\right). \tag{11.26}$$

This operation has a time complexity of $\mathcal{O}(N^2)$ but occurs during the offline stage, so it does not impact the online time complexity.

Next, in the online phase described in Algorithm 11.1, first, we initialize a candidate dataset as the entire dataset (Line 4). We then sequentially add to the online dataset $\mathbb{D}_M$ the data point that has the maximum value of $n_i\Big(x, \widehat{L_g C}(x|\mathbb{D}_M)\Big)$ among those in the

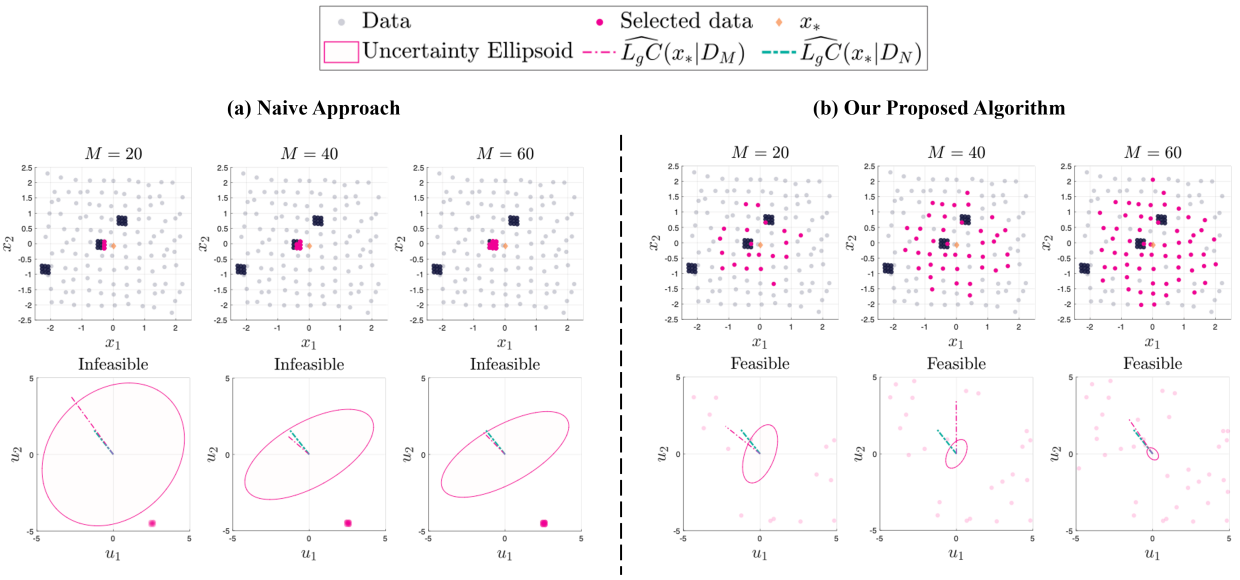



Figure 11.2: Comparison between the two data selection strategies–(a) naive approach described in Section 11.3.3 and (b) our main algorithm described in Section 11.3.4, on the Polysys running example system, with a varying number of online selected data points ($M = 20, 40, 60$). In each case, the first row visualizes the entire dataset $\mathbb{D}_N$ (grey dots) projected on the state space and the data points selected online $\mathbb{D}_M$ (magenta dots) according to the data selection algorithm at the query state $x = [0.011 \ \ -0.0756]^T$ (orange diamond). The second row visualizes the selected points projected on the control input space (magenta dots), and the prediction uncertainty $\beta\sigma_C(x, u)$'s growth in the control space as an ellipse. We also visualize $\widehat{L_gC}(x|\mathbb{D}_N)$ and $\widehat{L_gC}(x|\mathbb{D}_M)$ as the dashed green and magenta lines, respectively. The ellipse and magenta line represent the growth of the right-hand side and left-hand side of (11.11), respectively. The feasibility of the chance certifying constraint can be deduced by comparing the length of the magenta line to the ellipse's radial distance in the magenta line's direction. A smaller relative ratio suggests that a larger control input in the $\widehat{L_gC}(x|\mathbb{D}_M)$ direction is required to satisfy the chance constraint.

candidate dataset (Line 6-7). As we select each data point, we remove from the candidate dataset the data points that have a correlation greater than $\epsilon$ relative to the selected point, by directly referring to the matrix $\mathcal{B}$ (Line 8).

Algorithm 11.1 has a time complexity of $\mathcal{O}(MN)$, as each operation in Line 6 and Line 8 inside the for loop is $\mathcal{O}(N)$. At each time step, after obtaining $\mathbb{D}_M$ from the proposed algorithm, we use this online dataset for the GP-CF-SOCP filter in (11.12) instead of using the entire dataset. This requires evaluating the matrix inverse in (11.7) and (11.8) online, which has a time complexity of $\mathcal{O}(M^3)$. Thus, with our proposed approach, obtaining the optimal filtered control input $\pi_{\text{GP-CF}}(x)$ has a total time complexity of $\mathcal{O}(NM + M^3)$, in

**Algorithm 11.1:** Online Data Selection for GP-CF-SOCP

**Input:** Current state $x$, entire dataset $\mathbb{D}_N$, $\mathcal{B}$ defined in (11.25)
**Output: Online dataset** $\mathbb{D}_M$
$\mathbb{D}_M \leftarrow \emptyset$
$\mathcal{I}_{\text{candidate}} \leftarrow \{1, 2, \cdots, N\}$
**for** $k = 0;\ k < M;\ k = k + 1$ **do**
  $i_* \leftarrow \arg\max_{i \in \mathcal{I}_{\text{candidate}}}\ n_i(x, \widehat{L_g C}(x|\mathbb{D}_N))$
  $\mathbb{D}_M \leftarrow \mathbb{D}_M \cup (x_{i_*}, u_{i_*}, z_{i_*})$
  $\mathcal{I}_{\text{candidate}} \leftarrow \mathcal{I}_{\text{candidate}} \setminus \{j \in \mathcal{I}_{\text{candidate}} \mid \mathcal{B}_{ij} == 0\}$
**end**

terms of $N$ and $M$. We are neglecting the time complexity of solving the SOCP since it does not depend on the number of data points. Given that we choose $M << N$ in practice, the time complexity of the GP-CF-SOCP safety filter combined with our online data selection algorithm is linear in $N$.

***Running Example–Polysys (Cont'd):*** We investigate how Algorithm 11.1 selects data online and improves the downstream objective of enhancing the feasibility of the GP-CLF-SOCP through its self-correlation remedy in the Polysys example. We use $\epsilon = 0.95$ in the example, which is the minimum correlation between data points within a data cluster. Using this value prevents our main algorithm from selecting more than one point per data cluster. The first row of Figure 11.2 (b) displays that our proposed algorithm selects at most one data from each data cluster even as $M$ increases. This correlation-aware behavior resulting from upper-bounding the maximum self-correlation of the selected data points induces the algorithm to select diverse data. Consequently, the prediction uncertainty, illustrated as the ellipse in the second row of the image, is reduced as $M$ increases in all directions of $u$ but, more importantly, it is primarily reduced in the direction of $\widehat{L_g C}(x|\mathbb{D}_N)$. Moreover, in the case of $M = 20$, it is notable that the algorithm prioritizes selecting data points whose control input values are well aligned in the direction of $\widehat{L_g C}(x|\mathbb{D}_M)$. This shows that $n_i\Big(x, \widehat{L_g C}(x|\mathbb{D}_M)\Big)$, the metric we use to select the data points, actually captures well the relevance of the data with respect to $\widehat{L_g C}(x|\mathbb{D}_M)$, the feasible direction of the certifying chance constraint. As a result, the GP-CLF-SOCP controller utilizing the online dataset constructed by our main algorithm is feasible for all $M$ in the figure.

### 11.3.5 Related Data Selection Methods

The point at issue of this chapter is very related to the information-theoretic data subset selection [52, 203] and sensor placement [48, 109] problems, which are known to be NP-hard for many different objective functions, such as mutual information and conditional entropy [106, 110]. While our focus is on optimizing a particular certification-oriented measure

(11.19) that differs from the information-theoretic objective functions typically used for these problems, our optimization problem still suffers from the same combinatorial challenges, and solving (11.18) to optimality would be intractable for large datasets.

A reasonable alternative to our proposed data selection algorithm, would be to form the online dataset $\mathbb{D}_M$ by greedily selecting, one at a time, the data points that maximize (11.18). This idea was applied to the sensor placement problem in [110] and has been used for data-driven control problems in [194, 116]. To approximately solve (11.18), this greedy selection method can be implemented with an asymptotic time complexity of $\mathcal{O}(NM^3)$. While this asymptotic complexity is only slightly worse than the $\mathcal{O}(NM)$ complexity of Algorithm 11.1, in practice we observe that the greedy method is too slow to perform the data selection online, even when using the locality and lazy evaluation speedups proposed in [109].

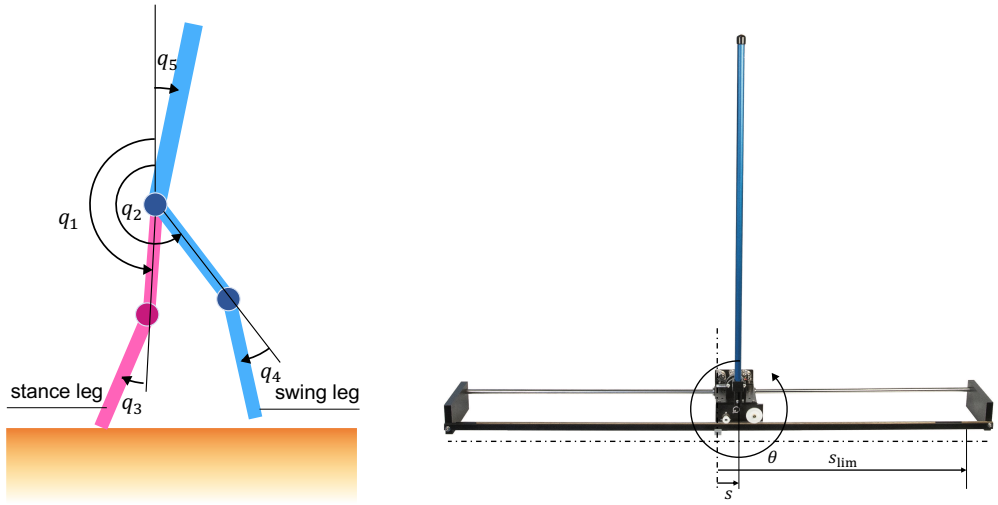
A simpler selection method would be to choose the $k$-nearest neighbors (k-NN) at each query state-action pair. However, given the control-affine structure of the target function $\Delta_C$, it is not immediately clear which distance metric should be used for the k-NN in order to capture the most relevant information. The authors in [220] propose to use the *kernel distance* [80, 155], which is the euclidean distance in the kernel feature space. Although simple, these k-NN selection approaches suffer from similar problems as our naive selection algorithm, as they do not consider the self-correlation of the dataset.

## 11.4 Examples

In this section, we apply our method to three specific examples, consisting of two numerical simulations and one hardware experiment. We refer to the GP-CF-SOCP filter using the full dataset as ***GP-CF-SOCP (Full)***, to the GP-CF-SOCP using the online data constructed by the naive approach in Section 11.3.3 as ***GP-CF-SOCP (Naive)***, and to the GP-CF-SOCP using the online data constructed by our main data selection algorithm as ***GP-CF-SOCP (Ours)***.

### 11.4.1 Running Example: Polynomial System (Cont'd)

The simulation results of Polysys under the GP-CLF-SOCP (Naive) and GP-CLF-SOCP (Ours) are evaluated in this study, extending the analysis in Section 11.2.5. To ensure a fair comparison, both controllers select $M = 40$ data points from the full dataset, which has a total of $N = 361$ data points, as constructed in Section 11.2.5. The results are presented in Figure 11.1, which shows that GP-CLF-SOCP (Ours) is feasible throughout the simulation period, imposing the probabilistic guarantee of stability to the closed-loop system. In contrast, the GP-CLF-SOCP (Naive) fails to do so, and the SOCP is infeasible very frequently with this approach. Note that when the SOCP is infeasible, the backup controller in (11.13) is deployed, and the stability property that the certifying constraint is trying to impose is not guaranteed anymore.




(a) (b)

Figure 11.3: (a) The configuration of the planar five-link bipedal robot RABBIT [37] (b) Cart-pole experiment setup based on Quanser Linear Servo Base Unit with Inverted Pendulum [161].

Further analysis of the trajectory of the two controllers in the left plot of Figure 11.1 reveals two important behaviors. First, the GP-CLF-SOCP (Ours) (blue) exhibits a similar trajectory to that of GP-CLF-SOCP (Full) (black). This implies that the effect of the information loss due to using only the online-selected data points is negligible when employing our algorithm.

Second, the GP-CLF-SOCP (Naive) controller (magenta) exhibits a more aggressive trajectory compared to GP-CLF-SOCP (Ours), as evidenced by the rapid decay of $V(x)$ and a swift change in state history starting at around $t = 1$s. This corresponds to the moment when the naive algorithm begins selecting most of the data points in the densely populated data cluster near the origin. These observations demonstrate that the naive data selection leads to a large prediction uncertainty in the direction of $\widehat{L_g C}(x|\mathbb{D}_N)$, resulting in a considerably more conservative control policy than necessary. This also makes the controller more susceptible to infeasibility.

Note that the Polysys example is devised to provide a detailed walk-through of our method; thus, we do not benchmark the computation time of each method in this example. Given the relatively small number of data points used in this example, the computational efficiency gained from our method would not be easily noticeable.

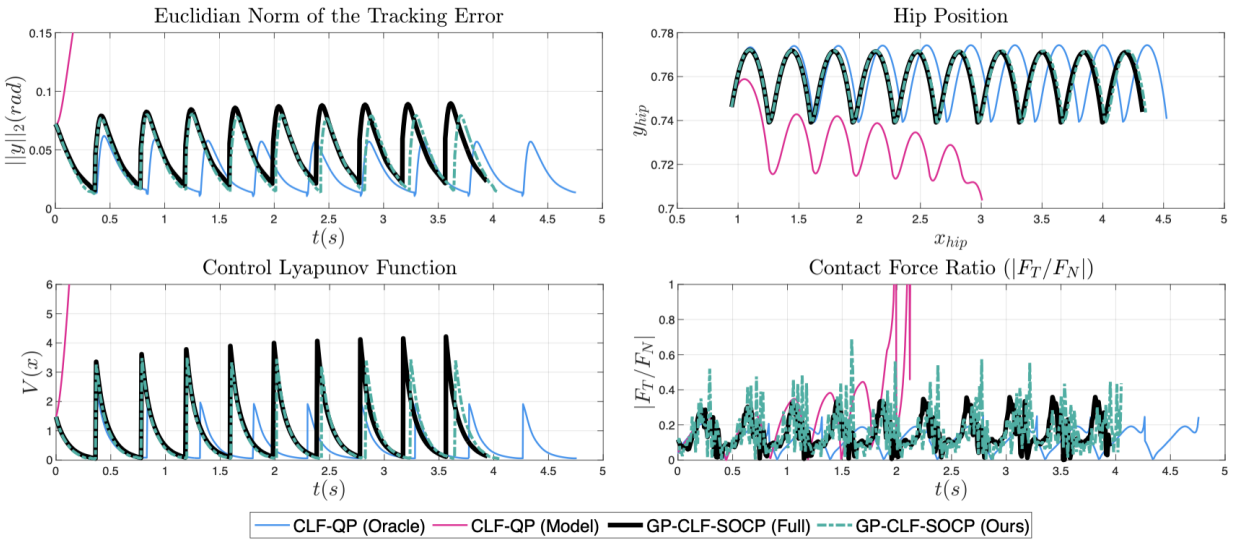



Figure 11.4: Simulation results of RABBIT achieving stable walking under various controllers: the nominal model-based CLF-QP (magenta), the oracle CLF-QP (blue), GP-CLF-SOCP (Full) (black), and GP-CLF-SOCP (Ours) (green). The left column depicts histories of the Euclidean norm of the tracking error with respect to the reference gait, $y$, and the value of the Control Lyapunov Function $V$. The right column shows the evolution of the hip's vertical position from the ground and the ratio between the tangential and normal contact forces, which should not surpass $k_f = 0.8$ for the robot to avoid slipping.

