LEARNING-BASED CONTROL UNDER CONSTRAINTS

TOWARDS SAFETY AND COMPUTATIONAL EFFICIENCY

Kanghui He

LEARNING-BASED CONTROL UNDER CONSTRAINTS

TOWARDS SAFETY AND COMPUTATIONAL EFFICIENCY

LEARNING-BASED CONTROL UNDER CONSTRAINTS

TOWARDS SAFETY AND COMPUTATIONAL EFFICIENCY

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates,
to be defended publicly
on Tuesday, January 20, 2026 at 10:00 AM

by

Kanghui HE

Master of Science in Aeronautical Engineering,
Beihang University, Beijing, China

born in Shaanxi, China

This dissertation has been approved by the promotors.

Promotor: Prof. dr. ir. B. De Schutter
Promotor: Dr. ir. A.J.J. van den Boom
Copromotor: Dr. ir. S. Shi

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. ir. B. De Schutter	Delft University of Technology, promotor
Dr. ir. A.J.J. van den Boom	Delft University of Technology, promotor
Dr. ir. S. Shi	Delft University of Technology, copromotor
Independent members:	
Prof. dr. R. Babuška	Delft University of Technology
Prof. dr. ir. T. Keviczky	Delft University of Technology
Prof. dr. ir. J. Suykens	Katholieke Universiteit Leuven, Belgium
Prof. dr. M. Zeilinger	Eidgenössische Technische Hochschule Zürich, Switzerland
Dr. M. Mazo Espinosa	Delft University of Technology, reserve member

The research described in this PhD thesis is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101018826 - CLariNet).

Keywords: Learning-based control, optimization-based control, reinforcement learning, safety-critical systems

Email: hekanghuily@gmail.com

ISBN/EAN: 978-90-361-0834-8 (Paperback)

ISBN/EAN: 978-94-6518-184-4 (E-book)

An electronic version of this dissertation is available at
https://repository.tudelft.nl/.

Enjoy and be content with the present.

CONTENTS

SUMMARY

While reinforcement learning (RL) and supervised learning provide powerful approaches for finding optimal controllers for complex systems, ensuring safety remains a critical challenge. In control problems, safety is typically defined as maintaining state and input constraint satisfaction throughout the system's evolution. The key issue lies in **balancing constraint satisfaction with computational efficiency in the presence of inevitable learning errors**. This PhD thesis addresses this challenge across linear, piecewise affine (PWA), and nonlinear systems with various constraint structures.

First, we consider linear systems with linear constraints. RL struggles with constraints, whereas model predictive control (MPC) excels in handling them but can be computationally demanding. The PhD thesis introduces a novel approach combining the two methodologies to overcome their individual limitations. Particularly, constrained approximate dynamic programming (ADP) associated with a novel convex and piecewise quadratic (PWQ) neural network is proposed to approximate the infinite-horizon value function, which is used as the terminal cost function in an MPC problem where the horizon is 1. A decomposition algorithm is developed to solve the PWQ MPC problem efficiently.

PWA systems can model hybrid behavior and approximate nonlinear dynamics. We address PWA systems with union-of-convex-sets constraints, where traditional hybrid MPC faces scalability issues. We consider two alternative learning-based approaches: safe learning with cost shaping and safe learning with policy filtering. The cost shaping method accommodates the state constraints into ADP by adding PWA penalty functions into the cost function. An actor-critic learning algorithm is developed, generating an explicit and computationally efficient ADP policy. We propose a systematic verification framework based on mixed-integer linear programming to certify the practical stability and safety. In contrast, the policy filtering method, which employs a safety filter, offers stronger safety guarantees and broader applicability than cost shaping. When applied to PWA systems, traditional safety filters usually solve a mixed-integer convex program, which compromises the computational benefit of learning-based control. We introduce an optimization-free safety filter that determines safe control inputs using algebraic and min-max operations, significantly reducing computational overhead. Additionally, to further reduce the conservatism of the safety filter, we design predictive safety filters using the concept of predictive control barrier functions (CBFs). Generalized Clarke derivatives are used to handle the non-smoothness of the optimization problem. We show that enforcing CBF constraints across all components of the generalized Clarke derivatives is sufficient to guarantee safety.

We also develop a general optimization-based safety framework for nonlinear systems with non-convex constraints. When applied to general nonlinear systems, traditional

safety filters and MPC typically involve a prediction model, which severely limits their application when the model is unknown. In contrast, the proposed optimization-based safe control framework can be used in model-based and model-free settings. Besides, it is computationally efficient to implement and preserves safety performance under learning errors. We introduce state-action control barrier functions (SACBFs), which facilitate efficient convex quadratic optimization-based safety filtering. We propose both model-based and data-driven approaches to synthesize SACBFs. The effect of learning errors on the effectiveness of SACBFs is addressed by developing an error-to-state safety analysis framework, which establishes a foundation for a constraint-tightening approach. As an extension to the model-free setting, direct data-driven safe control is proposed, enabling safe control design and implementation purely from state transition data. The proposed optimization-based control framework bridges the gap between model-free learning-based control and constrained control by decoupling performance optimization from safety enforcement.

In summary, the PhD thesis integrates machine learning and optimization to develop safe and online efficient control methods for linear, PWA, and nonlinear systems. Simulations on tasks such as inverted pendulum stabilization, adaptive cruise control, control of room temperature, and vehicle trajectory planning demonstrate significant improvements in computational efficiency and constraint satisfaction over conventional control methods.

SAMENVATTING

Hoewel *reinforcement learning* (RL) en *supervised learning* krachtige methoden bieden voor het vinden van optimale regelstrategieën voor complexe systemen, blijft het waarborgen van veiligheid een cruciale uitdaging. In regelproblemen wordt veiligheid doorgaans gedefinieerd als het handhaven van beperkingen op de toestanden en de ingangen gedurende de evolutie van het systeem. De kern van het probleem ligt in het balanceren van het naleven van beperkingen en het verhogen van de computationele efficiëntie in de aanwezigheid van onvermijdelijke leerfouten. Dit proefschrift pakt deze uitdaging aan voor lineaire, stuksgewijze affiene (in het Engels: *piecewise affine* (PWA)) en niet-lineaire systemen met diverse beperkingsstructuren.

Allereerst beschouwen we lineaire systemen met lineaire beperkingen. RL heeft moeite met beperkingen, terwijl modelgebaseerde voorspellende regeling (in het Engels: *model predictive control* (MPC)) daar juist sterk in is, maar vaak veel rekenkracht vereist. Dit proefschrift introduceert een nieuwe aanpak die beide methoden combineert om hun individuele beperkingen te overwinnen. In het bijzonder wordt een benadering met *constrained approximate dynamic programming* (ADP) voorgesteld, in combinatie met een nieuw convex en stuksgewijs kwadratisch (in het Engels: *piecewise quadratic* (PWQ)) neuraal netwerk om de oneindige-horizon waardefunctie te benaderen. Deze benadering wordt gebruikt als eindkostenfunctie in een MPC-probleem met een horizon van 1. We ontwikkelen ook een decompositie-algoritme om het PWQ-MPC-probleem efficiënt op te lossen.

PWA-systemen kunnen hybride gedrag modelleren en niet-lineaire dynamica benaderen. We beschouwen PWA-systemen met beperkingen gedefinieerd als de unie van convexe verzamelingen, waarbij traditionele hybride MPC tegen schaalbaarheidsproblemen aanloopt. We bekijken twee alternatieve leergebaseerde benaderingen: veilig leren met kostenaanpassing en veilig leren met regelbeleidsfiltering. De kostenaanpassingsmethode verwerkt toestandsbeperkingen in ADP door PWA-boetefuncties aan de kostenfunctie toe te voegen. Een *actor-critic* leeralgoritme wordt ontwikkeld dat een expliciet en computationeel efficiënt ADP-regelbeleid oplevert. We stellen ook een systematisch verificatiekader voor op basis van gemengd-gehele lineaire optimalisatie om praktische stabiliteit en veiligheid te garanderen. De regelbeleidsfiltermethode, die gebruikmaakt van een veiligheidsfilter, biedt echter sterkere veiligheidswaarborgen en bredere toepasbaarheid dan kostenaanpassing. Bij toepassing op PWA-systemen vereisen traditionele veiligheidsfilters doorgaans het oplossen van een gemengd-geheel convex optimalisatie-probleem, wat het computationele voordeel van leergebaseerde controle teniet doet. Daarom introduceren we een optimalisatievrij veiligheidsfilter dat veilige regelcommando's bepaalt via algebraïsche en min-max-bewerkingen, wat de rekenlast aanzienlijk vermindert. Om het conservatisme van het veiligheidsfilter te verminderen,

onderwerpen we bovendien voorspellende veiligheidsfilters op basis van het concept van voorspellende *control barrier functions* (CBF's). Gegeneraliseerde Clarke-afgeleiden worden gebruikt om de niet-gladheid van het optimalisatieprobleem te adresseren. We tonen aan dat het afdwingen van CBF-beperkingen over alle componenten van de gegeneraliseerde Clarke-afgeleiden voldoende is om veiligheid te garanderen.

Daarnaast ontwikkelen we een algemeen optimalisatie-gebaseerd veiligheidskader voor niet-lineaire systemen met niet-convexe beperkingen. Bij toepassing op algemene niet-lineaire systemen vereisen traditionele veiligheidsfilters en MPC meestal een voorspellingsmodel, wat hun toepasbaarheid beperkt wanneer het model onbekend is. In tegenstelling hiermee kan het voorgestelde optimalisatie-gebaseerde kader voor veilige regeling zowel in modelgebaseerde als modelvrije omgevingen worden gebruikt. Bovendien is het computationeel efficiënt te implementeren en blijft het veilig presteren bij leerfouten. We introduceren *state-action control barrier functions* (SACBF's), waarmee efficiënte convex-kwadratische optimalisatie-gebaseerde veiligheidsfiltering mogelijk is. Zowel modelgebaseerde als data-gedreven methoden worden voorgesteld voor het construeren van SACBF's. Het effect van leerfouten op de effectiviteit van SACBF's wordt aangepakt via een fout-naar-toestand veiligheidsanalyse, die de basis legt voor een aanpak met aangescherpte beperkingen. Als uitbreiding naar de modelvrije context wordt directe data-gedreven veilige regeling voorgesteld, waarmee veilige regeling enkel en alleen op basis van toestandstransitiedata mogelijk is. Het voorgestelde optimalisatie-gebaseerde regelkader overbrugt de kloof tussen modelvrije leergebaseerde regeling en regeling met beperkingen door optimalisatie van de prestatie los te koppelen van het afdwingen van veiligheid.

Samengevat integreert dit proefschrift *machine learning* en optimalisatie om veilige en online efficiënte regelmethoden te ontwikkelen voor lineaire, PWA- en niet-lineaire systemen. Simulaties van taken zoals stabilisatie van een omgekeerde slinger, adaptieve snelheidsregeling, temperatuurregeling van een kamer en voertuigtrajectplanning tonen aanzienlijke verbeteringen in rekenefficiëntie en naleving van beperkingen ten opzichte van conventionele regelmethoden aan.

PREFACE

Completing this PhD has been one of the most challenging yet rewarding journeys of my life. This achievement is not mine alone. It is the result of the collective support, wisdom, and encouragement of many people who have accompanied me along the way. I would like to take this opportunity to express my gratitude to those who have given me help and support during my PhD period.

First, I would like to express my sincere gratitude to my supervisors, Prof. Bart De Schutter, Prof. Ton van den Boom, and Dr. Shengling Shi. Thank you, Bart, for accepting me and being my main supervisor. Your sharp mind, efficient working style, and passion for scientific research have always inspired me and taught me how to become a dedicated and competent researcher. Thank you, Ton, for offering me insightful guidance and valuable feedback on my research. I truly enjoyed our in-depth discussions on model predictive control and max-plus-linear networks during our meetings. Shengling has given me innovative ideas in research and also great support in improving the quality of my research papers. Thank you, Shengling, for always being there to provide timely feedback on my research. Your insightful feedback ensured that my research stayed on the right track.

Second, I would like to thank Prof. Johan Suykens for offering me the wonderful opportunity to conduct my research visit at KU Leuven. I greatly appreciated the opportunity to learn from you and your group. Our stimulating discussions, especially on topics such as neural network control and duality in reinforcement learning, broadened my perspective and enriched my understanding. Thank you for your mentorship, your time, and your kind support during my stay.

Third, I want to express my gratitude to my friends and (former) colleagues at TU Delft. Thanks to Xiaoyu Liu, Dingshan Sun, Yun Li, Changrui Liu, Yixuan Liu, Luyao Zhang, Diyou Liu, Zhixin Feng, Yang Wang, Ying Ma, Ruiyuan Li, Yifei Li, Shijie Huang, Jingwei Dong, Tian Tao, Jianfeng Fu, Ximan Wang for your companionship and help. I enjoyed the time with you during dinner, playing board games, playing badminton, traveling, and discussing scientific problems. Thanks to Dr. Anil Alan and Francesco Cordiano for our scientific discussions and cooperation in our collaborative work. Also, thanks to the other nice colleagues in Bart's group, including Kaya ter Burg, Giray Önür, Reza Riahi Samani, Sam Mallick, Gianpietro Battocletti, Alessandro Riccardi, Caio Fabio Oliveira da Silva, Athina Ilioudi, and Amala Mary Vincent. Special thanks to our secretaries for their kind support.

Finally, I owe my deepest gratitude to my family, whose unconditional love and unwavering support have been my greatest source of strength throughout this journey. To my parents, thank you for always believing in me and encouraging me to start this journey.

Your guidance, through both words and actions, has profoundly shaped who I am. Even when I was far from home, your values and strength gave me the courage to stand on my own and face challenges with confidence. It is your support that allowed me to plant my feet firmly in a foreign land and grow, both as a researcher and as a person. I carry your love and teachings with me in everything I do. To my wife, Yue, thank you for always encouraging me when I felt depressed. Thank you for making my life colorful and filling my life with love, joy, and purpose. Last but not least, thanks to my newborn baby, Zhixing Lucas He. Your presence gave me the courage to keep going, even in the most difficult moments. I wish you a lifetime of happiness and good health, and I hope all your dreams come true.

Kanghui He
Delft, August 2025

1

INTRODUCTION

1.1. BACKGROUND

The advancement of modern control systems has brought remarkable improvements in various engineering fields, ranging from autonomous vehicles [91] to smart multi-energy networks [159] and beyond. Some of these advancements in the last 10 years can largely be attributed to the enhanced computing capabilities of microcomputers, the increasing richness of data, and the emergence of machine learning. Machine learning, especially deep learning using neural networks [125], can capture complex mappings directly from data, generalize to unforeseen scenarios, and optimize multiple learning objectives. These benefits make machine learning a powerful tool for both system modeling and controller synthesis. Now, there is a growing body of research, from both theoretical and practical perspectives, focusing on integrating machine learning into controller design, a field commonly referred to as learning-based control [112]. However, to optimize performance and to explore unknown scenarios, learning-based controllers could unavoidably produce radical and risky actions, which may have harmful effects on system components.

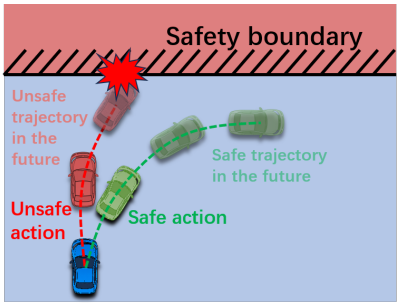
1.1.1. SAFETY IN SYSTEMS AND CONTROL

Safety in systems and control is a multifaceted concept that can vary depending on the context and application. For example, safety can mean resilience against denial-of-service attacks in cyber-security [43], fault tolerance to ensure reliable operation despite component failures [9], satisfaction of constraints on cumulative rewards in constrained Markov decision processes [189], or adherence to temporal logic specifications [146]. In this PhD thesis, safety is specifically defined as the ability of *satisfying given hard state and input constraints over the entire evolution of the system.* The main challenge of pursuing safety in this context is that unsafe control inputs may not immediately cause state constraint violations at the moment, but could lead to violations of some constraints in the future. Let us take a simple obstacle avoidance problem as an example (see Fig. 1.1a).

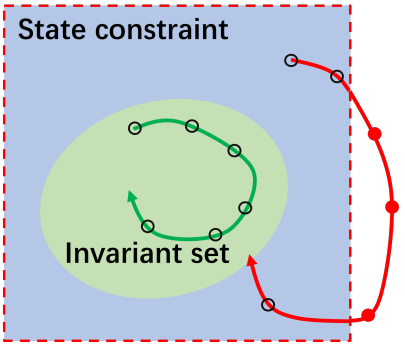
1

Suppose that an autonomous vehicle approaches an obstacle closely at a high speed. In that case, it does not record any constraint violation at the moment, but it could happen that the vehicle will eventually crash into the obstacle even if the maximum braking force or the maximum steering angle is applied.

To ensure safety, various methods have been proposed in the control community, including control barrier functions (CBFs) [8], reachability-based methods [4], and predictive control methods [198]. All three approaches are fundamentally rooted in the theory of set invariance [30]. An invariant set (also referred to as a safe set) is a set of states such that the system will remain within that set for all future time steps. CBFs and reachability-based methods find explicit representations of invariant sets, while predictive control methods implicitly define invariant sets through the feasible set of an open-loop constrained optimal control problem. In this PhD thesis, we investigate the interaction of these three methods with machine learning, in the aspects of learning safe sets as well as learning safe controllers.



(a) Safety problems in vehicle control. Unsafe control actions can lead to crashes in the future, depicted by the red curve. The green curve represents the trajectory from safe actions.



(b) An illustration of an invariant set, usually a subset of the state constraint set. Systems starting within the invariant set remain in this set, while systems starting outside the maximum invariant set (i.e., the largest maximum invariant set containing all invariant sets) can not always satisfy the constraint.

Figure 1.1: Geometric illustration of safety and invariant sets.

It should be noted that ensuring safety alone is not a challenging issue, as it is usually possible to keep the system stationary within the most trustworthy region. For example, drivers can always park their cars in the parking space to avoid dangerous accidents. The main difficulty, on the other hand, is achieving optimal control performance and simultaneously guaranteeing safety. To this end, model predictive control (MPC) [166] and safe or constrained learning-based control [10], [39] have been developed.

1.1.2. MODEL PREDICTIVE CONTROL

MPC is an optimization-based control method that solves a finite-horizon optimal control problem in real time to determine control inputs. MPC is a promising approach

for ensuring safety by including state and input constraints directly in the online optimization problem. Meanwhile, MPC can achieve sub-optimal performance[1] by directly considering the desired performance criteria as a (multi-objective) cost function in the optimization problem. To predict the system's behavior over future time steps, MPC requires a prediction model, which can be derived from either physical principles or machine learning techniques.

Despite its advantages, MPC also has significant limitations. The main roadblock lies in the fact that the online optimization problem is, in general, nonlinear and non-convex for nonlinear and hybrid systems. This results in a **huge computational burden**, which makes MPC unsuitable for situations where **swift decision-making** is crucial. Explicit MPC for linear and piecewise affine (PWA) systems addresses this issue by precomputing the mapping from the current state to the optimal control input offline using multiparametric optimization [35]. This mapping is then stored for online implementation. However, explicit MPC suffers from exponentially increasing offline computational complexity and storage requirements as the problem size expands, posing a critical limitation for large-scale systems [5]. Another limitation of MPC is its reliance on an accurate prediction model. For complex systems, modeling errors inherently exist and their effects on performance tend to be amplified in MPC optimization problems [180], particularly when a long prediction horizon is necessary to ensure long-term safety and to approach the infinite-horizon optimal policy. Over the past two decades, several robust and stochastic MPC methods have been developed to handle uncertainty. By requiring specific assumptions on uncertainty, these methods can offer robust or probabilistic guarantees on safety and performance [42], [150].

1.1.3. LEARNING-BASED CONTROL

Learning-based control leverages machine learning techniques to develop adaptive control policies. Similar to human learning, where children often imitate the behaviors of their parents or receive rewards when they perform well independently, learning-based control can be categorized into two main groups: supervised learning (SL)-based control and reinforcement learning (RL)-based control. In SL-based control, the controller learns from labeled data, which is collected from, e.g., an expert or experiments on optimal control. The control policy is trained using state-input pairs to minimize the difference between predicted and target control signals from the expert. In contrast, RL-based control learns through interactions with the controlled system and environment or from previously collected offline data, obtaining rewards or penalties based on its actions. RL can handle model uncertainty effectively because it learns directly from such interactions. Over many iterations, the RL agent can refine its policy to maximize a user-specified long-term cumulative reward.

Learning-based control can also be broadly classified into online and offline variants. Online learning methods [56], primarily considered in RL, improve the policy by interacting with the controlled system and environment using the latest learned policy (referred to as on-policy methods) or any other policy (referred to as off-policy meth-

[1] The sub-optimality can arise from several factors, including the finite prediction horizon, model uncertainty, the receding horizon strategy, and the sub-optimality of numerical optimization.

ods). The key advantage of online methods is that they are adaptive to a changing environment. However, online policy updates often require substantial computational resources. Besides, for safety-critical problems, online interaction with the real system and environment could be dangerous and thus impractical. These challenges have led to the development of offline learning-based control, in which the agent learns policies merely from previously collected data [126]. Typically, the dataset is generated from simulators, allowing for large-scale data collection and enabling the inclusion of potentially unsafe policies during training without real-world consequences. Once the training is finished, the policy will not be altered anymore in the online phase, making it more efficient for online implementation. In this PhD thesis, to address the online computational issues of MPC, **we focus on offline learning methods.**

Meanwhile, learning-based control can also be categorized into model-based methods and model-free methods. Most RL methods, including approximate value and policy iteration, deep deterministic policy gradient, trust region policy optimization, and proximal policy optimization, have both model-based and model-free variants [41]. In the model-free RL setting [70], the controller interacts directly with the real system to collect data, which is then used to train the control policy. In contrast, model-based RL first constructs a nominal model of the system and uses this model to generate training data. Model-based RL is generally more sample-efficient than its model-free counterpart, as it enables learning through the approximation of classical state value functions rather than the state–action value functions typically used in model-free RL. However, the convergence analysis of model-based RL is more challenging, as it must account for approximation errors arising from both model inaccuracies and value/policy function approximations. When safety constraints are considered, model-based RL becomes particularly appealing because it allows for potential constraint violations through simulations of the nominal model. In contrast, model-free RL requires safe exploration, i.e., ensuring that safety constraints are satisfied throughout the learning process, as it interacts directly with the real system. For SL, a representative model-free example is learning behaviors of humans [158], as humans typically do not rely on explicit mathematical models. Recently, there has been increasing interest in learning explicit MPC policies [44], [114], [122], which belongs to the class of model-based SL methods. In this PhD thesis, to improve sampling efficiency, we primarily focus on explicit model utilization, while in Chapter 7, we explore the extension to a model-free setting.

The main advantage of learning-based control over MPC is that it can enable fast computation of control inputs using policy function approximation. Moreover, by updating policies from state-input data, learning-based control bypasses the necessity for accurate models or hand-crafted controller design. By leveraging techniques from constrained stochastic optimization, RL can achieve probabilistic safety guarantees under model uncertainty [157]. Nevertheless, a notable drawback of learning-based control is its potential lack of deterministic safety, i.e., it struggles to deal with complex hard constraints on states and inputs. This drawback stems from the intrinsic randomness of the learning process, the limited capability of function approximation, as well as data insufficiency.

To enhance safety in learning-based control, existing research typically follows two main

methodologies: 1. **offline safety enforcement** and 2. **online constrained action optimization**.

1. Offline safety enforcement directly enforces safety constraints throughout the learning process, including methods such as constrained policy optimization [2], reward or cost shaping [29], [189], and Lagrangian RL [128]. These methods are mostly inspired by constrained optimization [26].

2. Online constrained action optimization utilizes safety filters to regulate potentially unsafe control actions generated by standard learning methods [123], [149], [196], [198].

We argue that **safe learning-based control cannot be truly achieved without using online optimization to guarantee explicit constraint satisfaction**. Therefore, this PhD thesis primarily focuses on online optimization methods to determine a safe policy. On the other hand, it is evident in the literature that including safety constraints during the learning process can improve task performance [53]. Therefore, in Chapters 2, 3, and 7, we investigate the combination of both methodologies.

1.2. MOTIVATION AND CHALLENGES

It can be observed that several advantages and disadvantages of MPC and learning-based control are nearly complementary. Specifically, weaknesses of either approach are strengths of the other. A detailed comparison of MPC and learning-based control is provided in Table 1.1, and also in some comprehensive surveys [19], [87], [168]. Given this observation, many researchers have explored the integration of MPC and learning-based control to overcome their respective limitations. For example, RL algorithms can be used to update parameterized MPC to produce safe and stabilizing policies [90], [154]. MPC, on the other hand, can serve as an expert or critic to guide the updates of learning-based control policies [84]. Besides, MPC policies can be used to warm start the training of RL policies [116]. Conversely, machine learning can be used to determine a part of the decision variables in MPC [145].

Even though these integrations make the resulting control frameworks more robust to uncertainties than MPC and safer and more reliable than learning-based control, we argue that **simultaneous safety guarantees and online computational efficiency have not yet been achieved in the literature related to such integrations**. This statement is based on the following reasoning: No matter how the two methods are integrated, control actions are determined either by parameterized MPC or by parameterized explicit functions. Learnable MPC can produce safe control actions but is still computationally expensive, while explicit functions offer low inference times but in general cannot ensure complete constraint satisfaction.

Given the limitations of MPC, learning-based control, and their integrations, we consider integrating learning-based control with efficient online optimization methods, such as constrained approximate dynamic programming (ADP) (which can be viewed as a one-step MPC with terminal value function approximation) and learning-based control with safety filters [81], [154], [197], [198]. As discussed in Section 1.1.1, safety filters in-

Table 1.1: Comparison of MPC and Learning-Based Control.

	Model Predictive Control (MPC)	**Learning-Based Control**
Online computational complexity	Typically high for PWA and nonlinear systems	Low
Safety and stability	Has formal guarantees under assumptions	Does not have guarantees unless combined with formal methods or safe exploration strategies
Handling uncertainties	Robust and stochastic MPC can handle model uncertainties.	Can inherently deal with uncertainties by exploring the environment
Model dependence	Requires a prediction model	Can learn directly from data, without dependency on explicit models
Adaptability	Can adapt to model changes by adjusting the prediction model	Needs re-training or online learning

volve using an invariant set to modify policies, aiming to preserve task performance by keeping the invariant set as large as possible. For nonlinear systems, it is in general intractable to synthesize a permissive invariant set. This motivates the development of using learning-based methods to learn invariant sets [4], [32], [47], [77], [82], [121], [173], [196], [201] (and safe policies in parallel [138], [211]). However, unavoidable learning errors may arise in the approximation of value functions, invariant sets, and policies, potentially compromising task performance and safety guarantees. This is one of the major open issues of safe learning-based control.

From a computational perspective, it should be remarked that for discrete-time PWA or nonlinear systems, online policy refinement in general includes solving a non-convex nonlinear optimization problem, which may require excessive computational time to obtain a sufficiently optimal solution [64], [103]. Moreover, achieving near-optimal control performance necessitates precise value function approximations and large invariant sets, often relying on sophisticated learning models such as deep neural networks. However, employing complex function approximations significantly increases the computational burden. In conclusion, there naturally exists a **trade-off among accuracy, optimality, and computational effort**.

1.3. SCOPE AND RESEARCH QUESTION

In this PhD thesis, we address the above challenges in the realm of constrained control of linear, PWA, and nonlinear systems. The central research question is

How to effectively integrate optimization and learning-based control, in such a way that performance optimality, safety, and computational tractability are preserved?

To answer this question, we capitalize on the adaptability of machine learning approaches and merge them with classical optimization and control approaches. In this PhD thesis,

we demonstrate how to combine these methods to develop computationally efficient and provably safe learning-based control algorithms.

Specifically, offline learning is employed to address unknown components in constrained optimal control problems, including optimal value functions, optimal policies, and invariant constraints. We study what kinds of learning objectives and function approximators should be adopted to improve control performance while adhering to computational time limits.

In our work, online optimization is mainly responsible for generating safe control actions. We study how to efficiently solve the optimization problem that determines the control actions or to approximate its solution. Besides, we also investigate how to use global optimization techniques, such as mixed-integer linear programming (MILP) [169], to offline certify the stability and safety of learning-based controllers. This helps us to assess at least whether a learned controller is practically stable and satisfies safety constraints, even when its performance is not guaranteed in theory.

Moreover, we utilize some control-theoretic approaches, such as control barrier functions, Lyapunov functions [118], reachability analysis, dynamic programming [25], (robust) MPC [106], and data-driven control [72], to establish formally sound performance guarantees for learning-based control.

Meanwhile, we advocate for the use of **PWA approximations on both systems and constraints**. PWA functions can approximate nonlinear functions and model hybrid phenomena and are the simplest extension of linear/affine functions [89]. By using PWA approximations, we can take advantage of rich control and optimization tools, such as polytopic reachability analysis [130] and mixed-integer programming, to balance between computational complexity and control accuracy.

1.4. CONTRIBUTIONS

The main contributions of the PhD thesis are introduced next:

1. We propose a novel approach combining ADP and MPC for optimal control of linear systems with linear constraints, more precisely, infinite-horizon constrained linear quadratic regulation problems. MPC uses a finite yet long horizon to approximate the infinite-horizon value function, but it struggles in high-speed applications due to computational demands. To effectively approximate the value function, we propose a novel convex and piecewise quadratic neural network, which is incorporated as the terminal cost in the MPC problem with the horizon equal to 1 to determine control inputs. When an accurate approximation is achieved, the proposed method guarantees both stability and feasibility. Additionally, we introduce an efficient decomposition algorithm to solve the online optimization problem, providing a rigorous analysis of finite-time termination and computational complexity.

2. We develop a constrained ADP method for optimal control of PWA systems with non-convex union-of-polyhedra state constraints. We utilize PWA penalty functions to manage these state constraints. Specifically, we propose two formulations

for integrating PWA penalties: one that adds penalties to the stage cost and another that embeds penalties directly into the cost-to-go. Our key finding is that embedding penalties into the cost-to-go, which is rarely considered in the literature, offers greater sampling efficiency and improved approximation accuracy. Besides, we conduct a systematic performance analysis to evaluate the feasibility, stability, and sub-optimality of the resulting learning-based controllers. The theoretical analysis provides guidance and recommendations for parameter tuning in the proposed ADP algorithms. Moreover, for scenarios where the theoretical results do not apply, we develop a mixed-integer optimization-based framework to verify the practical stability and safety of the closed-loop system.

3. We design safety filters for PWA systems with state constraints represented as a union of polyhedra and ellipsoids. We develop, for the first time, an optimization-free safety filter that can rapidly determine safe control inputs based on algebraic operations only. A piecewise affine and quadratic classification algorithm is proposed to learn the invariant set, which is a crucial component of the safety filter.