### 11.4.2 High-dimensional System in Simulation: Five-link Walker

In this section, we explore the performance of our algorithm in a high-dimensional system, RABBIT [37], a planar five-link bipedal robot consisting of ten state variables. We demonstrate the effectiveness of our algorithm in achieving the robot's stable walking, which is set as the desired system-critical constraint for the robot. The significance of our algorithm in reducing the computational demands of executing the GP-CF-SOCP certifying filter is highlighted.

RABBIT is a testbed system developed to study bipedal robot locomotion [37]. As depicted in Figure 11.3a, when one foot is in contact with the ground its configuration can be represented by the generalized coordinate vector $q = [q_1\ q_2\ q_3\ q_4\ q_5]^T$ consisting of the robot's joint angle variables. We adopt the mathematical model for RABBIT locomotion presented in [37] to design the simulation model of this system, where the state of the robot is defined as $x = [q\ \dot{q}] \in \mathbb{R}^{10}$, and the control input is defined as $u \in \mathbb{R}^4$, consisting of the hip and knee motor torques for both legs. The torque saturation is set at $150Nm$. The hybrid system description of the robot's walking process consists of a single-support swing

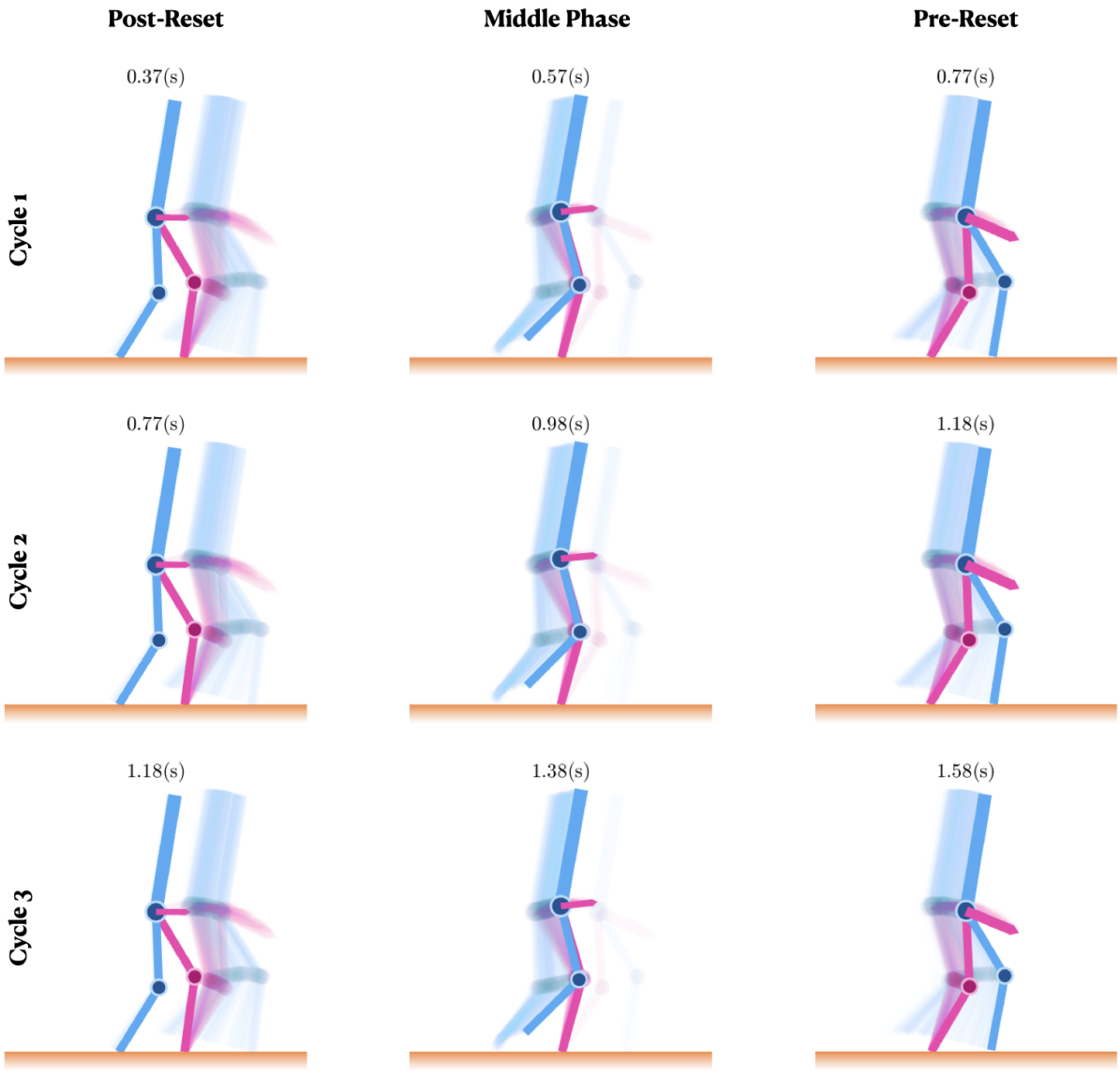



Figure 11.5: Visualization of a series of snapshots of the online data selected by our algorithm in the RABBIT simulation. Each snapshot illustrates the robot with its stance leg highlighted in magenta and the velocity of the torso indicated by the magenta arrow. Each row depicts a single walking cycle, while the same phase is aligned in the same column. The start and end of the swing phase are denoted by post-reset and pre-reset, respectively, and the middle phase represents the point at which $\theta(q) = 0.5$. The robot's state at each phase is displayed without transparency, while the selected data points are shown as the corresponding robot's configuration and torso velocity, drawn with transparency in the background.

Table 11.1: Total execution time (data selection, GP inference, numerical optimization) of the GP-CF-SOCP controller with different datasets in the RABBIT simulation and the cart-pole experiment. The table presents mean and standard deviations in milliseconds.

| | **GP-CF-SOCP (Ours)** | | | **GP-CF-SOCP (Full)** | | |
|---|---|---|---|---|---|---|
| System | mean | stdev | M | mean | stdev | N |
| RABBIT | 33.2 | 3.2 | 130 | 226.2 | 19.9 | 12569 |
| Cart-Pole | 11.8 | 0.75 | 40 | 60.4 | 4.1 | 6957 |

phase under a control-affine Lagrangian dynamics and a reset map defined by the rigid impact model, which switches the robot's state to the post-impact state upon the swing foot's impact with the ground.

The objective of the certifying filter is to achieve an exponentially stabilizing periodic gait for the RABBIT, despite the effect of the impacts. To accomplish this, we employ a Rapidly Exponentially Stabilizing Control Lyapunov Function (RES-CLF) [9] as our certificate function. We also set $\pi_{\text{ref}}(x) \equiv 0$ since this naturally captures the objective of minimizing the energy spent to produce the motor torques. In order to construct RES-CLFs, we first input-output linearize the continuous dynamics of the system by defining the output functions:

$$y(q) = y_a(q) - y_d(\theta(q)), \tag{11.27}$$

where $\theta(q)$ is a variable that defines the phase along the gait, which monotonically increases within each walking step, $y_a$ are the hip and knee coordinates of both legs, and $y_d(\cdot)$ is a desired trajectory of the hip and leg joints represented by a Bezier polynomial. This reference trajectory is generated offline using the Fast Robot Optimization and Simulation Toolkit (FROST) [81]. We can then decompose the state of the system into the transverse coordinates $\eta = [y\ \dot{y}]^T \in \mathcal{T} \subseteq \mathbb{R}^8$ and the zero coordinates $z = [\theta(q)\ \dot{\theta}(q)] \in \mathcal{Z} \subseteq \mathbb{R}^2$. After applying the input-output linearizing controller, we can represent the transverse dynamics as:

$$\dot{\eta} = f(\eta, z) + g(\eta, z)\mu, \tag{11.28}$$

where $\mu$ is the virtual input. By stabilizing $\eta$ to zero, we enforce the joint trajectory to converge to the desired stable walking gait defined by $y_d(\theta(q))$.

Model uncertainty is introduced in the simulation by scaling the mass and inertia values of the robot by a factor of 2, which poses a challenge for the controller to maintain stability during walking. Note that a payload is one of the most common sources of model uncertainty for legged robots in their practical applications. As illustrated in Figure 11.4, while the oracle CLF-QP (blue), which assumes access to the true plant dynamics, successfully completes ten steps, the nominal model-based CLF-QP (magenta), which is unaware of the change in mass and inertia, fails to stabilize the robot and it eventually falls down during the sixth step. This observation motivates using the GP-CLF-SOCP controller.

We collect the data points represented as $\bar{x}_j = ([\eta_j, z_j], \mu_j)$ since we aim to learn the effect of model uncertainty in the transverse dynamics (11.28). The dataset is collected in an episodic learning fashion, similar to our previous work [30]. The nominal model-based CLF-QP is run in the first episode to create an initial dataset for the GP regression. Following this, the GP-CLF-SOCP is executed, and the data collected from the new trajectory is iteratively added to the dataset. For the GP-CLF-SOCP, we initially use the full dataset; however, when the execution time of the SOCP controller approaches the limit of the target sampling time, we activate the data selection algorithm. It is essential to acknowledge that high-dimensional systems are more susceptible to the out-of-distribution problem, as data is inherently more scarce. To address this challenge, we introduce perturbations to the reset map at every impact event and create variations in the control policies executed in each episode, for example, by altering the number of $M$, in order to enhance the dataset's coverage. As a result, we obtain a comprehensive dataset comprising $N = 12,569$ data points.

When assuming the ability to deploy the GP-CLF-SOCP controller using the full dataset at a sampling rate of 20Hz (50ms), it can achieve ten successful steps without falling, as shown in Figure 11.4 (black). However, this would not be achievable in reality, as the average execution time of the controller using the full dataset is 226.2ms (4.4Hz), which significantly exceeds the target sampling time.

Instead, we employed our main data selection algorithm to choose $M = 130$ data points from the full dataset. As demonstrated in Table 11.1, this algorithm significantly reduced the execution time to an average of 33.2ms. Figure 11.5 displays the snapshots of selected online data points using our main algorithm along the walking gaits. We observe that throughout different walking cycles, the data selection follows a periodic pattern by consistently choosing similar data points at each walking phase.

As shown in Figure 11.4, the GP-CLF-SOCP (Ours) (green) is able to keep the RABBIT model's hip position at around a constant height without violating the implicit constraint on the contact force, which is imposed by a friction coefficient of 0.8. In other words, the controller enables the robot to successfully complete ten steps without a slip. This is further evidenced by the Control Lyapunov Function and tracking error plot, where the controller consistently and exponentially stabilizes the tracking error close to zero after the repeated state resets. It is worth noting that the resulting walking gait of the GP-CLF-SOCP controller differs from the oracle controller, as the SOCP controller chooses control inputs that are robust to the prediction uncertainty. Consequently, the controller behaves more conservatively; in this case, it leads to a slightly faster walking gait than the ideal scenario of using the oracle CLF-QP controller.

### 11.4.3 Hardware Experiment: Cart-pole System

The importance of the method presented in this work is most notable for real hardware systems, as we can use the data collected from the real system to account for the inevitable inaccuracies that even our best possible mathematical description of its dynamics might

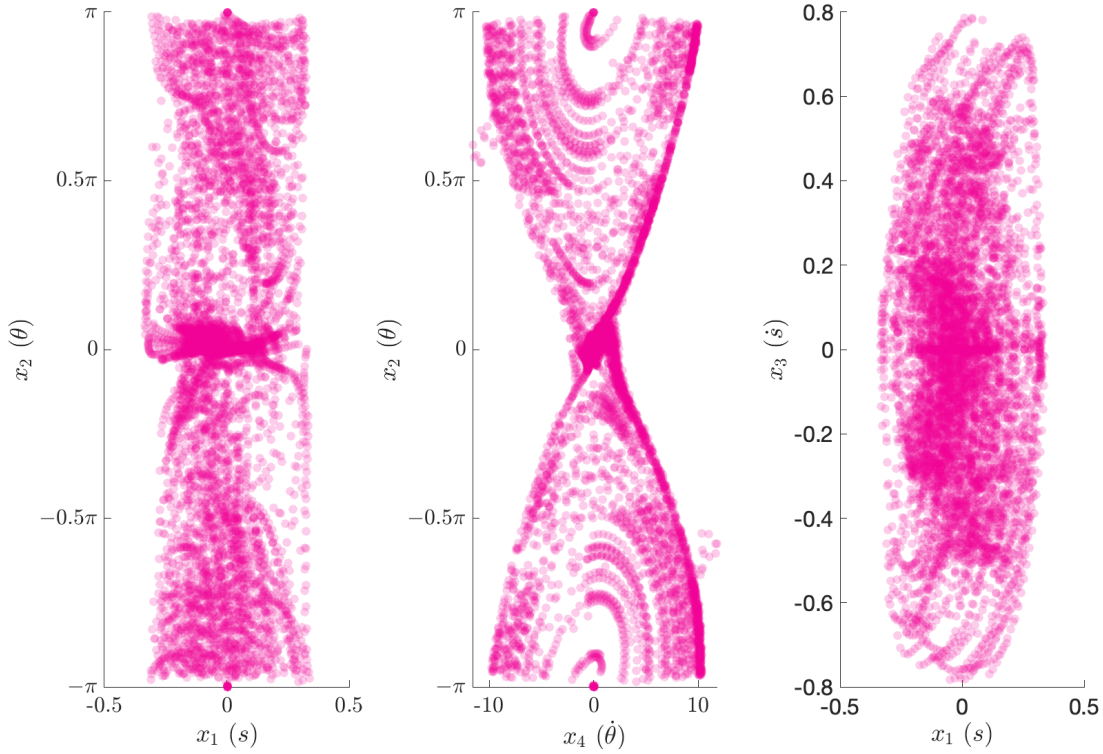



Figure 11.6: Training data for the cart-pole experiment collected from running 18 rollouts of the trajectories under the nominal CBF-QP and GP-CBF-SOCP from various initial states.

suffer from. This is precisely what is observed in the experiment we conducted on a Quanser Linear Servo Base Unit with Inverted Pendulum [161] hardware (Figure 11.3b). This cart-pole system consists of a linearly-actuated cart and an un-actuated pendulum. The state of the system can be described as $x = [s, \dot{s}, \theta, \dot{\theta}] \in \mathbb{R}^4$, where $s$ and $\dot{s}$ are the cart's position and velocity, and $\theta$ and $\dot{\theta}$ correspond to the pole's relative angle with respect to the upright position and its angular velocity. The control input $u \in \mathbb{R}$ is the voltage applied to the linear actuator of the cart.

The control objective of this experiment is to swing-up the pole to the upright position and balance it at the top, while respecting a safety constraint on the cart's position, given as $|s| \leq s_{\text{lim}} = 0.35\text{m}$. In particular, this constraint is placed to avoid the cart from colliding against the limits of the linear guide. The CBF we designed, which is then used as the certificate function, is based on the exponential CBF design methods for high relative-degree constraints [144]; in our case, the original cart position constraint has a relative degree of two. This results in a CBF expressed as

$$C(x) = -2s\dot{s} + k(s_{\text{lim}}^2 - s^2). \tag{11.29}$$

The zero-levelset of the CBF is depicted in red in the left plot of Figure 11.7.

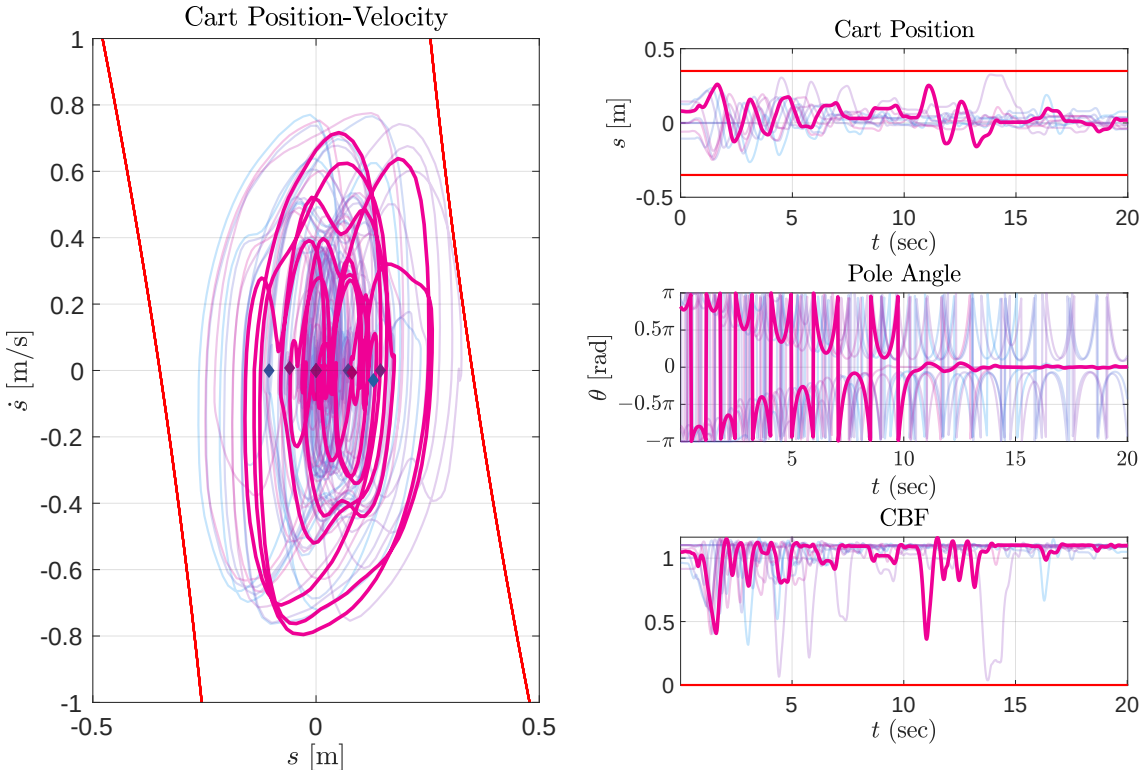



Figure 11.7: 10 episodes of the cart-pole experiment under GP-CBF-SOCP (Ours). We highlight one of the ten trajectories in magenta with thick curves and the rest in thin transparent curves. On the left is the phase plot of the trajectories in the cart position and velocity space $(s-\dot{s})$, where the region between the red curves indicates the zero-super level set of the CBF. The diamond markers indicate the initial states of the trajectories. No trajectory exits the zero-super level set of the CBF. On the right are the plots of cart position$(s)$, pole angle $(\theta)$, and CBF value of the trajectories in time. The highlighted trajectory successfully swings up the pole while maintaining the safety constraint. A video with the experimental results can be found in this link.

For the swing-up task, we design a reference policy $\pi_{\text{ref}}$, which is a hybrid controller that switches between an energy-based feedback controller that increases the total energy of the system until it matches the level of the potential energy of the unstable equilbrium, and a stabilizing controller to which the system switches when the pole is at the vicinity of the equilibrium.

We then apply the reference policy filtered by the nominal model-based CBF-QP certifying filter. For the nominal model, we use the high-fidelity dynamics model provided by the manufacturer [161]. We apply the computed filtered control input to the actual system every 25ms, which is the target sampling time for the real-time execution of the controller. Even though the provided dynamics model tries to capture many of the complex nonlinearities present in the system, we observe that, when deployed on the real system, the nominal model-based CBF-QP still fails to satisfy the safety constraints at several trials and the CBF values become negative.

This motivates us to employ the GP-CBF-SOCP certifying filter to achieve the swing-up task while adhering to the cart position limit, after learning the model uncertainty effect from the data. In order to maintain feasibility of the GP-CBF-SOCP filter, the dataset must sufficiently cover the state and control input space where the system operates. We collect these data points in an episodic fashion following the procedure described below. Initially, we run the first episode with the nominal CBF-QP and create an initial dataset for the GP regression. Subsequently, in the following episodes, we utilize the GP-CBF-SOCP, which uses the data collected in previous episodes, and aggregate the dataset with the data collected during the current episode. The GP-CBF-SOCP deployed during this process does not provide a robust safety guarantee since we do not yet have adequate data coverage. Consequently, the SOCP is often infeasible, resulting in a negative CBF value.

Furthermore, as more data points are aggregated, the GP inference takes longer, eventually exceeding the 25ms limit of our sampling time. Thus, we conduct this episodic procedure twice, each time collecting nine trajectories, and then combine the two datasets into the full dataset. With the full dataset comprising $N = 6957$ data points visualized in Figure 11.6, we observe that the GP-CBF-SOCP controller takes too long to perform the inference, causing an average 60.4ms execution time (Table 11.1), which does not meet the target sampling rate requirement. This effect is evident in the experiment, as the cart-pole fails to swing up properly due to the delay.

On the contrary, using our data selection algorithm with $M = 40$ points selected online, the total execution time becomes much smaller, resulting in an average of 11.8ms. Over 10 experiments using our main algorithm and the GP-CBF-SOCP, we achieve 100% constraint satisfaction. These trajectories are shown in Figure 11.7. Although not all of these experiments result in a successful balance at the upright position within the allocated 20 seconds (achieved in 6 out of 10 experiments), the GP-CBF-SOCP successfully prioritizes safety over performance, ensuring the cart never exits the defined limits imposed by the CBF-based certifying constraint. In the video showcasing the results, it is clear that the learned certifying constraint forces the cart to drop the pole when it approaches the position limit. Moreover, we demonstrate that even when an external user introduces disturbances by pushing the

pole, the system remains safe.

## 11.5 Chapter Summary

The primary innovation introduced in this chapter is a data selection algorithm, which operates online every time the SOCP safety filter is executed, to select the most relevant data points for ensuring the feasibility of the SOCP. These data points carry valuable information for determining the control input direction which guarantees system-critical constraint satisfaction, while avoiding redundancy among the chosen data points.

Most importantly, our proposed algorithm allows data-driven certifying filters to be applied to high-dimensional real robotic systems handling vast datasets. Previously proposed methods for implementing data-driven filters faced scalability limitations of non-parametric regression, restricting their usage to low-dimensional toy examples not requiring extensive datasets. Notably, the inference time complexity of GP regression increases quadratically with the dataset size. Our devised algorithm significantly enhances GP inference efficiency, reducing time complexity to linear in relation to the dataset size. We successfully demonstrate our method in a real cart-pole experiment, ensuring the cart position stays within the safety limits, and a 10-dimensional bipedal robot simulation attempting stable walking while subjected to parametric model errors.

# Chapter 12

# Conclusions and Future Vision

In conclusion, this dissertation presents a fundamental perspective on the development of reliable and intelligible controllers for real-world autonomous systems. Our research effectively explores the integration of model-based and data-driven approaches, taking advantage of approximate model knowledge when available and leveraging data to adapt to the intricacies of real-world systems. Through the introduction of several novel ideas, including principled reward shaping methods, distributional shift prevention mechanisms, uncertainty-aware model-based controllers, and safe active learning strategies, we address the challenges associated with uncertainty, safety, and adaptability in these systems.

The key contributions of this research not only enhance our understanding of autonomous systems but also provide practical, real-world applications in areas such as robotic locomotion and autonomous driving. Furthermore, the principles and methods presented herein can serve as building blocks for tackling more complex, uncertain, and dynamically changing environments, bringing us closer to realizing the full potential of robotic systems in real-world applications. Additionally, as advances in artificial intelligence span across numerous disciplines beyond robotics, the need to invest time and resources to minimize costly failures becomes increasingly important. We believe that the insights and techniques presented in this dissertation serve as valuable stepping stones for the much-needed continued research and advancement in this field.

Finally, we elaborate on some of the key technical challenges concerning safe autonomy that will require future research efforts.

### Scalable Data-Driven Safe Control Methods for Complex Multi-Agent Systems

While our research provides a foundation for future development and deployment of safer and more efficient autonomous systems, it is essential to recognize the limitations and complexities that arise in increasingly uncertain and dynamic settings. In multi-agent environments, it is crucial for autonomous systems to be able to reason online and react to changes in the behavior of other agents in order to stay safe.

Using online observations to constantly adapt to the changes in the environment seems

like the only way to effectively tackle this problem. However, highly expressive and scalable data-driven models are still not well-understood and providing meaningful uncertainty quantification techniques for these models should be a priority for future work.

### Uncertainty Propagation through Different Modules of the Software Stack

The prevalent approach to controlling autonomous systems involves modular pipelines that break down the control problem into several distinct modules, spanning from perception to decision making. This dissertation primarily focuses on developing principled methods for decision making. Nonetheless, it is evident that control and planning modules should take into account the uncertainties of the models employed in earlier layers of the stack to make robust decisions. As such, effective uncertainty quantification and propagation across all individual modules of the stack are essential for the safe deployment of these systems.

Considering that deep learning models offer state-of-the-art results in perception and prediction, quantifying uncertainty in these complex, overparameterized models and effectively propagating it to subsequent modules remains a significant research challenge.

### Safety of other Machine Learning Systems

As artificial intelligence systems continue to advance, they are unlocking a wider range of automation applications beyond just robotics. At the same time, concerns about privacy violations, algorithmic biases, and the unchecked spread of fake news have become more prevalent, especially with the recent rise of large language models. These concerns emphasize the importance of making machine learning systems reliable and transparent. Adapting research on physical safety to address broader constraint satisfaction problems could help these systems proactively evaluate the potential impact of their decisions when interacting with humans.

However, identifying appropriate constraints for these problems is challenging, which makes it difficult to even think about safety when considering these systems. Control-theoretic ideas should serve as a source of self-supervision for these models, helping to embed mathematical principles that guide the optimization process towards desired, but hard to analytically express, behaviors. Increased efforts should be devoted to the development of machine learning models that incorporate these principles and operate more securely and responsibly when employed for real-life interactions with humans.

# Bibliography


[1] A. Abate et al. "FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks". In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 2021, pp. 1–11.

[2] A. Agarwal et al. "Legged locomotion in challenging terrains using egocentric vision". In: *Conference on Robot Learning*. PMLR. 2023, pp. 403–415.

[3] A. K. Akametalu et al. "Reachability-based safe learning with Gaussian processes". In: *IEEE Conference on Decision and Control*. 2014, pp. 1424–1431.

[4] A. Alan et al. "Control Barrier Functions and Input-to-State Safety with Application to Automated Vehicles". In: *arXiv preprint arXiv:2206.03568* (2022).

[5] T. Alpcan and I. Shames. "An information-based learning approach to dual control". In: *IEEE transactions on neural networks and learning systems* 26.11 (2015), pp. 2736–2748.

[6] A. D. Ames, J. W. Grizzle, and P. Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 6271–6278.

[7] A. D. Ames and M. Powell. "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs". In: *Control of Cyber-Physical Systems*. Springer, 2013, pp. 219–240.

[8] A. D. Ames et al. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876.

[9] A. D. Ames et al. "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[10] N. Andréasson, A. Evgrafov, and M. Patriksson. *An introduction to continuous optimization: foundations and fundamental algorithms*. Courier Dover Publications, 2020.

[11] Z. Artstein. "Stabilization with relaxed controls". In: *Nonlinear Analysis: Theory, Methods & Applications* 7.11 (1983), pp. 1163–1173.

[12] S. Bansal et al. "Goal-driven dynamics learning via Bayesian optimization". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 5168–5173.

[13] S. Bansal et al. "Hamilton-jacobi reachability: A brief overview and recent advances". In: *2017 56th IEEE Conference on Decision and Control (CDC)*. 2017.

[14] S. Battilotti. "Robust stabilization of nonlinear systems with pointwise norm-bounded uncertainties: a control Lyapunov function approach". In: *IEEE Transactions on Automatic Control* 44.1 (1999), pp. 3–17.

[15] G. Beintema, R. Toth, and M. Schoukens. "Nonlinear state-space identification using deep encoder networks". In: *Learning for Dynamics and Control*. PMLR. 2021, pp. 241–250.

[16] S. Belkhale et al. "Model-based meta-reinforcement learning for flight with suspended payloads". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1471–1478.

[17] R. Bellman. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[18] F. Berkenkamp et al. "Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes". In: *IEEE Conference on Decision and Control*. 2016, pp. 4661–4666.

[19] F. Berkenkamp et al. "Safe Model-based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 908–918.

[20] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[21] M. Black and D. Panagou. "Adaptation for Validation of a Consolidated Control Barrier Function based Control Synthesis". In: *arXiv preprint arXiv:2209.08170* (2022).

[22] Boston Dynamics. *Spot*. URL: `https://www.bostondynamics.com/products/spot`.

[23] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[24] L. Brunke, S. Zhou, and A. P. Schoellig. "Barrier Bayesian Linear Regression: Online Learning of Control Barrier Conditions for Safety-Critical Control of Uncertain Systems". In: *Learning for Dynamics and Control*. 2022, pp. 881–892.

[25] J. Buch, S.-C. Liao, and P. Seiler. "Robust control barrier functions with sector-bounded uncertainties". In: *IEEE Control Systems Letters* 6 (2021), pp. 1994–1999.

[26] A. Capone et al. "Data selection for multi-task learning under dynamic constraints". In: *IEEE Control Systems Letters* 5.3 (2020), pp. 959–964.

[27] F. Castañeda et al. "Gaussian Process-based Min-norm Stabilizing Controller for Control-Affine Systems with Uncertain Input Effects and Dynamics". In: *2021 American Control Conference (ACC)*. 2021, pp. 3683–3690.

[28] F. Castañeda et al. "Improving Input-Output Linearizing Controllers for Bipedal Robots via Reinforcement Learning". In: *Learning for Dynamics and Control*. Berkeley, CA, June 2020, pp. 990–999.

[29] F. Castañeda et al. "In-Distribution Barrier Functions: Self-Supervised Policy Filters that Avoid Out-of-Distribution States". In: *Learning for Dynamics and Control*. Philadelphia, PA, June 2023.

[30] F. Castañeda et al. "Pointwise Feasibility of Gaussian Process-based Safety-Critical Control under Model Uncertainty". In: *IEEE Conference on Decision and Control*. 2021, pp. 6762–6769.

[31] F. Castañeda et al. "Recursively feasible probabilistic safe online learning with control barrier functions". In: *arXiv preprint arXiv:2208.10733* (2022).

[32] Y.-C. Chang, N. Roohi, and S. Gao. "Neural lyapunov control". In: *Advances in neural information processing systems* 32 (2019).

[33] G. Chen et al. "A review of lower extremity assistive robotic exoskeletons in rehabilitation therapy". In: *Critical Reviews™ in Biomedical Engineering* 41.4-5 (2013).

[34] R. T. Chen et al. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018).

[35] C.-A. Cheng, A. Kolobov, and A. Swaminathan. "Heuristic-guided reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13550–13563.

[36] R. Cheng et al. "Safe Multi-Agent Interaction through Robust Control Barrier Functions with Learned Uncertainties". In: *IEEE Conference on Decision and Control*. 2020, pp. 777–783.