4. We also design, for the first time, predictive safety filters for continuous-time PWA systems with general constraints. Predictive safety filters reduce the conservatism of standard safety filters, but require the smoothness of constraints and dynamics [52], which is not the case in our problem setting. To bridge this gap, we construct a predictive safety filter that leverages generalized Clarke sensitivity [58] to address the possibly non-smooth nature of the constraints and dynamics. We rigorously prove the recursive feasibility and long-term safety of the predictive safety filter. Moreover, we develop a computationally tractable way to calculate the generalized Clarke sensitivity, which is essential for practical implementation. Furthermore, we derive an explicit approximation of the safety filter solution, contributing to the further reduction of the online computational demand.

5. We propose a novel concept called state-action control barrier functions (SACBFs) to handle optimal control of nonlinear systems with nonlinear constraints. The superiority of SACBFs over existing safety certificates lies in their flexibility in parameterization. In particular, SACBFs can enforce safety constraints through a convex quadratic optimization problem, significantly reducing the computational burden typically faced by classical safety certificates such as CBFs. In addition, we develop a learning-based method to approximate SACBFs that is primarily built on reachability analysis. Furthermore, we propose a constraint-tightening approach to eliminate the effect of learning errors on the validity of SACBFs.

6. Motivated by the challenges of accurately modeling complex nonlinear systems, we propose an optimization-based, direct data-driven safe control scheme for nonlinear constrained control in the model-free setting. The scheme makes use of the aforementioned SACBF to enforce safety constraints. The advantage is that the control scheme can be trained and implemented using state transition data only. We develop three different learning-based methods for synthesizing SACBFs directly from data. We propose a systematic framework, called Error-to-State Safety (ESSf), to quantify how learning-induced errors in SACBFs affect overall safety per-

formance. This motivates the approach of state constraint tightening followed by SACBF constraint relaxation to ensure safety. Compared to the constraint-tightening approach in the fifth contribution, the newly developed approach exploits the inherent robustness of SACBFs and is thus less conservative. Furthermore, we explore the interaction between task performance and safety under the proposed control scheme. The main advantage of the proposed integrated optimization and learning control framework is that it separates the learning of the optimal controller into two components: optimizing performance through learning the objective function, and ensuring safety through learning the constraints.

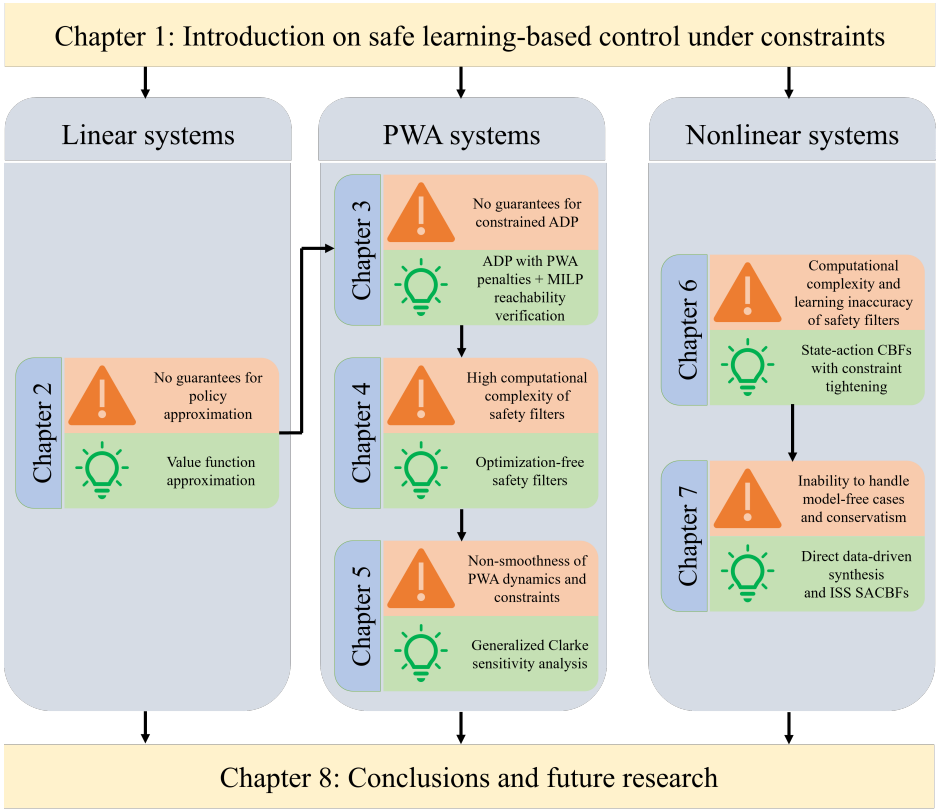


Figure 1.2: Structure of this PhD thesis. Orange triangles mark key open challenges identified in the literature, while lightbulbs represent the core contributions of this PhD thesis addressing those challenges. "ISS" means "input-to-state safety".

1.5. ORGANIZATION

The structure of the PhD thesis is illustrated in Fig. 1.2. Chapters 2-7 represent a collection of individual papers that have either been submitted for publication or have already been published. As different chapters focus on different aspects of safe learning-based control, the mathematical notations are defined for each chapter separately.

Table 1.2: Connections and comparisons of Chapters 2 to 7.

Chapters	**Learning method**	**Handling of constraints**	**Safety assurance**	**Computational considerations**
Chapter 2	Value function approximation	Computation of invariant sets	✓ under assumptions	Efficient piecewise quadratic optimization algorithms
Chapter 3	ADP	External penalty functions	✓ under assumptions	Explicit control policies
Chapter 4	Not specified	Use of classification to learn invariant sets	✗	Optimization-free safety filters
Chapter 5	Not specified	Predictive CBFs	✓	Approximate explicit solutions
Chapter 6	SL	SACBFs	✓ under assumptions	Convex quadratic parameterization
Chapter 7	Not specified	SACBFs	✓ under assumptions	No considerations

In Table 1.2, we present the connections and comparisons of Chapters 2 to 7, focusing on key aspects such as the learning method employed, the approach to handling constraints, the assurance of safety, and the consideration of computational efficiency.

The main contents of all chapters are introduced as follows. Chapter 1 gives a brief introduction to safety, model predictive control, and learning-based control, highlights the unsolved questions, and lists the contributions of the PhD thesis. Chapter 2 develops an ADP algorithm for linear systems with linear constraints. In Chapter 3, ADP algorithms are proposed for constrained PWA systems. Chapter 4 proposes an optimization-free safety filter for constrained PWA systems. In Chapter 5, a predictive safety filter for constrained PWA systems is proposed. Chapter 6 considers safety filter design for general nonlinear constrained control problems. Chapter 7 further presents an optimization-based safe control framework for unknown nonlinear systems.

2

APPROXIMATE DYNAMIC PROGRAMMING FOR CONSTRAINED LINEAR SYSTEMS

Approximate dynamic programming (ADP) faces challenges in dealing with constraints in control problems. Model predictive control (MPC) is, in comparison, well-known for its accommodation of constraints and stability guarantees, although its computation is sometimes prohibitive. This chapter introduces an approach combining the two methodologies to overcome their individual limitations. The predictive control law for constrained linear quadratic regulation (CLQR) problems has been proven to be piecewise affine (PWA) while the value function is piecewise quadratic. We exploit these formal results from MPC to design an ADP method for CLQR problems with a known model. A novel convex and piecewise quadratic neural network with a local-global architecture is proposed to provide an accurate approximation of the value function, which is used as the cost-to-go function in the online dynamic programming problem. An efficient decomposition algorithm is developed to generate the control policy and speed up the online computation. Rigorous stability analysis of the closed-loop system is conducted for the proposed control scheme under the condition that a good approximation of the value function is achieved. Comparative simulations are carried out to demonstrate the potential of the proposed method in terms of online computation and optimality.

This chapter is based on the paper [101] and its extended version [97].

2.1. INTRODUCTION

Model-based reinforcement learning (RL), also known as (approximate) dynamic programming ((A)DP) [204], has received much attention for the synthesis of controllers. Compared to its diverse industrial applications, the theoretical analysis on the stability and safety for RL faces great challenges. Thanks to Lyapunov stability theory, the stability issue of ADP has been comprehensively addressed, both for linear [119] and nonlinear systems [24], [186]. Although ADP approaches consider stability, safety is another important issue that needs further study. Safety means that the states and inputs of closed-loop systems satisfy some constraints. Techniques employed by RL or ADP approaches for dealing with constraints can be grouped into two categories: policy-projection-based methods and policy-optimization-based methods. Policy-projection-based methods consider the RL formulation in the unconstrained case and involve a regulator to check and modify the policies that may violate the constraints [50], [198]. These indirect methods, however, sometimes fail to capture the optimal solution of the constrained problem and lack stability guarantees. In comparison, optimization-based methods intend to directly get the optimal value function for the constrained problems by solving the constrained Bellman equation. With the optimal value function available, the optimal control policy can thereby be produced by solving a constrained policy optimization problem. [46] was the first investigation of this kind of method, but it is limited to searching for the best linear feedback control law and needs an initial stabilizing policy. Following this direction, [45] explores an estimation approach to find an initial stabilizing policy even when there are uncertain nonlinear dynamics.

In constrained cases, neither the optimal policy nor the optimal value function is readily available, even for the most basic infinite-horizon linear quadratic regulation (LQR) problems. This, to some extent, restricts the development of the policy-optimization-based RL methods for constrained control problems. In comparison, model predictive control (MPC) [35], an optimization-based control scheme widely adopted in the control community, has a mature stability and robustness theory as well as an inherent constraint handling. For infinite-horizon LQR problems, MPC can render the exact optimal control law due to the equivalence of finite and infinite optimal control as long as the horizon is sufficiently long [54]. With this fundamental property, infinite-horizon LQR problems can be solved by either implementing MPC online or explicit MPC [22], and the solution is proven to be piecewise affine (PWA) in the state space [22]. However, the computational complexity of both online MPC and explicit MPC grows dramatically with the increase of the problem size. This is one of the main drawbacks of MPC compared with RL or ADP.

Dedicated to overcoming this drawback, approximation methods of predictive control laws have received much attention in recent years. An emerging methodology is using specific function regression techniques such as PWA neural networks (NNs) [50], [114], [139]. Nevertheless, no guarantees of closed-loop stability are conveniently available, even though the final approximation error is small enough. Using NN-based controllers to warm start the solver in online MPC [51] can inherently guarantee stability, and learning Lyapunov functions to verify the stability of NN-based controllers [48] is also an alternative way. However, additional computation is required in the optimization or learning

procedure.

Observing that MPC has computational limitations and approximation in the policy space lacks performance guarantees, we aim to attain a computationally inexpensive control scheme for linear systems with stability and feasibility guarantees. To this end, we approximate the value function via ADP and shorten the prediction horizon to one. We focus on the infinite-horizon LQR problem with state and input constraints. Different from the research on approximating the MPC policy [50], [114], [139], we propose a value function approximation scheme. The optimal value function that is characterized by explicit MPC is approximated by a piecewise quadratic (PWQ) NN. The synthesis of the control law is conducted in a DP problem, where stability, feasibility, and suboptimality can be guaranteed if a good approximation is obtained. Meanwhile, note that one disadvantage of approximating the value function is that it needs online policy optimization. To address this issue, we develop algorithms so that online optimization can be done efficiently.

The contributions of the chapter are highlighted as follows:

1. We propose a novel NN structure to approximate the solution of the constrained LQR problem based on ADP. The proposed NN structure can capture the PWQ and convex properties of the value function. Different from policy-based approximation approaches [50], [114], [139], our approach has the important advantage that the resulting controller has safety and stability guarantees.

2. We propose an efficient algorithm to solve the policy optimization problem, which is a convex piecewise quadratic program. In particular, this program is simplified to a collection of quadratic programs (QPs). Note that the main difficulty that prevents one from considering more complex approximation structures is the increase in online computational time. We solve this problem in Algorithm 1. Complexity analysis and simulation results show that the proposed method requires much less online computation time than implicit MPC.

3. Compared to [46], the first exploration of ADP in a constrained LQR setting, the proposed approach eliminates the restriction of searching for a linear feedback law and does not require an initially stabilizing policy nor an initial state belonging to an invariant set.

4. We do a rigorous stability analysis, give stability conditions, and provide a tractable way to verify them.

2.2. PRELIMINARIES

Let $\mathbb{N} = \{0, 1, 2, ...\}$ and let $\lambda_{\max}(P)$ and $\lambda_{\min}(P)$ represent the maximum and minimum eigenvalues of a symmetric positive definite matrix P. The boundary of the set $\mathcal{P}$ is $\partial\mathcal{P}$, and $\text{int}(\mathcal{P})$ stands for the interior of $\mathcal{P}$. We use $A_{i,\cdot}$ to represent the ith row of the matrix A.

2.2.1. INFINITE-HORIZON OPTIMAL CONTROL AND MPC

We study the infinite-horizon constrained linear quadratic regulation (CLQR) problem

$$\begin{aligned} J_\infty^*(x) = \min_{U^\infty} & \left\{ J_\infty\left(x_0, U^\infty\right) := \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \right\} \\ \text{s.t. } & x_{k+1} = Ax_k + Bu_k, k = 0,1,\ldots, \; x_0 = x \\ & x_k \in \mathcal{X}, u_k \in \mathcal{U}, k = 0,1,\ldots, \end{aligned} \tag{2.1}$$

where $x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$, $\mathcal{X}$ and $\mathcal{U}$ are polyhedra that contain the origin in their interior, and $U^\infty = [u_0, u_1, ..., u_\infty]$ is the infinite-dimensional decision variable. Matrices A and B are known. Like [50], [117], the following assumption is supposed to hold throughout the chapter.

Assumption 2.2.1. (A, B) is stabilizable, $Q \succeq 0$, and $R \succ 0$. Moreover, there exists an initial state, such that there exists a sequence of admissible input vectors u_0, u_1,... that can steer the state to the origin, i.e., $\bar{X} := \{x \in \mathbb{R}^n | \exists U^\infty$ s.t. $x_k \in \mathcal{X}$, $u_k \in \mathcal{U}$, and $J_\infty^*(x) < \infty$, $\forall k \in \mathbb{N}\}$ is not empty.

With Assumption 2.2.1, let $K \in \mathbb{R}^{m\times n}$ be a stabilizing gain matrix for the unconstrained plant $x_{k+1} = Ax_k + Bu_k$. As in [54], we consider the admissible set of states under a given stabilizing control gain K as $\bar{\mathcal{X}}_K = \{x \in \mathbb{R}^n \mid x \in \mathcal{X}, -Kx \in \mathcal{U}\}$.

If the constraints in (2.1) are removed, the problem admits a unique linear solution $u_k^{\text{LQR}} = -K^* x_k$, $k = 0,1,\ldots$ where $K^* = \left(R + B^T P^* B\right)^{-1} B^T P^* A$, and $P^* = P^{*T}$ is the unique positive-definite solution of the algebraic Riccati equation [119]. Then, $J_\infty^*(x) = x^T P^* x$ in the absence of constraints.

In the constrained case, if the state is close to the origin, the unconstrained LQR solution $u_k^{\text{LQR}} = -K^* x_k$, $k = 0,1,\ldots$ may not violate the constraints so that the solution to (2.1) is identical to the unconstrained LQR solution $-K^* x_k$ as if there is no constraint at all. This motivates the consideration of the maximal LQR invariant set $\mathcal{O}_\infty^{\text{LQR}} = \{x \in \mathbb{R}^n \mid (A - BK^*)^k x \in \bar{\mathcal{X}}_{K^*}, \forall k \in \mathbb{N} \subseteq \mathbb{R}^n\}$ for the autonomous constrained linear system $x_{k+1} = (A - BK^*) x_k$, $x_k \in \mathcal{X}, \forall k \in \mathbb{N}$ [35], [54]. With this definition, the existing literature [54] considers using the finite-horizon problem

$$\begin{aligned} J_N^*(x) = \min_{\{u_k\}_{k=0}^{N-1}} & \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P^* x_N \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \; k = 0,1,\ldots,N-1, x_0 = x \\ & x_k \in \mathcal{X}, \; u_k \in \mathcal{U}, \; k = 0,1,\ldots,N-1 \end{aligned} \tag{2.2}$$

to approximate the infinite-horizon LQR problem 2.1. In (2.2), $U = [u_0, u_1, ..., u_{N-1}]$ is the decision variable and $U^*(x)$ is the solution. The rationale behind this design is that after a sufficiently long horizon, the resulting x_N will fall into $\mathcal{O}_\infty^{\text{LQR}}$ where the unconstrained LQR solution $u_N = -K^* x_N$ will not violate the constraints. As a result, the terminal cost $x_N^T P^* x_N$ is an exact representation for $J_\infty^*(x_N)$. The following theorem, proposed in [54], formally demonstrates the validity of this design.

Theorem 2.2.1 ([54]). Let $\bar{J}$ be an upper bound on $J_\infty^*(x)$. Suppose that Assumption 2.2.1 holds. For any $x \in \bar{X}$, if $N > (\bar{J} - p)/q$ where $0 < q \leq q_\mathrm{m} := \inf_{x \notin \mathcal{O}_\infty^{\mathrm{LQR}}} \{x^\mathrm{T} Q x\}$ and $0 < p \leq p_\mathrm{m} \hat{=} \inf_{x \notin \mathcal{O}_\infty^{\mathrm{LQR}}} \{x^\mathrm{T} P x\}$, then $x_N \in \mathcal{O}_\infty^{\mathrm{LQR}}$ and problem (2.2) is equivalent to problem 2.1 in the sense of $J_\infty^*(x) = J_N^*(x)$, $\forall x \in \bar{X}$.

According to Theorem 2.2.1, the horizon N is essential to the equivalence of the finite-horizon and infinite-horizon problems. For a given x, one can repeatedly solve problem (2.2) with an increasing N until the terminal state falls into $\mathcal{O}_\infty^{\mathrm{LQR}}$ [179]. This approach only works for a certain state value. Alternatively, [54] gives a conservative estimate of the horizon for all x belonging to a polyhedral set X_0 based on the result in Theorem 2.2.1. Specifically, [54] computes p_m, q_m, and J_∞^* at the vertices of X_0. Next, $\bar{J}$ is set to be maximum of all J_∞^*, and N is selected to be the smallest integer such that $N \geq (\bar{J} - p_\mathrm{m})/q_\mathrm{m}$. Throughout the chapter, we assume that N in (2.2) is chosen such that the equivalence in Theorem 2.2.1 is satisfied.

By substituting the state update equation into $J_N^*(x)$ and the constraints, (2.2) can be reformulated as a multi-parametric quadratic program (mpQP). See [35], [97] for details. Explicit MPC aims to directly solve the mpQP for all possible x in a feasible set, and find the relationship between the optimizer $U^*(x)$ and x. The results from explicit MPC show that the optimizer $U^*(\cdot) : \mathbb{R}^n \to \mathbb{R}^{mN}$ is PWA. So, the first element $u_0^*(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ of U^* satisfies

$$u_0^*(x) = F_j x + g_j, \text{ if } x \in \mathcal{R}_j, \quad j = 1, \ldots, N_\mathrm{r}, \tag{2.3}$$

where the polyhedral sets $\mathcal{R}_j = \{x \in \mathbb{R}^n : H_j x \leq h_j\}$, $j = 1, \ldots, N_\mathrm{r}$ constitute a finite partition of a compact set of initial conditions $X_0 \subseteq \bar{X}$.

2.2.2. PROBLEM FORMULATION

The explicit controller is easy to compute offline in the case of a short horizon and a low-dimensional input vector. However, with the increase of the horizon and the system's dimension, the number of regions in (2.3) grows exponentially [22] and the representations of these regions become more complex, which may make the offline computation and online implementation intractable.

Artificial NNs with at least one hidden layer have the capability of universal approximation for any continuous function, provided that the hidden layer has enough units. Herein, we can use NNs to represent the explicit MPC law, without any need to identify the regions in (2.3). Related work is reported in [50], [114], [139], combined with supervised learning or policy gradient methods. The NN-based controllers proposed in [50], [114], [139] require an online feasibility certificate to determine whether the outputs of the NNs are safe. These NN-based policy approximators, however, inherently lack stability guarantees.

To address the computational burden of MPC and the lack of guarantees of policy-based approximation, we adopt value function approximation and shorten the MPC horizon to one. The main challenges are thereby (i) the design of the value function approximator, which is expected to output a function akin to the optimal value function, (ii) the relation between those guarantees and the quality of the approximation, and (iii) further

reduction of online computation time concerning that the one-step problem contains a function approximator.

Before establishing our approximation structure, we concentrate on the properties of J_∞^*.

Theorem 2.2.2 ([35])**.** With Assumption 2.2.1 satisfied, in a compact polyhedral set of the initial conditions $X_0 \subseteq \bar{X}$, J_∞^* is continuous, convex, and PWQ over polyhedra

$$J_\infty^*(x) = J_i(x) := x^T P_i x + q_i^T x + v_i, \text{ if } x \in \mathcal{R}_i,\ i = 1, \dots, N_\mathrm{r}. \tag{2.4}$$

Moreover, if the mpQP problem (2.2) is not degenerate, then J_∞^* is continuously differentiable.

Theorem 2.2.3 ([17])**.** Assume that (2.1) results in a non-degenerate mpQP. Let $\mathcal{R}_i, \mathcal{R}_j$ be two neighboring polyhedra and $\mathcal{A}_i, \mathcal{A}_j$ be the corresponding sets of active constraints at the optimum of (2.2). Then, (i) $P_i - P_j \leq 0$ if $\mathcal{A}_i \subset \mathcal{A}_j$, and (ii) $P^* - P_j \leq 0$, $\forall j \in \{1, 2, \dots, N_\mathrm{r}\}$.

For the detailed description of degeneracy, see [97], [192].

2.3. ADP DESIGN USING CONVEX PIECEWISE QUADRATIC NEURAL NETWORKS

2.3.1. NN DESIGN FOR APPROXIMATION IN VALUE SPACE

According to Theorems 2.2.2 and 2.2.3, the value function approximator, denoted by $\hat{J}(\cdot, \theta)$ where θ refers to some parameters, is expected to have the following features: (F1) It can partition its input space into polyhedral regions. (F2) It can produce a convex and PWQ function partitioned by polyhedra. (F3) For x in a small region containing the origin, the approximator can provide the exact representation for the value function, i.e., $\hat{J}(x, \theta) = x^T P^* x$.

We will use a feed-forward NN to capture the relationship between the state x and $J_\infty^*(x)$. A feed-forward NN is composed of one or more hidden layers and an output layer, where each hidden layer contains an affine map $f_l(\kappa_{l-1}) = W_l \kappa_{l-1} + b_l$, followed by a nonlinear map $\kappa_l = g_l(f_l)$. Here, $W_l \in \mathbb{R}^{M_l \times M_{l-1}}$ and $b_l \in \mathbb{R}^{M_l}$ are the weights and biases, $g_l(\cdot) : \mathbb{R}^{M_l} \to \mathbb{R}^{M_l}$ is a nonlinear activation function, M_l is the width of the lth layer, referring to the number of units in the layer, and $M_0 = n$. Based on these definitions, an NN with L hidden layers and M_l units in the lth layer can be represented by $f_{\mathrm{NN}}(x, \theta) = [f_{L+1} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1](x)$, where θ contains all the weights and biases of the affine functions in all the layers, and the symbol $\circ$ means the layers are connected in series.

The activation function plays an important role in the approximation of NNs. In this chapter, we consider the rectifier linear unit (ReLU), which is defined as $g_{\mathrm{ReLU}}(x) = \max\{0, x\}$. An important property of the ReLU is that it can produce a series of PWA functions with polyhedral partitions, combined with an affine transformation [152]. Actually, the output of an NN with ReLUs as activation functions is PWA. However, as the optimal value function is PWQ, we are interested in producing a class of PWQ basis functions that can efficiently represent the value function.

Let us focus on the last hidden layer. All activation units in this layer can still be written as a continuous PWA function over the input space of the network [79]. The element-wise product of any two vector-valued continuous PWA functions with the same number of components is a continuous PWQ function. We therefore calculate the element-wise product of all the units κ_L in the last hidden layer

$$\varphi = p(\kappa_L) := \text{diag}(\kappa_L)\kappa_L \in \mathbb{R}^{M_L}$$

to generate a series of PWQ functions $\varphi(x) = [\varphi_1(x) \;\; \varphi_2(x) \cdots \;\; \varphi_{M_L}(x)]^T$. This can be viewed as a layer in the NN, denoted by the product layer $p(\kappa_L)$.

The output layer is a weighted sum of the outputs of the previous layer $f_{L+1}(\varphi) = r^T\varphi = \sum_{i=1}^{M_L} r_i\varphi_i$.

To make the proposed approximator satisfy F3, we develop a "local-global" architecture, which decomposes the outputs of the approximator into two parts

$$\hat{J}(x, W, b, r) = x^T P^* x + [f_2 \circ p \circ g_1 \circ f_1](x) \tag{2.5}$$

with P^* the solution to the algebraic Riccati equation. The term $x^T P^* x$ is included to capture "global" aspects of J_∞^*, while the neural-network-based term is exploited to identify the polyhedral partition in (2.4) and capture the local residuals $J_\infty^*(x) - x^T P^* x$. Since the known term $x^T P^* x$ that dominates the value function is extracted and fixed, using such a "local-global" architecture is possible to enhance the quality of approximation.

We hereafter denote M_1 by M, and $\hat{J}(\cdot, W, b, r)$ by $\hat{J}(\cdot, \theta)$, with all the parameters condensed in θ.

2.3.2. NN TRAINING AND CONVEXITY ANALYSIS

Training data needs to be generated offline by solving (2.2) for N_x different initial states $\{x^{(i)}\}_{i=1}^{N_x}$, $x^{(i)} \in X_0$, and getting the state-value pairs $\{(x^{(i)}, J_\infty^*(x^{(i)}))\}_{i=1}^{N_x}$. Let $\{u_k^{(i)*}\}_{k=0}^{N-1}$ and $\{x_k^{(i)*}\}_{k=0}^{N}$ denote the solution to the MPC problem for the initial state $x^{(i)}$ and the corresponding trajectories of the closed-loop controlled system. We present an efficient sampling strategy by leveraging the equivalence in Theorem 2.2.1. Suppose that we have obtained the optimal control sequence $\{u_k^{(i)*}\}_{k=0}^{N-1}$ and the corresponding value $J_N^*(x^{(i)})$ for different $x^{(i)}$, consider the sub-problems whereby we start at $x_k^{(i)*}$, for $k = 1, \ldots, N-1$ and wish to minimize $J_\infty(x_k^{(i)*}, U)$ in (2.1). According to the principle of optimality [28], the truncated optimal control sequence $\{u_j^{(i)*}\}_{j=k}^{N-1}$ is also optimal for these sub-problems. Hence, the optimal value functions for these subsequent trajectories $x_k^{(i)*}$, $k = 1, \ldots, N-1$ can directly be computed as $J_\infty^*(x_k^{(i)*}) = J_N^*(x_k^{(i)*}) = J_N^*(x^{(i)}) - \sum_{j=0}^{k-1} x_j^{(i)*T} Q x_j^{(i)*} + u_j^{(i)*T} R u_j^{(i)*}$, with no need to solve (2.2) repeatedly.

With this design, we can offline generate $N_x N$ state-value pairs by only solving (2.2) N_x times. With the $N_x N$ state-value pairs available, the NN is trained so that its parameters approximate the solution to

$$\min_{b<0, W, r\geq 0} \frac{1}{NN_x} \sum_{i=1}^{N_x} \sum_{k=0}^{N-1} e(x_k^{(i)*}, \theta), \tag{2.6}$$

where $e(x_k^{(i)*},\theta) = (\hat{J}(x_k^{(i)*},\theta) - J_\infty^*(x_k^{(i)*}))^2$ is the square of the approximation error for each training pair, and $x_0^{(i)*} = x^{(i)}$, $\forall i \in \{1,\dots,N_x\}$. The constraint $b < 0$ is introduced to guarantee that no units in the hidden layer are activated when x is near the origin, i.e., to fulfill F3, while the constraint $r \geq 0$ is responsible for maintaining the convexity of $\hat{J}$.

Problem (2.6) is a nonlinear least-squares problem, which can be successfully solved by the gradient descent method [86]. The constraints on the NN parameters can be handled by constraint elimination, i.e., by letting $r = \left(\bar{r}_1^2, \bar{r}_2^2, \dots, \bar{r}_M^2\right)^T$, which can always guarantee $r \geq 0$, penalizing the constraint violation in the loss function, or reducing the number of hidden units if constraint violation is detected.

After the NN is trained, the system can be run, and the control signals are computed by solving a DP problem. With a specific structure, our proposed NN allows for producing a convex function $\hat{J}$ so that any locally optimal point is also globally optimal. Suppose that the output of the proposed approximator has the following PWQ form:

$$\hat{J}(x,\theta) = \hat{J}^j(x) := x^T \hat{P}_j x + \hat{q}_j^T x + \hat{v}_j, \text{ if } x \in \hat{\mathcal{R}}_j,\ j = 1,\dots,\hat{N}_\mathrm{r}, \tag{2.7}$$

where $\hat{\mathcal{R}}_j$ are polyhedra defined by the hyperplanes $\{W_{l,\cdot}x + b_l = 0\}_{l=1}^M$. Define $\bar{R} = \mathrm{diag}(r)$ and we can rewrite (2.5) as $\hat{J}(x,\theta) = x^T P^* x + \kappa^T \bar{R}\kappa$, where κ, the output of the hidden layer, is PWA w.r.t. x. In the following, we will show that any feasible solutions to (2.6) can ensure the convexity of $\hat{J}(\cdot,\theta)$.

Proposition 2.3.1. Consider the PWQ NN (2.5). For any $\theta = \{W, b, r\}$ such that r is non-negative and that b is negative, the function $\hat{J}(\cdot,\theta)$ is continuously differentiable and convex w.r.t. its input.

Proof. In the interior of any $\hat{\mathcal{R}}_j, j = 1,\dots,\hat{N}_\mathrm{r}$, continuous differentiability of $\hat{J}$ is clear since $\hat{J}$ has a quadratic form, and convexity of $\hat{J}$ follows from the positive semi-definiteness of $\bar{R}$. Then, we have $\hat{P}_j - P^* \geq 0$ for all $j = 1,\dots,\hat{N}_\mathrm{r}$. At the boundary of any neighboring $\hat{\mathcal{R}}_i$ and $\hat{\mathcal{R}}_j$, without loss of generality, suppose that $\hat{\mathcal{R}}_i, \hat{\mathcal{R}}_j$ are partitioned by the hyperplane $W_{1,\cdot}x + b_1 = 0$. It follows from (2.7) that $\hat{J}^j(x) = \hat{J}^i(x) + r_1(W_{1,\cdot}x + b_1)^2$. Differentiating both sides yields $\nabla_x \hat{J}_j(x,\theta) = \nabla_x \hat{J}_i(x,\theta)$, $\forall x \in \{x \in \mathbb{R}^n | W_{1,\cdot}x + b_1 = 0\}$ which proves continuous differentiability of $\hat{J}(x,\theta)$ at the boundary.