[37] C. Chevallereau et al. "Rabbit: A testbed for advanced control theory". In: *IEEE Control Systems Magazine* 23.5 (2003), pp. 57–79.

[38] J. Choi et al. "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions". In: *Robotics: Science and Systems*. Corvalis, OR, 2020.

[39] J. J. Choi et al. "Robust control barrier–value functions for safety-critical control". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 6814–6821.

[40] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 539–546.

[41] S. R. Chowdhury and A. Gopalan. "On Kernelized Multi-Armed Bandits". In: *International Conference on Machine Learning*. Sydney, NSW, Australia, 2017, pp. 844–853.

[42] K. Chua et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models". In: *Advances in neural information processing systems* 31 (2018).

[43] M. H. Cohen and C. Belta. "Safe Exploration in Model-based Reinforcement Learning using Control Barrier Functions". In: *arXiv preprint arXiv:2104.08171* (2021).

[44] E. Contal, V. Perchet, and N. Vayatis. "Gaussian process optimization with mutual information". In: *International Conference on Machine Learning*. PMLR. 2014, pp. 253–261.

[45] R. K. Cosner et al. "Self-Supervised Online Learning for Safety-Critical Control using Stereo Vision". In: *arXiv preprint arXiv:2203.01404* (2022).

[46] E. Coumans and Y. Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org`. 2019.

[47] J. J. Craig, P. Hsu, and S. S. Sastry. "Adaptive control of mechanical manipulators". In: *The International Journal of Robotics Research* 6.2 (1987), pp. 16–28.

[48] C. Currin et al. "Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments". In: *Journal of the American Statistical Association* 86.416 (1991), pp. 953–963.

[49] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314.

[50] X. Da et al. "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion". In: *Conference on Robot Learning*. PMLR. 2021, pp. 883–894.

[51] H. Dai and F. Permenter. "Convex synthesis and verification of control-Lyapunov and barrier functions with input constraints". In: *arXiv preprint arXiv:2210.00629* (2022).

[52] M. Daszykowski, B. Walczak, and D. Massart. "Representative subset selection". In: *Analytica chimica acta* 468.1 (2002), pp. 91–103.

[53] C. Dawson, S. Gao, and C. Fan. "Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control". In: *IEEE Transactions on Robotics* (2023), pp. 1–19.

[54] C. Dawson et al. "Safe nonlinear control using robust neural lyapunov-barrier functions". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1724–1735.

[55] S. Dean et al. "Guaranteeing Safety of Learned Perception Modules via Measurement-Robust Control Barrier Functions". In: *Conference on Robot Learning*. 2021.

[56] V. Dhiman et al. "Control barriers in bayesian learning of system dynamics". In: *IEEE Transactions on Automatic Control* (2021).

[57] F. Ebert et al. "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control". In: *arXiv preprint arXiv:1812.00568* (2018).

[58] D. D. Fan et al. "Bayesian Learning-Based Adaptive Control for Safety Critical Systems". In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 4093–4099.

[59] C. Fiedler, C. W. Scherer, and S. Trimpe. "Practical and rigorous uncertainty bounds for gaussian process regression". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 8. 2021, pp. 7439–7447.

[60] J. F. Fisac et al. "A general safety framework for learning-based control in uncertain robotic systems". In: *IEEE Transactions on Automatic Control* 64.7 (2018), pp. 2737–2752.

[61] V. François-Lavet, R. Fonteneau, and D. Ernst. "How to discount deep reinforcement learning: Towards new dynamic strategies". In: *arXiv preprint arXiv:1512.02011* (2015).

[62] R. Freeman and P. V. Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science and Business Media, 2008.

[63] S. Frennert and B. Östlund. "The domestication of robotic vacuum cleaners among seniors". In: *Gerontechnology* 3.12 (2015), pp. 159–168.

[64] V. Gaitsgory, L. Grüne, and N. Thatcher. "Stabilization with discounted optimal control". In: *Systems and Control Letters* 82 (2015), pp. 91–98.

[65] K. Galloway et al. "Torque Saturation in Bipedal Robotic Walking Through Control Lyapunov Function-Based Quadratic Programs". In: *IEEE Access* 3 (2015), pp. 323–332.

[66] M. Goslin and M. R. Mine. "The Panda3D graphics engine". In: *Computer* 37.10 (2004), pp. 112–114.

[67] R. B. Gramacy and D. W. Apley. "Local Gaussian process approximation for large computer experiments". In: *Journal of Computational and Graphical Statistics* 24.2 (2015), pp. 561–578.

[68] M. Greeff, A. W. Hall, and A. P. Schoellig. "Learning a Stability Filter for Uncertain Differentially Flat Systems using Gaussian Processes". In: *IEEE Conference on Decision and Control*. 2021, pp. 789–794.

[69] G. Grimm et al. "Examples when nonlinear model predictive control is nonrobust". In: *Automatica* 40.10 (2004), pp. 1729–1738.

[70] G. Grimm et al. "Model predictive control: for want of a local control Lyapunov function, all is not lost". In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 546–558.

[71] J. W. Grizzle, G. Abba, and F. Plestan. "Asymptotically stable walking for biped robots: Analysis via systems with impulse effects". In: *IEEE Transactions on automatic control* 46.1 (2001), pp. 51–64.

[72] J. W. Grizzle et al. "Models, feedback control, and open problems of 3D bipedal robotic walking". In: *Automatica* 50.8 (2014), pp. 1955–1988.

[73] S. Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3389–3396.

[74] X. Guo and Y. Zhang. "Maturity in automated driving on public roads: a review of the six-year autonomous vehicle tester program". In: *Transportation research record* 2676.11 (2022), pp. 352–362.

[75] M. Gutmann and A. Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304.

[76] M. U. Gutmann and J.-i. Hirayama. "Bregman divergence as general framework to estimate unnormalized statistical models". In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. 2011, pp. 283–290.

[77] T. Haarnoja et al. "Learning to walk via deep reinforcement learning". In: *arXiv preprint arXiv:1812.11103* (2018).

[78] T. Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). eprint: `1801.01290`.

[79] D. Hafner et al. "Learning latent dynamics for planning from pixels". In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.

[80] M. Hein and O. Bousquet. "Hilbertian metrics and positive definite kernels on probability measures". In: *International Workshop on Artificial Intelligence and Statistics*. PMLR. 2005, pp. 136–143.

[81] A. Hereid and A. D. Ames. "FROST: Fast robot optimization and simulation toolkit". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 719–726.

[82] J. P. Hespanha, D. Liberzon, and A. R. Teel. "Lyapunov conditions for input-to-state stability of impulsive systems". In: *Automatica* 44.11 (2008), pp. 2735–2744.

[83] G. E. Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[84] R. A. Horn. "The hadamard product". In: *Proceedings of Symposia in Applied Mathematics*. Vol. 40. 1990, pp. 87–169.

[85] K. Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

[86] J. Hwangbo et al. "Control of a Quadrotor With Reinforcement Learning". In: *Robotics and Auto. Letters* 2.4 (Oct. 2017), pp. 2096–2103.

[87] J. Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (2019).

[88] A. Jadbabaie and J. Hauser. "On the stability of receding horizon control with a general terminal cost". In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 674–678.

[89] A. Jadbabaie, J. Yu, and J. Hauser. "Receding horizon control of the Caltech ducted fan: A control Lyapunov function approach". In: *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*. Vol. 1. IEEE. 1999, pp. 51–56.

[90] A. Jadbabaie, J. Yu, and J. Hauser. "Unconstrained receding-horizon control of nonlinear systems". In: *IEEE Transactions on Automatic Control* 46.5 (2001), pp. 776–783.

[91] P. Jagtap, G. J. Pappas, and M. Zamani. "Control barrier functions for unknown nonlinear systems using gaussian processes". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 3699–3704.

[92] M. Jankovic. "Robust control barrier functions for constrained stabilization of nonlinear systems". In: *Automatica* 96 (2018), pp. 359–367.

[93] Z. Jarvis-Wloszek et al. "Some controls applications of sum of squares programming". In: *42nd IEEE international conference on decision and control*. Vol. 5. IEEE. 2003, pp. 4676–4681.

[94] N. Jiang et al. "The dependence of effective planning horizon on model accuracy". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. Citeseer. 2015, pp. 1181–1189.

[95] W. Jin et al. "Neural certificates for safe control policies". In: *arXiv preprint arXiv:2006.08465* (2020).

[96] R. Julian et al. "Efficient adaptation for end-to-end vision-based robotic manipulation". In: (2020).

[97] R. Julian et al. "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning". In: *arXiv preprint arXiv:2004.10190* (2020).

[98] G. Kahn et al. "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5129–5136.

[99] K. Kang et al. "Lyapunov density models: Constraining distribution shift in learning-based control". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 10708–10733.

[100] K. Kazemian and S. Dean. "Random Features Approximation for Fast Data-Driven Control". In: *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*. Dec. 2023.

[101] C. M. Kellett and A. R. Teel. "Results on discrete-time control-Lyapunov functions". In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*. Vol. 6. IEEE. 2003, pp. 5961–5966.

[102] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*. Vol. 3. Prentice Hall, Upper Saddle River, NJ, 2002.

[103] C. Khazoom et al. "Humanoid Self-Collision Avoidance Using Whole-Body Control with Control Barrier Functions". In: *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE. 2022, pp. 558–565.

[104] M. J. Khojasteh et al. "Probabilistic safety constraints for learned high relative degree system dynamics". In: *Learning for Dynamics and Control*. 2020, pp. 781–792.

[105] H. Kim et al. "Emi: Exploration with mutual information". In: *arXiv preprint arXiv:1810.01176* (2018).

[106] C.-W. Ko, J. Lee, and M. Queyranne. "An exact algorithm for maximum entropy sampling". In: *Operations Research* 43.4 (1995), pp. 684–691.

[107] S. Kolathaya and A. D. Ames. "Input-to-state safety with control barrier functions". In: *IEEE control systems letters* 3.1 (2018), pp. 108–113.

[108] T. Koller et al. "Learning-based model predictive control for safe exploration". In: *2018 IEEE conference on decision and control (CDC)*. IEEE. 2018, pp. 6059–6066.

[109] A. Krause. "Optimizing sensing: Theory and applications". PhD thesis. Carnegie Mellon University, 2008.

[110] A. Krause, A. Singh, and C. Guestrin. "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies." In: *Journal of Machine Learning Research* 9.2 (2008).

[111] M. Krstic. "Inverse Optimal Safety Filters". In: *arXiv preprint arXiv:2112.08225* (2021).

[112] A. Kumar et al. "Rma: Rapid motor adaptation for legged robots". In: *arXiv preprint arXiv:2107.04034* (2021).

[113] A. Kumar et al. "Stabilizing off-policy q-learning via bootstrapping error reduction". In: *Advances in Neural Information Processing Systems* 32 (2019).

[114] A. Lederer, J. Umlauft, and S. Hirche. "Uniform error bounds for gaussian process regression with application to safe control". In: *Advances in Neural Information Processing Systems* 32 (2019).

[115] A. Lederer et al. "How training data impacts performance in learning-based control". In: *IEEE Control Systems Letters* 5.3 (2020), pp. 905–910.

[116] A. Lederer et al. "The Impact of Data on the Stability of Learning-Based Control". In: *Learning for Dynamics and Control*. 2021, pp. 623–635.

[117] J. Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science robotics* 5.47 (2020).

[118] I. Lenz, R. A. Knepper, and A. Saxena. "DeepMPC: Learning deep latent features for model predictive control." In: *Robotics: Science and Systems*. Vol. 10. Rome, Italy. 2015.

[119] S. Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[120] T. Lew et al. "On the problem of reformulating systems with uncertain dynamics as a stochastic differential equation". In: *arXiv preprint arXiv:2111.06084* (2021).

[121] F. L. Lewis and D. Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control". In: *IEEE circuits and systems magazine* 9.3 (2009), pp. 32–50.

[122] Z. Li et al. "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots". In: *arXiv preprint arXiv:2103.14295* (2021).

[123] Y. Lin and E. D. Sontag. "Control-lyapunov universal formulas for restricted inputs". In: *Control-Theory and Advanced Technology* 10 (1995), pp. 1981–2004.

[124] B. Lincoln and A. Rantzer. "Relaxing dynamic programming". In: *IEEE Transactions on Automatic Control* 51.8 (2006), pp. 1249–1260.

[125] L. Lindemann et al. "Learning robust output control barrier functions from safe expert demonstrations". In: *arXiv preprint arXiv:2111.09971* (2021).

[126] C. Liu and M. Tomizuka. "Control in a safe set: Addressing safety in human-robot interactions". In: *Dynamic Systems and Control Conference*. Vol. 46209. American Society of Mechanical Engineers. 2014.

[127] C. Liu et al. "Algorithms for verifying deep neural networks". In: *Foundations and Trends in Optimization* 4.3-4 (2021), pp. 244–404.

[128] H. Liu et al. "When Gaussian process meets big data: A review of scalable GPs". In: *IEEE transactions on neural networks and learning systems* 31.11 (2020), pp. 4405–4423.

[129] K. Long et al. "Learning barrier functions with memory for robust safe navigation". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4931–4938.

[130] B. T. Lopez, J.-J. E. Slotine, and J. P. How. "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety". In: *IEEE Control Systems Letters* 5.3 (2020), pp. 1031–1036.

[131] J. Lygeros et al. "Dynamical properties of hybrid automata". In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 2–17.

[132] A. Majumdar, A. A. Ahmadi, and R. Tedrake. "Control design along trajectories with sums of squares programming". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 4054–4061.

[133] Z. Mandi, P. Abbeel, and S. James. "On the Effectiveness of Fine-tuning Versus Meta-reinforcement Learning". In: *arXiv preprint arXiv:2206.03271* (2022).

[134] H. Mania, A. Guy, and B. Recht. "Simple random search provides a competitive approach to reinforcement learning". In: *arXiv preprint arXiv:1803.07055* (2018).

[135] A. Marco et al. "Automatic LQR tuning based on Gaussian process global optimization". In: *2016 IEEE International conference on robotics and automation (ICRA)*. 2016, pp. 270–277.

[136] R. McAllister et al. "Robustness to out-of-distribution inputs via task-aware generative uncertainty". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2083–2089.

[137] V. Mendez et al. "Current solutions and future trends for robotic prosthetic hands". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021), pp. 595–627.

[138] C. D. Meyer. *Matrix analysis and applied linear algebra*. Vol. 71. Siam, 2000.

[139] T. G. Molnar and A. D. Ames. "Safety-Critical Control with Bounded Inputs via Reduced Order Models". In: *arXiv preprint arXiv:2303.03247* (2023).

[140] B. Morris and J. W. Grizzle. "A restricted Poincaré map for determining exponentially stable periodic orbits in systems with impulse effects: Application to bipedal robots". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE. 2005, pp. 4199–4206.

[141] R. Munos and C. Szepesvári. "Finite-Time Bounds for Fitted Value Iteration." In: *Journal of Machine Learning Research* 9.5 (2008).

[142] A. Nagabandi et al. "Deep dynamics models for learning dexterous manipulation". In: *Conference on Robot Learning*. PMLR. 2020, pp. 1101–1112.

[143] A. Y. Ng, D. Harada, and S. Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *International conference on machine learning*. Vol. 99. 1999, pp. 278–287.

[144] Q. Nguyen and K. Sreenath. "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints". In: *American Control Conference*. 2016, pp. 322–328.

[145] Q. Nguyen and K. Sreenath. "L1 Adaptive Control for Bipedal Robots with Control Lyapunov Function based Quadratic Programs". In: *American Control Conference*. Chicago, IL, July 2015, pp. 862–867.

[146] Q. Nguyen and K. Sreenath. "Optimal Robust Control for Bipedal Robots through Control Lyapunov Function based Quadratic Programs." In: *Robotics: Science and Systems*. Rome, Italy. 2015.