Furthermore, since $\hat{J}(x,\theta)$ is differentiable, the convexity of $\hat{J}(x,\theta)$ at the boundary can be checked through the first-order condition [37]. Without loss of generality, let $x_1 \in \hat{\mathcal{R}}_i$ and $x_2 \in \hat{\mathcal{R}}_j$, then we have

$$\begin{aligned}
&\hat{J}(x_2,\theta) - \hat{J}(x_1,\theta) - \nabla_x \hat{J}(x_1,\theta)^T (x_2 - x_1) \\
=& x_2^T P_i x_2 + q_i^T x_2 + r_1 \left(W_{1,\cdot}x_2 + b_1\right)^2 - x_1^T P_i x_1 + q_i^T x_1 - \left(x_1^T P_i + q_i^T\right)(x_2 - x_1) \\
=& (x_2 - x_1)^T P_i (x_2 - x_1) + r_1 \left(W_{1,\cdot}x_2 + b_1\right)^2 \geq 0,
\end{aligned}$$

which demonstrates that $\hat{J}$ satisfies the first-order condition at the boundary. □

2.3.3. SUBOPTIMAL CONTROL LAW BASED ON DP

With a well-fitted $\hat{J}$ available, at each time step $t \in \mathbb{N}$, we can obtain a suboptimal control policy by solving the following one-step DP problem:

$$\begin{aligned} \min_{u_t} \ & \{\hat{Q}(x_t, u_t) := x_t^T Q x_t + u_t^T R u_t + \hat{J}(Ax_t + Bu_t, \theta)\} \\ \text{s.t. } & u_t \in \mathcal{U}, Ax_t + Bu_t \in \mathcal{C}, \end{aligned} \tag{2.8}$$

where $\hat{Q}(x_t, u_t)$ can be viewed as the approximated optimal Q-function [28] for (2.1). Denote the solution to (2.8) by $\hat{u}_t^*$. Here, we use another subscript $(\cdot)_t$ to indicate that (2.8) is solved online at each time step t.

In (2.8), $\mathcal{C}$ is chosen as $\bar{X}$ or $\mathbb{R}^n$, depending on whether $\bar{X}$ is computable. In particular, if the iterative algorithm for computing $\bar{X}$, e.g., Algorithm 10.3 in [35] does not terminate in finite time, we drop the constraint on $Ax_t + Bu_t$. This could happen, e.g., when there is no state constraint in (2.1). In both cases, problem (2.8) is recursively feasible since $\bar{X}$ is a control invariant set (CIS) [35].

The objective function in (2.8) is nonlinear and contains an NN. Standard solvers such as the ellipsoid algorithm or the interior-point algorithm require the computation of the Hessian $\nabla_u^2 \hat{Q}(x, u)$ or the gradient $\nabla_u \hat{Q}(x, u)$ in each iteration. Such computation can only be carried out by visiting all units in the hidden layers and extracting the activated ones. The advantage of low computational complexity brought by DP will then inevitably diminish.

In view of this, we intend to avoid frequently calculating $\nabla_u^2 \hat{Q}(x, u)$ and $\nabla_u \hat{Q}(x, u)$ by decomposing the Q-function $\hat{Q}(x_t, u_t)$ into some quadratic functions. In particular, we develop an optimization algorithm in which problem (2.8) is reduced to a QP problem in each iteration. For a given x_t, $t \in \mathbb{N}$, consider the set of activated ReLU units in $\hat{J}(Ax + Bu, \theta)$, defined by

$$\bar{\mathcal{A}}(u) = \left\{ i \in \{1, \dots, M\} \mid W_{i,\cdot}(Ax_t + Bu) + b_i > 0 \right\}. \tag{2.9}$$

With (2.9), $\hat{Q}(x_t, u)$ can thus be computed as

$$\hat{Q}(x_t, u) = u^T \bar{P}(\bar{\mathcal{A}}(u)) u + \bar{q}^T(\bar{\mathcal{A}}(u)) u + \bar{v}(\bar{\mathcal{A}}(u)), \tag{2.10}$$

where

$$\begin{aligned} \bar{P}(\bar{\mathcal{A}}(u)) &= R + B^T (P^* + \sum_{i \in \bar{\mathcal{A}}(u)} r_i W_{i,\cdot}^T W_{i,\cdot}) B \\ \bar{q}^T(\bar{\mathcal{A}}(u)) &= 2 x_t^T A^T P^* B + 2 (\sum_{i \in \bar{\mathcal{A}}(u)} r_i \left(W_{i,\cdot} A x_t + b_i \right) W_{i,\cdot}) B \\ \bar{v}(\bar{\mathcal{A}}(u)) &= x_t^T (Q + A^T P^* A) x_t + \sum_{i \in \bar{\mathcal{A}}(u)} r_i \left(W_{i,\cdot} A x_t + b_i \right)^2. \end{aligned} \tag{2.11}$$

Since $\bar{P}(\bar{\mathcal{A}}(u)) > 0$, the right-hand side of (2.10) is a convex quadratic function if $\bar{\mathcal{A}}(u)$ is fixed. An algorithm that can effectively solve general piecewise convex programs (PCP) is proposed in [137], called the PCP Algorithm. We adopt it to solve the problem (2.8).

For extra details, see [97]. Nevertheless, the PCP Algorithm is not the ideal choice for solving our DP problem, because it needs to iteratively compute the intersection of some sets, and one of the auxiliary QPs contains numerous constraints if the NN has a large number of hidden units. This motivates us to consider the following design. In each iteration s, $s \in \mathbb{N}^+$, let $u^{(s)}$ be an initial input (for $s = 1$) or the input calculated from the last iteration (for $s > 1$). We compute the set of activated units $\bar{\mathcal{A}}(u^{(s)})$ at $u^{(s)}$, and thereby get $\bar{P}(\bar{\mathcal{A}}(u^{(s)}))$ and $\bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))$ from (2.11). Then, we solve the QP

$$\begin{aligned} u^{(s+1)} = \arg\min_{u} \; & u^T \bar{P}(\bar{\mathcal{A}}(u^{(s)}))u + \bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))u \\ \text{s.t. } & u \in \mathcal{U}, Ax_t + Bu \in \mathcal{C}, \end{aligned} \tag{2.12}$$

which returns $u^{(s+1)}$ for the next iteration. In the next iteration, after computing $\bar{\mathcal{A}}(u^{(s+1)})$, we can terminate and output $u^{(s+1)}$ if $\bar{\mathcal{A}}(u^{(s+1)})$=$\bar{\mathcal{A}}(u^{(s)})$. If a cycle occurs, i.e., $\bar{\mathcal{A}}(u^{(s+1)})$=$\bar{\mathcal{A}}(u^{(k)})$, $\exists k \in \{1,\dots,s-1\}$, we arbitrarily choose another $\bar{\mathcal{A}}(u^{(s+1)})$ that has not been involved in previous iterations. The proposed method is summarized in Algorithm 1.

Algorithm 1 Decomposition algorithm for solving (2.8)

Input: State x_t at time step t, input u_{t-1} at the last time step $t-1$ (if $t > 0$), the function $\hat{J}$, the matrices Q and R, and the sets $\mathcal{C}$ and $\mathcal{U}$
Output: Control input u_t
Initialize a starting point $u^{(1)} \leftarrow u_{t-1}$
for $s = 1, 2, \dots$ **do**
  Update the set of activated units $\bar{\mathcal{A}}(u^{(s)})$ by (2.9).
  if $\bar{\mathcal{A}}(u^{(s)}) = \bar{\mathcal{A}}(u^{(s-1)})$ and $s > 1$ **then**
    Let $s_{\mathrm{m}} \leftarrow s$, **return** $u_t \leftarrow u^{(s_{\mathrm{m}})}$, and **break**.
  else
    if $\bar{\mathcal{A}}(u^{(s)}) = \bar{\mathcal{A}}(u^{(k)}), \exists k \in \{1,\dots,s-2\}$ and $s > 2$ **then**
      Reset $\bar{\mathcal{A}}(u^{(s)})$ to be a new set of activated units that never occurred previously.
    end if
    Compute the coefficients $\bar{P}(\bar{\mathcal{A}}(u^{(s)}))$ and $\bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))$ associated with $\bar{\mathcal{A}}(u^{(s)})$ through (2.11).
    Update the policy through (2.12) and get $u^{(s+1)}$.
  end if
end for

In Algorithm 1, the starting point $u^{(1)} = u_{t-1}$ is initialized with the last control input, which can be viewed as a warm start for the algorithm.

Compared to the PCP Algorithm, Algorithm 1 only needs to solve one QP per iteration, in which the constraints are the same as those in (2.8). Additionally, Algorithm 1 circumvents the calculations of some sets that are necessary in the PCP Algorithm. Meanwhile, Algorithm 1 can achieve finite termination as well as the optimality for problem (2.8).

Theorem 2.3.1. Consider the DP problem (2.8). For any x_t that makes (2.8) feasible, the decomposition algorithm (Algorithm 1) terminates in a finite number of iterations, i.e.,

there exists a finite s_{m} such that $\bar{\mathcal{A}}\left(u^{(s_{\mathrm{m}})}\right)=\bar{\mathcal{A}}\left(u^{(s_{\mathrm{m}}-1)}\right)$. If $\bar{\mathcal{A}}\left(u^{(s_{\mathrm{m}})}\right)=\bar{\mathcal{A}}\left(u^{(s_{\mathrm{m}}-1)}\right)$, then $u^{(s_{\mathrm{m}})}$ is the solution to (2.8).

Proof. We first show that if Algorithm 1 stops, it outputs a solution to (2.8). According to Algorithm 1, $u^{(s_{\mathrm{m}})}$ minimizes $\bar{Q}_{u^{(s_{\mathrm{m}})}}(x_t,u):=u^T\bar{P}(\bar{\mathcal{A}}(u^{(s_{\mathrm{m}})}))u+\bar{q}^T(\bar{\mathcal{A}}(u^{(s_{\mathrm{m}})}))u+\bar{v}(\bar{\mathcal{A}}(u^{(s_{\mathrm{m}})}))$ subject to $u\in\mathcal{U},Ax_t+Bu\in\mathcal{C}_\infty$ if $\bar{\mathcal{A}}(u^{(s_{\mathrm{m}})})=\bar{\mathcal{A}}(u^{(s_{\mathrm{m}}-1)})$. In this case, the inequality

$$\nabla_u^T\bar{Q}_{u^{(s_{\mathrm{m}})}}(x_t,u^{(s_{\mathrm{m}})})(u-u^{(s_{\mathrm{m}})})\geq 0 \tag{2.13}$$

holds for all $u\in\mathcal{U}_1:=\{u|u\in\mathcal{U},Ax_t+Bu\in\mathcal{C}$. Using the fact that $\nabla_u^T\bar{Q}_{u^{(s_{\mathrm{m}})}}\left(x_t,u^{(s_{\mathrm{m}})}\right)=\nabla_u^T\hat{Q}\left(x_t,u^{(s_{\mathrm{m}})}\right)$, the gradient inequality for the convex PWQ function $\hat{Q}(x_t,\cdot)$ can be applied at $u^{(s_{\mathrm{m}})}$: $\hat{Q}(x_t,u)\geq\hat{Q}(x_t,u^{(s_{\mathrm{m}})})+\nabla_u^T\hat{Q}_{u^{(s_{\mathrm{m}})}}(x_t,u^{(s_{\mathrm{m}})})(u-u^{(s_{\mathrm{m}})})$, which, combined with (2.13), shows that $\hat{Q}(x_t,u)\geq\hat{Q}\left(x_t,u^{(s_{\mathrm{m}})}\right)\ \forall u\in\mathcal{U}_1$. Thus, optimality of $u^{(s_{\mathrm{m}})}$ is proven.

If a cycle occurs, i.e., $\exists k\in\{1,2,\ldots,s-2\}$ such that $\bar{\mathcal{A}}(u^{(s)})=\bar{\mathcal{A}}(u^{(k)})$ for some s, according to Algorithm 1, we select another set of activated units that has never been considered in problem (2.12). As the number of combinations of activated units is finite, the algorithm must stop in a finite number of iterations. □

In general, the amount of data needed in our method is in general less than that in policy-based methods [114] because the value function is a scalar, but the policy may be multi-dimensional.

The proposed PWQ approximation is motivated by our prior knowledge of the value function of the considered constrained infinite-horizon LQR problem, which has been proven to be PWQ. The proposed PWQ approximation can be applied to any regression problem where the target function is known to be PWQ. For example, PWQ Lyapunov functions are widely used to analyze the stability of piecewise affine systems [113]. In addition, in bi-level optimization, when a convex quadratic program appears as a lower-level problem, its value function, which is PWQ, is needed to represent the optimal lower-level response compactly. Moreover, the PWQ approximation has been recognized as a tractable approach for approximating globally optimal solutions to nonlinear optimization problems [184]. These examples highlight the broad potential of the proposed neural network-based PWQ approximation method.

2.4. ANALYSIS OF THE PROPOSED METHOD

2.4.1. STABILITY ANALYSIS

In this section, we investigate the stability of the proposed control law. Since both J_∞^* and $\hat{J}$ are PWQ on polyhedra, the intersection of any $\mathcal{R}_i$ and $\hat{\mathcal{R}}_j$, $i=1,\ldots,N_{\mathrm{r}}$, $j=1,\ldots,\hat{N}_{\mathrm{r}}$ is still polyhedral. Let $\mathcal{R}_{i,j}$ denote this intersection if such intersection represents a full-dimensional region, i.e., $\mathcal{R}_{i,j}:=\mathcal{R}_i\cap\hat{\mathcal{R}}_j$, $i\in\{1,...,N_{\mathrm{r}}\}$, $j\in\{1,...,\hat{N}_{\mathrm{r}}\}$ and $\dim(\mathcal{R}_i\cap\hat{\mathcal{R}}_j)=n$. It is clear that all $\mathcal{R}_{i,j}$ constitute a partition of X_0.

To certify the stability, we need to know the upper bound of the approximation error for all x belonging to X_0, i.e., we will compute a positive constant ζ such that

$$|e(x)| := \left| \frac{\hat{J}(x,\theta)}{J^*_\infty(x)} - 1 \right| \leq \zeta, \quad \forall x \in X_0 \backslash \{0\}. \tag{2.14}$$

We propose to leverage the Lipschitz continuity of $\nabla \hat{J}(\cdot,\theta)$ and $\nabla J^*_\infty(\cdot)$. Before doing this, we need the following assumption to ensure the density of samples.

Assumption 2.4.1. For the partition $\mathcal{R}_{i,j}$ of X_0, there exists at least one training point in each $\text{int}(\mathcal{R}_{i,j})$.

Since we can compute the approximation error of the proposed NN at the training points, it is possible to obtain an upper bound of the approximation error for all x in X_0. In the interior of each $\mathcal{R}_{i,j}$, both J^*_∞ and $\hat{J}$ are quadratic and hence twice continuously differentiable. According to [35, Lemma 3.2], $\nabla_x J^*_\infty$ and $\nabla_x \hat{J}$ are locally Lipschitz on $\text{int}(\mathcal{R}_{i,j})$, i.e., there exist non-negative constants L_i and $\hat{L}_j$ such that for all $(x,y) \in \text{int}(\mathcal{R}_{i,j}) \times \text{int}(\mathcal{R}_{i,j})$, we have

$$\begin{aligned} J^*_\infty(y) &\leq J^*_\infty(x) + \nabla^T J^*_\infty(x)(y-x) + \frac{L_i}{2}\|y-x\|_2^2 \\ \hat{J}(y,\theta) &\leq \hat{J}(x,\theta) + \nabla^T \hat{J}(x,\theta)(y-x) + \frac{\hat{L}_j}{2}\|y-x\|_2^2, \end{aligned} \tag{2.15}$$

where L_i and $\hat{L}_j$ can be chosen as the largest eigenvalue of P_i and $\hat{P}_j$ [35, Lemma 3.2], respectively.

We define a measure of the gradient error as $e_{\text{grad}}(x) := \frac{\|\nabla \hat{J}(x,\theta) - \nabla J^*_\infty(x)\|_2}{J^*_\infty(x)}$, and let $\bar{e}$ and $\bar{e}_{\text{grad}}$ stand for the maximum values of $|e|$ and e_{grad} over all samples.

Remark 2.4.1. Assumption 2.4.1 is used to compute a global bound ζ to guarantee (2.14). Assumption 2.4.1 is not required to be satisfied when training the NN. To guarantee stability, if Assumption 2.4.1 is not satisfied after the NN is trained, we can add one testing point in each region $\text{int}(\mathcal{R}_{i,j})$ where there is no training point. Then, we evaluate the approximation errors $|e|$ and e_{grad} at these testing points, and adjust $\bar{e}$ and $\bar{e}_{\text{grad}}$ if necessary so that $|e| \leq \bar{e}, e_{\text{grad}} \leq \bar{e}_{\text{grad}}$ also hold at these testing points. Nevertheless, we should point out that one of the limitations is that we sometimes need carefully selected testing data to satisfy Assumption 2.4.1, and the data should be dense enough if the NN or the optimal value function has many polyhedral regions.

Let $\mathcal{R}_1$ refer to the polyhedron where no constraints in the mpQP derived from (2.2) are active, i.e., $\mathcal{R}_1 = \mathcal{O}^{\text{LQR}}_\infty$, and accordingly let $\hat{\mathcal{R}}_1$ represent the polyhedron where no ReLU units in $\hat{J}$ are activated. In the region $\mathcal{R}_{1,1} = \mathcal{R}_1 \cap \hat{\mathcal{R}}_1$, we have $|e(x)| \equiv 0$ due to (2.5). For every $\mathcal{R}_{i,j}$ except $\mathcal{R}_{1,1}$, the following lemma presents an upper bound of $|e(x)|$.

Lemma 2.4.1. With Assumption 2.4.1 satisfied, for any $x \in \text{int}(\mathcal{R}_{i,j})$, $(i,j) \neq (1,1)$, $|e(x)|$ is upper bounded by

$$\zeta_{i,j} := \frac{1}{1-\beta(y)d_{i,j}} \left(\bar{e} + \bar{e}_{\text{grad}} d_{i,j} + \frac{\max\{\hat{L}_j, L_i\} d_{i,j}^2}{2J^*_\infty(y)} \right), \tag{2.16}$$

where $y \in \text{int}(\mathcal{R}_{i,j})$ is the training or testing point closest to x, $\beta(y) = \left\|\nabla^T J^*_\infty(y)\right\|_2 / J^*_\infty(y)$, and $d_{i,j}$ denotes the maximum Euclidean distance between any nearest training or testing points in $\mathcal{R}_{i,j}$.

Proof. Consider the following three cases:

Case 1: $e(x) \geq 0,\ \forall x \in \text{int}(\mathcal{R}_{i,j})$. In this case, for any $x \in \text{int}(\mathcal{R}_{i,j})$, let $y \in \text{int}(\mathcal{R}_{i,j})$ denote the training or testing point closest to x. Then (2.15) results in

$$\begin{aligned}|e(x)| &\leq \frac{\hat{J}(y,\theta) + \nabla^T \hat{J}(y,\theta)(x-y) + \hat{L}_j \|x-y\|_2^2/2}{J^*_\infty(y) + \nabla^T J^*_\infty(y)(x-y)} - 1 \\ &\leq \frac{\hat{J}(y,\theta) - J^*_\infty(y) + (\nabla \hat{J}(y,\theta) - \nabla J^*_\infty(y))^T (x-y) + \hat{L}_j \|x-y\|_2^2/2}{J^*_\infty(y) - \left\|\nabla^T J^*_\infty(y)\right\|_2 \|x-y\|_2} \\ &\leq \frac{|e(y)| + e_{\text{grad}}(y)\|x-y\|_2 + \hat{L}_j \|x-y\|_2^2 / (2J^*_\infty(y))}{1 - \beta(y)\|x-y\|_2},\end{aligned}$$

where $\beta(y) = \left\|\nabla^T J^*_\infty(y)\right\|_2 / J^*_\infty(y)$. The first inequality is true due to the second inequality of (2.15) and the convexity of $J^*_\infty(\cdot)$. Using the fact that $|e(y)| \leq \bar{e}, e_{\text{grad}}(y) \leq \bar{e}_{\text{grad}}$, we have

$$|e(x)| \leq \frac{1}{1 - \beta(y) d_{i,j}} \left(\bar{e} + \bar{e}_{\text{grad}} d_{i,j} + \frac{\hat{L}_j d_{i,j}^2}{2 J^*_\infty(y)} \right) \tag{2.17}$$

holds for any $x \in \text{int}(\mathcal{R}_{i,j})$. Note that $\beta(y)$ is bounded on all $\mathcal{R}_{i,j}$ except $\mathcal{R}_{1,1}$, since J^*_∞ can only equal 0 at origin, which is in $\mathcal{R}_{1,1}$. Therefore, one can always make $d_{i,j}$ sufficiently small so that $1 - \beta(y) d_{i,j} > 0$.

Case 2: $e(x) \leq 0,\ \forall x \in \text{int}(\mathcal{R}_{i,j})$. Similarly to Case 1, we can readily get almost the same expression of the upper bound as the right-hand side of (2.17), and the only difference is that $\hat{L}_j$ is replaced by L_i.

Case 3: $\exists x_1, x_2 \in \text{int}(\mathcal{R}_{i,j})$ such that $e(x_1) > 0$ and $e(x_2) < 0$. In this case, for all $x \in \text{int}(\mathcal{R}_{i,j})$ subject to $\hat{J}(x,\theta) \geq J^*_\infty(x)$, we can get the same upper bound for $|e(x)|$ as in (2.17), and for all $x \in \text{int}(\mathcal{R}_{i,j})$ subject to $\hat{J}(x,\theta) < J^*_\infty(x)$, (2.17) holds with $\hat{L}_j$ replaced by L_i. □

Finally, since e is continuous on the closure of $\mathcal{R}_{i,j}$, the bound $\zeta_{i,j}$ applies to all x in $\mathcal{R}_{i,j}$. In addition, thanks to Assumption 2.4.1, the value of ζ in (2.14) can be determined by computing the right-hand side of (2.16) at all training/testing points and choosing the largest one.

Computation of $\nabla \hat{J}(y,\theta)$, $\nabla J^*_\infty(y)$, L_i, and $\hat{L}_j$. The computation of $\nabla \hat{J}(y,\theta)$ and $\hat{L}_j$ is straightforward since the analytical form of $\hat{J}$ is known. The value of $\nabla J^*_\infty(y)$ can be obtained by the sensitivity analysis of problem (2.2) [35, Theorem 6.9]. The Lipschitz constant L_i can be selected as the largest eigenvalue of P_i.

With the property of boundedness for e established, we can assess the stability for the closed-loop system with the approximate controller $\hat{u}^*$. Corresponding to the selection of $\mathcal{C}$ in (2.8), we consider two cases: (1) $\mathcal{C} = \bar{X}$ and (2) $\mathcal{C} = \mathbb{R}^n$.

Theorem 2.4.1. Let $\hat{u}_t^*$ be the solution to the DP problem (2.8) with $\mathcal{C} = \bar{X}$. Then, problem (2.8) is recursively feasible for the initial condition $x_0 \in \bar{X}$. Furthermore, suppose that Assumptions 2.2.1 and 2.4.1 hold with $X_0 = \bar{X}$. If ζ in (2.14) satisfies

$$\frac{1-\zeta^2}{\zeta} > 2 \sup_{x \in X_0 \setminus \{0\}} \frac{\hat{J}(x,\theta)}{x^T Q x}, \tag{2.18}$$

then the origin of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \ldots$ is asymptotically stable with the domain of attraction $\bar{X}$.

Proof. The recursive feasibility of (2.8) follows from the definition of $\bar{X}$. In the rest, the stability will be proved.

First, consider the solution to (2.2). For every x_t, $t = 0, 1, \ldots$, the MPC control law u_t^*, which contains the first m elements of $U^*(x_t)$, will be applied to the system. From the equivalence in Theorem 2.2.1, $\{u_t^*\}_{t=0}^{\infty}$ is also a solution to the (2.1). As a result, $Ax_t + Bu_t^* \in \bar{X}$ for any $x_t \in \bar{X}$, (else, $J_\infty^*(Ax_t + Bu_t^*) = \infty$ and $J_\infty^*(x_t) = \infty$, which contradicts the claim that $x_t \in \bar{X}$).

Suppose that $x_t \in \bar{X} \setminus \{0\}$. Applying the Bellman Optimality Equation [28] for (2.1) leads to

$$J_\infty^*\left(Ax_t + Bu_t^*\right) = J_\infty^*(x_t) - x_t^T Q x_t - u_t^{*T} R u_t^* < J_\infty^*(x_t) - x_t^T Q x_t. \tag{2.19}$$

Combining (2.19) with (2.14), we have

$$\begin{aligned}
&x_t^T Q x_t + \hat{u}_t^{*T} R \hat{u}_t^* + \hat{J}\left(Ax_t + B\hat{u}_t^*, \theta\right) \\
\leq &x_t^T Q x_t + u_t^{*T} R u_t^* + \hat{J}\left(Ax_t + Bu_t^*, \theta\right) \\
\leq &x_t^T Q x_t + u_t^{*T} R u_t^* + J_\infty^*\left(Ax_t + Bu_t^*\right) + \zeta J_\infty^*\left(Ax_t + Bu_t^*\right) \\
< &J_\infty^*(x_t) + \zeta J_\infty^*(x_t) - \zeta x_t^T Q x_t \\
< &\hat{J}(x_t, \theta) + 2\zeta J_\infty^*(x_t) - \zeta x_t^T Q x_t.
\end{aligned}$$

The first inequality is true since $\hat{u}_t^*$ is a minimizer of (2.8). The second and last inequalities hold due to (2.14), and the third line is true thanks to the optimal Bellman equation in (2.19). Then, combining (2.14) and (2.18) yields

$$\frac{1+\zeta}{\zeta} > 2 \sup_{x \in \bar{X} \setminus \{0\}} \frac{\hat{J}(x,\theta)}{(1-\zeta) x^T Q x} > 2 \sup_{x \in \bar{X} \setminus \{0\}} \frac{J_\infty^*(x)}{x^T Q x},$$

which implies

$$\hat{J}\left(Ax_t + B\hat{u}_t^*, \theta\right) - \hat{J}(x_t, \theta) < -(1+\zeta) x_t^T Q x_t - \hat{u}_t^{*T} R \hat{u}_t^* + 2\zeta J_\infty^*(x_t) < 0, \ \forall x_t \in \bar{X} \setminus \{0\}.$$

Let $x_0 \in \bar{X} \setminus \{0\}$ and $x_1, x_2, \ldots$ be the trajectory of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \ldots$ The sequence $\hat{J}(x_0, \theta), \hat{J}(x_1, \theta), \ldots$ is strictly decreasing. Besides, it is easy to check that $\hat{J}(0, \theta) = 0$, $\hat{J}(x, \theta) > 0$, $\forall x \in \bar{X} \setminus \{0\}$, and $\hat{J}(\cdot, \theta)$ is continuous at the origin, finite in $\bar{X}$. Then, $\hat{J}(\cdot, \theta)$ is therefore a Lyapunov Function according to [35, Theorem 7.2]. So, the asymptotic stability of the origin for any $x \in \bar{X}$ follows. □

In the case of $X_0 \subset \bar{X}$, where the set X_0 may not be invariant for the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \ldots$, the corresponding stability conditions are described in the following corollary:

Corollary 2.4.1. Let $\hat{u}_t^*$ be the solution to the DP problem (2.8) with $\mathcal{C} = \mathbb{R}^n$. For a given $X_0 \subset \bar{X}$, suppose that Assumptions 2.2.1 and 2.4.1 hold. Define a compact set as $\Omega := \{x \in \mathbb{R}^n | \hat{J}(x,\theta) \leq \chi\}$ where $\chi := \inf_{x \in \partial X_0} \hat{J}(x,\theta)$. If ζ in (2.14) satisfies (2.18), then problem (2.8) is recursively feasible for the initial condition $x_0 \in \Omega$, and the origin of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \ldots$ is asymptotically stable with the domain of attraction Ω.

Proof. From the definition of Ω and the continuity of $\hat{J}(\cdot,\theta)$, we know that $\Omega \subseteq X_0$. Following the proof of Theorem 2.4.1, we can show that for all $x_t \in \Omega$, $\hat{J}\left(Ax_t + B\hat{u}_t^*, \theta\right) - \hat{J}(x_t,\theta) < 0$ holds. As a result, the set Ω is positively invariant w.r.t. the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$. The recursive feasibility thus follows from $\Omega \subseteq X_0 \subseteq \mathcal{X}$. Similar to the proof of Theorem 2.4.1, it can be shown that $\hat{J}$ is a Lyapunov function and the stability of the origin follows. □

According to Theorem 2.4.1 and Corollary 2.4.1, asymptotic stability is achieved if the condition in (2.18) holds. (2.18) can be satisfied by making ζ small enough since the right-hand side of (2.18) is upper bounded. Besides, in view of (2.16), ζ is determined mainly by $\bar{e}$, $\bar{e}_{\text{grad}}$, and $d_{i,j}$. The condition (2.18) can thereby be guaranteed in two ways. One is to add more hidden units into the NN so that $\bar{e}$ and $\bar{e}_{\text{grad}}$ could be smaller according to the universal approximation theorem [109]. Another possibility is to involve more state-value/gradient data to test the upper bounds $\bar{e}$ and $\bar{e}_{\text{grad}}$ so that $d_{i,j}$ is reduced.

Remark 2.4.2. The stability analysis is related to the sum-of-squares (SOS) framework for stability verification. In particular, we establish stability by verifying that the PWQ function $\hat{J}(x,\theta) = x^T P^* x + \kappa(x)^T \bar{R}\kappa(x)$ is a Lyapunov function. Since κ is a PWA function, $\hat{J}$ can be equivalently expressed as a piecewise SOS polynomial $Z_j^T(x)\bar{P}Z_j(x)$, for $x \in \hat{\mathcal{R}}_j$, where each $Z_j(x)$ is a vector of monomials and $\bar{P}$ is positive semi-definite. However, it is generally difficult to apply SOS programming to verify the Lyapunov condition for $\hat{J}(\cdot,\theta)$, as done in [12], because the vector $Z_j(x)$ changes when x and $Ax + Bu^*$ belong to different regions. Instead, we propose a sampling-based method and leverage the Lipschitz continuity of $\hat{J}$ to derive sufficient conditions for $\hat{J}$ to be a valid Lyapunov function.