[147] Q. Nguyen and K. Sreenath. "Robust Safety-Critical Control for Dynamic Robotics". In: *IEEE Transactions on Automatic Control* 67.3 (2022), pp. 1073–1088.

[148] Q. Nguyen and K. Sreenath. "Safety-Critical Control for Dynamical Bipedal Walking with Precise Footstep Placement". In: *IFAC Analysis and Design of Hybrid Systems*. Atlanta, GA, Oct. 2015.

[149] Q. T. Nguyen. "Robust and Adaptive Dynamic Walking of Bipedal Robots". PhD thesis. Carnegie Mellon University, 2017.

[150] P. Ogren, M. Egerstedt, and X. Hu. "A control Lyapunov function approach to multi-agent coordination". In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*. Vol. 2. 2001, 1150–1155 vol.2.

[151] A. v. d. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[152] X. B. Peng et al. "Learning agile robotic locomotion skills by imitating animals". In: *arXiv preprint arXiv:2004.00784* (2020).

[153] X. B. Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[154] M. Petrik and B. Scherrer. "Biasing approximate dynamic programming with a lower discount factor". In: *Advances in neural information processing systems* 21 (2008), pp. 1265–1272.

[155] J. M. Phillips and S. Venkatasubramanian. "A gentle introduction to the kernel distance". In: *arXiv preprint arXiv:1103.1625* (2011).

[156] R. Postoyan et al. "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost". In: *IEEE Transactions on Automatic Control* 62.6 (2016), pp. 2736–2749.

[157] R. Postoyan et al. "Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 487–492.

[158] Public Utilities Commision of the State of California. *Resolution Approving Cruise Llc's Application For Phase I Driverless Autonomous Vehicle Passenger Service Deployment Program*. 2022.

[159] F. Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

[160] Z. Qin, D. Sun, and C. Fan. "SABLAS: Learning Safe Control for Black-box Dynamical Systems". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1928–1935.

[161] Quanser. *Linear servo base unit with inverted pendulum*. Apr. 2021. URL: `https://www.quanser.com/products/linear-servo-base-unit-inverted-pendulum/`.

[162] J. Quinonero-Candela and C. E. Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[163] H. Ravanbakhsh and S. Sankaranarayanan. "Learning Control Lyapunov Functions from Counterexamples and Demonstrations". In: *Autonomous Robots* 43.2 (Feb. 2019), pp. 275–307.

[164] J. Reher, C. Kann, and A. D. Ames. "An Inverse Dynamics Approach to Control Lyapunov Functions". In: *American Control Conference*. 2020.

[165] S. M. Richards, F. Berkenkamp, and A. Krause. "The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems". In: *Proceedings of The 2nd Conference on Robot Learning*. Vol. 87. Proceedings of Machine Learning Research. Oct. 2018, pp. 466–476.

[166] C. Richter and N. Roy. "Safe visual navigation via deep learning and novelty detection". In: *Robotics: Science and Systems*. Cambridge, MA. 2017.

[167] A. Robey et al. "Learning control barrier functions from expert demonstrations". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 3717–3724.

[168] R. M. Sanner and J.-J. Slotine. "Gaussian networks for direct adaptive control". In: *IEEE Transactions on Neural Networks* 3.6 (1992), pp. 837–863.

[169] S. Sastry. *Nonlinear systems: analysis, stability, and control*. Vol. 10. Springer Science & Business Media, 1999.

[170] S. Sastry and M. Bodson. *Adaptive control: stability, convergence and robustness*. Courier Corporation, 1989.

[171] S. S. Sastry and A. Isidori. "Adaptive control of linearizable systems". In: *IEEE Transactions on Auto. Control* 34.11 (1989), pp. 1123–1131.

[172] F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[173] J. Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[174] J. Siekmann et al. "Sim-to-real learning of all common bipedal gaits via periodic reward composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7309–7315.

[175] D. Silver et al. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. Proceedings of Machine Learning Research. 2014, pp. 387–395.

[176] L. Smith et al. "Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World". In: *arXiv preprint arXiv:2110.05457* (2021).

[177] E. D. Sontag. "A 'universal' construction of Artstein's theorem on nonlinear stabilization". In: *Systems and Control Letters* 13.2 (1989), pp. 117–123.

[178] E. D. Sontag. "On the Input-to-State Stability Property". In: *European Journal of Control* 1.1 (1995), pp. 24–36.

[179] K. Sreenath et al. "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL". In: *The International Journal of Robotics Research* 30.9 (2011), pp. 1170–1193.

[180] N. Srinivas et al. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design". In: *International Conference on Machine Learning*. Haifa, Israel, 2010.

[181] M. Srinivasan et al. "Synthesis of control barrier functions using a supervised machine learning approach". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 7139–7145.

[182] G. Still. "Lectures on parametric optimization: An introduction". In: *Optimization Online* (2018).

[183] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[184] A. J. Taylor and A. D. Ames. "Adaptive safety with control barrier functions". In: *American Control Conference*. 2020, pp. 1399–1405.

[185] A. J. Taylor et al. "A control lyapunov perspective on episodic learning via projection to state stability". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 1448–1455.

[186] A. J. Taylor et al. "Episodic learning with control lyapunov functions for uncertain robotic systems". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6878–6884.

[187] A. J. Taylor et al. "Learning for safety-critical control with control barrier functions". In: *Learning for Dynamics and Control*. 2020, pp. 708–717.

[188] A. J. Taylor et al. "Towards Robust Data-Driven Control Synthesis for Nonlinear Systems with Actuation Uncertainty". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 6469–6476.

[189] C. Tessler and S. Mannor. "Reward Tweaking: Maximizing the Total Reward While Planning for Short Horizons". In: *arXiv preprint arXiv:2002.03327* (2020).

[190] V. Tresp. "A Bayesian committee machine". In: *Neural computation* 12.11 (2000), pp. 2719–2741.

[191] J. Umlauft, L. Pöhler, and S. Hirche. "An Uncertainty-Based Control Lyapunov Approach for Control-Affine Systems Modeled by Gaussian Process". In: *IEEE Control Systems Letters* 2 (2018), pp. 483–488.

[192] J. Umlauft et al. "Feedback linearization using Gaussian processes". In: *IEEE Conference on Decision and Control*. 2017, pp. 5249–5255.

[193] J. Umlauft and S. Hirche. "Feedback linearization based on Gaussian processes with event-triggered online learning". In: *IEEE Transactions on Automatic Control* 65.10 (2019), pp. 4154–4169.

[194] J. Umlauft et al. "Smart forgetting for safe online learning with Gaussian processes". In: *Learning for Dynamics and Control*. 2020, pp. 160–169.

[195] Unitree Robotics. *A1*. URL: `https://www.unitree.com/products/a1/`.

[196] R. Urtasun and T. Darrell. "Sparse probabilistic regression for activity-independent human pose inference". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[197] H. Van Hoof et al. "Stable reinforcement learning with autoencoders for tactile and visual data". In: *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2016, pp. 3928–3934.

[198] M. Verhaegen and V. Verdult. *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[199] K. P. Wabersich et al. "Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems". In: *Preprint* (2023).

[200] K. P. Wabersich and M. N. Zeilinger. "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems". In: *Automatica* 129 (2021), p. 109597.

[201] H. Wang, K. Margellos, and A. Papachristodoulou. "Safety Verification and Controller Synthesis for Systems with Input Constraints". In: *arXiv preprint arXiv:2204.09386* (2022).

[202] M. Watter et al. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems* 28 (2015).

[203] K. Wei, R. Iyer, and J. Bilmes. "Submodularity in data subset selection and active learning". In: *International conference on machine learning*. PMLR. 2015, pp. 1954–1963.

[204] T. Wei and C. Liu. "Safe Control Algorithms Using Energy Functions: A Unified Framework, Benchmark, and New Directions". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 238–243.

[205] T. Wei et al. "Persistently Feasible Robust Safe Control by Safety Index Synthesis and Convex Semi-Infinite Programming". In: *IEEE Control Systems Letters* 7 (2023), pp. 1213–1218.

[206] K. Q. Weinberger and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification." In: *Journal of machine learning research* 10.2 (2009).

[207] H. Wendland. *Scattered data approximation*. Vol. 17. Cambridge university press, 2004.

[208] T. Westenbroek et al. "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers". In: *IFAC-PapersOnLine* 54.5 (2021), pp. 19–24.

[209] T. Westenbroek et al. "Feedback linearization for uncertain systems via reinforcement learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1364–1371.

[210] T. Westenbroek et al. "Learning min-norm stabilizing control laws for systems with unknown dynamics". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 737–744.

[211] T. Westenbroek et al. "Lyapunov Design for Robust and Efficient Robotic Reinforcement Learning". In: *6th Annual Conference on Robot Learning*. PMLR. 2022.

[212] E. Westervelt. "Toward a Coherent Framework for the Control of Planar Biped Locomotion". PhD thesis. University of Michigan, 2003.

[213] E. R. Westervelt et al. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007.

[214] E. Westervelt, J. Grizzle, and D. Koditschek. "Zero dynamics of underactuated planar biped walkers". In: *IFAC Proceedings Volumes* 35.1 (2002), pp. 551–556.

[215] A. Wilcox et al. "LS3: Latent space safe sets for long-horizon visuomotor control of sparse reward iterative tasks". In: *Conference on Robot Learning*. PMLR. 2022, pp. 959–969.

[216] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press, Cambridge, MA, 2006.

[217] G. Wu and K. Sreenath. "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 2038–2044.

[218] Y. Wu, G. Tucker, and O. Nachum. "Behavior regularized offline reinforcement learning". In: *arXiv preprint arXiv:1911.11361* (2019).

[219] X. Xu et al. "Robustness of control barrier functions for safety critical control". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 54–61.

[220] K. Yu, L. Ji, and X. Zhang. "Kernel nearest-neighbor algorithm". In: *Neural Processing Letters* 15.2 (2002), pp. 147–156.

[221] F. Zhang. *The Schur complement and its applications*. Vol. 4. Springer Science & Business Media, 2006.

[222] M. Zhang et al. "Solar: Deep structured representations for model-based reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7444–7453.

[223] L. Zheng et al. "Learning-Based Safety-Stability-Driven Control for Safety-Critical Systems under Model Uncertainties". In: *International Conference on Wireless Communications and Signal Processing*. 2020, pp. 1112–1118.

# Appendix A

# Proofs and Intermediate Results

## A.1 Chapter 3 Proofs

### A.1.1 Proof of Lemma 3.1

To prove the desired result, we demonstrate that for each $\theta^* \in \Theta \setminus \Xi$ there exists a finite $\bar{\rho} \in \mathbb{R}_+$ such that $\theta^* \notin S_\rho$ for each $\rho > \bar{\rho}$. For a fixed $\theta^* \in \Theta \setminus \Xi$, define $M_1^{\theta^*} = E_{x\sim X}[\|\hat{\pi}(x,\theta^*)\|_2^2]$ and $M_2^{\theta^*} = E_{x\sim X}[[\Psi(x,\theta^*)]^+]$ so that for each $\rho > 0$ we have $L_\rho(\theta^*) = M_1^{\theta^*} + \rho M_2^{\theta^*}$. Since $\theta^* \notin \Xi$, there must exist $x^* \in \Omega_c$ such that $[\Psi(x^*,\theta^*)]^+ > 0$. Under our standing assumptions, the map $[\Psi(\cdot,\theta^*)]^+$ can be seen to be continuous, since the space of continuous functions is closed under addition, multiplication and composition. Putting these two facts together, there must exist a $\delta > 0$ such that for each $x \in B^\delta(x^*) \cap \Omega_c$ we have $[\Psi(x,\theta^*)]^+ > 0$. This in turn implies that $M_2^{\theta^*} > 0$. Thus, we see that $L_\rho(\theta^*) \to \infty$ as $\rho \to \infty$.

Next, letting $\bar{\theta}$ be defined as in the statement of the lemma, for each $\rho \in \mathbb{R}_+$ we have $L_\rho(\bar{\theta}) = M_1^{\bar{\theta}}$ where $M_1^{\bar{\theta}} = E_{x\sim X}[\|\hat{\pi}(x,\bar{\theta})\|_2^2]$ and we note that the term $E_{x\sim X}[[\Psi(x,\theta^*)]^+]$ contributes nothing to $L_\rho(\bar{\theta})$ since $\bar{\theta} \in \Xi$. Thus, if we set $\bar{\rho} = \max\left\{0, \frac{M_1^{\bar{\theta}} - M_1^{\theta^*}}{M_2^{\theta^*}}\right\}$ we see that $L_\rho(\theta^*) > L_\rho(\bar{\theta})$ for each $\rho > \bar{\rho}$, proving the desired statement for our fixed $\theta^*$. □

### A.1.2 Proof of Theorem 3.1

Let $\bar{\rho}$ be defined as in the statement of Lemma 3.1. Then for each $\rho > \bar{\rho}$ we have $S_\rho \subset \Xi$, where $\Xi$ is defined as in (3.9). This implies that for each $\theta \in S_\rho$ we have $L(\theta) = E_{x\sim X}[\|\hat{\pi}(x,\theta)\|_2^2]$. Let $\bar{\theta}$ be defined as in the statement of the theorem, and let $\theta \in S_\rho$ be arbitrary. By the definition of the min-norm control law we have $\|\hat{\pi}(x,\bar{\theta})\|_2 \leq \|\hat{\pi}(x,\theta)\|_2$ for each $x \in \Omega_c$, which in turn implies that $L(\bar{\theta}) \leq L(\theta)$. Next, suppose that $\hat{\pi}(x^*,\theta) \neq \pi_{\text{CLF}}(x^*)$ for some $x^* \in \Omega_c$. Again, using the definition of $u_p^*$ we have $\|\hat{\pi}(x^*,\bar{\theta})\|_2 < \|\hat{\pi}(x^*,\theta)\|_2$. By the continuity of $\hat{\pi}(\cdot,\theta)$, we know that there exists $\delta > 0$ such that for each $x \in B^\delta(x^*) \cap \Omega_c$ we have $\|\hat{\pi}(x,\bar{\theta})\|_2^2 < \|\hat{\pi}(x,\theta)\|_2^2$. This implies that $L(\bar{\theta}) < L(\theta)$, demonstrating the desired result. □

### A.1.3 Proof of Lemma 3.2

To prove the claim, we will first consider the two maps $\theta \to E_{x\sim X}\left[\|\hat{\pi}(x,\theta)\|_2^2\right]$ and $\theta \to E_{x\sim X}\left[\rho\left[\Psi(x,\theta)\right]^+\right]$ separately. In particular, we will show that the first term is strongly convex in $\theta$ while the second term is simply convex. The result of the theorem then follows from the fact that the addition of a strongly convex function and a convex function yields a strongly convex function.

First, we rewrite $\|\hat{\pi}(x,\theta)\|_2^2$ as $\theta^T W(x)^T W(x)\theta$ where $W(x) = [u_1(x), u_2(x), \ldots, u_K(x)]^T$ collects the basis of control functions. Note that the positive semi-definite matrix $\bar{W} = E_{x\sim X}\left[W(x)^T W(x)\right]$ is the Grammian for $\{\pi_k\}_{k=1}^K$ on $C(\Omega_c, \mathbb{R}^m)$, and thus will be full-rank and positive definite iff $\{\pi_k\}_k^K$ is linearly independent on this space. Based on these facts, we see that $E_{x\sim X}\left[\|\hat{\pi}(x,\theta)\|_2^2\right] = \theta^T \bar{W}\theta$ is a strongly convex quadratic function of the parameters.

Next, we turn to the term $E_{x\sim X}\left[\rho\left[\Psi(x,\theta)\right]^+\right]$. We demonstrate that for a fixed $x^* \in \Omega_c$ and each $\rho \in \mathbb{R}_+$ the mapping $\theta \to \|\hat{\pi}(x^*,\theta)\|_2^2 + \rho\left[\Psi(x^*,\theta)\right]^+$ is strongly convex using basic properties of convex functions [23]. We begin by examining the term $[\Psi(x,\theta)]^+$. Examining equations (3.10) and (3.6) we see that the map $\theta \to \Psi(x^*,\theta)$ is affine in $\theta$ for each fixed $x^* \in \Omega_c$. Furthermore, we may rewrite the term $\rho\left[y\right]^+ = \max\{0, \rho y\}$. Since the pointwise maximum of two affine functions defines a convex function, we see that $\theta \to \rho\left[\Psi(x^*,\theta)\right]^+$ is convex, implying that

$$\rho\left[\Psi(x, \alpha\theta_3)\right]^+ \leq \alpha\rho\left[\Psi(x,\theta_1)\right]^+ + (1-\alpha)\rho\left[\Psi(x,\theta_2)\right]^+$$

for each $x \in \Omega_c$, $\theta_1, \theta_2 \in \mathbb{R}^K$ and $\theta_3 = \alpha\theta_1 + (1-\alpha)\theta_2$ for some $\alpha \in [0,1]$. This pointwise fact implies that

$$\begin{aligned} E_{x\sim X}\left[\rho\left[\Psi(x,\theta_3)\right]^+\right] \leq{}& \alpha E_{x\sim X}\left[\rho\left[\Psi(x,\theta_1)\right]^+\right] \\ &+ (1-\alpha)E_{x\sim X}\left[\rho\left[\Psi(x,\theta_2)\right]^+\right]. \end{aligned}$$

Thus, $\theta \to E_{x\sim X}\left[\rho\left[\Psi(x,\theta)\right]^+\right]$ is convex, as desired. □

## A.2 Chapter 4 Proofs

### A.2.1 Proof of Theorem 4.1

The overall loss can be written as

$$\mathbb{E}_{x\sim X} L^{(\rho_1,\rho_2)}(x,\theta) = M_u(\theta) + \rho_1 M_1(\theta) + \rho_2 M_2(\theta).$$

For convenience we write

$$\begin{aligned} \overline{M}_u &= \max_{\theta\in\Theta} M_u(\theta) & \underline{M}_u &= \min_{\theta\in\Theta} M_u(\theta) \\ \overline{M}_1 &= \max_{\theta\in\Theta} M_1(\theta) & \underline{M}_1 &= \min_{\theta\in\Theta} M_1(\theta) \\ \overline{M}_2 &= \max_{\theta\in\Theta} M_2(\theta) & \underline{M}_2 &= \min_{\theta\in\Theta} M_2(\theta) \end{aligned}$$

We first demonstrate that there exists $C_1, C_2 \geq 0$ such that if $\rho_1 \geq \frac{1}{\epsilon_1} C_1 \rho_2 + \frac{1}{\epsilon_1} C_2$ then for each global optimizer $\theta^* \in \Theta$ of $\mathbf{P}_{(\rho_1,\rho_2)}$ we must have $\theta^* \in \Theta_{\epsilon_1}$. To show this consider two points $\theta_1 \in \Theta_0$ and $\theta_2 \notin \Theta_{\epsilon_1}$. Let $L_k = \mathbb{E}_{x\sim X} L_{(\rho_1,\rho_2)}(x, \theta_k)$ for $k \in \{1, 2\}$. We have that:

$$L_1 \leq \overline{M}_u + \rho_2 \overline{M}_2 \quad \text{and} \quad \underline{M}_u + \rho_1 \epsilon_1 + \rho_2 \underline{\mathrm{M}}_2 \leq L_2.$$

Here, the first inequality follows from the fact that $M_1(\theta_1) = 0$ and the second inequality follows from the fact that $M_1(\theta_2) \geq \epsilon_1$. Combining the inequalities yields:

$$L_1 \leq \left(\overline{M}_u - \underline{M}_u\right) - \rho_1 \epsilon_1 + \rho_2 \left(\overline{M}_2 - \underline{M}_2\right) + L_2$$

Thus, we see that if we set $\rho_1 > \frac{1}{\epsilon_1} C_1 \rho_2 + \frac{1}{\epsilon_1} C_2$ with $C_1 = \overline{M}_2 - \underline{M}_2$ and $C_2 = \overline{M}_u - \underline{M}_u$, then we must have that $L_1 < L_2$. Thus, any $\theta_2 \notin \Theta_{\epsilon_1}$ cannot be a global minimizer if we choose the constants $C_1, C_2 \geq 0$ as above.

Next, we demonstrate that when we fix $\rho_1 \geq \frac{1}{\epsilon_1} C_1 \rho_2 + \frac{1}{\epsilon_1} C_2$ as we vary $\rho_2$, then there exists $C_3 \geq 0$ such that if we choose $\rho_2 \geq \frac{1}{\epsilon_2} C_3$ then any global optimizer $\theta^*$ of $\mathbf{P}_{(\rho_1,\rho_2)}$ must lie in $\Theta_{\epsilon_1,\epsilon_2}$. As established above, we already know that all optimizers of $\mathbf{P}_{(\rho_1,\rho_2)}$ must lie in $\Theta_{\epsilon_1}$ in this case. Thus, we will now consider the two points $\theta_3 \in \Theta_{0,0}$ and $\theta_4 \in \left\{\theta \in \Theta_{\epsilon_1} \colon M_2(\theta) > \tilde{M}_2 + \epsilon_2\right\}$, with $\tilde{M}_2$ defined as in (4.4), so that $\theta_4$ satisfies the desired tolerance for the CBF constraint but not the desired tolerance for the CLF constraint. Again let $L_k = \mathbb{E}_{x\sim X} L_{(\rho_1,\rho_2)}(x, \theta_k)$ for $k \in \{3, 4\}$. We then have that

$$L_3 \leq \overline{M}_u + \rho_2 \tilde{M}_2 \quad \text{and} \quad \underline{M}_u + \rho_2(\tilde{M}_2 + \epsilon_2) \leq L_4$$

where we have used the fact that $M_1(\theta_3) = 0$, $M_2(\theta_3) = \tilde{M}_2$ and $M_2(\theta_4) \geq \tilde{M}_2 + \epsilon_2$. Again combining the inequalities and rearranging terms we see that

$$L_3 \leq \left(\overline{M}_u - \underline{M}_u\right) - \rho_2 \epsilon_2 + L_4.$$

Thus, we see that if we select $C_3 = \overline{M}_u - \underline{M}_u$ and put $\rho_2 \geq \frac{1}{\epsilon_2} C_3$ then it must be the case that $L_3 < L_4$ so that $\theta_4$ is not a minimizer. Thus, we see that if we choose $\rho_1 \geq \frac{1}{\epsilon_1} C_1 \rho_2 + \frac{1}{\epsilon_1} C_2$ and $\rho_2 \geq \frac{1}{\epsilon_2} C_3$ with all of the constants chosen as above then all minimizers of $\mathbf{P}_{(\rho_1,\rho_2)}$ must lie in $\Theta_{\epsilon_1,\epsilon_2}$, as desired. □

## A.3 Chapter 5 Proofs and Intermediate Results

### A.3.1 Intermediate Results

**Lemma A.1.** *The composite function $\tilde{\boldsymbol{\nu}}_\gamma^\pi = W + \gamma \tilde{V}_\gamma^\pi \colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is positive definite.*

*Proof.* Note that we can re-write the reshaped cost (5.7) as

$$\tilde{V}_\gamma^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k \Big( [W(x_{k+1}) - W(x_k) + \ell(x_k, \pi(x_k))] \Big), \tag{A.1}$$

where $\{x_k\}_{k=0}^{\infty}$ is the state trajectory generated by the policy $\pi$ from the initial condition $x_0 \in \mathcal{X}$. By rearranging terms we can rewrite this expression as:

$$\tilde{V}_\gamma^\pi(x_0) = -W(x_0) + (1-\gamma)\sum_{k=0}^{\infty}\gamma^k W(x_{k+1}) + \sum_{k=0}^{\infty}\gamma^k \ell(x_k, \pi(x_k)) > -W(x_0) + Q(x_0) \quad \text{(A.2)}$$

where we have used the fact that $W$ and $\ell$ are both non-negative, and that $\ell(x_0, \pi(x_0)) > Q(x_0)$. Thus, using this expression we see that

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x_0) = W(x_0) + \gamma\tilde{V}_\gamma^\pi(x_0) > (1-\gamma)W(x_0) + \gamma Q(x_0), \quad \text{(A.3)}$$

Since $Q$ and $W$ are assumed to be positive definite functions this demonstrates that $\boldsymbol{\mathcal{V}}_\gamma^\pi$ is in fact positive definite, since a convex combination of positive definite functions is positive definite. The proof is concluded by noting that the choice of $\gamma$ and $\pi$ is arbitrary, and thus the conclusion that $\boldsymbol{\mathcal{V}}_\gamma^\pi$ is positive definite holds for all policies and discount factors. □

### A.3.2 Proof of Theorem 5.2

Lemma A.1 demonstrates that $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi = W + \gamma\tilde{V}_\gamma^\pi \colon \mathcal{X} \to \mathbb{R}\cup\{\infty\}$ is a positive definite function. Using the hypotheses of the results with the inequality (5.15) we obtain

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi\big(F(x,\pi(x))\big) - \tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) \leq (-1 + (1-\gamma)[\tilde{C} + \tilde{\delta}])Q(x). \quad \text{(A.4)}$$

Note that if $\tilde{C} + \tilde{\delta} < \frac{1}{1-\gamma}$ then the right hand side of (5.2.2) will be negative definite, which establishes that $\pi$ asymptotically stabilizes the system. □

### A.3.3 Proof of Lemma 5.1

Consider a policy $\bar{\pi} \in \Pi$ defined for each $x \in \mathcal{X}$ by:

$$\bar{\pi}(x) \in \arg\inf_{u\in\mathcal{U}} W(F(x,u)) - W(x) + \ell(x,u) \leq 0, \quad \text{(A.5)}$$

where the preceding inequality follows directly from the assumptions made in the Lemma. Next, for a given initial condition $x_0 \in \mathcal{X}$ let $\{x_k\}_{k=0}^{\infty}$ be the state trajectory generated by $\bar{\pi}$. The corresponding reshaped cost is given by

$$\begin{aligned}
\tilde{V}_\gamma^{\bar{\pi}}(x_0) &= \sum_{k=0}^{\infty}\gamma^k\Big([W\big(F(x_k,\bar{\pi}(x_k))\big) - W(x_k)] + \ell(x_k,\bar{\pi}(x_k))\Big) && \text{(A.6)}\\
&\leq \sum_{k=0}^{\infty}\gamma^k(0) && \text{(A.7)}\\
&\leq 0, && \text{(A.8)}
\end{aligned}$$

which demonstrates the desired result, since the initial condition and discount factor were chosen arbitrarily. □

## A.4 Chapter 10 Proofs and Intermediate Results

### A.4.1 An Intermediate Result of an Equivalent Formulation of the CBF Chance Constraint

We first provide an additional reformulation of the CBF chance constraint, equivalent to those of (10.8b) and (10.16c), which will be useful for the proofs of the feasibility results.

**Lemma A.2.** *The CBF chance constraint* (10.8b) *is feasible at a point $x \in \mathcal{X}$ if and only if there exists a control input $u \in \mathbb{R}^m$ that satisfies both of the following conditions:*

$$\begin{cases} [1 \ u^T] H(x|\mathbb{D}_N) \begin{bmatrix} 1 \\ u \end{bmatrix} \leq 0, & \text{(A.9a)} \\ \widehat{L_g B}(x|\mathbb{D}_N) u + \widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)) \geq 0, & \text{(A.9b)} \end{cases}$$

*where*

$$H(x|\mathbb{D}_N) := \begin{bmatrix} H_{11} & H_{1u} \\ H_{1u}^T & H_{uu} \end{bmatrix}, \quad \textit{with} \tag{A.10}$$

$$\begin{aligned} H_{11} =& \beta^2 \Sigma_{L_f B}(x|\mathbb{D}_N) - (\widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)))^2, \\ H_{1u} =& \beta^2 \Sigma_{L_f B}^{1/2}(x|\mathbb{D}_N)^T \Sigma_{L_g B}^{1/2}(x|\mathbb{D}_N) - \big(\widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x))\big) \widehat{L_g B}(x|\mathbb{D}_N), \\ H_{uu} =& \beta^2 \Sigma_{L_g B}(x|\mathbb{D}_N) - \widehat{L_g B}(x|\mathbb{D}_N)^T \widehat{L_g B}(x|\mathbb{D}_N). \end{aligned}$$

In (A.10), we have used the following relations:

$$\Sigma_{L_f B}(x|\mathbb{D}_N) = \Sigma_{L_f B}^{1/2}(x|\mathbb{D}_N)^T \Sigma_{L_f B}^{1/2}(x|\mathbb{D}_N) \ \in \mathbb{R}, \tag{A.11}$$

$$\Sigma_{L_g B}(x|\mathbb{D}_N) = \Sigma_{L_g B}^{1/2}(x|\mathbb{D}_N)^T \Sigma_{L_g B}^{1/2}(x|\mathbb{D}_N) \ \in \mathbb{R}^{m \times m}. \tag{A.12}$$

*Proof.* The first inequality (A.9a) is directly obtained by squaring both sides of (10.16c). The second inequality (A.9b) is required to check that the right-hand side of (10.16c) is non-negative, as the left-hand side is trivially non-negative. □

### A.4.2 An Intermediate Reusult of a Necessary Condition for Pointwise Feasibility

**Lemma A.3.** *If for a given dataset $\mathbb{D}_N$, the GP-CBF-SOCP* (10.8) *is feasible at a point $x \in \mathbb{R}^n$, then it must hold that the symmetric matrix $H(x|\mathbb{D}_N)$ defined in* (A.10) *cannot be positive definite.*

*Proof.* Positive definiteness of $H(x|\mathbb{D}_N)$ would mean that there does not exist any control input $u \in \mathbb{R}^m$ such that $[1 \ u^T] H(x|\mathbb{D}_N) \begin{bmatrix} 1 \\ u \end{bmatrix} \leq 0$. However, this is a contradiction to Equation (A.9a) in Lemma A.2. Therefore, $H(x|\mathbb{D}_N)$ cannot be positive definite if the GP-CBF-SCOP is feasible. □

### A.4.3 Proof of Lemma 10.2

In this proof, we show that the condition (10.17) of Lemma 10.2 is equivalent to $H(x|\mathbb{D}_N)$ of (A.10) not being positive definite for the same state $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$. By Lemma A.3, this would mean that (10.17) is a necessary condition for pointwise feasibility of the GP-CBF-SOCP, which is the desired result.