2.4.2. COMPLEXITY ANALYSIS

We analyze the offline storage requirement as well as the online computational complexity of the proposed control scheme, and compare them with other methods, such as implicit MPC, explicit MPC, and the policy approximation methods of MPC [51], [114]. Storage space is dominated by the number of regions and control laws (for explicit MPC), or the structure of the NN (for approximate MPC). The storage of some system's parameters, such as A, B, $\mathcal{X}$, $\mathcal{U}$, Q, and R, are neglected for consistency.

Explicit MPC requires the storage of N_{r} regions and affine feedback laws. Suppose that each region $\mathcal{R}_j$ is defined by $n_{\text{c}}^{(j)}$ constraints. Then, explicit MPC needs to store $(n+1)\sum_{j=1}^{N_{\text{r}}} n_{\text{c}}^{(j)} + N_{\text{r}}(mn+m)$ real numbers. As for the proposed method, it needs to

construct a PWQ NN before running the system. The NN contains 3 parameters: W, b, and r, so the storage of the proposed NN requires $nM+2M$ real numbers in total. In addition, as for the policy approximation methods reported in [51], [114], the total storage demand of the NNs is $(n+n_0+1)M+(L-1)(M+1)M$, with $n_0=m$ for [114] and $n_0=N(m+2n+n_c+m_c)$ for [51], respectively. Here, L denotes the number of hidden layers, and n_c and m_c denote the number of constraints specified by $\mathcal{X}$ and $\mathcal{U}$. In general, using an NN that requires much smaller storage space than explicit MPC can get an acceptable performance.

Online computation time will be evaluated in terms of floating-point operations (flops) for the computations that should be performed online. Implicit MPC needs to solve the QP (2.2) at each time step. Solving (2.2) for a given x requires $f_{\mathrm{QP}}(Nm,\ N(m_c+n_c))$ flops in the worst case ((2.2) has no redundant constraints). Here, $f_{\mathrm{QP}}(n_{\mathrm{D}},\ n_{\mathrm{I}})$ represents the number of flops needed to solve a QP with n_{D} decision variables and n_{I} linear inequalities. So in an interior-point method, solving (2.2) requires $O\left(N^3m^3\right)$ flops per iteration. In comparison, the number of flops for explicit MPC is $2n\sum_{j=1}^{N_r}n_c^{(j)}$ [35].

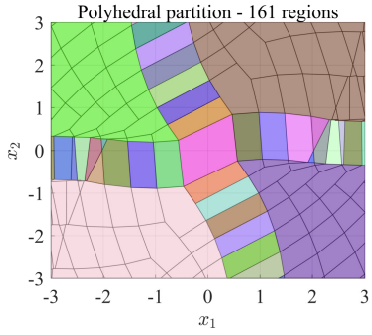
In our proposed control scheme, solving the DP problem (2.8) causes computational complexity. In the PCP Algorithm and Algorithm 1, the number of flops to determine $\bar{A}\left(u^{(s)}\right)$ and to compute the coefficients $\bar{P}\left(\bar{A}\left(u^{(s)}\right)\right), \bar{q}^T\left(\bar{A}\left(u^{(s)}\right)\right)$ is bounded by $f_{\mathrm{act}}=M(2n^2+n+m^2+5m+2mn+2)+2mn+m(m+3)/2$ in total. Then, in each iteration the PCP Algorithm has to solve 2 different QPs, which need $f_{\mathrm{QP}}(m,\ n_s)$ and $f_{\mathrm{QP}}(m,\ M+n_c+l_c)$ flops. Here, n_s stands for the number of inequalities that define $\mathcal{U}_s$. Algorithm 1 needs only $f_{\mathrm{QP}}(m,\ n_c+l_c)$ flops to find $u^{(s)}$ in each iteration. Besides, some other calculation includes the update of $\mathcal{U}_s$ ($2m^2+2m-1$ flops per iteration) and the comparison of $\bar{A}(u^{(s)})$ and $\bar{A}(u^{(s-1)})$ (M flops per iteration). Therefore, the total number of flops to implement the PCP Algorithm is $f_{\mathrm{pcp}}=s_{\mathrm{m}}(f_{\mathrm{act}}+f_{\mathrm{QP}}(m,\ M+n_c+l_c)+2m^2+2m-1)+\sum_{s=1}^{s_{\mathrm{m}}}f_{\mathrm{QP}}(m,\ n_s)$ and the number of flops to calculate the control input by using Algorithm 1 is $f_{\mathrm{algo1}}=s_{\mathrm{m}}(f_{\mathrm{act}}+f_{\mathrm{QP}}(m,\ n_c+l_c)+M)$, which can be much less than those in the PCP Algorithm. Theoretically, the maximum value of s_{m} can be the number of polyhedral regions in the partition for the NN, and thus can grow exponentially with the system's dimension. However, the above analysis largely overestimates the complexity, as it does not take into account that most regions are never visited, and that we use a warm start for the proposed algorithm.

2.5. CASE STUDY

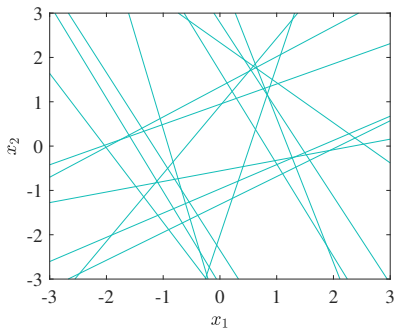
Consider a 2-dimensional linear system with $A=\begin{bmatrix}1 & 0.1\\ -0.1 & 1\end{bmatrix}$, $B=\begin{bmatrix}1 & 0.05\\ 0.5 & 1\end{bmatrix}$ and input constraints $\mathcal{U}=\{u\in\mathbb{R}^2\mid ||u||_\infty\le 0.5\}$. We are interested in stabilizing the system at the origin and meanwhile minimizing the cost $\sum_{k=0}^{\infty}x_k^TQx_k+u_k^TRu_k$ with $Q=I_2$ and $R=0.1I_2$. We choose the region of interest $\mathcal{X}_0=\left\{x\in\mathbb{R}^2\mid \|x\|_\infty\le 3\right\}$. By applying the algorithm in [54], it can be verified that the vertices of $\mathcal{X}_0$ can be steered to an ellipsoidal subset of $\mathcal{O}_\infty^{\mathrm{LQR}}$ with the horizon $N=10$. By solving the explicit MPC problem with $N=10$, the partition of the state space for the optimal control law on the region $\mathcal{X}_0$ can be obtained, and is depicted in Fig. 2.1(a).

To illustrate that the proposed PWQ NN has a good approximation performance, different types of NNs are compared in Fig. 2.1(c). Besides, a strictly global architecture without the quadratic term $x^T P^* x$ is also compared. 4410 state samples are selected to train the NNs. We use multi-start local optimization [171] with 20 starting points to reduce sub-optimality. Choosing the learning rate $\alpha = 0.1$, we compare the absolute mean square errors in Fig. 2.1(c). From Fig. 2.1(c), it is observed that the training objectives for the NNs decrease consistently over epochs, while the proposed approach has the smallest mean square error during the training process. Additionally, increasing the width does not necessarily improve the approximation ability of the proposed NN.

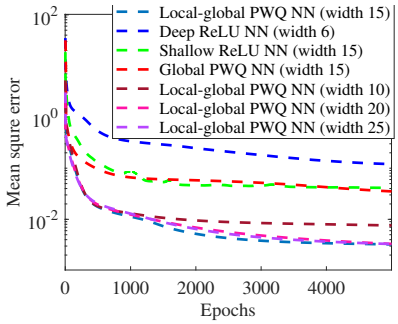
Fig. 2.1(b) depicts the polyhedral partition for the proposed NN with width 15. It is noticed that the partition of the NN concentrates in the areas where the explicit MPC law changes rapidly. We also compare the output of the PWQ NN with the real optimal value function in Fig. 2.1(d), from which one can observe that the proposed method is able to closely approximate the optimal value function with a very simple network architecture.



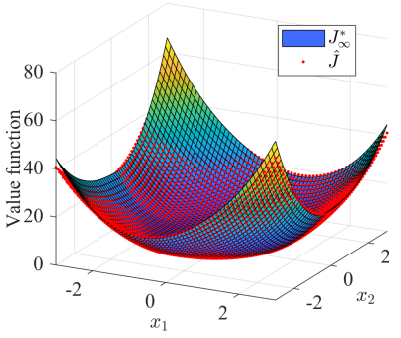
(a) Regions of optimal explicit MPC.



(b) Regions of the PWQ NN.



(c) Comparison of the different types of NNs.



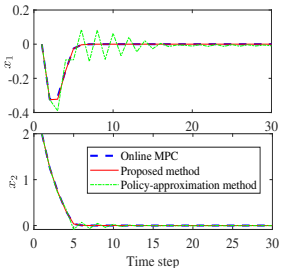
(d) The comparison of the approximated and real value functions.

Figure 2.1: Performance of the proposed PWQ NN.

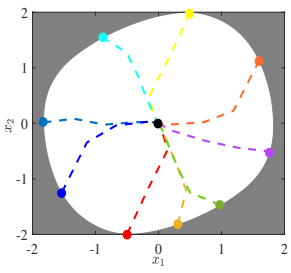
To verify the stability of the closed-loop system, we note that the maximal stabilizable

set $\bar{X}$ is open and thereby not computable. So, we concentrate on checking (2.18) with X_0 replaced by Ω. The approximation error bounds at the samples are $\bar{e} = 0.122$ and $\bar{e}_{\text{grad}} = 0.395$. Based on (2.16), $\zeta = 0.188$. Next, the value of the right-hand side of (2.18) is 4.904. As a result, it can be readily verified that (2.18) holds.

In the closed-loop simulation, the proposed method, implicit MPC, and the policy-approximation method of [114] are compared. The implementation details are given in [97]. From Fig. 2.2(a), it can be seen that the proposed method can properly approximate the MPC controller, and that the corresponding trajectory is stabilized at the origin. In comparison, the policy-approximation method experiences some fluctuations when the state is close to the origin. This is probably because the value of the optimal control input is small if x is around the origin, so a slight approximation error of the policy may lead to drastic changes in the dynamic response. Besides, we can verify the stability by illustrating the invariance of the sub-level set $\Omega = \{x \in \mathbb{R}^n | \hat{J}(x,\theta) \leq \chi\}$. We select some initial states that are at the boundary of Ω, and plot the behavior of the closed-loop system with the proposed controller in Fig. 2.2(b). It is seen that the trajectories starting from the boundary of Ω will always stay in Ω, which is computed by Corollary 2.4.1.



(a) Comparison of the different methods.



(b) Verification of the stability.

Figure 2.2: Closed-loop simulation. In (b), the white region represents Ω, the colored lines refer to the trajectories, and the colored points are the starting state points.

Moreover, in [97], the stability conditions in Theorem 2.4.1 and the comparison between the proposed method and another ADP method in [46] are illustrated through another problem with both state and input constraints. The results in [97] can clearly show that the proposed controller is stabilizing and feasible even for those initial states on the boundary of $\bar{X}$. Combined with Algorithm 1, the proposed method takes less total computation time (0.142 s) than the ADP method (0.270 s) in [46] and implicit MPC (0.161 s).

2.6. CONCLUSIONS

We have developed an ADP control framework for infinite-horizon optimal control of linear systems subject to state and input constraints. Compared to some common NNs, such as ReLU NNs, the proposed NN maintains the PWQ property and convexity of the

real value function and has a much better approximation performance. These properties and superiority contribute to the reduction of the online computation as well as the construction of explicit stability criteria. Therefore, the advantages of our method include low computational requirements, stability assurance, and excellent approximation of the optimal control law.

3

APPROXIMATE DYNAMIC PROGRAMMING FOR CONSTRAINED PIECEWISE AFFINE SYSTEMS

Infinite-horizon optimal control of constrained piecewise affine (PWA) systems has been approximately addressed by hybrid model predictive control (MPC), which, however, has computational limitations, both in offline design and online implementation. In this chapter, we consider an alternative approach based on approximate dynamic programming (ADP), an important class of methods in reinforcement learning. We accommodate non-convex union-of-polyhedra state constraints and linear input constraints into ADP by designing PWA penalty functions. PWA function approximation is used, which allows for a mixed-integer encoding to implement ADP. The main advantage of the proposed ADP method is its online computational efficiency. Particularly, we propose two control policies, which lead to solving a smaller-scale mixed-integer linear program than conventional hybrid MPC, or a single convex quadratic program, depending on whether the policy is implicitly determined online or explicitly computed offline. We characterize the stability and safety properties of the closed-loop systems, as well as the sub-optimality of the proposed policies, by quantifying the approximation errors of value functions and policies. We also develop an offline mixed-integer linear programming-based method to certify the reliability of the proposed method. Simulation results on an inverted pendulum with elastic walls and on an adaptive cruise control problem validate the control performance in terms of constraint satisfaction and CPU time.

This chapter is based on the paper [102] and its extended version [99].

3.1. INTRODUCTION

3.1.1. BACKGROUND

There has been an increasing interest in control of piecewise affine (PWA) systems due to their capability of representing hybrid models and approximating nonlinear dynamics [134]. Many practical control problems can be modelled as PWA systems with constraints, including emergency evasive maneuvers [83], robotic manipulation that has multi-contact behaviors [175], and traffic control [89]. PWA systems are a special class of switched systems where the subsystem in each mode is affine and the transitions are based on the state belonging to different regions. Tractable controller design methods for PWA systems include synthesizing piecewise linear control laws via Linear Matrix Inequalities [124], adaptive control [134], and model predictive control (MPC) [124]. The challenges of controlling a PWA system include ensuring stability and achieving optimality guarantees [124], as well as addressing both offline and online computational complexity [34], [175]. These difficulties primarily arise from the system's hybrid structure and inherent nonlinearity. For suboptimal control of PWA systems with constraints, MPC is widely applied. However, MPC for PWA systems still faces challenges in computational complexity [34], because it involves solving a mixed-integer linear programming (MILP) problem. The complexity of solving MILP MPC problems is, in general, dominated by the number of integers, which is proportional to the prediction horizon. Explicit MPC [34], an offline version of MPC, requires solving a parametric MILP problem, which also suffers from computational complexity issues. These issues make MPC only suitable for slow PWA processes or for small-scale problems [175].

In contrast to MPC, reinforcement learning (RL) can learn a policy that minimizes a finite-/infinite-horizon cost and could have a much lower online computational burden than MPC. In RL, two different methodologies can be distinguished: policy search [208] and dynamic programming [40]. Dynamic programming has the advantage over policy search that it reduces the policy optimization problem to a one-step look-ahead problem. When applied to systems with continuous state and input spaces, approximate dynamic programming (ADP) has been developed [111]. In this chapter, we consider approximate value iteration (VI), the most basic and direct way to solve the Bellman equation. Moreover, we provide comprehensive performance guarantees for stability, safety, and sub-optimality of the developed ADP approach. In the context of RL, safety can have various definitions. In this chapter, we specifically address safety as ensuring that the state and input of the system satisfy predefined constraints throughout the entire system's evolution after the learning process is finished.

3.1.2. RELATED WORK

To reduce the computational cost of MPC, two different types of approaches have been extensively studied: approximate MPC and reinforcement learning under constraints.

Approximate MPC: Approximate MPC parameterizes a policy and then uses supervised learning or gradient-based methods to mimic a predictive control policy. The online computational cost is thus significantly reduced because approximate MPC directly computes control actions based on the learned parameters, rather than online solving

an optimization problem. Some approximate MPC work considers linear systems, with different focuses on, e.g., stability [177] and constraint satisfaction [50]. Some approximate MPC approaches can handle nonlinear control problems with constraints, e.g., by using constraint tightening [106].

RL for constrained control: RL for constrained control can be roughly categorized into two groups: policy-projection RL and policy-optimization RL. For the first group, a predictive safety filter [130], [198] can be adopted to modify the learned policy, which can be derived from any RL algorithm. For PWA systems with linear constraints, this group needs to solve mixed-integer convex problems online [130], and as a result, it is not suitable for large-scale systems or for systems requiring fast computation. For the second group, constrained policy optimization [19], [78], [189] is often used. Most methods [189] consider constrained Markov Decision Process (MDP) problems, in which constraints are on expected cumulative costs. Recent developments have been made to transform instantaneous constraints into constraints on expected cumulative costs. The stability property of RL controllers for nonlinear systems has been investigated recently [107], [154], [162]. Nevertheless, these references consider unconstrained problems and do not address the sub-optimality of the RL policy.

Based on these observations, for the policy optimization methods, one could know that no work has been done for PWA systems, and comprehensive performance of RL-based controllers regarding online computing convenience, stability, and constraint fulfillment cannot be achieved simultaneously.

Performance verification of learning-based controllers: In addition to controller design, there is some related work on performance analysis and verification of RL or any learning-based controllers for PWA systems, by using a learner/verifier framework or explicitly computing the range of trajectories in a finite horizon. However, for PWA systems with learning-based controllers, there is currently no systematic way to verify different properties, including practical and asymptotic stability as well as state constraint satisfaction.

3.1.3. METHODS AND CONTRIBUTIONS OF THIS CHAPTER

In conclusion, using RL to produce a reliable learning-based controller for constrained PWA systems with performance guarantees and low online computational requirements is still an open problem. The main challenge is to concurrently ensure the stability, safety, and efficiency of the online computations. Existing work can either provide stability/safety guarantees [19], [90], [97], [130], [154], [198] or achieve low computational cost [128], [211]. In this chapter, we develop ADP algorithms under linear and UoP constraints. We propose two formulations for the inclusion of PWA penalties in dynamic programming, i.e., adding penalties to the stage cost and integrating penalties into the cost-to-go. We then present two different controllers: an implicit controller that is obtained online by solving an MILP problem that is much simpler than the one of implicit hybrid MPC, and an explicit controller that is learned offline by policy gradient. We provide rigorous analysis on the closed-loop stability, safety, as well as sub-optimality of the controllers. We establish a systematic, MILP-based procedure that allows us to certify the reliability of the closed-loop system. The chapter contributes to the state of the art

as follows.

1. This work is the first research on designing policy optimization RL methods for constrained PWA systems. Systematic performance analysis on the feasibility, stability, and sub-optimality of the RL-based controllers is provided. The analysis suggests several ways to employ the proposed algorithms in practice.
2. Compared to MPC, our method exhibits superiority in terms of online computational simplicity. In particular, the resulting online policy optimization problem is either an MILP problem with significantly fewer integer variables than the hybrid MPC MILP problem, or a single convex quadratic programming (QP) problem.
3. We develop a mixed-integer optimization-based framework to exactly verify the stability and safety of the closed-loop system. The framework extends the verification techniques of [67], [115], [177] with a comprehensive scheme that addresses both practical and asymptotic stability properties and the enlargement of stable and safe regions.

3.2. PRELIMINARIES

Notations: Let $\mathbb{R} = (-\infty, +\infty)$, $\mathbb{R}_{\geq 0} = [0, +\infty)$, and $\mathbb{R}_{>0} = (0, +\infty)$. The boundary of the set $\mathcal{S}$ is $\partial\mathcal{S}$, and $\text{int}(\mathcal{S})$ stands for the interior of $\mathcal{S}$. We utilize $A_{i\cdot}$ to represent the ith row of the matrix A. We define the sub-level set $\mathcal{B}(J, S)$ for a continuous function $J : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and a compact set $S \subseteq \mathbb{R}^n$ as $\mathcal{B}(J, S) := \{x \in \mathbb{R}^n | J(x) \leq \rho\}$, where $\rho = \min_{x \in \partial S} J(x)$. For $a \in \mathbb{R}$, denote by $\lceil a \rceil$ the smallest integer larger than or equal to a.

3.2.1. OPTIMAL CONTROL OF PWA SYSTEMS

We consider discrete-time PWA systems of the form

$$x_{t+1} = f_{\text{PWA}}(x_t, u_t) = A_i x_t + B_i u_t + f_i \quad \text{if} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \in \mathcal{C}_i, \tag{3.1}$$

with input and state constraints $x \in X$, $u \in U$[1]. In (3.1), $\{\mathcal{C}_i\}_{i=1}^s$ is a polyhedral partition of the state-input space $\mathcal{X} \times \mathcal{U}$. The matrices A_i, B_i and the vectors f_i define the affine dynamics in the regions $\mathcal{C}_i$. Including the offset vector f_i allows for the representation of an affine transformation instead of just a linear one in each region of the state space. We want to design a control policy $u_t = \pi(x_t)$, $t = 0, 1, \ldots$ with $\pi : \mathcal{X} \to \mathcal{U}$ to simultaneously satisfy the constraints and minimize the infinite-horizon cost

$$J(x_0, \mathbf{u}) = J_\pi(x_0) := \sum_{t=0}^{\infty} l(x_t, \pi(x_t)), \tag{3.2}$$

where $\mathbf{u} = \{u_t\}_{t=0}^{\infty} = \{\pi(x_t)\}_{t=0}^{\infty}$ stands for the infinite-horizon input sequence and the function $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$ is the stage cost. Throughout the chapter, we assume that the dynamics, constraints, and stage costs satisfy the following assumption.

[1] All the results of this chapter also apply to the case when there is a coupled constraint: $[x^T \ u^T]^T \in D$ with D a polyhedron in $\mathcal{X} \times \mathcal{U}$, by letting X be the projection of D in $\mathcal{X}$ and by letting U be a time-varying set depending on x.

Assumption 3.2.1.
- *Dynamics:* The function $f_{\text{PWA}}(\cdot,\cdot): \mathcal{X}\times\mathcal{U}\to\mathcal{X}$ is a continuous PWA function, with $\mathcal{X}\subseteq\mathbb{R}^{n_x}$ and $\mathcal{U}\subseteq\mathbb{R}^{n_u}$. Besides, $f_{\text{PWA}}(0,0)=0$.
- *Constraints:* The state constraint set $X=\bigcup_{i=1}^{r_0} X^{(i)}$ is a UoP, where $X^{(i)}$ is a polyhedron for each $i=1,\dots,r_0$ and r_0 is the number of polyhedra. The input constraint set U is a polytope. Moreover, $X\times U\subset\bigcup_{i=1}^{s}\mathcal{C}_i=\mathcal{X}\times\mathcal{U}$.
- *Stage cost:* The stage cost is based on the 1-/∞-norm: $l(x_t,u_t)=\|Qx_t\|_{q_1}+\|Ru_t\|_{q_2}$, where $q_1,q_2\in\{1,\infty\}$ and Q,R have full column rank.

Similar assumptions can be found in other papers on control of PWA systems, e.g., [16], [209]. In most literature [16], [124], [209], it is usually assumed that X is a polyhedron, while we generalize it to a UoP. It should be mentioned that many practical nonlinear and non-convex state constraints, such as collision avoidance encountered in robotics, can be modeled or outer-approximated by constraint sets that are UoPs [190].

By combining (3.1) and (3.2), our control objective is to solve the constrained infinite-horizon optimal control problem

$$\begin{aligned} J^*(x_0) = \min_{\pi,\mathbf{u},\mathbf{x}}\ & J_\pi(x_0) \\ \text{s.t. } & x_{t+1}=f_{\text{PWA}}(x_t,u_t),\ u_t=\pi(x_t) \\ & x_t\in X,\ u_t\in U,\ t=0,1,\dots, \end{aligned} \tag{3.3}$$

where $\mathbf{x}=\{x_t\}_{t=0}^{\infty}$, and $J^*(\cdot):\mathcal{X}\to[0,\infty]$ is the optimal value function. An optimal policy, denoted by $\pi^*(\cdot):\mathcal{X}\to\mathcal{U}$, minimizes $J_\pi(x_0)$ subject to the constraints in (3.3) for any initial state x_0 that makes (3.3) feasible. In this chapter, we denote by $\bar{X}$ the set of feasible initial states x_0 that make $J^*(x_0)$ finite. In [35], $\bar{X}$ is called the maximal stabilizable set for (3.1).

Assumption 3.2.2. The set $\bar{X}$ is non-empty. Furthermore, for any $x_0\in\bar{X}$, there exists a policy $\pi(\cdot):\mathcal{X}\to\mathcal{U}$ such that the system (3.1) with $u_t=\pi(x_t)$, starting from x_0, will reach the origin in a finite number of time steps.

Assumption 3.2.2 is a standard stabilizability assumption for discrete-time systems. Similar assumptions for PWA systems can be found, e.g., in [16], [124].

For any $x\in\mathcal{X}$, according to Bellman's principle of optimality [28], the value function J^* and the optimal policy π^* satisfy the following equations:

$$\begin{aligned} J^*(x) &= \Gamma J^*(x) := \min_{u\in U} l(x,u)+J^*(f_{\text{PWA}}(x,u)) \\ \pi^*(x) &\in \arg\min_{u\in U} l(x,u)+J^*(f_{\text{PWA}}(x,u)), \end{aligned} \tag{3.4}$$

where Γ is called the Bellman operator [28]. In (3.4), the domain of J^* is the whole state space $\mathcal{X}$, which means that the value of J^* goes to infinity outside $\bar{X}$. In general, the equation for J^* in (3.4) may have multiple solutions. Nevertheless, it follows from [27, Proposition 1] that J^* can be the unique solution that satisfies $J^*(0)=0$ under Assumption 3.2.2.

3.2.2. EXACT VALUE ITERATION

Solving the Bellman equations is, in general, computationally prohibitive for nonlinear systems. Usually, an MPC problem with a finite horizon is solved online to approximate the infinite-horizon optimal policy. However, computational complexity also remains a hurdle in the application of MPC to PWA systems. For the PWA system, the equations in (3.4), on the other hand, can be solved by using an exact value iteration (VI) method [16], which solves multiple multi-parametric linear programs (mp-LPs). To motivate our ADP methods, we summarize the exact VI method and discuss its limitations in this section. The exact VI algorithm starts from an initial value function $J_0(\cdot)$ that is either zero in $\mathcal{X}$ (case 1) or a control Lyapunov function defined on a subset of $\mathcal{X}$ (case 2). In case 2, the following assumption should be satisfied.

Assumption 3.2.3. A continuous and PWA control Lyapunov function $J_{\mathrm{CL}}(\cdot) : X_{\mathrm{CI}} \to \mathbb{R}_{\geq 0}$ in a polyhedral control-invariant set X_{CI} is available. In other words, $\min_{u\in U, f_{\mathrm{PWA}}(x,u)\in X_{\mathrm{CI}}} l(x,u) + J_{\mathrm{CL}}(f_{\mathrm{PWA}}(x,u)) - J_{\mathrm{CL}}(x) \leq 0,\ \forall x \in X_{\mathrm{CI}}$.

Assumption 3.2.3 frequently appears in the stability analysis of MPC [35], where J_{CL} is chosen as the terminal cost and X_{CI} is specified as the terminal constraint. To satisfy Assumption 3.2.3, it is sufficient to compute a stabilizing piecewise linear feedback law on X_{CI}, and then J_{CL} can be computed by solving some nonlinear inequalities that contain 1-/∞-norm of some linear functions [124].

With the initialization $X_0 = X$ and $J_0(x) = 0,\ \forall x \in X_0$ (case 1), or $X_0 = X_{\mathrm{CI}}$ and $J_0(x) = J_{\mathrm{CL}}(x),\ \forall x \in X_0$ (case 2), the exact VI method iterates as follows:

$$J_k(x) = \min_{u\in U, f_{\mathrm{PWA}}(x,u)\in X_{k-1}} l(x,u) + J_{k-1}(f_{\mathrm{PWA}}(x,u)) \tag{3.5}$$

for $k = 1, 2, \ldots$ Here, $X_k = \mathrm{Pre}(X_{k-1}) \cap X_0$, where $\mathrm{Pre}(\mathcal{S}) = \{x \in \mathcal{X} | \exists u \in U \text{ s.t. } f_{\mathrm{PWA}}(x,u) \in \mathcal{S}\}$ is the backward-reachable set to a set S. Even for PWA systems with polytopic state constraints, the backward-reachable set to a polyhedral set can be a non-convex UoP because of the nonlinear dynamics (3.1), which means that $X_k, k = 1, 2, \ldots$ can be non-convex UoPs [34].

The resulting J^* and π^* are both PWA functions sharing the same polyhedral partition of the feasible region $\bar{X}$ [16]. However, the complexity (i.e., the number of polyhedral regions or affine functions) of J^* and π^* is exponential in the dimension of the system [33], so that storing the affine functions and regions of J^* and π^* needs a huge amount of memory. Secondly, the number of mp-LPs that need to be solved per iteration also grows exponentially with the problem dimension. Moreover, the online implementation of [16] needs to search which polyhedron the measured state belongs to. For high-dimensional systems, these regions may have complex representations. Based on these observations, it is thus necessary to simplify both the procedure of solving the Bellman equation and the control policy, by using some approximation methods. However, the VI formulation in (3.5) is not suitable for approximation because the probably non-convex constraint $f_{\mathrm{PWA}}(x,u) \in X_{k-1}$ leads to too complex optimization problems.

3.3. VALUE ITERATION WITH PENALTY FUNCTIONS

To deal with this issue, we consider soft state constraints by defining a penalty function. Suppose that each $X^{(i)}$ of the UoP constraint set $X = \bigcup_{i=1}^{r_0} X^{(i)}$ has the half-space representation $X^{(i)} := \{x \in \mathbb{R}^{n_x} | E_X^{(i)} x \leq g_X^{(i)}\}$, where $E_X^{(i)} \in \mathbb{R}^{m_x^{(i)} \times n_x}$, $g_X^{(i)} \in \mathbb{R}^{m_x^{(i)}}$, and $m_x^{(i)}$ is the number of rows of $E_X^{(i)}$. We design the penalty function $P(\cdot,\cdot)$ as the following min-max forms:

$$P(x,X) = p \min_i \max \left\{0, (E_X^{(i)})_1.x - (g_X^{(i)})_1, \ldots, (E_X^{(i)})_{m_x^{(i)}}.x - (g_X^{(i)})_{m_x^{(i)}}\right\} \text{ or}$$

$$P(x,X) = p \min_i \sum_{j=1}^{m_x^{(i)}} \max \left\{0, (E_X^{(i)})_j.x - (g_X^{(i)})_j\right\}, \tag{3.6}$$

where $(g_X^{(i)})_j$ is the jth element of the vector $g_X^{(i)}$, $m_x^{(i)}$ is the number of rows of $E_X^{(i)}$, and $p > 0$ is the constraint violation penalty weight. When X reduces to a polyhedron, i.e., $r_0 = 1$, the minimum operator in (3.6) will be removed.

An important property of P is that P is a PWA function w.r.t. its first argument. This means that adding such a penalty function into the cost function in (3.3) will not change the PWA properties of the optimal value function and the optimal policy. Note that we avoid barrier functions such as the logarithmic barrier function because they can go to infinity in any compact constraint sets and will deprive the value function of the PWA property. Besides, we do not penalize the input constraint violation because the input constraints are linear constraints that can be readily handled.

In case 2 of (3.5), in addition to enforcing a soft penalty for X, we need to reconstruct the initial value function since J_{CL} is undefined outside X_{CI}. To achieve this, we need to penalize $J_{\mathrm{CL}}(x)$ for x outside X_{CI} by finite values, i.e., define the penalized initial value function

$$J_0^{\mathrm{soft}}(x) := \begin{cases} J_{\mathrm{CL}}(x), & x \in X_{\mathrm{CI}} \\ J_{\mathrm{CL}}(\bar{z}) + P(x, X_{\mathrm{CI}}), & x \notin X_{\mathrm{CI}}, \end{cases} \tag{3.7}$$

where $\bar{z}(\cdot) : \mathcal{X} \to \mathcal{X}$ is one of the optimizers of the following mp-LP:

$$\bar{z}(x) \in \arg\min_{z \in X_{\mathrm{CI}}} ||z - x||_\infty. \tag{3.8}$$

Based on the defined penalty function, a VI algorithm with penalty is developed as follows:

In Algorithm 2, we consider two options for the VI, in which we define two Bellman operators for $J : \mathcal{X} \to \mathbb{R}$. The first one used in option 1 is

$$\Gamma_{\mathrm{p},1} J(x) := \min_{u \in U} \; l_{\mathrm{p}}(x,u) + J\left(f_{\mathrm{PWA}}(x,u)\right), x \in \mathcal{X}, \tag{3.9}$$

where $l_{\mathrm{p}}(x,u) = l(x,u) + P(x,X)$. The second one used in option 2 is

$$\Gamma_{\mathrm{p},2} J(x) := \min_{u \in U} \; l(x,u) + J\left(f_{\mathrm{PWA}}(x,u)\right) + P_{k-1}\left(f_{\mathrm{PWA}}(x,u)\right), \; x \in \mathcal{X}, \tag{3.10}$$

Algorithm 2 Value iteration with penalty

Input: The PWA system f_{PWA}, the state and input constraints $x \in X$ and $u \in U$, the stage cost l, and J_{CL} and X_{CI} (for (option (b)).
Output: A value function $J_{k-1}^{\text{soft}}(\cdot) : \mathcal{X} \to \mathbb{R}_{\geq 0}$.

1: Initialize the value function (option (a)) $J_0^{\text{soft}}(x) \leftarrow 0$, $\forall x \in \mathcal{X}$, or (option (b)) by (3.7).
2: **for** $k = 1, 2, \ldots$ **do**
3: the value iteration $J_k^{\text{soft}}(x) \leftarrow \Gamma_{\text{p},\alpha} J_{k-1}^{\text{soft}}(x)$, $\forall x \in \mathcal{X}$, where $\alpha = 1$ if option 1 is chosen, or $\alpha = 2$ if option 2 is chosen, and $\Gamma_{\text{p},\alpha}$ is defined in (3.9) and (3.10).
4: If $J_k^{\text{soft}}(x) = J_{k-1}^{\text{soft}}(x)$, $\forall x \in \mathcal{X}$, **break**.
5: **end for**

where $P_0(x) = 0, \forall x \in \mathcal{X}$ and $P_k(x) = P(x, X), \forall x \in \mathcal{X}$ and $\forall k > 0$. Since the state constraints are removed, the working region of VI is the whole state space $\mathcal{X}$. Besides, we also consider two options (options (a) and (b)) for the initialization of the value function. The combinations of the above options result in four different options: options 1(a), 1(b), 2(a), and 2(b). Note that adding a penalty to the stage cost, which is applied in option 1 of Algorithm 2, is common in existing constrained ADP methods [210]. In comparison, in option 2 of the algorithm, we propose a novel scheme in which we add a penalty into the cost-to-go $J_{k-1}^{\text{soft}}\left(f_{\text{PWA}}(x,u)\right)$. In the following theorem, we will analyze the PWA property and continuity of each J_k^{soft} as well as the convergence of the sequence $\{J_k^{\text{soft}}\}_{k=0}^{\infty}$ to a fixed optimal value function in all options.

Theorem 3.3.1. Considering Algorithm 2, if Assumptions 3.2.1-3.2.2 hold in option (a) and Assumptions 3.2.1-3.2.3 hold in option (b), then each J_k^{soft}, $k < \infty$ is a continuous PWA function on $\mathcal{X}$ and the value function sequence $\{J_k^{\text{soft}}(x)\}_{k=0}^{\infty}$ converges point-wise to

$$
\begin{aligned}
J^{\text{soft}*}(x) := \min_{\{u_i, x_i\}_{i=0}^{\infty}} \; & \sum_{i=0}^{\infty} l_{\text{p}}(x_i, u_i) \\
\text{s.t.} \quad & x_{i+1} = f_{\text{PWA}}(x_i, u_i),\ u_i \in U,\ i = 0, 1, \ldots \\
& x_0 = x
\end{aligned}
$$

for all $x \in X$.

Proof. The proof contains three parts.

Part 1: Convergence of the VI sequence in option 1. For Option 1(a), in the nonlinear parametric problem in (3.9), U is closed, and the set $\left\{u \in U \middle| J_k^{\text{soft}}(x) = l(x,u) + P(x,X) + J_{k-1}^{\text{soft}}\left(f_{\text{PWA}}(x,u)\right)\right\}$ is nonempty for any x in $\mathcal{X}$. Then, according to [13, Theorem 4.2.1], J_k^{soft} is lower-semicontinuous on $\mathcal{X}$. Hence, $\{x \in \mathcal{X} | J_k^{\text{soft}}(x) \leq \lambda\}$ with $\lambda \in \mathbb{R}$ are closed. As l_p and f_{PWA} are continuous on $\mathcal{X} \times \mathcal{U}$, the set $U_k(x, \lambda) = \{u \in U | \; l(x,u) + P(x,X) + J_{k-1}^{\text{soft}}\left(f_{\text{PWA}}(x,u)\right) \leq \lambda\}$ is closed and thus compact for all $x \in \mathcal{X}$, $\lambda \in \mathbb{R}$, and $k \geq 1$. Therefore, the compactness assumption in [27] is satisfied and $\{J_k^{\text{soft}}\}_{k=0}^{\infty}$ converges point-wise to $J^{\text{soft}*}$ [27, Proposition 2].

For Option 1(b), we define $J_0^{\infty}(x) = \begin{cases} J_{\mathrm{CL}}(x), & x \in X_{\mathrm{CI}} \\ \infty & x \notin X_{\mathrm{CI}} \end{cases}$. The point-wise convergence of the VI $J_k^{\infty}(x) = \Gamma_{\mathrm{p},1} J_{k-1}^{\infty}(x)$ to $J^{\mathrm{soft}*}(x)$ is always guaranteed by [27, Proposition 2] since $J_0^{\infty}(x) \geq J^{\mathrm{soft}*}(x)$, $\forall x \in \mathcal{X}$. By applying the monotonicity of the Bellman operator $\Gamma_{\mathrm{p},1}$ [28], we get the convergence of the VI sequence $\{J_k^{\mathrm{soft}}\}_{k=0}^{\infty}$ to $J^{\mathrm{soft}*}$ in option 1(b).

Part 2: Continuity and PWA property of the value function in option 1. By recursively iterating (3.9), it is observed that $J_k^{\mathrm{soft}}(x)$ can be derived via the following batch approach:

$$\begin{aligned} J_k^{\mathrm{soft}}(x) = \min_{u_0,\ldots,u_{k-1},\, x_0,\ldots,x_k} & \sum_{i=0}^{k-1} l_{\mathrm{p}}(x_i, u_i) + J_0^{\mathrm{soft}}(x_k) \\ \text{s.t. } & x_{i+1} = f_{\mathrm{PWA}}(x_i, u_i), u_i \in U, i = 0, \ldots, k-1 \\ & x_0 = x. \end{aligned} \tag{3.11}$$

For Option 1(a), yhe proof of continuity follows from the proof of [35, Corollary 17.2], because the objective function in (3.11) is continuous and there is no state constraint. The proof of the PWA property of J_k^{soft} follows a similar approach to that of [35, Theorem 17.3], with detailed exposition given in [99].

For Option 1(b), we can prove that the initial value function J_0^{soft} is continuous and PWA on $\mathcal{X}$ (the detailed proof is given in [99]). As a result, the remainder of the proof of the continuity and PWA property of J_k^{soft} is similar to that in option 1(a). Thus, we have completed the proof of the statements of Theorem 3.3.1 in option 1.

Part 3: Equivalence of the VI sequences for options 1 and 2. In option 2, by iterating $J_k^{\mathrm{soft}}(x) = \Gamma_{\mathrm{p},2} J_{k-1}^{\mathrm{soft}}(x)$ from k to 0, we can get the expression of J_k^{soft} via the batch approach:

$$J_k^{\mathrm{soft}}(x) = \text{the optimal value of (3.11)} - P(x, X). \tag{3.12}$$

From (3.12), we notice that the difference between the value functions in options 1 and 2 at the same iteration is $P(x, X)$, which is always continuous in x and equals zero if $x \in X$. Combining (3.12) with the first and second parts of the proof proves the statements of Theorem 3.3.1. □

Option 2, incorporating penalties into the cost-to-go, yields the equivalent value function J_k^{soft} as option 1, which adds penalties to stage costs. Both options can alleviate the violation of state constraints by adding penalties to the overall infinite-horizon cost function. A detailed comparison between options 1 and 2 is given in Section 3.4.3.

3.4. CONSTRAINED ADP ALGORITHM

3.4.1. ALGORITHM DESIGN

Continuity of the value functions, established in Theorem 3.3.1, is desired since it enables a universal approximation capability [94]. With Algorithm 2, a tractable ADP ap-

proach can be developed to approximate each J_k^{soft}. In particular, a function approximator (critic) $\hat{J}_k(\cdot,\theta_k)$, which is parameterized by θ_k, is constructed to replace J_k^{soft}. In each iteration k, a set $X_s = \{x^{(i)}\}_{i=1}^{N_x}$ of state samples is collected from a compact region of interest $\Omega_k \subseteq \mathcal{X}$, according to some strategies such as sampling from a uniform grid and random sampling [40]. Here, N_x is the number of samples. The update of the parameter θ_k minimizes $\sum_{i=1}^{N_x}[\Gamma_{p,\alpha}\hat{J}_{k-1}(x,\theta_{k-1})|_{x=x^{(i)}} - \hat{J}_k(x^{(i)},\theta_k)]^2$. The iterative procedure stops when the difference between θ_k and θ_{k-1} is small enough. The detailed procedure is given in Algorithm 3.

3

Algorithm 3 Constrained approximate value iteration

Input: The PWA system f_{PWA}, the state and input constraints $x \in X$ and $u \in U$, the stage cost l, and J_{CL} and X_{CI} (for (option (b)), $\mathcal{X}$, the sample set X_s, ρ_v, and ϵ.
Output: A value function $\hat{J}_{k-1}(\cdot,\theta_k)$

1: Option (a): Initialize the value function $\hat{J}_0(\cdot,\theta_0) \leftarrow 0, \forall x \in \Omega_0$, where $\Omega_0 = \mathcal{X}$ in option 1(a) or $\Omega_0 = X$ in option 2(a).
Option (b): Initialize the value function $\hat{J}_0(\cdot,\theta_0)$ by

$$\theta_0 \leftarrow \arg\min_{\theta} \sum_{x^{(i)} \in X_s \cap \Omega_0} \rho_v(x^{(i)}) \left(J_0^{\text{soft}}(x^{(i)}) - \hat{J}_0(x^{(i)},\theta) \right)^2, \tag{3.13}$$

where J_0^{soft} is from (3.7), and $\Omega_0 = \mathcal{X}$ in option 1(b) or $\Omega_0 = X_{\text{CI}}$ in option 2(b).
2: **for** $k = 1,2,\ldots$ **do**
3: **if** option 1 is chosen, **then** let $\Omega_k \leftarrow \mathcal{X}$ and $\alpha \leftarrow 1$.
4: **end if**
5: **if** option 2 is chosen, **then** let $X_k \leftarrow \text{Pre}(X_{k-1}) \cap X$, $\Omega_k \leftarrow X_k$, and $\alpha \leftarrow 2$.
6: **end if**
7: Obtain the target value $v_k^{(i)}$ by

$$v_k^{(i)} \leftarrow \Gamma_{p,\alpha}\hat{J}_{k-1}(x,\theta_{k-1})|_{x=x^{(i)}}, \forall x^{(i)} \in X_s \cap \Omega_k. \tag{3.14}$$

8: Find θ_k such that

$$\theta_k \leftarrow \arg\min_{\theta} \sum_{x^{(i)} \in X_s \cap \Omega_k} \rho_v(x^{(i)}) \left(v_k^{(i)} - \hat{J}_k(x^{(i)},\theta) \right)^2. \tag{3.15}$$

9: **if** $|\hat{J}_k(x,\theta_k) - \hat{J}_{k-1}(x,\theta_{k-1})| \le \epsilon(x),\ \forall x \in \Omega_k \cap \Omega_{k-1}$, **then break**.
10: **end if**
11: **end for**

In (3.15) of step 8 of Algorithm 3, $\rho_v(\cdot) : \mathcal{X} \to \mathbb{R}_{>0}$ is the state relevance weighting function. In step 9, $\epsilon(\cdot) : \mathcal{X} \to \mathbb{R}_{\ge 0}$ is a tolerance function, determining whether $\hat{J}_{k-1}$ is satisfactory. Both of ρ_v and ϵ will be designed later in Section 3.5.1. Besides, in practice, one would also need a limit on the maximum number of iterations.

With $\hat{J}_{k-1}$ available, a sub-optimal control policy $\hat{\pi}^{\text{im}}(x)$ can be implicitly determined by

$$\hat{\pi}^{\text{im}}(x) \in \arg\min_{u\in U} \; l(x,u) + \hat{J}_{k-1}(f_{\text{PWA}}(x,u),\theta_{k-1}),\ \forall x \in \Omega_k. \tag{3.16}$$

In step 1 of Algorithm 3, the function approximator is initialized by regressing J_0^{soft}, provided that the explicit form of J_{CL} is known. If the explicit form of J_{CL} is not available but a stabilizing and safe piecewise linear feedback law $\pi_{\text{PWL}}(\cdot): \mathcal{X} \to \mathcal{U}$ on X_{CI} is known, we can initialize $\hat{J}_0$ as an approximation of $J_{\pi_{\text{PWL}}}$, which is also a control Lyapunov function, by doing a policy evaluation [40].

To carry out the iterative procedure in Algorithm 3 more efficiently, we need to use a proper function approximator at each iteration. Since each J_k^{soft} is a PWA function, it is preferable that the candidate approximator can also output a PWA function. Suitable choices are thereby NNs with (leaky) rectifier linear units (ReLUs) as activation functions, difference of two max-affine functions, Min-Max NNs [136], and so on. Detailed descriptions of these function approximators are given in [99].

Computation of X_k under the UoP state constraint: In option 2, one is required to compute the reachable sets X_k. The following lemma states that we are able to obtain X_k by performing polyhedral operations even if the state constraint is UoP.

Lemma 3.4.1. For the PWA system (3.1) with a polytopic input constraint $u \in U$ and a UoP state constraint $x \in X = \bigcup_{i=1}^{r_0} X^{(i)}$, the set iterates $X_k = \text{Pre}(X_{k-1}) \cap X,\ k = 1,2,\ldots$ make each X_k be a UoP. Then, if X_{k-1} is in the form of $X_{k-1} = \bigcup_{i=1}^{r_{k-1}} X_{k-1}^{(i)}$, where each $X_{k-1}^{(i)}$ is a polyhedron, then X_k is given by

$$X_k = \bigcup_{j=1}^{r_0} \bigcup_{i=1}^{r_{k-1}} \text{Pre}(X_{k-1}^{(i)}) \cap X^{(j)}. \tag{3.17}$$

Proof. The proof can be constructed by induction. Noticing that $X_0 = X$ is a UoP, we assume that X_{k-1} is a UoP in the form of $X_{k-1} = \bigcup_{i=1}^{r_{k-1}} X_{k-1}^{(i)}$. Then, the backward-reachable set $\text{Pre}(X_{k-1})$ can be computed as $\text{Pre}(X_{k-1}) = \bigcup_{i=1}^{r_{k-1}} \text{Pre}(X_{k-1}^{(i)})$. This results in

$$\begin{aligned} X_k &= \text{Pre}(X_{k-1}) \cap X = \bigcup_{j=1}^{r_0} \text{Pre}(X_{k-1}) \cap X^{(j)} \\ &= \bigcup_{j=1}^{r_0} \left(\bigcup_{i=1}^{r_{k-1}} \text{Pre}(X_{k-1}^{(i)}) \right) \cap X^{(j)} = \bigcup_{j=1}^{r_0} \bigcup_{i=1}^{r_{k-1}} \text{Pre}(X_{k-1}^{(i)}) \cap X^{(j)}. \end{aligned} \tag{3.18}$$

In the last line of (3.18), $\text{Pre}(X_{k-1}^{(i)}) \cap X^{(j)}$ is a UoP for each i and j because $X_{k-1}^{(i)}$ and $X^{(j)}$ are polyhedra. As a result, X_k is a UoP. This proves the statement for k. Finally, by induction, we complete the proof of Lemma 3.4.1. □

Mixed-integer formulations of problems (3.14) and (3.16): Problems (3.14) and (3.16) have similar forms. They can be transformed into MILP problems since both the PWA system and PWA function approximators are MILP representable, which is shown in [99].

Here, we say a function J is MILP representable if J can be represented by a set of mixed-integer linear equations and inequalities containing additional variables. We note this set by $\mathrm{gr}_{\mathrm{MILP}}(J)$. As a result, we obtain mixed-integer formulations of problems (3.14) and (3.16). Precisely, if, e.g., the first type of the penalty function in (3.6) and the infinity norm of the stage cost are chosen, (3.14) in option 1 can be equivalently written as the following MILP:

$$\begin{aligned}\min_{\substack{u,\varepsilon^{(x)},\varepsilon^{(u)},\\ \varepsilon^{(J)},\varepsilon^{(P)},x^+}}\ & \varepsilon^{(x)}+\varepsilon^{(u)}+\varepsilon^{(J)}+\varepsilon^{(P)}\\ \text{s.t.}\ & -1_{n_x}\varepsilon^{(x)}\le \pm Qx,\ -1_{n_u}\varepsilon^{(u)}\le \pm Ru,\ u\in U\\ & (x^+,\varepsilon^{(J)})\in \mathrm{gr}_{\mathrm{MILP}}(\hat{J}_{k-1}),\ (x,\varepsilon^{(P)})\in \mathrm{gr}_{\mathrm{MILP}}(P(\cdot,X)),\ ([x^T\ u^T]^T,x^+)\in \mathrm{gr}_{\mathrm{MILP}}(f_{\mathrm{PWA}}).\end{aligned}$$

Similar transformations can be achieved for problem (3.16), for option 2 of Algorithm 3, and for the other choice of P and l.

MILP problems can be effectively solved by using the branch-and-bound approach [206], which is a global optimization algorithm.

Different from (3.14), Problem (3.16) is solved online. Even though it still belongs to an MILP problem, (3.16) can be solved more rapidly than a general hybrid MPC problem with a long horizon, because (3.16) in general results much fewer auxiliary and binary variables. That is one of the main benefits of using ADP or RL.

Remark 3.4.1. When there are approximation errors, the convergence of $\hat{J}_k$ to J^{soft^*} is in general not guaranteed because the infinite-horizon cost (3.3) is undiscounted and does not induce a contraction property for the Bellman operator. In general, one can add a discount factor to (3.3) to ensure the convergence of the value iteration under approximation errors, but this may come at the cost of weakening the stability [161].

3.4.2. APPROXIMATING EXPLICIT POLICIES

Since two PWA functions f_{PWA} and $\hat{J}_{k-1}$ are coupled in (3.16), (3.16) may still have many auxiliary and binary variables, if, e.g., a multiple-layer (deep) NN is used. As (3.16) needs to be solved online, the advantage of low computational complexity brought by ADP is not obvious. To avoid solving complex MILP problems online, the policy $\hat{\pi}^{\mathrm{im}}$ can also be represented explicitly, in which case it usually needs to be approximated by a second function approximator (actor). The actor is also recommended to have a PWA form since the optimal control policy π^* is PWA.

As the optimizer $\hat{\pi}^{\mathrm{im}}$ can be discontinuous and not unique, instead of using supervised learning methods to train the actor, we can directly construct a parameterized policy $\hat{\pi}^{\mathrm{ex}}(\cdot,\omega)$ with parameter ω and update ω to minimize the expectation of the objective function in (3.16) w.r.t. the sample distribution d_s used in Algorithm 3. This results in the following policy optimization problem

$$\begin{aligned}\omega^*\in \arg\min_{\omega}\ & \mathbb{E}_{x\sim d_s}[\rho_\pi(x)(l(x,\hat{\pi}^{\mathrm{ex}}(x,\omega))+\hat{J}_{k-1}(f_{\mathrm{PWA}}(x,\hat{\pi}^{\mathrm{ex}}(x,\omega)),\theta_{k-1}))]\\ \text{s.t.}\ & \mathbb{E}_{x\sim d_s}[\hat{\pi}^{\mathrm{ex}}(x,\omega)]\in U,\end{aligned}\tag{3.19}$$

where $\rho_\pi(\cdot): \mathcal{X} \to \mathbb{R}_{>0}$ is another state relevance weighting function to be specified later in Section 3.5.1. Similar to the critic, we specify $\hat{\pi}^{\text{ex}}$ as a PWA approximator.

To solve (3.19), the policy gradient method, combined with the Lagrangian relaxation approach [189] or the augmented Lagrangian approach [128] for constraint handling, can be employed.

The above procedures are conducted offline. Ideally, if there are no approximation errors on both the critics and the actor, and the penalty weight p and the number of iterations are infinite, we have $X_\infty = \bar{X}$ and $\hat{\pi}^{\text{ex}}(\cdot, \omega^*) \equiv \hat{\pi}^{\text{im}}$. Consequently, $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$ will always make the system satisfy all the constraints for the initial condition $x_0 \in \bar{X}$. However, due to approximation errors and the finite penalty weight, the policy cannot always satisfy the state and input constraints. As the input constraints are usually hard constraints, in the online setting we project $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$ onto U when the current state x_t is received. This results in a convex quadratic program:

$$\hat{\pi}^{\text{ex}}_{\text{proj}}(x) = \arg\min_{u \in U} ||u - u_t^{\text{ex}}||_2, \tag{3.20}$$

where $u_t^{\text{ex}} = \hat{\pi}^{\text{ex}}(x_t, \omega^*)$. Problem (3.20) can be treated as a parametric quadratic program with the parameter u_t^{ex}. Therefore, the optimizer of (3.20) is unique, PWA [35, Theorem 6.7], and also MILP representable [177, Lemma 4].

Remark 3.4.2. The number of decision variables and the number of constraints in the convex QP problem (3.20) are not larger than those in the MILP problem (3.16). It is known that convex QP problems are P problems, while MILP problems are NP-hard problems. Therefore, the consideration of the explicit policy $\hat{\pi}^{\text{ex}}(x_t, \omega^*)$ further enhances the online computational efficiency compared to (3.16).

3.4.3. DISCUSSIONS ON THE ADP METHOD

Comparison between options 1 and 2 of Algorithm 3: The main differences between options 1 and 2 are the training region Ω_k and the way each $\hat{J}_{k-1}$ iterates. These differences lead to differences in the adaptability and efficiency of options 1 and 2:

- Compared to option 1, option 2 is more efficient in sampling and can result in a better approximation accuracy, because the working region Ω_k in option 2 is in general much smaller than $\mathcal{X}$. This is also the main advantage of adding penalties into the cost-to-go over adding penalties into the stage cost. We note that to implement option 1, one should choose a region of interest for sampling, and the region must be larger than X, so that the constraint violation can be penalized in the critic. However, the states that can be steered to the origin and have zero constraint violation are all contained in $\bar{X}$, which is much smaller than $\mathcal{X}$. Accordingly, in option 2, we concentrate on X_k, which converges to $\bar{X}$ as k goes to infinity.
- On the other hand, option 2 needs to compute the k-step controllable set Ω_k (X_k) while option 1 does not. Therefore, for large-scale PWA systems, for which the exact computation of each X_k is computationally very demanding, option 1 is preferable.

Comparison to RL with a safety filter: In the schemes of [130], [198], RL policies are pro-

jected onto a safe set where both state and input constraints are considered. That design is motivated by the fact that the policies in [130], [198] are derived from standard RL algorithms that do not account for constraints. In comparison, our proposed ADP algorithms incorporate the state constraints into the cost function by adding penalty terms for violating the constraints. Besides, input constraints are regarded as hard constraints in the optimization problems (3.14), (3.16), and (3.20).

3.5. PERFORMANCE ANALYSIS AND VERIFICATION

In this section, we will characterize the stability and safety of the closed-loop system with the policies $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$, and also the sub-optimality properties of these policies. Firstly, we provide general conditions under which stability and safety hold. These conditions can guide the parameter tuning of Algorithm 3. Then, we give sub-optimality guarantees, i.e., a bound on the mismatch between the infinite cost of the policies and real value functions. Finally, we develop verifiable stability and safety conditions. We say that a closed-loop system is safe in a set if its states and inputs satisfy the constraints for all trajectories starting from the set.

3.5.1. STABILITY AND SAFETY ANALYSIS

First, we state a useful lemma that gives some properties of the value function J^{soft}_k.

Lemma 3.5.1. Consider Algorithm 2. Suppose that Assumptions 3.2.1-3.2.2 hold in option (a) and Assumptions 3.2.1-3.2.3 hold in option (b).

(i) Then, there exists a positive constant $\gamma < \infty$ such that for all $k \geq 0$, $J^{\text{soft}}_k(x) \leq \gamma l(x,0)$, $\forall x \in \bar{X}$;

(ii) there exists a finite $\bar{k} > 0$ such that for all $k \geq \bar{k}$, we have

$$J^{\text{soft}}_k(x) - J^{\text{soft}}_{k-1}(x) \leq \beta l(x,0),\ \forall x \in \bar{X}, \text{ with } \beta \in (0,1). \tag{3.21}$$

Proof. Firstly, the result in (i) is straightforward, based on the fact that both J^{soft}_k and $l(\cdot,0)$ are continuous PWA functions on $\bar{X}$ and equal to zero iff $x = 0$.

For (ii), we will prove by contradiction the existence of a finite $\bar{k}$ such that $J^{\text{soft}}_{\bar{k}}(x) - J^{\text{soft}}_{\bar{k}-1}(x) \leq \beta l(x,0)$, $\forall x \in \bar{X}$ holds in option 1. Suppose that it does not hold, then for any $k > 0$, there exists a $\bar{x} \in \bar{X}$ such that $J^{\text{soft}}_k(\bar{x}) - J^{\text{soft}}_{k-1}(\bar{x}) \geq l(\bar{x},0)$. According to the contraction property of the Bellman operator [28], we have $J^{\text{soft}}_i(\bar{x}) - J^{\text{soft}}_{i-1}(\bar{x}) \geq l(\bar{x},0)$, $\forall i \in \{1,...,k\}$. Summing over i yields $J^{\text{soft}}_k(\bar{x}) \geq k l(\bar{x},0)$. Letting $k \to \infty$ makes $J^{\text{soft}}_\infty(\bar{x})$ contradict (i) of Lemma 3.5.1. Moreover, again using the contraction property of $\Gamma_{\text{p},1}$ leads to the conclusion in (ii). Finally, (3.12) tells that the difference of $J^{\text{soft}}_{\bar{k}}(x)$ in options 1 and 2 is $P(x,X)$, which is independent of k. Therefore, (3.21) also holds for option 2 of Algorithm 2. □

Then, we are ready to state the main result in this subsection.

Theorem 3.5.1. Consider Algorithm 3 and the proposed policies $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$. Let Ω be a compact subset of X. Suppose that Assumptions 3.2.1-3.2.2 hold in option (a) and Assumptions 3.2.1-3.2.3 hold in option (b). Consider the following conditions:

(C1): There exist a constant $\zeta \in (0,1)$ and a positive integer k such that $\left|\hat{J}_{k-1}(x) - J^{\text{soft}}_{k-1}(x)\right| \leq \zeta J^{\text{soft}}_{k-1}(x),\ \forall x \in \Omega$.

(C2): There exist a constant $e_{\text{p}} > 0$ and a positive integer k such that $\hat{J}_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) - \hat{J}_{\hat{\pi}^{\text{im}}}(x) \leq e_{\text{p}} l(x,0),\ \forall x \in \Omega$. Here, $\hat{J}_{\pi}(\cdot)$ is defined by $\hat{J}_{\pi}(x) := l(x,\pi(x)) + \hat{J}_{k-1}(f_{\text{PWA}}(x,\pi(x)))$.

As a result, we have:

(i) If C1 holds with $k \geq \bar{k}$ and

$$(1+\zeta)(1-\beta) > \max(2\zeta\gamma,\ 1), \tag{3.22}$$

where $\bar{k}$, β, and γ come from Lemma 3.5.1, then the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \hat{\pi}^{\text{im}}(x_t)),\ t = 0,1,\ldots$ is asymptotically stable and safe in $\mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$.

(ii) If C1 and C2 hold with $k \geq \bar{k}$, and

$$(1+\zeta)(1-\beta) > \max(2\zeta\gamma + e_{\text{p}},\ 1), \tag{3.23}$$

then the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \hat{\pi}^{\text{ex}}_{\text{proj}}(x_t)),\ t = 0,1,\ldots$ is asymptotically stable and safe in $\mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$.

Proof. For each $k \geq \bar{k}$, we define the policy $\pi_k(\cdot) : \mathcal{X} \to \mathcal{U}$ as (one of) the optimizer(s) of $\Gamma_{\text{p},\alpha} J^{\text{soft}}_{k-1}(\cdot)$, $\alpha = 1$ or 2. For every $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \subseteq X$, according to (3.5.1),

$$J^{\text{soft}}_{k}(x) = \Gamma_{\text{p},\alpha} J^{\text{soft}}_{k-1}(x) \geq l(x,\pi_k(x)) + J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\pi_k(x))) \tag{3.24}$$

holds in both options. Together with (3.21), (3.24) yields

$$\begin{aligned} J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\pi_k(x))) - J^{\text{soft}}_{k-1}(x) &\leq -(1-\beta) l(x,\pi_k(x)) \\ &\leq -(1-\beta) l(x,0), \end{aligned} \tag{3.25}$$

which means that for every $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega)$, the policy π_k will make $f_{\text{PWA}}(x,\pi_k(x)) \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega)$.