Let $\psi(x|\mathbb{D}_N) \coloneqq [\widehat{L_f B}(x|\mathbb{D}_N) + \gamma(B(x)),\ \widehat{L_g B}(x|\mathbb{D}_N)]$. Then, condition (10.17) does not hold if and only if

$$1 - \psi(x|\mathbb{D}_N)\frac{1}{\beta^2}\Sigma_B(x|\mathbb{D}_N)^{-1}\psi(x|\mathbb{D}_N)^T > 0. \tag{A.13}$$

Note that (A.13) is equivalent to

$$M(x|\mathbb{D}_N)/(\beta^2\Sigma_B(x|\mathbb{D}_N)) > 0, \tag{A.14}$$

where we use the operator / for the Schur complement, and $M(x|\mathbb{D}_N) := \begin{bmatrix} 1 & \psi(x|\mathbb{D}_N) \\ \psi(x|\mathbb{D}_N)^T & \beta^2\Sigma_B(x|\mathbb{D}_N) \end{bmatrix}$. From [221, Thm. 1.12], since $\Sigma_B(x|\mathbb{D}_N)$ is positive definite, (A.14) holds if and only if $M(x|\mathbb{D}_N)$ is also positive definite. We now apply again [221, Thm. 1.12], but this time to $M(x|\mathbb{D}_N)/1$. Then, (A.14) is equivalent to the positive definiteness of $M(x|\mathbb{D}_N)/1 = \beta^2\Sigma_B(x|\mathbb{D}_N) - \psi(x|\mathbb{D}_N)^T\psi(x|\mathbb{D}_N) = \beta^2[\Sigma_{L_fB}^{1/2}(x|\mathbb{D}_N)\ \Sigma_{L_gB}^{1/2}(x|\mathbb{D}_N)]^T[\Sigma_{L_fB}^{1/2}(x|\mathbb{D}_N)\ \Sigma_{L_gB}^{1/2}(x|\mathbb{D}_N)] - \psi(x|\mathbb{D}_N)^T\psi(x|\mathbb{D}_N) = H(x|\mathbb{D}_N)$. Therefore, (10.17) does not hold if and only if $H(x|\mathbb{D}_N)$ is positive definite, and the inverse statement completes the proof. □

### A.4.4 Proof of Lemma 10.3

For a point $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$, let $e_\dagger(x|\mathbb{D}_N) \in \mathbb{R}^{m\times 1}$ be the unit eigenvector of $\mathcal{F}(x|\mathbb{D}_N) \in \mathbb{R}^{m\times m}$ associated with the minimum eigenvalue $\lambda_\dagger(x|\mathbb{D}_N) \in \mathbb{R}$. Then, clearly,

$$\lambda_\dagger(x|\mathbb{D}_N) < 0 \implies e_\dagger(x|\mathbb{D}_N)^T\mathcal{F}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) < 0. \tag{A.15}$$

Using (A.15) and taking into account the definition of $\mathcal{F}(x|\mathbb{D}_N)$ in (10.18), the fact that $\Sigma_{L_gB}(x|\mathbb{D}_N)$ is positive definite indicates that $\lambda_\dagger(x|\mathbb{D}_N) < 0 \implies \widehat{L_gB}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) \neq 0$.

Next, take a control input $\pi_{\text{safe}}(x)$ in the direction of $e_\dagger(x|\mathbb{D}_N)$, as defined in (10.19). Plugging $\pi_{\text{safe}}(x)$ into (A.9a), the left-hand side of (A.9a) becomes a polynomial in $\alpha$, of the form $\alpha^2 e_\dagger(x|\mathbb{D}_N)^T\mathcal{F}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) + \mathcal{O}(\alpha)$, where $\mathcal{O}(\alpha)$ denotes terms with degree lower than or equal to 1. Note that the value of the polynomial can be made negative by choosing a large-enough constant $\alpha$, since from (A.15) we know that $e_\dagger(x|\mathbb{D}_N)^T\mathcal{F}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) < 0$. Lastly, also plugging $\pi_{\text{safe}}(x)$ into (A.9b) yields $\alpha|\widehat{L_gB}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N)| + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) \geq 0$, which again holds for a sufficiently large $\alpha$. Therefore, by Lemma A.2 the GP-CBF-SOCP (10.8) is feasible when $\lambda_\dagger(x|\mathbb{D}_N) < 0$ and $\pi_{\text{safe}}(x)$ is a feasible control input for large-enough $\alpha$. □

### A.4.5 Proof of Theorem 10.2

We first provide a geometric interpretation of the probabilistic CBF second-order cone constraint (10.16c). For a point $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$, $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$ is the positive sheet of an $m$-dimensional hyperboloid in $\mathbb{R}^{m+1}$ (illustrated in Fig. 10.1). This surface asymptotically converges to the conical surface $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)(u-u_0) \right\|_2 = t$ as $\|u\|_2 \to \infty$, where

$$u_0 = -\Sigma_{L_gB}(x|\mathbb{D}_N)^{-1}\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T\Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \tag{A.16}$$

is the least-squares control input that minimizes $\left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2$. We will refer to the conical surface $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)(u-u_0) \right\|_2 = t$ as the *asymptote* of $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$.

Since $\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N) \succ 0$, the GP-CBF-SOCP (10.8) is feasible if and only if an intersection between the hyperboloid $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$ and the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = t$ exists. We now analyze each of the individual cases of Theorem 10.2.

Case 1 (Fig. 10.1-Hyperbolic): This case matches the sufficient condition of Lemma 10.3. Note that this condition implies that the necessary condition (10.17) is trivially satisfied. In this case, the slope of the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = t$ is greater than the slope of the asymptote of the hyperboloid for the direction of $u$ corresponding to $\pi_{\text{safe}}$.

Case 2 (Fig. 10.1-Elliptic): Given that the smallest eigenvalue of $\mathcal{F}(x|\mathbb{D}_N)$ is positive, then $\mathcal{F}(x|\mathbb{D}_N) \succ 0$. Note that $\mathcal{F}(x|\mathbb{D}_N)$ is the lower-right block of the matrix $H(x|\mathbb{D}_N)$ in (A.10). Therefore, $\mathcal{F}(x|\mathbb{D}_N) \succ 0$ implies that the left-hand side of Equation (A.9a) must be strictly convex, with a unique global minimum at some $u = u_1 \in \mathbb{R}^m$. The first-order optimality condition gives

$$\begin{aligned} u_1 =& -\mathcal{F}(x|\mathbb{D}_N)^{-1}h, \quad \text{with} \\ h :=& \beta^2\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T\Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) - \widehat{L_gB}(x|\mathbb{D}_N)^T\big(\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x))\big). \end{aligned}$$

Since at $u_1$ the minimum is attained, Equation (A.9a) holds if and only if

$$[1 \; u_1^T]H(x|\mathbb{D}_N)\begin{bmatrix} 1 \\ u_1 \end{bmatrix} \leq 0. \tag{A.17}$$

Plugging (A.10) and $u_1 = -\mathcal{F}(x|\mathbb{D}_N)^{-1}h$ into (A.17), we get

$$\begin{aligned} \beta^2\Sigma_{L_fB}(x|\mathbb{D}_N) - \big(\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x))\big)^2 - \\ h^T\mathcal{F}(x|\mathbb{D}_N)^{-1}h = H(x|\mathbb{D}_N)/\mathcal{F}(x|\mathbb{D}_N) \leq 0. \end{aligned} \tag{A.18}$$

Since for this case $\mathcal{F}(x|\mathbb{D}_N)$ is positive definite, and $H(x|\mathbb{D}_N)$ cannot be positive definite by the necessary condition (10.17), then from [221, Thm. 1.12] the inequality (A.18) must be satisfied. Consequently, (A.9a) holds for $u = u_1$. Now, plugging $u_1$ into (A.9b) we have: $\widehat{L_gB}(x|\mathbb{D}_N)u_1 + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) - \widehat{L_gB}(x|\mathbb{D}_N)\,\mathcal{F}(x|\mathbb{D}_N)^{-1}\big[\beta^2\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T\Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) - \widehat{L_gB}(x|\mathbb{D}_N)^T(\widehat{L_fB}(x|\mathbb{D}_N)+\gamma(B(x)))\big]$, which matches exactly the left-hand side of (10.20). Thus, from Lemma A.2 the feasible set is non-empty if and only if (10.20) is non-negative (see Fig. 10.1-Elliptic). On the other hand, (10.20) being negative would mean that the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u+\widehat{L_fB}(x|\mathbb{D}_N)+\gamma(B(x))=t$ intersects the hyperboloid's negative sheet, $-\beta\left\|\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u+\Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N)\right\|_2 = t$, forming an ellipse, and therefore cannot intersect the positive sheet. Consequently, when $\lambda_\dagger(x|\mathbb{D}_N) > 0$, the GP-CBF-SOCP (10.8) is feasible if and only if (10.20) holds.

Case 3 (Fig. 10.1-Parabolic): For this case, note that $\lambda_\dagger(x|\mathbb{D}_N)=0$ means that there exists some control input direction for which the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u+\widehat{L_fB}(x|\mathbb{D}_N)+\gamma(B(x))=t$ and the asymptote of $\beta\left\|\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N)\right\|_2 = t$ have the same slope (see Fig. 10.1-Parabolic). Let us define

$$p \coloneqq \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) - \widehat{L_gB}(x|\mathbb{D}_N)\Sigma_{L_gB}(x|\mathbb{D}_N)^{-1}\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)^T\Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N).$$

Then, condition (10.21) is satisfied if and only if $p > 0$. Consider the control input $u = u_0$ from (A.16) that minimizes $\left\|\Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N)\right\|_2$. Then, we can rewrite $p = \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) + \widehat{L_gB}(x|\mathbb{D}_N)u_0$.

Furthermore, let $e_\dagger(x|\mathbb{D}_N)$ denote the unit eigenvector of $\mathcal{F}(x|\mathbb{D}_N)$ associated with the eigenvalue $\lambda_\dagger(x|\mathbb{D}_N) = 0$. Then, clearly, $e_\dagger(x|\mathbb{D}_N)^T\mathcal{F}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) = 0$. Based on the definition of $\mathcal{F}(x|\mathbb{D}_N)$ (10.18), since $\Sigma_{L_gB}(x|\mathbb{D}_N) \succ 0$ then it must hold that $\widehat{L_gB}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N) \neq 0$. Next, plugging a control input of the form

$$u = u_0 + \alpha\text{sgn}(\widehat{L_gB}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N))e_\dagger(x|\mathbb{D}_N), \quad \alpha > 0, \tag{A.19}$$

into the left-hand side of (A.9a), we have

$$\begin{aligned}\beta^2\Sigma_{L_fB}(x|\mathbb{D}_N) - (\widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)))^2 + 2h^Tu_0 + \\ u_0^T\mathcal{F}(x|\mathbb{D}_N)u_0 - 2\alpha p|\widehat{L_gB}(x|\mathbb{D}_N)e_\dagger(x|\mathbb{D}_N)|.\end{aligned} \tag{A.20}$$

And plugging (A.19) into the left-hand side of (A.9b), we obtain

$$p + \alpha \cdot |\widehat{L_gB}(x|\mathbb{D}_N)e_\dagger|. \tag{A.21}$$

If $p$ is positive, then there exists a large-enough positive constant $\alpha$ such that (A.20) is non-positive and (A.21) positive. Therefore, from Lemma A.2, the GP-CBF-SOCP (10.8) is feasible in this case.

Note that the geometric interpretation of the condition $p > 0$ is that the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = t$, which has the same slope as the asymptote of $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$ along the direction of $e_\dagger(x|\mathbb{D}_N)$, should be placed over the asymptote in order for it to intersect the positive sheet of $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)u + \Sigma^{1/2}_{L_fB}(x|\mathbb{D}_N) \right\|_2 = t$. Furthermore, at $u = u_0$, the asymptote $\beta \left\| \Sigma^{1/2}_{L_gB}(x|\mathbb{D}_N)(u - u_0) \right\|_2 = t$ takes value $t = 0$, and $p$ is the value of the hyperplane $\widehat{L_gB}(x|\mathbb{D}_N)u + \widehat{L_fB}(x|\mathbb{D}_N) + \gamma(B(x)) = t$ at $u = u_0$. Therefore, when $p \leq 0$, the hyperplane is always under the positive sheet of the hyperboloid, and never intersects it. Consequently, both the constraint (10.16c) and the GP-CBF-SOCP (10.8) are not feasible when $p \leq 0$. □

### A.4.6 Proof of Lemma 10.4

Let us consider the trajectory generated by running Algorithm 10.1 from any $x_0 \in \mathcal{X}$, which we assume locally exists and is unique (as stated in the hypothesis of the Lemma). For a fixed dataset $\mathbb{D}_N$, $\lambda_\dagger\big(x|\mathbb{D}_N\big)$ is a continuous function of the state $x$ by basic continuity arguments. For the event-triggered updates of the dataset, if at time $t$ we have $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)}\big) \geq -\varepsilon$, then Algorithm 10.1 applies a control input $\pi_{\text{safe}}(x(t))$ from (10.19), collects the resulting measurement, and adds it to $\mathbb{D}_{N(t)}$, forming $\mathbb{D}_{N(t)+1}$. Note that from the posterior variance expression (9.7), after adding the new data point we have $e_\dagger(x|\mathbb{D}_{N(t)})^T \Sigma_{L_gB}(x|\mathbb{D}_{N(t)+1}) e_\dagger(x|\mathbb{D}_{N(t)}) \to 0$ for large $\alpha$. Therefore, using Assumptions 10.3 and 10.4, we can choose $\alpha > 0$ such that $e_\dagger(x|\mathbb{D}_{N(t)})^T \big(\beta^2 \Sigma_{L_gB}(x|\mathbb{D}_{N(t)+1}) - \widehat{L_gB}(x|\mathbb{D}_{N(t)+1}) \widehat{L_gB}(x|\mathbb{D}_{N(t)+1})^T\big) e_\dagger(x|\mathbb{D}_{N(t)}) < 0$, leading to $\lambda_\dagger\big(x(t)|\mathbb{D}_{N(t)+1}\big) < 0$. An equivalent argument proves that with the time-triggered updates $\lambda_\dagger$ stays negative after the new data point is added. □

### A.4.7 Proof of Lemma 10.5

We use [182, Thm. 6.4] which provides a sufficient condition for local Lipschitz continuity of solutions of parametric optimization problems. Twice differentiability of the objective and constraints with respect to both state and input trivially follows from Assumption 10.5 and the structure of (10.8). For a given state $x \in \mathcal{X}$ and dataset $\mathbb{D}_N$, $\lambda_\dagger(x|\mathbb{D}_N) < 0$ means that there exists a control input $\pi_{\text{safe}}$ from (10.19) that strictly satisfies constraint (10.8b), meaning that in this case (10.8) satisfies Slater's Condition (SC), since the problem is convex. [182, Thm. 6.4] requires satisfaction of the Mangasarian Fromovitz Constraint Qualification (MFCQ) and the Second Order Condition (SOC2) of [182, Def. 6.1] at the solution of (10.8). In [10, Prop. 5.39], it is shown that SC implies MFCQ. Furthermore, since we have a strongly convex objective function in the decision variables $(u, d)$, and the constraints are convex in $(u, d)$, the Lagrangian of (10.8) is strongly convex in $(u, d)$, implying SOC2 satisfaction. □

### A.4.8 Proof of Theorem 10.4

The proof trivially follows from [131, Thm. III.1] using local Lipschitz continuity of the continuous dynamics (from Lemma 10.5 and the expression of $\pi_{\text{safe}}$ in (10.19)) instead of global Lipschitz continuity, therefore establishing local existence and uniqueness of executions of the closed-loop switched system. □

## A.5 Chapter 11 Proofs

For notational convenience, we will drop $(\cdot|\mathbb{D}_N)$ or $(\cdot|\mathbb{D}_M)$ when the dependency is obvious.