Now, we show that J^{soft}_{k-1} and $\hat{J}_{k-1}$ are Lyapunov functions for the system with $\hat{\pi}^{\text{im}}$. In particular, for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$, if C1 and (3.22) hold, we have

$$\begin{aligned} & l(x,\hat{\pi}^{\text{im}}(x)) + \hat{J}_{k-1}(f_{\text{PWA}}(x,\hat{\pi}^{\text{im}}(x))) \\ \leq & l(x,\pi_k(x)) + \hat{J}_{k-1}(f_{\text{PWA}}(x,\pi_k(x))) \\ \leq & l(x,\pi_k(x)) + (1+\zeta) J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\pi_k(x))) \\ \leq & (1+\zeta) J^{\text{soft}}_{k-1}(x) + \left(1 - (1+\zeta)(1-\beta)\right) l(x,\pi_k(x)) \\ \leq & \hat{J}_{k-1}(x) + 2\zeta J^{\text{soft}}_{k-1}(x) + \left(1 - (1+\zeta)(1-\beta)\right) l(x,\pi_k(x)). \end{aligned} \tag{3.26}$$

In (3.26), the first inequality is true since $\hat{\pi}^{\text{im}}$ is an optimizer of problem (3.16); the second and the last inequalities hold because x and $f_{\text{PWA}}(x,\pi_k(x))$ are all in Ω, in which C1 holds; and the third inequality is correct owing to (3.25). Since $1-(1+\zeta)(1-\beta)<0$, considering Lemma 3.5.1, (3.26) results in

$$\hat{J}_{k-1}(x^+)-\hat{J}_{k-1}(x)\leq\left(2\zeta\gamma-(1+\zeta)(1-\beta)\right)l(x,0), \tag{3.27}$$

with $x^+ = f_{\text{PWA}}(x,\hat{\pi}^{\text{im}}(x))$. The right-hand side of (3.27) is strictly negative except for $x=0$ according to (3.22). Therefore, we get that $\forall x\in\mathcal{B}(J^{\text{soft}}_{k-1},\Omega)\cap\mathcal{B}(\hat{J}_{k-1},\Omega)$, $f_{\text{PWA}}(x,\hat{\pi}^{\text{im}}(x))\in\mathcal{B}(\hat{J}_{k-1},\Omega)$ holds. Condition C1 implies that $\hat{J}_{k-1}(x)>0$, $\forall x\in(\mathcal{B}(J^{\text{soft}}_{k-1},\Omega)\cap\mathcal{B}(\hat{J}_{k-1},\Omega))\backslash\{0\}$ and $\hat{J}_{k-1}(0)=0$. This shows that J^{soft}_{k-1} is a Lyapunov function for the system $x_{t+1}=f_{\text{PWA}}(x_t,\hat{\pi}^{\text{im}}(x_t))$.

Similarly to (3.26) and (3.27), we can get the following inequality for J^{soft}_{k-1}:

$$\begin{aligned}&l(x,\hat{\pi}^{\text{im}}(x))+J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\hat{\pi}^{\text{im}}(x)))\\ \leq&\frac{(1+\zeta)J^{\text{soft}}_{k-1}(x)+(1-(1+\zeta)(1-\beta))l(x,\pi_k(x))-\zeta\, l(x,\hat{\pi}^{\text{im}}(x))}{1-\zeta}.\end{aligned} \tag{3.28}$$

With (3.22) and Lemma 3.5.1, (3.28) readily leads to

$$J^{\text{soft}}_{k-1}(x^+)-J^{\text{soft}}_{k-1}(x)\leq\frac{2\zeta\gamma-(1+\zeta)(1-\beta)}{1-\zeta}l(x,0), \tag{3.29}$$

which means that J^{soft}_{k-1} is strictly decreasing from any $x\in(\mathcal{B}(J^{\text{soft}}_{k-1},\Omega)\cap\mathcal{B}(\hat{J}_{k-1},\Omega))\backslash\{0\}$ to the next state. Combining (3.27) and (3.29), we note that $\mathcal{B}(J^{\text{soft}}_{k-1},\Omega)\cap\mathcal{B}(\hat{J}_{k-1},\Omega)$ is a positively invariant set. This, together with the Lyapunov functions $\hat{J}_{k-1}$ and J^{soft}_{k-1}, leads to the asymptotic stability and safety of the closed-loop system.

Next, we analyze the behavior of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$. With C1 and C2, $\hat{J}_{k-1}(x^+)-\hat{J}_{k-1}(x)$ with $x^+=f_{\text{PWA}}(x,\hat{\pi}^{\text{ex}}_{\text{proj}}(x))$ is upper bounded by the right-hand side of (3.27) plus $e_{\text{p}}l(x,0)$, and $J^{\text{soft}}_{k-1}(x^+)-J^{\text{soft}}_{k-1}(x)$ is also upper bounded by the right-hand side of (3.29) plus $e_{\text{p}}l(x,0)$. Together with (3.23), this results in the asymptotic stability and safety of the closed-loop system with the policy $\hat{\pi}^{\text{ex}}_{\text{proj}}$. □

Remark 3.5.1. Although Theorem 3.5.1 provides sufficient conditions for stability, some of them (e.g., C1 and C2) are difficult to verify. A method that can verify C1 and C2 in a probabilistic way is reported in [106]. On the other hand, Theorem 3.5.1 suggests several ways to design the parameters and function approximators in Algorithm 3. For practical verification of stability and safety, in Section 3.5.3, we move beyond using Conditions C1 and C2. Instead, we propose an offline verification framework based on solving MILP problems. As will be demonstrated in the case study, this framework effectively verifies safe and stable regions.

(i) All results in Theorem 3.5.1 require (3.21) to hold. The left-hand side of (3.21) is about the residual error of VI. It indicates that a suitable tolerance function ϵ, which determines the stopping condition at step 9 of Algorithm 3, could be $\epsilon(x)=e_{\text{tole}}l(x,0)$, for some $e_{\text{tole}}\in(0,1)$.

(ii) Condition C1 limits the mismatch between $\hat{J}_{k-1}$ and J^{soft}_{k-1}, which further limits the approximation error $v_i(x) := \hat{J}_i(x) - \Gamma_{\text{p},\alpha}\hat{J}_{i-1}(x)$, $i = 1,...,k$ of VI. To make ζ as small as possible, which helps to fulfill (3.22) and (3.23), ρ_v in (3.15) could be specified by $\rho_v(x) = 1/l^2(x,0)$. To understand this, we consider the state trajectory x_0, $x_1,...,x_k$ that is generated from the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \pi_{k-t}(x_t))$, $t = 0,...,k-1$, where π_i denotes the optimizer of $\Gamma_{\text{p},\alpha}J^{\text{soft}}_{i-1}$. Then, we have $\hat{J}_k(x_0) - J^{\text{soft}}_k(x_0) \le \hat{J}_{k-1}(x_1) - J^{\text{soft}}_{k-1}(x_1) + v_k(x_0) \le \hat{J}_0(x_k) - J^{\text{soft}}_0(x_k) + \sum_{i=1}^{k} v_i(x_{k-i})$. Suppose that $|\hat{J}_0(x_k) - J^{\text{soft}}_0(x_k)| \le e_{\text{v}} l(x_k,0)$ and $|v_i(x_{k-i})| \le e_{\text{v}} l(x_{k-i},0)$, $i = 1,...,k$, where $e_{\text{v}} > 0$, we obtain

$$\hat{J}_k(x_0) - J^{\text{soft}}_k(x_0) \le e_{\text{v}} \sum_{i=0}^{k} l(x_i,0) \le e_{\text{v}} J^{\text{soft}}_k(x_0). \tag{3.30}$$

Similar procedures can be applied to upper bound the value of $J^{\text{soft}}_k(x_0) - \hat{J}_k(x_0)$. From (3.13) and (3.15), we see that compared to letting $\rho_v(x) = 1$, our choice of ρ_v is more likely to lead to a smaller e_{v}, which can contribute to the reduction of ζ, according to (3.30). Moreover, to circumvent the singularity of ρ_v at the origin, we let $\rho_v(x) = 1/(l^2(x,0) + \rho)$, with a small constant $\rho > 0$.

(iii) Condition C2 requires the policy approximation error $\hat{J}_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) - \hat{J}_{\hat{\pi}^{\text{im}}}(x)$ is constrained by the state cost $l(x,0)$ scaled by a constant e_{p} and that the constant e_{p} is small enough. Based on this requirement, we take $\rho_\pi(x) = 1/(l(x,0) + \rho)$ to make e_{p} small.

Remark 3.5.2. Different from the existing stability results on ADP [107], [133], [154], [162], Theorem 3.5.1 considers the effects of state constraints. Besides, condition C1 allows us to analyze the sub-optimality of $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$, which has not been addressed in existing research [107], [133], [154], [162].

3.5.2. SUB-OPTIMALITY ANALYSIS

Based on Theorem 3.5.1, we can compute upper bounds on $J_{\hat{\pi}^{\text{im}}}$ and $J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}$, which are the infinite-horizon costs of $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$, defined in (3.2).

Corollary 3.5.1. Consider Algorithm 3 and the proposed policies $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$. Let Ω be a compact subset of X.

(i) Let the assumptions in (i) of Theorem 3.5.1 hold and $2\zeta\gamma < 1$. Then, for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$, we have the inequality

$$J^{\text{soft}*}(x) \le J_{\hat{\pi}^{\text{im}}}(x) \le \frac{1-\zeta}{1-2\zeta\gamma} J^{\text{soft}}_{k-1}(x). \tag{3.31}$$

(ii) Let the assumptions in (ii) of Theorem 3.5.1 hold and $2\zeta\gamma + e_{\text{p}} < 1$. Then, for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$, we have the inequality

$$J^{\text{soft}*}(x) \le J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) \le \frac{1-\zeta}{1-2\zeta\gamma-e_{\text{p}}} J^{\text{soft}}_{k-1}(x). \tag{3.32}$$

Proof. We note that the first inequality in (3.31) and the first inequality in (3.32) directly

follow from the optimality of $J^{\text{soft}*}$. We derive from (3.28) that

$$\frac{1-2\zeta\gamma}{1-\zeta} l(x,\hat{\pi}^{\text{im}}(x)) \leq J^{\text{soft}}_{k-1}(x) - J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\hat{\pi}^{\text{im}}(x))) \tag{3.33}$$

holds for any $x \in (\mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega))\backslash\{0\}$. Consider the trajectory $x_0, x_1, \ldots$ that is generated by applying $\hat{\pi}^{\text{im}}(x_t)$ to system (3.1) at each time step t, $t = 0,1,\ldots$ Letting $x = x_t$ and summing up both sides of (3.33) from $t = 0$ to $t = \infty$, we get $J_{\hat{\pi}^{\text{im}}}(x_0) \leq \frac{1-\zeta}{1-2\zeta\gamma}(J^{\text{soft}}_{k-1}(x_0) - J^{\text{soft}}_{k-1}(x_\infty))$. The asymptotic stability in (i) of Theorem 3.5.1 indicates that $J^{\text{soft}}_{k-1}(x_\infty) = 0$, so (i) of Corollary 3.5.1 is proved. Similarly, we can upper bound the stage cost when applying $\hat{\pi}^{\text{ex}}_{\text{proj}}$ by $\frac{1-2\zeta\gamma-e_{\text{p}}}{1-\zeta} l(x,\hat{\pi}^{\text{ex}}_{\text{proj}}(x)) \leq J^{\text{soft}}_{k-1}(x) - J^{\text{soft}}_{k-1}(f_{\text{PWA}}(x,\hat{\pi}^{\text{ex}}_{\text{proj}}(x)))$. (ii) of Corollary 3.5.1 will be obtained by summing up the above inequality along the trajectory controlled by $\hat{\pi}^{\text{ex}}_{\text{proj}}$. □

It is almost impossible to get an optimal control policy from (3.19) due to the approximation error. However, (3.31) and (3.32) confirm the intuition that a smaller approximation error of the critic (and the actor) leads to tighter sub-optimality guarantees. Namely, as $\zeta \to 0$, $e_{\text{p}} \to 0$, and $k \to \infty$, we have $J_{\hat{\pi}^{\text{im}}}(x) \to J^{\text{soft}*}(x)$ and $J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) \to J^{\text{soft}*}(x)$ for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1},\Omega) \cap \mathcal{B}(\hat{J}_{k-1},\Omega)$. Moreover, if option (a) of Algorithm 3 is used, $J^{\text{soft}}_{k-1}(x)$ in (3.31) and (3.32) can be replaced by $J^{\text{soft}*}(x)$ because $J^{\text{soft}}_{k-1}(x) \leq J^{\text{soft}*}(x)$, $\forall x \in \mathcal{X}$.

3.5.3. STABILITY AND SAFETY VERIFICATION

As mentioned in the previous subsection, conditions C1 and C2 in Theorem 3.5.1 can only be evaluated statistically. Moreover, the conditions in Theorem 3.5.1 are sufficient conditions for $\hat{J}_{k-1}$ and J^{soft}_{k-1} to be Lyapunov functions, and thus these conditions can be conservative. Besides, sometimes only practical stability can be ensured for nonlinear systems with neural controllers [162]. In this section, we propose an offline verification framework to simultaneously verify the practical stability and safety of the system controlled by the projected policy $\hat{\pi}^{\text{ex}}_{\text{proj}}$ in a deterministic manner, based on MILP. A small adaptation that can be used to verify the asymptotic stability for $\hat{\pi}^{\text{ex}}_{\text{proj}}$ and $\hat{\pi}^{\text{im}}$ is provided at the end of this section.

The proposed verification procedure contains 3 steps. Different from [49], [67], [115] that directly verify asymptotic stability, for practical stability we need to first verify the convergence of the closed-loop system to a neighborhood containing the origin, and then verify the invariance of the neighborhood. These will be formulated as two MILP problems. Finally, to enlarge the inner-estimated region of attraction, which is a sub-level set of $\hat{J}_{k-1}$, the third MILP problem will be formulated.

To verify the stability in any sub-level set $\mathcal{B}_{r_1} = \{x \in \mathcal{X} | \hat{J}_{k-1}(x) \leq r_1,\ r_1 > 0\}$ that is contained in X, we can directly extend the verifier in [49], in which a mixed-integer quadratic program is solved to test a quadratic candidate Lyapunov function for PWA systems with input constraints. In particular, after implementing Algorithm 3 and (3.19), we have the value function $\hat{J}_{k-1}$ and the explicit policy $\hat{\pi}^{\text{ex}}(\cdot,\omega^*)$ at our disposal. Then, we formulate

the following optimization problem:

$$\begin{aligned} a_1^* := \max_{x,u} \; & \hat{J}_{k-1}(f_{\text{PWA}}(x,u)) - \hat{J}_{k-1}(x) + c_1 l(x,0) \\ \text{s.t. } & u = \hat{\pi}^{\text{ex}}_{\text{proj}}(x),\ r_2 \le \hat{J}_{k-1}(x) \le r_1, \end{aligned} \tag{3.34}$$

where c_1 is a small positive parameter, and $r_2 \in (0, r_1)$. If $a_1^* \le 0$, we can conclude that the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is safe in $\mathcal{B}_{r_1}$, and that any trajectories starting in $\mathcal{B}_{r_1}$ will enter $\mathcal{B}_{r_2} = \{x \in \mathcal{X} | \hat{J}_{k-1}(x) \le r_2,\}$ in finite time. The formal result is included in Theorem 3.5.2. If $a_1^* > 0$, we can reduce the values of r_1 and c_1. In this way, the objective function of problem (3.34) and the feasible region of x become smaller, which will make a_1^* smaller.

As all functions in (3.34) are PWA and thus MILP representable, problem (3.34) can be formulated as an MILP problem.

After the trajectories reach $\mathcal{B}_{r_2}$, we need to verify that they will always stay in $\mathcal{B}_{r_2}$, i.e., we need to prove the positive invariance of $\mathcal{B}_{r_2}$. This leads to the second MILP problem:

$$\begin{aligned} a_2^* := \max_{x,u} \; & \hat{J}_{k-1}(f_{\text{PWA}}(x,u)) - c_2 \hat{J}_{k-1}(x) - r_2 + r_2 c_2 \\ \text{s.t. } & u = \hat{\pi}^{\text{ex}}_{\text{proj}}(x),\ 0 \le \hat{J}_{k-1}(x) \le r_2, \end{aligned} \tag{3.35}$$

where $c_2 \in [0,1]$. One can directly take $c_2 = 0$ to minimize a_2^*. Clearly, $a_2^* \le 0$ implies the positive invariance of $\mathcal{B}_{r_2}$, which will be proven in Theorem 3.5.2. If $a_2^* > 0$, similarly we can make r_2 smaller.

However, the stable and safe region $\mathcal{B}_{r_1}$ derived from (3.34) and (3.35) is usually small, as $\mathcal{B}_{r_1}$ is a sub-level set of $\hat{J}_{k-1}$. In particular, if the weights in Q on different states vary greatly, the resulting $\mathcal{B}_{r_1}$ will be rather narrow and much smaller than the real region of attraction. It could also happen that the evolution of the closed-loop system does not make $\hat{J}_{k-1}$ decrease at the beginning, but will drive the state into $\mathcal{B}_{r_1}$ in a finite number of time steps. Besides, in most cases, we are interested in the performance of a policy in a polyhedron (or a UoP), rather than a sub-level set.

Therefore, we further develop the third optimization problem that evaluates the range of trajectories of the closed-loop system in a finite number of time steps for all initial states in a polyhedron X_{in} or a UoP of interest. The problem is

$$\begin{aligned} & \text{Check if } x_t \in X,\ t = 1, \dots, N-1, \forall x_0 \in X_{\text{in}} \text{and if } \hat{J}_{k-1}(x_N) \le r_1, \forall x_0 \in X_{\text{in}} \\ & \text{s.t. } u_t = \hat{\pi}^{\text{ex}}_{\text{proj}}(x_t),\ x_{t+1} = f_{\text{PWA}}(x_t, u_t),\ t = 0, \dots, N-1, \end{aligned} \tag{3.36}$$

where N is the number of time steps and r_1 is such that it makes $a_1^* \le 0$ in (3.34). If (3.36) returns "Yes", we can conclude that for any initial state in X_{in}, the states of the closed-loop system will satisfy the constraints from $t = 0$ to $t = N-1$, and the final state x_N will reach the stable and safe region $\mathcal{B}_{r_1}$ computed in (3.34). Similar to (3.34) and (3.35), (3.36) can be exactly expressed in an MILP form. If X_{in} is a UoP, we also need some additional binary variables to formulate the initial condition $x_0 \in X_{\text{in}}$ (see [99]).

The integration of (3.34), (3.35), and (3.36) constitutes the proposed verification framework, which computes the exact evolution of the closed-loop system. It does not need

any sampling or statistical testing procedure. The effectiveness of the proposed verification framework is stated in the following theorem.

Theorem 3.5.2. Consider the policy $\hat{\pi}^{\text{ex}}_{\text{proj}}$ and the proposed verification framework consisting of (3.34), (3.35), and (3.36). If $\hat{J}_{k-1}(0) = 0$, $a_1^* \leq 0$, $a_2^* \leq 0$, and (3.36) returns "Yes", then the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is safe in X_{in}, and any trajectory starting from X_{in} will approach $\mathcal{B}_{r_1}$ in at most $N + \lceil (r_1 - r_2)\hat{\gamma}/(c_1 r_2) \rceil$ time steps and stay in $\mathcal{B}_{r_1}$ thereafter. Here $\hat{\gamma}$ is a positive constant independent of the initial condition.

3

Proof. If (3.36) returns "Yes", the trajectories of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ with initial condition $x_0 \in X_{\text{in}}$ will be contained in $\mathcal{B}_{r_1}$ after N time steps.

Then, for any initial state $x_0 \in \mathcal{B}_{r_1}$, suppose that the $t_{\text{f}} - 1$-step trajectory $x_0, x_1, \ldots, x_{t_{\text{f}}-1}$ of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is not contained in $\mathcal{B}_{r_2}$. Since $a_1^* \leq 0$, we have

$$\hat{J}_{k-1}(x_{t+1}) - \hat{J}_{k-1}(x_t) \leq -c_1 l(x_t, 0), \ t = 0, \ldots, t_{\text{f}} - 1. \tag{3.37}$$

Summing (3.37) over time yields

$$\hat{J}_{k-1}(x_{t_{\text{f}}}) \leq \hat{J}_{k-1}(x_0) - c_1 \sum_{t=0}^{t_{\text{f}}-1} l(x_t, 0). \tag{3.38}$$

Meanwhile, since both $\hat{J}_{k-1}$ and $l(\cdot, 0)$ are continuous PWA functions on their domains, similarly to (i) of Lemma 3.5.1, there exists a positive and finite constant $\hat{\gamma}$ such that $\hat{J}_{k-1}(x) \leq \hat{\gamma} l(x, 0)$ for all $x \in X$. As a result, (3.38) implies that $\hat{J}_{k-1}(x_{t_{\text{f}}}) \leq r_1 - \frac{c_1 r_2 t_{\text{f}}}{\hat{\gamma}}$. Specifying $t_{\text{f}} = \lceil (r_1 - r_2)\hat{\gamma}/(c_1 r_r) \rceil$, which is finite and does not depend on x_0, we have $\hat{J}_{k-1}(x_{t_{\text{f}}}) \leq r_2$. Combining the above statements, we can conclude that any trajectory of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ starting from X_{in} will reach $\mathcal{B}_{r_2}$ in less than $N + t_{\text{f}}$ time steps. Finally, the positive invariance of $\mathcal{B}_{r_2}$ is straightforward if $a_2^* \leq 0$, since we have $\hat{J}_{k-1}(x) \leq r_2 \Rightarrow \hat{J}_{k-1}(f_{\text{PWA}}(x, \hat{\pi}^{\text{ex}}_{\text{proj}}(x))) \leq r_2$ from (3.35). This completes the proof of Theorem 3.5.2.

□

With the results in Theorems 3.5.1 and 3.5.2, we now make practical suggestions on how to implement Algorithm 3 and the proposed verification framework.

After the explicit policy $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$ is obtained, we first find r_2 that makes $a_2^* \leq 0$. Then, one can use (3.35) to compute the safe and stable region and then use (3.36) to enlarge it. If $a_2^* \leq 0$ but $a_1^* > 0$ whatever r_1 is chosen, one needs to apply (3.36) with r_1 replaced by r_2. The cost is that one may have to choose a large horizon N since r_2 is small. The complexity of (3.36) grows exponentially w.r.t. N. If $a_1^* > 0$ and $a_2^* > 0$, no matter what r_1 and r_2 are chosen, one has to refine the learning process. To this end, Theorem 3.5.1 implies that one can either (i) increase the number of iterations by tightening the stopping condition, i.e., making e_{tole} smaller, or (ii) improve the approximation quality of function approximators and restart Algorithm 3.

Remark 3.5.3. The proposed verification framework generalizes the verification methods in [67], [115], [177], and is an extension of [67], [115], [177] from linear systems to PWA systems and from asymptotic stability to practical stability. Specifically, if we remove (3.35) and (3.36) and fix $r_2 = 0$, the proposed method verifies asymptotic stability, which is stronger than the properties in Theorem 3.5.2, and it is then similar to the method in [67]. For the comparison with [115], we note that to guarantee state constraint satisfaction, [115] needs to find a positively invariant set. In comparison, X_{in} in (3.36) is not necessarily positively invariant. Besides, (3.36) includes the case when X and the searching region X_{in} are UoPs.

In addition, (3.34) can be adjusted to verify the asymptotic stability and safety of $\hat{\pi}^{\text{im}}$. With $r_2 = 0$ and $c_1 l(x,0)$ replaced by $c_1 l(x,u)$, $a_1^* \leq 0$ tells that $\hat{J}_{k-1}$ is a control Lyapunov function, which further validates the asymptotic stability and safety of $\hat{\pi}^{\text{im}}$ in $\mathcal{B}_{r_2}$.

3.5.4. OVERALL DESIGN PROCEDURE

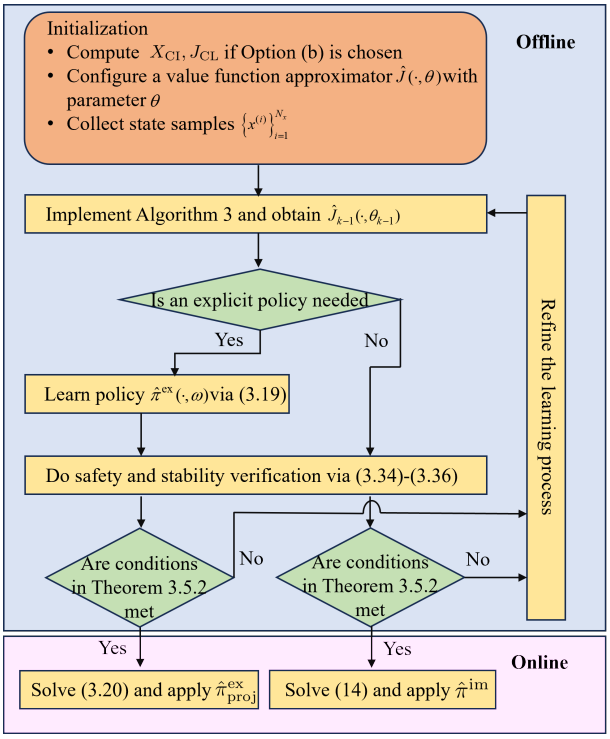


Figure 3.1: The schematic diagram of the design procedure.

We provide a roadmap (see Fig. 3.1) outlining the procedures for implementing the proposed methods. The flowcharts within the blue box depict offline processes, while the flowcharts in the pink box represent online steps. After Algorithm 3 is implemented, two options can be chosen. One can either learn the explicit policy via (3.19) or bypass this step. Then, the proposed verification tool in Section 3.5.3 is used to assess the perfor-

mance of the policy. If the policy falls short of expectations, it becomes necessary to refine the learning process and restart Algorithm 3. After verifying the safety and stability of the policy $\hat{\pi}^{\text{ex}}_{\text{proj}}$ or $\hat{\pi}^{\text{im}}$, in the online phase we solve (3.20) or (3.16) to compute the control input $\hat{\pi}^{\text{ex}}_{\text{proj}}(x_t)$ or $\hat{\pi}^{\text{im}}(x_t)$ when the state measure x_t is received.

The control framework involves four types of parameters: the hyper-parameters of the learning models for the value function and the policy, the penalty weight p, the parameter e_{tole} of the tolerance function ϵ determining the stopping criterion of Algorithm 3, and the parameters r_1, r_2, and N of the verification framework. Guidelines for setting hyper-parameters of the learning models can be found in machine learning literature, such as [86]. The principle for tuning r_1, r_2, and N has been provided after Theorem 3.5.2. To refine the learning process, one can adjust the first three kinds of parameters by the following steps: (i) improving the approximation quality of the learning models by tuning the hyper-parameters or amplifying samples; (ii) raising p; (iii) increasing the number of iterations by tightening the stopping condition (e_{tole} smaller).

3.6. CASE STUDY

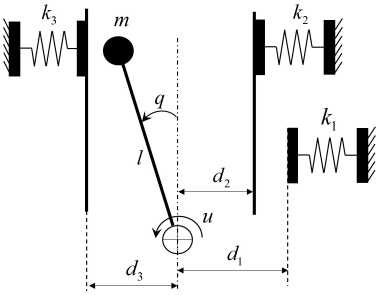


Figure 3.2: An inverted pendulum with elastic walls.

We validate the proposed methods in two examples. We illustrate the convergence of different options of the proposed ADP method, the effectiveness of the verification method, as well as the online computational advantage of the proposed controllers.

3.6.1. EXAMPLE 1: INVERTED PENDULUM WITH ELASTIC WALLS

We consider a physical system in which an active inverted pendulum is placed between elastic walls to maintain the vertical position (see Fig. 3.2). By linearizing the dynamics around the vertical configuration $q = \dot{q} = 0$ and discretizing the system with a sampling time 0.05 s, we obtain a discrete-time PWA model with 4 modes, where the system matrices are given by

$$A_i = \begin{bmatrix} 1 & 0.05 \\ \alpha_i & 1 \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix}, f_i = \begin{bmatrix} 0 \\ \beta_i \end{bmatrix}, i = 1,2,3,4,$$

with $\alpha_1 = -29.5$, $\alpha_2 = -14.5$, $\alpha_3 = 0.5$, $\alpha_4 = -24.5$, $\beta_1 = -3.3$, $\beta_2 = -1.5$, $\beta_3 = 0$ and $\beta_4 = 2.5$, and where the partition $\{\mathcal{C}_i\}_{i=1}^4$ is given by $\mathcal{C}_1 = \{(x,u) \mid [1\ \ 0]x \leq -0.12\}$, $\mathcal{C}_2 = \{(x,u) \mid$

$-0.12 \le [1\ \ 0]x \le -0.1\}$, $\mathcal{C}_3 = \{(x,u) \mid -0.1 \le [1\ \ 0]x \le 0.1\}$ and $\mathcal{C}_4 = \{(x,u) \mid [1\ \ 0]x \ge 0.1\}$. The detailed description of the system is given in [99].

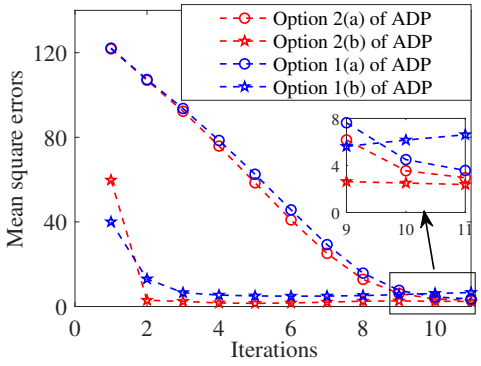
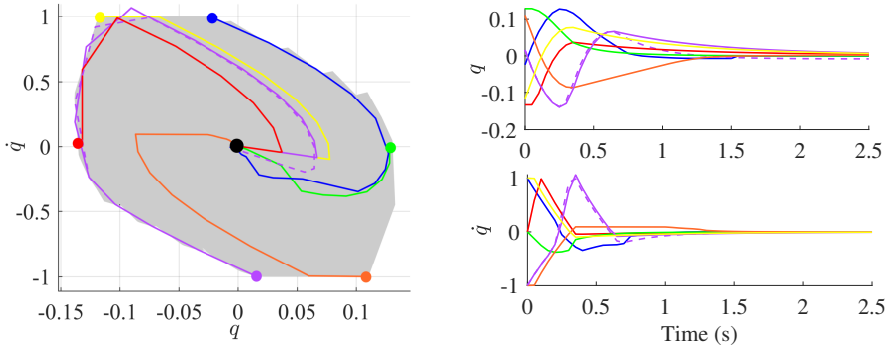


Figure 3.3: The learning process of the ADP algorithm with different options.



(a) Closed-loop trajectories of the inverted pendulum. The shaded region is $\bar{X}$.

(b) Time-domain system responses.

Figure 3.4: Simulation results under linear state constraints.

Due to safety considerations and the limited capability of the torque motor, the inverted pendulum system is supposed to satisfy the constraints $[-0.15\ \ -1]^T \le x \le [0.15\ \ 1]^T$ and $-4 \le u \le 4$. The constrained infinite-horizon control problem is constructed with $l(x,u) = ||\text{diag}([20\ \ 1])x||_\infty + ||u||_\infty$. The overall control objective is to solve the infinite-horizon optimal control problem given in (1). The offline procedure for computing the proposed control policies $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}$ includes solving several optimization problems. In particular, implementing Algorithm 3 requires solving the MILP problem (3.14) and the nonlinear regression problem (3.15). Computing $\hat{\pi}^{\text{ex}}$ requires solving the policy optimization problem (3.19). Besides, the verification step contains solving three MILP problems (3.34)-(3.36).

A 61 × 61 state data grid is constructed to cover the region $\{x|[-0.17\ \ -1.2]^T \le x \le [0.17\ \ 1.2]^T\}$. These training points are exploited to train the actor and the critic. Both of

them are ReLU NNs that have two hidden layers of widths 8. Starting from a zero value function and after 10 iterations, the closed-loop behavior of the system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is illustrated in Fig. 3.4. Fig. 3.4(a) plots the trajectories of the closed-loop system (controlled by $\hat{\pi}^{\text{ex}}_{\text{proj}}$) starting from some vertices of $\bar{X}$ in the state space, while Fig. 3.4(b) displays the time-domain responses corresponding to the state-space trajectories of Fig. 3.4(a). Although states starting from these vertices are in general the most difficult to regulate to the origin, they converge rapidly under $\hat{\pi}^{\text{ex}}_{\text{proj}}$. Meanwhile, we should mention that the state constraints could be slightly violated: the trajectory depicted by the purple curves violates the constraints by about 5 percent. To avoid this, one can tighten the state constraints and restart the ADP algorithm. The dashed purple curve describes the trajectory starting from the purple vertex with 10 percent constraint tightening[2]. We can observe that constraint violation is avoided.

To carry out option (b) of Algorithm 3, we consider the linear subsystem $(A_3,\ B_3)$ and readily get a linear stabilizing feedback law $u_0 = [-40\ \ -10]x$, which is used to initialize the value function of option (b). The optimal value function J^* is computed by the MPT3 toolbox [105] in 9.6 hours. The learning processes of different versions of Algorithm 3 are compared in Fig. 3.3, in which the mean square errors between $\hat{J}_{k-1}\left(x^{(i)},\theta_k\right)$ and $J^*(x^{(i)})$ are depicted. One can see that the value function approximation in option (b) has a much faster convergence rate than that in option (a), and option 2 results in lower approximation errors than option 1. The accelerated convergence rate in option (b) is mainly attributed to its initial value function approximation $\hat{J}_0$ being considerably closer to $J^{\text{soft}*}$ compared to option (a). The ultimately reduced approximation errors in option 2 likely stem from a more densely sampled state space.

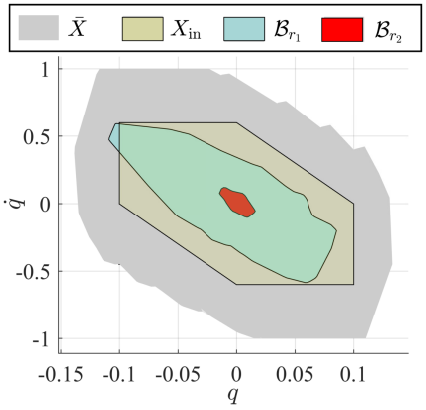


Figure 3.5: Safe and stable regions verified by the verification framework.

The conditions for the stability and safety of the policy $\hat{\pi}^{\text{ex}}_{\text{proj}}$ are verified in Fig. 3.5. Fig. 3.5 depicts the safe and stable regions that are analyzed by the proposed verification

[2] We leverage a backtracking strategy to incrementally increase the constraint tightening factors until the safety performance is successfully verified by our proposed verification method.

framework. The input constraints are always satisfied in the simulation because the optimization problems (3.16) and (3.20) contain the input constraints as hard constraints. The blue region represents $\mathcal{B}_{r_1}$, which is computed by (3.34) with $r_1 = 18$, $c_1 = 0.1$, $r_2 = 3$. We note that for some states in $\mathcal{B}_{r_2}$, which is colored red, the objective function in (3.34) becomes positive, so the verification method in [67] fails. However, (3.35) outputs a negative a_2^* with $c_2 = 0.1$, which means that any trajectory of the closed-loop system starting from $\mathcal{B}_{r_1}$ will reach in the neighborhood $\mathcal{B}_{r_2}$ containing the origin in finite number of time steps. Furthermore, the safe and stable region $\mathcal{B}_{r_1}$ is enlarged to X_{in} (the yellow region) by (3.36) with $N = 3$. However, we observe that the verified safe and stable polytope X_{in} may be conservative compared to our trajectory simulation in Fig. 3.4. Suppose that $X_{\text{in}} := \{x \in \mathbb{R}^{n_x} | E_{X_{\text{in}}} x \leq g_{X_{\text{in}}}\}$. The conservatism is primarily attributed to the naive choice of $E_{X_{\text{in}}}$. Practical strategies to mitigate this conservatism involve using a UoP X_{in} and extending the horizon N. These two methods, on the other hand, will increase the complexity of the MILP problem (3.36).

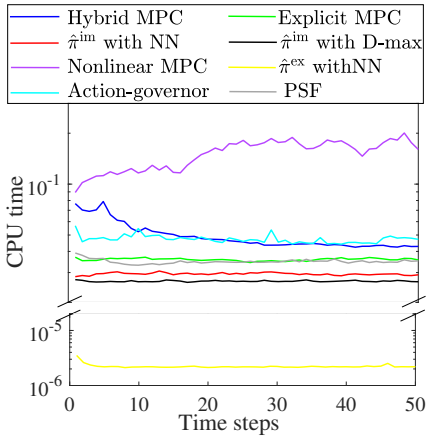
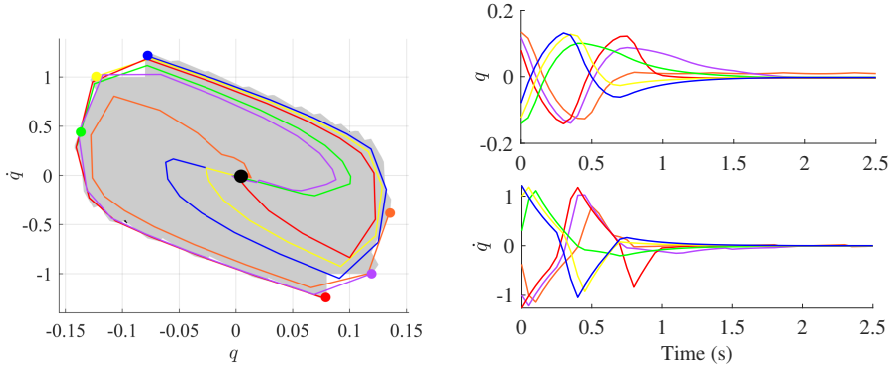


Figure 3.6: Statistical analysis of different methods regarding CPU times. D-max means the difference of two max-affine functions.

We compare the CPU time of running the ADP-based controllers, hybrid MPC, explicit MPC computed by the MPT3 toolbox, nonlinear MPC that uses the nonlinear model, action-governor RL in [130], and the predictive safety filter (PSF) in [198]. The horizons of MPC and the PSF are taken as 8. The implementation of hybrid MPC and the PSF requires to solve an MILP problem. In comparison, for $\hat{\pi}^{\text{im}}$ with the value function approximator chosen as the ReLU NN (or the difference of two max-affine functions that has 15 and 5 terms in the first and second max blocks, respectively), one should solve a smaller MILP problem than the ones of hybrid MPC and the PSF. For statistical analysis, we randomly select 100 initial states and run the system for 50 time steps. According to the results in Fig. 3.6, the computation of $\hat{\pi}^{\text{ex}}_{\text{proj}}$ requires the least amount of CPU time, which is around 2.2×10^{-6} s per time step. Besides, $\hat{\pi}^{\text{im}}$ performs better than hybrid MPC, explicit MPC, action-governor RL, and PSF, regarding online computation time, both when using the ReLU NN (about 0.028 s) or the difference of two max-affine func-



(a) Closed-loop trajectories of the inverted pendulum with UoP state constraints.

(b) Time-domain system responses under UoP state constraints.

Figure 3.7: Simulation results under UoP state constraints.

tions (about 0.026 s) for value function approximation.

We further consider the case when the state constraints are a UoP, namely, $x \in X^{(1)} \cup X^{(2)}$ with $X^{(1)} = \{x | [-0.15 \quad -1]^T \leq x \leq [0.15 \quad 1]^T\}$ and $X^{(2)} = \{x | [-0.08 \quad -1.5]^T \leq x \leq [0.08 \quad 1.5]^T\}$. This means a higher angular velocity is allowed when the pendulum is close to its vertical configuration. After implementing option 2(a) of Algorithm 3 in 10 iterations, $\bar{X}$ is depicted in Fig. 3.7(a). The trajectories of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}$ starting from some vertices of $\bar{X}$ are plotted in Fig. 3.7, from which one can find that the proposed scheme is still valid even when the state constraints are a UoP.

3.6.2. EXAMPLE 2: CENTRALIZED ADAPTIVE CRUISE CONTROL

We consider an adaptive cruise control problem in which we drive three vehicles (followers) to follow a leading vehicle (leader) in a highway environment. The control objective is to keep a desired distance $d = 20$ m between each two adjacent vehicles.

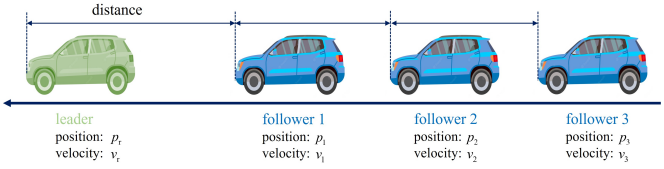


Figure 3.8: Adaptive cruise control set up.

Model: The differential equation for the velocity $v_i(t)$, $i = 1,2,3$ of each following vehicle is $m\dot{v}_i(t) + cv_i^2(t) + \mu mg = bf_i(t)$, where $bf_i(t)$ is the input traction/brake force, and where $c = 0.5$ kg/m, $m = 800$ kg, $\mu = 0.01$, $b = 3700$ N, $g = 9.8$ m/s^2. The maximum allowable velocity for followers is $v_{\max} = 35$ m/s.

Following [62], a least-squares PWA approximation of the nonlinear damping force $V(v) = cv^2$ results in the continuous-time PWA dynamics with 2 modes for each vehicle

$$m\dot{v}_i(t) + c_j v_i(t) + a_j = bf_i(t), \text{ if } v_i(t) \in [\alpha_{j-1}, \alpha_j),\ j = 1,2, \tag{3.39}$$

where $\alpha_0 = 0$, $\alpha_1 = v_{\max}/2$, $\alpha_2 = v_{\max}$, $c_1 = 3cv_{\max}/8$, $c_2 = 13cv_{\max}/8$, $a_1 = \mu mg$, and $a_2 = -5cv_{\max}^2/8 + \mu mg$. The leader is assumed to have a constant velocity $v_r = 20$ m/s. By defining a new state $x = [p_1 - p_r - d \quad v_1 - v_r \quad p_2 - p_1 - d \quad v_2 - v_r \quad p_3 - p_2 - d \quad v_3 - v_r]^T$, where p_1, p_2, p_3 and p_r are the positions of the three followers and the leader, respectively, the control goal is then making x zero. The dynamics of x thereby have 2^3 different modes. Noticing that when x is zero, $v_i \in [\alpha_1, \alpha_2), i = 1,2,3$, so we introduce an input transformation $u = [f_1 - \frac{v_r c_2 + a_2}{b} \quad f_2 - \frac{v_r c_2 + a_2}{b} \quad f_3 - \frac{v_r c_2 + a_2}{b}]^T$ to make the origin of x-u space the equilibrium of the state equations for x. By discretizing the differential equations for x with the sampling time 1 s, we get a discrete-time PWA model with 6 states, 3 inputs, and 8 modes.

As for constraints, we consider limitations on the distance between each two adjacent vehicles, the velocity, and the traction/brake force input. In particular, we require that $10\text{ m} \le |p_i - p_{i-1}| \le 30\text{ m}$, $5\text{ m/s} \le v_i \le 35\text{ m/s}$, and $|f_i| \le 1$ N, where $i = 1,2,3$ and $p_0 = p_r$. All of these constraints can be readily converted into polyhedral constraints on x and u.

The stage cost is chosen as $l(x,u) = \left\|\text{diag}\left(\begin{bmatrix} 1 & 0.5 & 1 & 0.5 & 1 & 0.5 \end{bmatrix}\right)x\right\|_\infty + \left\|\text{diag}\left(\begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}\right)u\right\|_\infty$.

Setups: We uniformly randomly sample 25000 state points from the state space $\{x \in \mathbb{R}^6 \mid [-12 \ -18 \ -12 \ -18 \ -12 \ -18]^T \le x \le [12 \ 18 \ 12 \ 18 \ 12 \ 18]^T\}$. We compare the performance of different controllers, including the proposed ADP controllers with different parameter settings, hybrid MPC with different prediction horizons, and the PSF-based RL [198]. The performance metrics include the average CPU time per time step, the cumulative stage cost over 15 time steps, and the percentage that the system trajectory satisfies all constraints. To test the performance of controllers, we generate 100 initial state scenarios that make the hybrid MPC with horizon 4 recursively feasible and stabilizing. Since controllable sets are difficult to compute in this example, we start from a zero value function and adopt option 2 of Algorithm 3 to get $\hat{J}_{k-1}$. All PWA function approximators used in the proposed method and the safety-filter-based RL are ReLU NNs with 2 hidden layers.

Results: Table 3.1 shows the results. In Table 3.1, N represents the prediction horizon chosen in hybrid MPC, M refers to the number of units in each hidden layer, and the safety rate is defined as the number of initial states leading to a safe closed-loop trajectory divided by the total number (100) of initial states. As expected, MPC provides the best performance regarding the total cost and safety rate, but it needs too much CPU time for computation. Compared with MPC, $\hat{\pi}^{\text{ex}}_{\text{proj}}$ and the PSF-RL in [198] can reduce the online computation time, especially for $\hat{\pi}^{\text{ex}}_{\text{proj}}$. With the increase of the number of units, the performance of $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$ regarding the total cost becomes better and close to that of MPC. As both $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$ have acceptable performance regarding the total cost and safety rate, using $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is preferable due to its significant reduction of CPU time. Be-

sides, Table 1 also indicates that with the same number of units, the PSF-RL in [198] can be computed faster than $\hat{\pi}^{\text{im}}$, but the performance of $\hat{\pi}^{\text{im}}$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}$ is superior to that of the RL method in [198] in terms of both cumulative cost and constraint violations.

Table 3.1: Performance of different controllers

Methods	CPU time (mean/max)	Total cost	Safety rate
MPC with $N = 4$	0.1592 / 0.4593	21.7664	100%
MPC with $N = 5$	0.2850 / 0.7703	21.6653	100%
MPC with $N = 6$	0.4630 / 1.4619	21.5997	100%
$\hat{\pi}^{\text{im}}$ with $M = 15$	0.0605 / 0.1036	23.0097	98%
$\hat{\pi}^{\text{im}}$ with $M = 20$	0.0916 / 0.1621	22.6613	100%
$\hat{\pi}^{\text{im}}$ with $M = 25$	0.1372 / 0.2991	22.3042	100%
$\hat{\pi}^{\text{im}}$ with $M = 30$	0.5487 / 1.3904	22.0716	100%
$\hat{\pi}^{\text{ex}}_{\text{proj}}$ with $M = 15$	0.0008 / 0.0028	23.4043	99%
$\hat{\pi}^{\text{ex}}_{\text{proj}}$ with $M = 20$	0.0007 / 0.0027	22.9568	99%
$\hat{\pi}^{\text{ex}}_{\text{proj}}$ with $M = 25$	0.0007 / 0.0027	22.7022	99%
$\hat{\pi}^{\text{ex}}_{\text{proj}}$ with $M = 30$	0.0008 / 0.0030	22.7562	99%
PSF-RL in [198] with $M = 30, N = 5$	0.0809 / 0.3570	29.7561	85%

3.7. CONCLUSIONS AND FUTURE WORK

We have proposed an ADP control scheme to deal with infinite-horizon optimal control of PWA systems subject to linear and union-of-polyhedra (UoP) constraints, based on MILP. With carefully designed PWA penalty functions, the probably non-convex UoP constraints during the learning process are removed while the PWA properties of the value functions are maintained. We have formally analyzed the PWA properties and continuity of the value function, as well as the closed-loop stability and safety under the approximation errors. We have also designed an offline verification tool to make the proposed method reliable. Simulation results show that the ADP-based policies are near-optimal and require much less online computational effort than conventional hybrid MPC.

The limitations of the proposed ADP method are threefold. The performance of the policies depends heavily on the approximation accuracy of value functions. Secondly, our method does not scale well to high-dimensional problems due to the dramatic growth of sampling complexity. Additionally, our method is limited to the cases involving UoP state constraints and linear input constraints. Therefore, topics for future work include eliminating the reliance on value function approximation, exploring more efficient sampling strategies, and considering general convex multi-time-step constraints.

4

LEARNING-BASED CONTROL OF CONSTRAINED PIECEWISE AFFINE SYSTEMS USING OPTIMIZATION-FREE SAFETY FILTERS

Control of piecewise affine (PWA) systems under complex constraints faces challenges in guaranteeing both safety and online computational efficiency. Learning-based methods can rapidly generate control signals with good performance, but rarely provide safety guarantees. A safety filter is a modular method to improve safety for any controller. When applied to PWA systems, a traditional safety filter usually needs to solve a mixed-integer convex program, which reduces the computational benefit of learning-based controllers. We propose a novel optimization-free safety filter designed to handle state constraints that involve a combination of polyhedra and ellipsoids. The proposed safety filter only utilizes algebraic and min-max operations to determine safe control inputs. This offers a notable advantage compared with traditional safety filters by allowing for significantly more efficient computation of control signals. The proposed safety filter can be integrated into various function approximators, such as neural networks, enabling safe learning throughout the learning process. Simulation results on a bicycle model with PWA approximation validate the proposed method regarding constraint satisfaction, CPU time, and the preservation of sub-optimality.

This chapter is based on the paper [103].

4.1. INTRODUCTION

Background. There is a growing interest in controlling piecewise affine (PWA) systems due to their ability to represent hybrid models and to approximate nonlinear dynamics [35]. Control of PWA systems with constraints faces challenges in balancing multiple performance measures, such as safety, optimality, and online computational efficiency. In particular, hybrid model predictive control (MPC) can be applied to PWA systems, with mature results on sub-optimality and safety guarantees [124]. With a PWA prediction model, the MPC problem can be converted into a mixed-integer convex programming (MICP) problem. However, this comes at a large online computational price to solve the MICP problem, especially when there are complex state constraints such as a combination of polyhedral and ellipsoidal constraints. Explicit MPC, which finds the analytical solution of the hybrid MPC problem, also suffers from computational complexity issues [141]. In contrast, other methods that use a simple control structure, such as adaptive control [134] and piecewise linear feedback control [124], can be implemented rapidly but lack formal performance guarantees.

In recent years, various learning-based methods have been employed to design controllers. There are two main classes of learning-based methods: learning models and learning control policy parameters. In this chapter, we assume the PWA model is given and thus focus on the second class of methods. A common issue of learning-based controllers under constraints is that they can lead to infeasible solutions. To deal with this issue, one can add a penalty to the learning objective [99] or design a safety filter [100], [130], [207]. The penalty method cannot always provide strict constraint satisfaction. In comparison, safety filters usually involve a constrained optimization problem to determine a safe control input. The main constraint of the problem is that the successor state should be within a controlled invariant set. Safety filters can be applied to any learning-based controllers, but can introduce significant computational complexity. In this chapter, our primary emphasis is on the safety filter approach, with a specific focus on mitigating its computational burden.

Related work. Simplifying hybrid MPC: To reduce the computational burden of hybrid MPC, research has been conducted to simplify or approximate the solution of the MICP problem. [142] considers the warm start of MPC. [144] focuses on predicting the discrete solution of the MICP problem by machine learning and solving convex programs online. These methods provide safety and optimality certificates, but still require some online convex or mixed-integer optimization. The computational issue is thus not completely addressed.

Safety filters for PWA systems: For PWA systems encompassing linear or PWA constraints, a less conservative controlled invariant set is usually a union of polyhedra, making the optimization problem in the safety filter an MICP problem [130]. This means that the process of imposing safety comes at the expense of high computation times. Some papers have investigated how to enforce constraints without solving any optimization problems. For linear systems with linear constraints, leveraging their vertex representation can help to produce safe control actions [215]. [213] finds the explicit solution of an RL safety filter comprised of a quadratic program. In [194], an output-constrained neural

network (NN) structure is proposed to approximate the solution of convex optimization problems. The optimization-free methods in [194], [213], [215] require the convexity of both the model and constraints, and thus are not applicable to PWA systems.

Contributions. In this chapter, we consider PWA systems with complex constraints consisting of a union of polyhedra and ellipsoids. We aim to impose safety for any reference controller with a low online computational price. We design a safety filter that computes the safe control input closest to the reference input along a predetermined direction. We obtain the analytical expression of the mapping from the reference input to the safe input. The computation of control inputs is thus based on some basic algebraic, min, and max operations. As a significant benefit, this usually in practice enables a much more efficient computation of control efforts compared to classical safety filters. The proposed safety filter requires a precomputed controlled invariant set consisting of a union of polyhedra and ellipsoids. To achieve this purpose, we transform the invariant set construction problem into a binary classification problem and propose a PWA and piecewise quadratic (PWQ) classification approach that can efficiently estimate a controlled invariant set with low conservativeness.

4.2. PRELIMINARIES AND PROBLEM FORMULATION

Notations. The sets of non-negative integers, positive integers, non-negative integers less than or equal to a, and positive integers less than or equal to a are denoted by $\mathbb{N}$, $\mathbb{N}^+$, $\mathbb{N}_a$, and $\mathbb{N}_a^+$, respectively. The boundary of the set S is denoted by ∂S. The operator $\min\{\cdot,\cdot\}$ returns the smaller of two scalars, $\min(\cdot)$ outputs the minimum element of a vector, $\text{step}(\cdot)$ is a unit step function.

4.2.1. PROBLEM FORMULATION

We consider a deterministic discrete-time PWA system

$$x_{t+1} = f_{\text{PWA}}(x_t, u_t) = A_i x_t + B_i u_t + f_i, \text{ if } x_t \in \mathcal{C}_i, \tag{4.1}$$

where $x_t \in \mathbb{R}^{n_x}$ and $u_t \in \mathbb{R}^{n_u}$ are the state and the input at time step t, $f_{\text{PWA}}(\cdot,\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ satisfies $f_{\text{PWA}}(0,0) = 0$, $\{\mathcal{C}_i\}_{i=1}^s$ is a strict polyhedral partition [35, Definition 4.5] of the state space, s is the number of polyhedral regions, and the matrices A_i, B_i and the vectors f_i define the affine dynamics in the regions $\mathcal{C}_i$. The system is required to satisfy the state and input constraints $x_t \in X \subseteq \mathbb{R}^{n_x}$, $u_t \in U(x_t) \subseteq \mathbb{R}^{n_u}$ for all $t \in \mathbb{N}$. We assume that $X = \cup_{i=1}^{m_x} X_i$ where each X_i is a polyhedron or ellipsoid, and m_x is the number of these sets. We also assume that $U(x)$ is a projection of a polyhedron or ellipsoid in $\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ onto $\mathbb{R}^{n_u}$. Note that $U(\cdot)$ accommodates the case when there is a state-dependent input constraint. Besides, we assume that f_{PWA} is known and there is no disturbance.

We will explore imposing safety on PWA systems in a computationally cheap manner, either by modifying a given unsafe controller or by directly learning a safe controller.

Main problem P1: Given a PWA system (4.1) and a reference controller $\pi_{\text{r}}(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$, adapt inputs $\pi_{\text{r}}(x)$ to $\pi^*(x)$ *online* in a computationally efficient way such that the

trajectory of the closed-loop system with π^* satisfies the constraints $x_t \in X$ and $\pi^*(x_t) \in U(x_t)$, $\forall\, t \in \mathbb{N}$.

Extended problem P2: Given a PWA system (4.1), synthesize *offline* a controller $\pi_\theta(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ with the parameter θ such that the trajectory of the closed-loop system with π_θ satisfies the constraints $\forall\, t \in \mathbb{N}$, and simultaneously minimizes a predefined cost $J(\theta)$.

The reference controller is a baseline control strategy intended to optimize performance in certain aspects, but does not provide safety guarantees. This includes, but is not limited to, controllers based on reinforcement learning.

4.2.2. SAFETY FILTER

4

Without considering the online computational complexity, problem P1 can be solved by constructing a safety filter [100], [130], [207]. In particular, suppose that a controlled invariant set $S \subseteq X$ [35, Definition 10.9] is available. In the online situation, when receiving $\pi_\mathrm{r}(x)$ and x, the safety filter solves the constrained optimization problem

$$\pi_\mathrm{p}(x) = \arg\min_{u \in U(x)} \|u - \pi_\mathrm{r}(x)\|_2, \text{s.t. } f_\mathrm{PWA}(x,u) \in S, \tag{4.2}$$

and applies $\pi_\mathrm{p}(x)$ to the system. The problem (4.2) thus implicitly determines the projected policy $\pi_\mathrm{p}(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$. The invariance of S guarantees the recursive feasibility of (4.2) for any $x_0 \in S$, and the relation $S \subseteq X$ ensures the safety of the closed-loop system with π_p.

Because of (4.2), π_p is usually different from π_r, and some other good performances of π_r could be sacrificed if π_p is applied. To reduce the conservatism of π_p, we expect to use a large S in (4.2), ideally the maximal controlled invariant set. However, for PWA systems, especially those with complex state constraints such as $x \in X$ considered in this chapter, a large S is usually a union of many polyhedra and ellipsoids, which has a very complex representation. As a result, *globally* solving (4.2) needs a mixed-integer encoding of the constraint $f_\mathrm{PWA}(x,u) \in S$. Therefore, the online computational issue is not completely solved by (4.2). To this end, we under-approximate the solution of (4.2) by finding an explicit map from π_r to a suboptimal solution of (4.2), without solving any mixed-integer program. This method will determine the safe control inputs through algebraic operations and thus solve the computational issue.

To solve problem P2, existing literature [207] has made (4.2) an optimization layer when parameterizing the policy, and let (4.2) be included in the training loop of π_θ. In particular, when training θ, this optimization layer is considered in the backward pass, such that the cost $J(\theta)$ is minimized for the projected policy. There are two advantages brought by this idea. First, safety can be ensured even during the learning process. Besides, some good performance of π_θ can be preserved. However, when performing the backward pass, it is difficult to compute the policy gradient $\nabla_\theta J$. To deal with this issue, our method for solving P1 allows for a closed-form representation of π_θ and thus makes π_θ end-to-end trainable. Consequently, problem P2 will be solved.

4.3. OPTIMIZATION-FREE SAFETY FILTER

Our main focus is on problem P1. We propose a safety filter that finds an analytical solution to a policy optimization problem. This solution can be directly applied to solve P2 by integrating the filter into the learning process of π_θ.

We need a stronger assumption than the availability of the controlled invariant set S.

Assumption 4.3.1. A controlled invariant set $S \subseteq X$ and a safe policy $\pi_s(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ on S are available. In other words, S is positive invariant [35, Definition 10.7] for the autonomous system $x_{t+1} = f_{\text{PWA}}(x_t, \pi_s(x_t))$ under the constraints $x \in X$ and $\pi_s(x) \in U(x)$.

Assumption 4.3.1 is not restrictive, because for many methods of synthesizing S such as the LMI-based method [124], polytopic trees [175], and learning control barrier functions [69], a safe policy π_s on S is obtained as a by-product. We assume that $S = \cup_{i=1}^{w} S_i$, where each S_i is a polyhedron or an ellipsoid, and $w \in \mathbb{N}^+$. For each polyhedral S_i, it can have the representation $S_i = \{x \in \mathbb{R}^{n_x} \mid H_i x \leq h_i\}$. For each ellipsoidal S_i, it has the representation $S_i = \{x \in \mathbb{R}^{n_x} \mid (x - e_i)^T E_i (x - e_i) \leq 1\}$. Here, $h_i \in \mathbb{R}^{n_{h_i}}$, $H_i \in \mathbb{R}^{n_{h_i} \times n_x}$, $E_i \in \mathbb{R}^{n_x \times n_x} \succ 0$, and $e_i \in \mathbb{R}^{n_x}$.

Inspired by [194], our method consists of a direction determination followed by a line search. In the online phase when the reference control signal $\pi_r(x)$ is received, we find a safe control input $u \in U(x)$ along the line segment with the endpoints $\pi_s(x)$ and $\pi_r(x)$, such that $f_{\text{PWA}}(x, u) \in S$ and the L_2 distance between u and $\pi_r(x)$ is minimized. In other words, we consider the optimizer of the problem

$$\lambda^*(x) := \max_{\lambda \in [0,1]} \lambda \tag{4.3a}$$

$$\text{s.t. } u = \pi_s(x) + \lambda(\pi_r(x) - \pi_s(x)) \tag{4.3b}$$

$$u \in U(x),\ f_{\text{PWA}}(x, u) \in S, \tag{4.3c}$$

and apply the input

$$\pi^*(x) = \pi_s(x) + \lambda^*(x)(\pi_r(x) - \pi_s(x)) \tag{4.4}$$

to the system.

The reasons why we consider polyhedral and ellipsoidal invariant sets are that (i) most existing methods [124], [130], [175] of synthesizing S for linear or PWA systems result in these types of sets, and that (ii) considering these types allows us to compute the optimizer λ^* in a closed form. The methods of synthesizing S will be discussed in the next section.

Let $R(\cdot) : \mathbb{R}^{n_x} \to \mathbb{N}_s^+$ represent the function that maps x to the index i of $\mathcal{C}_i$ such that $x \in \mathcal{C}$. Since $\{\mathcal{C}_i\}_{i=1}^{s}$ is a strict polyhedral partition of the state space, $R(x)$ is well-defined. The constraint $f_{\text{PWA}}(x, u) \in S$ in (4.3) is equivalent to $\exists i \in \mathbb{N}_w^+$ s.t. $A_{R(x)}x + B_{R(x)}u + f_{R(x)} \in S_i$. After substituting the expression (4.3b) of u into (4.3c), constraints (4.3b)-(4.3c) are equivalent to $\lambda \in \Pi_u(x)$ and $\lambda \in \cup_{i \in \mathbb{N}_w^+} \Pi_i(x)$, where $\Pi_u(x) = \{\lambda \in \mathbb{R} \mid \pi_s(x) + \lambda(\pi_r(x) - \pi_s(x)) \in U(x)\}$, and $\Pi_i(x) = \{\lambda \in \mathbb{R} \mid u = \pi_s(x) + \lambda(\pi_r(x) - \pi_s(x)),\ A_{R(x)}x + B_{R(x)}u + f_{R(x)} \in S_i\}$. Each Π_i represents the set of λ making $f_{\text{PWA}}(x, u) \in S$, while the successor state $f_{\text{PWA}}(x, u)$ is restrained in S_i.