### A.5.1 Proof of Lemma 11.1

From (11.11), GP-CF-SOCP is feasible under $u = \alpha' \widehat{L_g C}(x)$ if

$$c\alpha \left\| \widehat{L_g C}(x) \right\|^2 - \beta \sigma_C \Big( x, c\alpha \widehat{L_g C}(x) \Big) \geq - \Big( \widehat{L_f C}(x) + \gamma(C(x)) \Big),$$

where $c := \alpha'/\alpha > 1$. First, we compare $\sigma_C \Big( x, c\alpha \widehat{L_g C}(x) \Big)$ and $\sigma_C \Big( x, \alpha \widehat{L_g C}(x) \Big)$ as below:

$$\begin{aligned}
&\frac{1}{c^2} \sigma_C^2 \Big( x, c\alpha \widehat{L_g C}(x) \Big) \\
&= \frac{1}{c^2} [1 \;\; c\alpha \widehat{L_g C}(x)] \Sigma_C(x) \begin{bmatrix} 1 \\ c\alpha \widehat{L_g C}(x)^\top \end{bmatrix} \\
&= [1/c \;\; \alpha \widehat{L_g C}(x)] \Sigma_C(x) \begin{bmatrix} 1/c \\ \alpha \widehat{L_g C}(x)^\top \end{bmatrix} \\
&= [1 \;\; \alpha \widehat{L_g C}(x)] \Sigma_C(x) \begin{bmatrix} 1 \\ \alpha \widehat{L_g C}(x)^\top \end{bmatrix} - \begin{bmatrix} 1 - \frac{1}{c^2} & 0 \end{bmatrix} \Sigma_C(x) \begin{bmatrix} 1 - \frac{1}{c^2} \\ 0 \end{bmatrix} \\
&= \sigma_C^2 \Big( x, \alpha \widehat{L_g C}(x) \Big) - \left( 1 - \frac{1}{c^2} \right)^2 \Sigma_C(x)_{[1,1]}.
\end{aligned}$$

Thus, it holds that

$$\sigma_C^2 \Big( x, c\alpha \widehat{L_g C}(x) \Big) = c^2 \left( \sigma_C^2 \Big( x, \alpha \widehat{L_g C}(x) \Big) - \left( 1 - \frac{1}{c^2} \right)^2 \Sigma_C(x)_{[1,1]} \right)$$

Using this expression, we can check that

$$\frac{c\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,c\alpha\widehat{L_gC}(x)\Big)}{c\left(\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,\alpha\widehat{L_gC}(x)\Big)\right)}$$
$$=\frac{\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sqrt{\sigma_C^2\Big(x,\alpha\widehat{L_gC}(x)\Big)-\left(1-\frac{1}{c^2}\right)^2\Sigma_C(x)_{[1,1]}}}{\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,\alpha\widehat{L_gC}(x)\Big)}>1.$$

Finally, since $\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,\alpha\widehat{L_gC}(x)\Big)$ is strictly positive from (11.16), by taking $c$ satisfying

$$c\geq\frac{-\Big(\widehat{L_fC}(x|\mathbb{D}_N)+\gamma(C(x))\Big)}{\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,\alpha\widehat{L_gC}(x)\Big)},$$

we get

$$\begin{aligned}&c\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,c\alpha\widehat{L_gC}(x)\Big)\\&>c\left(\alpha\left\|\widehat{L_gC}(x)\right\|^2-\beta\sigma_C\Big(x,\alpha\widehat{L_gC}(x)\Big)\right)\\&\geq-\Big(\widehat{L_fC}(x|\mathbb{D}_N)+\gamma(C(x))\Big),\end{aligned}$$

which completes the proof. □

## A.5.2 Proof of Lemma 11.2

We begin the proof by noting that

$$\lim_{\alpha\to\infty}\frac{1}{\alpha}\begin{bmatrix}\mathbf{k}_{*1}(x,\alpha\widehat{L_gC}(x))\\\vdots\\\mathbf{k}_{*M}(x,\alpha\widehat{L_gC}(x))\end{bmatrix}=\lim_{\alpha\to\infty}\frac{1}{\alpha}\begin{bmatrix}k_f(x_1,x_1)+\mathbf{k}^u_{*1}(x,\alpha\widehat{L_gC}(x))\\\vdots\\k_f(x_M,x_M)+\mathbf{k}^u_{*M}(x,\alpha\widehat{L_gC}(x))\end{bmatrix}=\begin{bmatrix}\mathbf{k}^u_{*1}(x,\widehat{L_gC}(x))\\\vdots\\\mathbf{k}^u_{*M}(x,\widehat{L_gC}(x))\end{bmatrix}.$$

Thus, we obtain

$$\begin{aligned}
&\arg\min_{\mathbb{D}_M} \lim_{\alpha\to\infty} \frac{1}{\alpha}\sigma_C\Big(x, \alpha\widehat{L_g C}(x)\Big) \\
&= \arg\min_{\mathbb{D}_M} \frac{1}{\alpha^2}\sigma_C^2\Big(x, \alpha\widehat{L_g C}(x)\Big) \\
&= \arg\max_{\mathbb{D}_M} \lim_{\alpha\to\infty} \frac{1}{\alpha^2}\left[\mathbf{k}_{*1}(x, \alpha\widehat{L_g C}(x)) \ \cdots \ \mathbf{k}_{*M}(x, \alpha\widehat{L_g C}(x))\right] \\
&\qquad\qquad (K_{\mathbb{D}_M} + \sigma_n^2 I)^{-1}\begin{bmatrix}\mathbf{k}_{*1}(x, \alpha\widehat{L_g C}(x)) \\ \vdots \\ \mathbf{k}_{*M}(x, \alpha\widehat{L_g C}(x))\end{bmatrix} \text{(from (11.16))} \\
&= \arg\max_{\mathbb{D}_M} \left[\mathbf{k}^u_{*1}(x, \widehat{L_g C}(x)) \ \cdots \ \mathbf{k}^u_{*N}(x, \widehat{L_g C}(x))\right] \\
&\qquad\qquad (K_{\mathbb{D}_M} + \sigma_n^2 I)^{-1}\begin{bmatrix}\mathbf{k}^u_{*1}(x, \widehat{L_g C}(x)) \\ \vdots \\ \mathbf{k}^u_{*M}(x, \widehat{L_g C}(x))\end{bmatrix},
\end{aligned}$$

which is precisely the objective function appearing in the Lemma. □

### A.5.3 Proof of Theorem 11.1

For notational convenience, we use the subscript $ij$ to indicate the $(i, j)$-th off-diagonal term of a matrix. We first present a few lemmas that will be used in the proof.

**Lemma A.4.** *Let $C = (c_{ij}) \in \mathbb{R}^{n\times n}$ be a non-negative matrix. Then, the maximal eigenvalue of $C$ is upper bounded by its maximal row sum, that is,*

$$\lambda_{\max}(C) \le \max_i \sum_{j=1}^{n} c_{ij}. \tag{A.22}$$

*Proof.* This is a corollary of Perron-Frobenius Theorem for nonnegative matrices [138, Ch.8]. □

**Lemma A.5** (Weyl's Inequality)**.** *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric matrices. Then,*

$$\lambda_{\min}(A + B) \ge \lambda_{\min}(A) + \lambda_{\min}(B).$$

**Lemma A.6.** *Let $B = (b_{ij}) \in \mathbb{R}^{n\times n}$ be a square matrix whose diagonal entities satisfy $b_{ii} = 1$, and whose off-diagonal entities satisfy $-1 \le b_{ij} \le 0$ for all $i \ne j$. Let $\bar{B} = (\bar{b}_{ij}) \in \mathbb{R}^{n\times n}$ be a matrix whose diagonal entities are all one, and whose off-diagonal entities are $\bar{b}_{ij} = \pm b_{ij}$, where the signs can be arbitrary. Then, if $B$ is positive definite, $\bar{B}$ is also positive definite.*

*Proof.* This can be proved by induction. The case when $n = 1$ is trivial since there is no off-diagonal term.

Assume the lemma holds for $n = k$, that is, if $B_k$ and $\bar{B}_k$ are constructed to satisfy the statement in the lemma, $B_k \succ 0 \Rightarrow \bar{B}_k \succ 0$ holds.

Next, consider

$$B_{k+1} = \begin{bmatrix} B_k & p_k \\ p_k^T & 1 \end{bmatrix} \succ 0,$$

where $p_k = [b_{1(k+1)} \ \cdots \ b_{k(k+1)}]^T$, and $-1 \leq b_{i(k+1)} \leq 0$ for $i = 1, \cdots, k$. Let $\bar{B}_{k+1}$ constructed according to the statement in the lemma as

$$\bar{B}_{k+1} = \begin{bmatrix} \bar{B}_k & \bar{p}_k \\ \bar{p}_k^T & 1 \end{bmatrix}.$$

Since $B_{k+1}$ is positive definite, by Schur complement lemma, the following holds.

$$B_k \succ 0, \ p_k^T B_k^{-1} p_k < 1.$$

Note that $\bar{B}_k \succ 0$ holds due to the assumption of the induction. Define

$$C_k = I - B_k = \begin{bmatrix} 0 & & b_i j \\ & \ddots & \\ b_j i & & 0 \end{bmatrix}, \ \bar{C}_k = I - \bar{B}_k = \begin{bmatrix} 0 & & \bar{b}_i j \\ & \ddots & \\ \bar{b}_j i & & 0 \end{bmatrix}.$$

Then

$$\begin{aligned} \bar{p}_k^T \bar{B}_k^{-1} \bar{p}_k = \bar{p}_k^T (I - \bar{C}_k)^{-1} \bar{p}_k = & \sum_{t=0}^{\infty} \bar{p}_k^T \bar{C}_k^t \ \bar{p}_k \leq \sum_{t=0}^{\infty} p_k^T C_k^t \ p_k \\ = & p_k^T (I - C_k)^{-1} p_k = p_k^T B_k^{-1} p_k < 1. \end{aligned}$$

Since $\bar{B}_k \succ 0$ and $\bar{p}_k^T \bar{B}_k^{-1} \bar{p}_k < 1$ holds, by Schur complement lemma, $\bar{B}_{k+1}$ is positive definite. This shows that the lemma holds for $n = k + 1$. The lemma is proved by induction. □

Presented next is the main Proof of Theorem 11.1. We will drop $(x, u)$ from $\mathbf{k}_{*i}^u$ and $n_i$ for notational convenience. We want to prove that

$$[\mathbf{k}_{*1}^u \ \cdots \ \mathbf{k}_{*N}^u] \, (K_{\mathbb{D}_M} + \sigma_n^2 I)^{-1} \begin{bmatrix} \mathbf{k}_{*1}^u \\ \vdots \\ \mathbf{k}_{*M}^u \end{bmatrix} \geq \frac{1 - \epsilon}{1 + \epsilon(N - 2)} \sum_{i=1}^{N} n_i^2 \tag{A.23}$$

is equivalent to

$$[\mathbf{k}_{*1}^u \ \cdots \ \mathbf{k}_{*N}^u] \, (K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1} \begin{bmatrix} \mathbf{k}_{*1}^u \\ \vdots \\ \mathbf{k}_{*M}^u \end{bmatrix} \geq \frac{1 - \epsilon}{1 + \epsilon(N - 2)} \times [\mathbf{k}_{*1}^u \ \cdots \ \mathbf{k}_{*N}^u] \, \mathrm{Diag} \left( \begin{bmatrix} \frac{1}{\mathbf{k}_1} \cdots \frac{1}{\mathbf{k}_N} \end{bmatrix} \right) \begin{bmatrix} \mathbf{k}_{*1}^u \\ \vdots \\ \mathbf{k}_{*M}^u \end{bmatrix}.$$

It is sufficient to prove that

$$(K_{\mathbb{D}_N} + \sigma_n^2 I)^{-1} \succeq \frac{1-\epsilon}{1+\epsilon(N-2)} \text{Diag}\left(\left[\frac{1}{\mathbf{k}_1} \cdots \frac{1}{\mathbf{k}_N}\right]\right),$$

and this is equivalent to

$$\frac{1+\epsilon(N-2)}{1-\epsilon} \text{Diag}\left([\mathbf{k}_1 \ \cdots \ \mathbf{k}_N]\right) - (K_{\mathbb{D}_N} + \sigma_n^2 I) \succeq 0. \tag{A.24}$$

We have

$$\begin{aligned}
&\frac{1+\epsilon(N-2)}{1-\epsilon} \text{Diag}\left([\mathbf{k}_1 \ \cdots \ \mathbf{k}_N]\right) - (K_{\mathbb{D}_N} + \sigma_n^2 I) \\
=&\begin{bmatrix} \frac{\epsilon(N-1)}{1-\epsilon}\mathbf{k}_1 - \sigma_n^2 & & -\mathbf{k}_{ij} \\ & \ddots & \\ -\mathbf{k}_{ji} & & \frac{\epsilon(N-1)}{1-\epsilon}\mathbf{k}_N - \sigma_n^2 \end{bmatrix} \\
=&\text{Diag}\left(\sqrt{\mathbf{k}_1}, \cdots, \sqrt{\mathbf{k}_N}\right) A \ \text{Diag}\left(\sqrt{\mathbf{k}_1} \ \cdots \ \sqrt{\mathbf{k}_N}\right),
\end{aligned} \tag{A.25}$$

where

$$A := \begin{bmatrix} \frac{\epsilon(N-1)}{1-\epsilon} - \frac{\sigma_n^2}{\mathbf{k}_1} & & -\frac{\mathbf{k}_{ij}}{\sqrt{\mathbf{k}_i \mathbf{k}_j}} \\ & \ddots & \\ -\frac{\mathbf{k}_{ji}}{\sqrt{\mathbf{k}_j \mathbf{k}_i}} & & \frac{\epsilon(N-1)}{1-\epsilon} - \frac{\sigma_n^2}{\mathbf{k}_N} \end{bmatrix}.$$

Thus, we just have to prove that $A$ is positive semidefinite. By Lemma A.5,

$$\lambda_{\min}(A) \geq \lambda_{\min}(\epsilon(N-1)\bar{B}) + \lambda_{\min}\left(A - \epsilon(N-1)\bar{B}\right),$$

where

$$\bar{B} := \begin{bmatrix} 1 & & -\frac{1}{\epsilon(N-1)}\frac{\mathbf{k}_{ij}}{\sqrt{\mathbf{k}_i \mathbf{k}_j}} \\ & \ddots & \\ -\frac{1}{\epsilon(N-1)}\frac{\mathbf{k}_{ji}}{\sqrt{\mathbf{k}_j \mathbf{k}_i}} & & 1 \end{bmatrix}.$$

Note that

$$A - \epsilon(N-1)\bar{B} = \text{Diag}\left(\frac{\epsilon^2(N-1)}{1-\epsilon} - \frac{\sigma_n^2}{\mathbf{k}_1}, \cdots, \frac{\epsilon^2(N-1)}{1-\epsilon} - \frac{\sigma_n^2}{\mathbf{k}_N}\right),$$

and from (11.23),

$$\frac{\epsilon^2(N-1)}{1-\epsilon} - \frac{\sigma_n^2}{\mathbf{k}_i} \geq 0$$

for all $i = 1, \cdots, N$, thus, $A - \epsilon(N-1)\bar{B}$ is positive semidefinite. Therefore, it is now sufficient to prove that $\bar{B}$ is positive semidefinite to prove (A.24).

We define

$$C := \begin{bmatrix} 0 & & \frac{1}{\epsilon(N-1)} \frac{|\mathbf{k}_{ij}|}{\sqrt{\mathbf{k}_i \bar{k}_j}} \\ & \ddots & \\ \frac{1}{\epsilon(N-1)} \frac{|\mathbf{k}_{ji}|}{\sqrt{\mathbf{k}_j \bar{k}_i}} & & 0 \end{bmatrix}. \tag{A.26}$$

By applying Lemma A.4 to $C$ which is non-negative, and by using condition (11.22):

$$\lambda_{\max}(C) \leq \max_i \sum_{j=1, j\neq i}^{N} \frac{1}{\epsilon(N-1)} \frac{|\mathbf{k}_{ij}|}{\sqrt{\mathbf{k}_i \mathbf{k}_j}} < \frac{1}{\epsilon(N-1)} \epsilon(n-1) = 1.$$

Thus, we have $\lambda_{\max}(C) < 1$. By applying Lemma A.5, we have

$$\lambda_{\min}(I - C) \geq \lambda_{\min}(I) + \lambda_{\min}(-C) = 1 - \lambda_{\max}(C) > 0.$$

Thus, $B := I - C$ is positive definite. Note that $\bar{B}$ and $B$ satisfy the conditions in Lemma A.6 since $0 \leq \frac{1}{\epsilon(N-1)} \frac{|\mathbf{k}_{ij}|}{\sqrt{\mathbf{k}_i \mathbf{k}_j}} < \frac{1}{N-1} \leq 1$ from (11.22). Thus, by Lemma A.6, $\bar{B}$ is positive definite.

□