If U is polyhedral (ellipsoidal), $\lambda \in \Pi_u(x)$ is a linear (quadratic) constraint on λ. As a result, problem (4.3) can be expressed as

$$\lambda^*(x) = \max_{\lambda} \lambda \tag{4.5a}$$
$$\text{s.t. } a\lambda^2 + b\lambda \le c \tag{4.5b}$$
$$\exists i \in \mathbb{N}_w^+,\ a^{(i)}\lambda^2 + b^{(i)}\lambda \le c^{(i)}, \tag{4.5c}$$

where all the coefficients depend on x, and (4.5b) and (4.5c) are element-wise inequalities. If $U(x)$ is polyhedral, $a = 0$. If S_i is polyhedral, $a^{(i)} = 0$. The coefficients a, b and c are vectors. If S_i is ellipsoidal, $a^{(i)}$, $b^{(i)}$, and $c^{(i)}$ are scalars; otherwise, they are usually vectors. The constraint $\lambda \in [0,1]$ is included in (4.5b) with the form of $[1\ -1]^T\lambda \le [1\ 0]^T$. Due to the positive definiteness of E_i, a and $a^{(i)}$ are non-negative. For any x, the value of these coefficients can be easily obtained by substituting (4.3b) and the affine dynamics into the inequalities of $U(x)$ and S_i.

The following proposition characterizes the optimizer λ^*. A detailed proof is given in the appendix.

Proposition 4.3.1. Consider problem (4.5), the optimizer λ^* has the following representation:

$$\lambda^*(x) = \min\left\{\min(\lambda^{(0)}),\ \max_{i\in\mathbb{N}_w^+}\{\min(\lambda^{(i)})\}\right\}, \tag{4.6}$$

with

$$\lambda_j^{(0)} = \begin{cases} c_j/b_j & \text{if} \quad a_j = 0,\ b_j > 0; \\ 1 & \text{if} \quad a_j = 0,\ b_j \le 0; \\ \frac{-b_j + \sqrt{b_j^2 + 4a_j c_j}}{2a_j} & \text{if} \quad a_j > 0, \end{cases} \tag{4.7}$$

$$\lambda_j^{(i)} = \begin{cases} c_j^{(i)}/b_j^{(i)} & \text{if } a_j^{(i)} = 0,\ b_j^{(i)} > 0; \\ \text{step}(c_j^{(i)}) & \text{if } a_j^{(i)} = 0,\ b_j^{(i)} = 0; \\ \text{step}\left(c_j^{(i)} - b_j^{(i)} \min\left\{\min(\lambda^{(0)}),\ \min\limits_{\lambda_k^{(i)}\in(0,1)} \lambda_k^{(i)}\right\}\right) & \text{if } a_j^{(i)} = 0,\ b_j^{(i)} < 0; \\ 0 & \text{if } a_j^{(i)} > 0, \delta_j^{(i)} := {b_j^{(i)}}^2 + 4a_j^{(i)}c_j^{(i)} < 0; \\ \frac{-b_j^{(i)} + \sqrt{\delta_j^{(i)}}}{2a_j^{(i)}} \text{step}\left(2a_j^{(i)}\min(\lambda^{(0)}) + b_j^{(i)} + \sqrt{\delta_j^{(i)}}\right) & \text{if } a_j^{(i)} > 0,\ \delta_j^{(i)} \ge 0. \end{cases} \tag{4.8}$$

In (4.6)-(4.8), λ^* is a scalar and λ^i, $i \in \mathbb{N}_w$ are vectors. The subscript j represents the jth element of a vector.

The equations (4.4), (4.6)-(4.8) constitute the proposed safe controller π^*, which renders the system (4.1) safe on S under Assumption 4.3.1. At the same time, $\pi^*(x)$ is the closest safe input along the line connecting $\pi_s(x)$ and $\pi_r(x)$ for any x. This means that if $\pi_r(x)$ is a safe control input, i.e., $\pi_r(x) \in U(x)$ and $f_{\text{PWA}}(x, \pi_r(x)) \in S$, we have $\pi^*(x) = \pi_r(x)$.

These mechanisms make (4.4) and (4.6)-(4.8) work as a safety filter for the system (4.1). Fig. 4.1 illustrates the geometric meanings of λ^* and π^*.

To solve Problem P2, since we have obtained the closed-form of the mapping from x, $\pi_r(x)$, and $\pi_s(x)$ to $\lambda^*(x)$, we can easily involve (4.4) in the training loop of the learning-based policy π_r. Compared to [207], which requires differentiating the optimizer of an optimization problem, our method enables efficient policy gradient computation by applying the chain rule to (4.4) and (4.6)-(4.8).

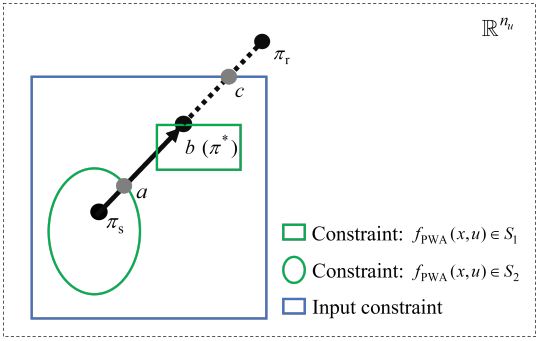


Figure 4.1: Geometric interpretation of the proposed safety filter. We consider the case when $S = S_1 \cup S_2$. Each region with a green boundary represents the constraint on u such that $f_{\text{PWA}}(x,u) \in S_i$, $i = 1,2$. The input from the reference policy π_r is outside U and thus unsafe. We search for the least conservative input along the segment from π_s to π_r. According to (4.6)-(4.8), we have $\min(\lambda^{(1)}) = \frac{|a-\pi_s|}{|\pi_r-\pi_s|}$, $\min(\lambda^{(2)}) = \frac{|b-\pi_s|}{|\pi_r-\pi_s|}$, and $\min(\lambda^{(0)}) = \frac{|c-\pi_s|}{|\pi_r-\pi_s|}$. Then, $\lambda^* = \frac{|b-\pi_s|}{|\pi_r-\pi_s|}$, and the filtered input is b.

Remark 4.3.1. Our design draws inspiration from [194], in which the solution of convexly constrained optimization problems is approximated while constraints are strictly satisfied. In contrast, our method accommodates non-convex constraints (a union of polyhedra and ellipsoids). Furthermore, unlike [194], we do not necessitate $\pi_s(x)$ to lie in the interior of each polyhedron and ellipsoid.

While the computational advantage of the proposed safe filter based on min-max operations cannot be formally or rigorously proved, it is typically true in practice. Besides, in [73], it is also verified by simulation that the min-max based approach is faster than MPC for PWA systems.

4.4. SYNTHESIZING S AND π_s

In this section, we discuss and present the methods of synthesizing the controlled invariant set S and the safe policy π_s for the PWA system (4.1).

Existing methods of synthesizing S for PWA systems include the LMI approach [124], polyhedral iteration [130], and polytopic trees [175]. In this section, we first inner approximate the original state constraint by a polyhedral constraint. The above-mentioned approaches can then be adopted to compute a controlled invariant set S_0 under the inner approximated constraint. Then, we propose a PWA and PWQ classification approach to estimate a large S that is a union of polyhedra and ellipsoids. The approach can ap-

proximate the maximum control-invariant set under the original constraint.

4.4.1. INNER APPROXIMATION OF THE STATE CONSTRAINT

To apply the existing methods of synthesizing $\mathcal{S}$, we need to inner approximate the state constraint by a polyhedron $\{x \in \mathbb{R}^{n_x} \mid Gx \leq g\}$ with $G \in \mathbb{R}^{n_g \times n_x}$ and $g \in \mathbb{R}^{n_g}$. The matrix G should be determined in advance. The choice of G is problem-specific and the simplest choice is $G = [\mathbb{I}_{n_x} \ -\mathbb{I}_{n_x}]^T$, which makes the polyhedron a hyper-rectangle. Then, we decrease g until the following condition is verified:

$$x \in X, \ \forall x \text{ s.t. } Gx \leq g. \tag{4.9}$$

The constraint $x \in \mathcal{C}_i$ or $x \in X_i$ can be easily represented by linear or quadratic inequalities, and the state constraint is the intersection of $\cup_{i \in \mathbb{N}_s^+} \mathcal{C}_i$ and $\cup_{i=1}^{m_x} X_i$. With this knowledge, for the state constraint we can compute a validation function $h_X(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}$ (as defined below) through some min and max operators [47].

Definition 4.4.1 (Validation function)**.** Given a closed set Z, a function $h_Z(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is called a *validation function* for the constraint $z \in Z$ if $h_Z(z) \leq 0 \Leftrightarrow z \in Z$.

Then, the condition (4.9) is equivalent to

$$\max_{x \in \mathbb{R}^{n_x}} \{h_X(x), \text{ s.t. } Gx \leq g\} \leq 0. \tag{4.10}$$

The included problem in (4.10) can be solved by mixed-integer linear or quadratic programming.

With a polyhedral inner approximation of the state constraint available, we can thereby apply the approaches [124], [130], [175] to obtain a union of polyhedra or a union of ellipsoids controlled invariant set S_0 and a safe policy under the inner approximated constraint with guarantees.

4.4.2. A PWA AND PWQ CLASSIFICATION APPROACH

In the previous subsection, there are several steps that can induce conservatism for S. Firstly, the state constraint is inner approximated. Besides, if the LMI approach [124] is used, we can only obtain S for the piecewise linear subsystem[1], rather than the original PWA system (4.1). Conservatism can also arise when applying the S-procedure to get tractable LMIs. These simplifications usually result in very small S. This observation motivates us to consider enlarging S.

According to the reachability analysis [35, Theorem 11.1], the T-step controllable set S_T of a controlled invariant set S_0 is still controlled invariant, and S_T expands to the maximum controlled invariant set as T grows to infinity. Exactly computing S_T for PWA systems with complex state constraints is impossible. Therefore, we present a data-driven approach to approximate S_T. First, we need an oracle to tell whether a particular state is

[1] The piecewise linear subsystem is obtained by removing every mode i of (4.1) such that $0 \notin \text{closure}(\mathcal{C}_i)$.

in S_T. To achieve this, we consider the following unconstrained optimization problem:

$$B_T(x) := \min_{\{u_t, x_t\}_{t=0}^{T}} \max \left\{ \max_{t \in \mathbb{N}_{T-1}} h_X(x_t), \max_{t \in \mathbb{N}_{T-1}} h_{U(x_t)}(u_t), h_{S_0}(x_T) \right\}$$
$$\text{s.t. } x_0 = x,\ x_{t+1} = f_{\text{PWA}}(x_t, u_t),\ t = 0, 1, ..., T-1, \tag{4.11}$$

representing a discrete-time version of the Hamilton-Jacobi reachability problem [15]. In (4.11), S_0 is a union of polyhedra and ellipsoids controlled invariant set, which can be obtained by the above-mentioned approaches [124], [130], [175]. The functions h_X, h_U, and h_{S_0} are PWA or PWQ validation functions. Problem (4.11) can be straightforwardly cast as an MICP problem.

Lemma 4.4.1. The optimal value function B_T of (4.11) is a validation function for the constraint $x \in S_T$, i.e., $S_T = \{x \in \mathbb{R}^{n_x} \mid B_T(x) \leq 0\}$.

Proof: According to (4.11), we can straightforwardly get the following equivalence: $B_T(x) \leq 0 \Leftrightarrow$ there exists $\{(u_t, x_t)\}_{t=0}^{T}$ such that $h_X(x_t) \leq 0$, $h_{U(x_t)}(u_t) \leq 0$, and $h_{S_0}(x_T) \leq 0 \Leftrightarrow x \in S_T$. ■

Lemma 4.4.1 implies that B_T can serve as the oracle by testing whether $B_T(x) \leq 0$ for any particular state x. Then, we can sample the state space, solve the problem (4.11) to generate a data set $\{(x_i, \text{step}(-B_T(x_i)))\}_{i=1}^{N_x}$, where N_x is the number of samples. To learn S_T, existing literature has used a NN binary classification model [32] or a support vector machine [185] to learn the mapping from x to $\text{step}(-B_T(x))$. These approaches, however, are not suitable in our case because we require the learned set to be a union of polyhedra and ellipsoids. To this end, we design the following parameterized function:

$$\hat{B}(x, \omega) = \min_{j \in \mathbb{N}_l^+} \{\hat{B}_j(x, \omega_j)\} - 1, \tag{4.12}$$

where $l \in \mathbb{N}^+$ and

$$\hat{B}_j(x, \omega_j) = ||P_j(x - c_j)||_\infty \text{ or } (x - c_j)^T P_j (x - c_j) \tag{4.13}$$

with the learning parameters $\omega_j = (P_j, c_j)$, $P_j \in \mathbb{R}^{n_x \times n_x}$ and $c_j \in \mathbb{R}^{n_x}$. All the parameters are condensed in ω. In (4.12), each $\hat{B}_j$ is called a component function. For each state, the smallest component function is called the active function. The equation (4.13) means that for each component, we can use either a PWA function or a quadratic function. Each P_j is full rank if $\hat{B}_j$ is PWA, or positively definite if $\hat{B}_j$ is quadratic. The ways of guaranteeing P_j to be full rank or positively definite can be found in [67], [100], which leverage singular value or Cholesky decomposition. The zero sub-level set of $\hat{B}$, denoted by $\hat{S}_B$, is a union of polyhedra and ellipsoids.

The parameters are trained to minimize the misclassification error over the state space. Intuitively, this can be formulated as the following unconstrained optimization problem:

$$\min_{\omega} \int_{S_T \setminus \hat{S}_B} 1 dx + \int_{\hat{S}_B \setminus S_T} 1 dx. \tag{4.14}$$

The objective function represents the volume of the regions that $\hat{B}$ misclassifies. However, it is hard to find the analytical relation between the objective function of (4.14)

and ω. This means that to solve (4.14), one can only use numerical gradient-based or gradient-free optimization algorithms. Since w is composed of several matrices and vectors, these algorithms become inefficient when the total number N_x of samples is large and w has numerous components. To deal with this issue, we propose to train the parameters such that the following optimization problem is solved[2]:

$$\min_{\omega} \int_{x\in\mathbb{R}^{n_x}} \frac{\operatorname{sign}(B_T(x))\max\{-B_T(x)\hat{B}(x,\omega),0\}}{B_T(x)}\,dx, \tag{4.15}$$

where $\operatorname{sign}(\cdot)$ returns the sign of a real number and specifically $\operatorname{sign}(0)/0=-1$.

Interpretation of (4.15). For any x, the term in the integral is a measure of the misclassification. In particular, if $x\in S_T\cap\hat{S}_B$ or if $x\notin S_T$ and $x_i\notin\hat{S}_B$, the term in the integral reaches its minimum (zero). Otherwise, the learning model (4.12) misclassifies x and the term in the integral is $-\operatorname{sign}(B_T(x))\hat{B}(x,\omega)$. This indicates that the objective of (4.14) is to make $\hat{B}(x,\omega)$ have the same sign as $B_T(x)$ for all x.

As the integral in (4.15) is difficult to compute, like existing supervised learning methods [32], we approximate the objective of (4.15) by an empirical loss, i.e., the sum of the terms in the integral over a finite number of samples $\{x_i\}_{i=1}^{N_x}$. Then, we use stochastic gradient descent [86, Section 5.9] to update ω and the complexity is independent on N_x.

We employ two important strategies to avoid highly sub-optimal solutions. The first strategy involves multi-start optimization with random initialization of ω [171]. For the second strategy, we refer to it as multi/random gradient descent. When applying a classical stochastic gradient descent algorithm to (4.12), ω is susceptible to falling into sub-optimal points where many component functions of $\hat{B}$ remain inactive for all states. The multi/random gradient descent is thereby proposed as follows. For the sample x_i that satisfies $\hat{B}(x_i,\omega)\le 0$ but $B_T(x_i)>0$, our algorithm updates all non-negative $\hat{B}_j$. This targeted update can accelerate the learning process. Conversely, for the sample x_i that satisfies $\hat{B}(x_i,\omega)>0$ but $B_T(x_i)\le 0$, we randomly select several positive $\hat{B}_j$ to update. This randomized selection prevents the repetition of updating the same component during the learning phase, promoting a more diverse exploration of the parameter space.

It should be pointed out that the proposed classification method cannot guarantee the invariance of $\hat{S}_B$. However, as T increases, the method can approach the maximal controlled invariant set, thereby reducing the conservatism compared to existing methods [124], [175].

4.4.3. LEARNING π_s

After the training of $\hat{B}$ has been finished, we further learn a safe policy $\hat{\pi}_s(\cdot,\beta)$ with the parameter β. The choice of $\hat{\pi}_s$ is flexible, and can accommodate various forms such as NNs. According to Assumption 4.3.1, $\hat{\pi}_s$ is expected to satisfy $f_{\text{PWA}}(x,\hat{\pi}_s(x,\beta))\in\hat{S}_B$ and $\hat{\pi}_s(x,\beta)\in U(x)$ for all $x\in\hat{S}_B$. This prompts us to optimize β to minimize the following objective:

$$\max_{x\in\hat{S}_B}\max\left\{\hat{B}(f_{\text{PWA}}(x,\hat{\pi}_s(x,\beta),\omega)),\ h_{U(x)}(\hat{\pi}_s(x,\beta))\right\}.$$

[2]It is hard to prove that any local optima of (4.15) is an optimum of (4.14), unless B_T and $\hat{B}$ are $\mathbb{C}^1$ functions.

Similar to the proposed method for solving (4.15), we replace the above objective by an empirical loss and use stochastic gradient descent to update β.

4.5. CASE STUDY

We consider the lateral and yaw dynamics of a bicycle model [80]:

$$\begin{aligned} m\ddot{y} &= -mv_x\dot{\psi} + 2(F_{y_f} + F_{y_r}) \\ I\ddot{\psi} &= 2(aF_{y_f} - bF_{y_r}), \end{aligned} \tag{4.16}$$

where $x = [\dot{y}\ \dot{\psi}]^T$ is the state, m is the mass, I is the inertia, a and b are the distances from the front and rear wheels to the center of gravity, ψ is the heading angle, v_x is the longitudinal speed, which is assumed to be constant, F_{y_f} and F_{y_r} are front and rear tire lateral forces. Based on the results of [80], F_{y_f} and F_{y_r} can be approximated as PWA functions of $[\dot{y}\ \dot{\psi}\ \delta]^T$, where δ is the steering angle (input). The PWA functions are given by

$$F_{y_f} = \begin{cases} C_2\left(\frac{\dot{y}+a\dot{\psi}}{v_x} + \alpha^*\right) - C_1\delta - C_1\alpha^* & \text{for } \frac{\dot{y}+a\dot{\psi}}{v_x} < -\alpha^* \\ C_1\left(\frac{\dot{y}+a\dot{\psi}}{v_x} - \delta\right) & \text{for } \frac{\dot{y}+a\dot{\psi}}{v_x} \in [-\alpha^*, \alpha^*] \\ C_2\left(\frac{\dot{y}+a\dot{\psi}}{v_x} - \alpha^*\right) - C_1\delta + C_1\alpha^* & \text{for } \frac{\dot{y}+a\dot{\psi}}{v_x} > \alpha^*, \end{cases}$$

$$F_{y_r} = \begin{cases} C_2\left(\frac{\dot{y}-b\dot{\psi}}{v_x} + \alpha^*\right) - C_1\alpha^* & \text{for } \frac{\dot{y}-b\dot{\psi}}{v_x} < -\alpha^* \\ C_1\frac{\dot{y}-b\dot{\psi}}{v_x} & \text{for } \frac{\dot{y}-b\dot{\psi}}{v_x} \in [-\alpha^*, \alpha^*] \\ C_2\left(\frac{\dot{y}-b\dot{\psi}}{v_x} - \alpha^*\right) + C_1\alpha^* & \text{for } \frac{\dot{y}-b\dot{\psi}}{v_x} > \alpha^*, \end{cases}$$

where C_1, C_2 and α^* are constants. After discretizing (4.16) by Euler Discretization with the sampling time 0.05 s, we get a discrete-time PWA system (4.1) with 9 polyhedral regions. The system is subject to the input and state constraints: $|\delta| \le 0.17$, $x \in X_1 \cup X_2$ with $X_1 = \{x \,|\, |\dot{y}| \le 1, |\dot{\psi}| \le 1\}$, $X_2 = \{x \,|\, \dot{y}^2 + \dot{\psi}^2 \le 1.2\}$.

The simulations are conducted in MATLAB R2021a on an AMD Core R7-5800H CPU @3.20GHz machine. All MICP problems are solved by Gurobi [92].

After applying the LMI approach, we obtain an ellipsoidal controlled invariant set $S_0 = \{x|\ x^T P x \le 1\}$ with $P = \begin{bmatrix} 4.606 & 0 \\ 0 & 7.597 \end{bmatrix}$. We sample the state space $\{x|\ |\dot{y}| \le 1.5,\ |\dot{\psi}| \le 1.5\}$ by a uniform grid and get 100^2 samples. Then, the problem (4.11) is solved with $T = 7$ for each state sample. Since the regions near the boundary of S_T are of more interest than the regions away from the boundary, we copy the samples x_i that satisfy $|B_T(x_i)| \le \epsilon$. Here ϵ is a small positive constant. As a result, we get the training data $\{(x_i, \text{step}(-B_T(x_i)))\}_{i=1}^{15215}$ for the proposed classifier (4.12).

Fig. 4.2 shows the performance of the proposed classifier (4.12) tested on 1000 samples that are randomly selected from the training samples. We take $N_\text{p} = N_\text{e} = 15$, where N_p (N_e) refers to the number of PWA (quadratic) component functions in (4.12). The classifier is trained by the proposed multi/random gradient descent. We observe that the proposed classifier can distinguish between the state samples inside and outside S_T

with high accuracy. We also compare the multi/random gradient descent with the particle swarm optimization as employed in [83] for $N_e = 10$ and $N_p = 0$. The result indicates that the proposed method achieves an impressive 96.51% classification accuracy within a training time of 29 seconds, whereas the particle swarm optimization yields a significantly lower accuracy of 70.32% after 573 seconds of training.

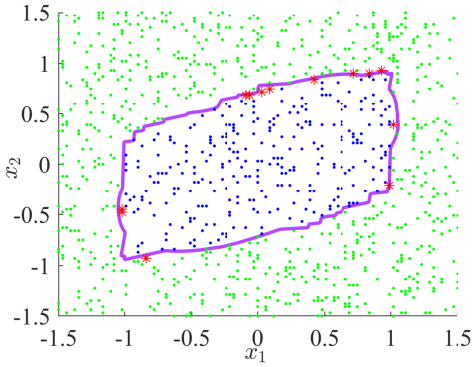


Figure 4.2: Performance of the proposed classifier $\hat{B}$. The blue and green points represent the states within and outside S_T, the purple boundary is $\{x|\hat{B}(x,\omega)=0\}$, and the red stars represent the states that $\hat{B}$ misclassifies.

Next, we verify the proposed safety filter. We consider three kinds of learning-based controllers: supervised learning of MPC [114], an approximate-dynamic-programming (ADP) controller [99], and supervised learning of MPC with the safety filter (4.4) included in the training loop. The learning-based controllers, as well as the learned safe policy $\hat{\pi}_s$, are represented by NNs with three hidden layers with hyperbolic tangent activation functions. The three layers contain 16, 64, and 16 neurons. To improve safety, $\hat{B}$ and π_s are trained with 5% state constraint tightening.

Table 4.1: Comparison of safety filters with different $\hat{S}_B$ and different solving algorithms.

Safety filter	Algorithm	Learning MPC			ADP			Learning MPC with (4.4) included		
		CPU (ms)	Cost	Safety rate	CPU (ms)	Cost	Safety rate	CPU (ms)	Cost	Safety rate
No		0.004	4.776	19.53%	0.002	4.728	15.62%			
$N_p = 0$ $N_e = 30$	Proposed	0.209	4.798	100%	0.207	4.728	100%	0.218	4.732	100%
	MIQCP	84.5	4.798	100%	84.7	4.728	100%			
	Nonlinear	8.61	4.798	100%	8.72	4.728	100%			
$N_p = 15$ $N_e = 15$	Proposed	0.339	4.808	100%	0.339	4.738	100%	0.364	4.748	100%
	MIQCP	64.7	4.858	100%	64.4	4.787	100%			
	Nonlinear	8.71	13.07	96.88%	8.43	8.536	96.88%			
$N_p = 5$ $N_e = 5$	Proposed	0.177	4.963	97.66%	0.180	4.954	97.66%	0.164	4.770	100%
	MIQCP	44.2	5.823	97.66%	44.3	5.754	97.66%			
	Nonlinear	8.53	21.33	90.62%	9.20	16.37	90.62%			

Table 4.1 shows the comparative results. The performance metrics include the total cost $\sum_{t=0}^{50} x_t^T x_t + \delta_t^2$, the rate of safety, and the average CPU time for generating the input per time step. For each $\hat{S}_B$, we compare three methods for generating safe control inputs. The first one is the proposed optimization-free safe controller (4.4). The last two ones are based on solving (4.2) (with S replaced by $\hat{S}_B$). "MIQCP" indicates that (4.2) is solved by mixed-integer quadratically constrained programming, while "nonlinear" indicates

that (4.2) is solved by the active-set nonlinear programming algorithm, implemented using the MATLAB function "fmincon"[3].

Without safety filters, both learning MPC and ADP show lower safety rates compared to when safety filters are applied, with the proposed method and MIQCP almost achieving a 100% safety rate. The CPU time of computing the proposed filter (4.4) is less than 0.4 ms, significantly lower than that of solving MIQCP and nonlinear programming problems. Additionally, the proposed filter (4.4) generally performs competitively with MIQCP in terms of cost. These results suggest that the choice of safety filter and its parameters can significantly impact the safety and computational efficiency of learning MPC and ADP policies, with the proposed filter showing promise for achieving high safety rates at low computational costs. Additionally, when considering (4.4) in the training loop of learning MPC, the total cost is slightly lower than that of learning MPC without (4.4) in the training loop. Besides, although the proposed safe filter (4.4) outputs a suboptimal solution to (4.2), it does not mean that the policy π^* results in a higher total cost than π_{p}.

4.6. CONCLUSIONS AND FUTURE WORK

This chapter has presented a new safety filter, called the optimization-free safety filter, which can enhance safety for PWA systems subject to the combination of polyhedral and ellipsoidal constraints. Compared to standard safety filters that need to solve a usually nonconvex optimization problem online, the proposed safety filter has very small online computational overhead. To construct the safety filter, we have developed a new PWA and PWQ classification approach to learn a controlled invariant set from data. Simulation on a bicycle model with hybrid approximation has demonstrated that the proposed method can achieve zero constraint violations and generate control signals at the millisecond level. Topics for future work include handling more complex constraints, improving the applicability in large-scale PWA systems, and testing the computational advantages in more complex and realistic problems.

4.A. APPENDIX: PROOF OF PROPOSITION 4.3.1

Let λ^+ denote the right-hand side of (4.6). The task is to prove $\lambda^+ = \lambda^*$.

(i) Feasibility of λ^+. Since $\pi_{\mathrm{s}}(x) \in U(x)$, we have $c \geq 0$. According to (4.7), we know $\lambda^{(0)} \geq 0$. Meanwhile, as the constraint $\lambda \in [0,1]$ is included in (4.5b), $\exists j$ such that $\lambda_j^{(0)} = 1$. Therefore, we also have $\lambda^{(0)} \leq 1$. Since there exists at least one $i \in \mathbb{N}_s^+$ such that $f_{\mathrm{PWA}}(x, \pi_{\mathrm{s}}(x)) \in S_i$, we have that there exists at least one $i \in \mathbb{N}_s^+$ such that $c^{(i)} \geq 0$. Similarly, we know $\max_{i \in \mathbb{N}_w^+} \{\min(\lambda^{(i)})\} \geq 0$. Therefore, we get $\lambda^+ \geq 0$.

If $\lambda^+ = 0$, by virtue of Assumption 4.3.1 we obtain the feasibility of λ^+. In the remaining proof of the feasibility, we consider $\lambda^+ > 0$. Firstly, we consider the case when $a_j = 0$. This case could occur when $U(x)$ is polyhedral, or when $U(x)$ is ellipsoidal and $\pi_{\mathrm{r}}(x) = \pi_{\mathrm{s}}(x)$. If $b_j > 0$, $b_j \lambda^+ \leq b_j c_j / b_j \leq c_j$. If $b_j \leq 0$, $b_j \lambda^+ \leq 0 \leq c_j$. Then, we consider the case when $a_j > 0$, i.e., when $U(x)$ is ellipsoidal and $\pi_{\mathrm{r}}(x) \neq \pi_{\mathrm{s}}(x)$. As $\pi_{\mathrm{s}}(x) \in U(x)$, the quadratic

[3] Since both f_{PWA} and $\hat{B}$ are continuous functions, we can treat them as general continuous nonlinear functions without considering the PWA property of them.

equation $a_j\lambda^2 + b_j\lambda - c_j = 0$ has one non-negative root $\lambda_j^{(0)}$ and one non-positive root. As $\lambda^+ \in (0, \lambda_j^{(0)}]$, it follows that $a_j\lambda^{+2} + b_j\lambda^+ - c_j \le 0$. Applying the above argument to all the elements of a leads to the feasibility of λ^+ regarding (4.5b).

Let $i' = \arg\max_{i\in\mathbb{N}_w^+}\{\min(\lambda^{(i)})\}$. Next, we prove the feasibility of λ^+ regarding (4.5c) by verifying (4.5c) for $i = i'$. For the jth element $a_j^{(i')}$ of $a^{(i')}$, we define $f_j^{(i')}(\lambda) := a_j^{(i')}\lambda^2 + b_j^{(i')}\lambda - c_j^{(i')}$. We consider the five cases in (4.8).

Cases (1&2): $a_j^{(i')} = 0$ and $b_j^{(i')} \ge 0$. In these two cases, if $c_j^{(i')} < 0$, we have $\lambda^+ = \lambda_j^{(i')} = 0$, which is certainly feasible for problem (4.3) because of Assumption 4.3.1. If $c_j^{(i')} \ge 0$, as $f_j^{(i')}(\cdot)$ is monotonically non-decreasing, we have $f_j^{(i')}(\lambda^+) \le f_j^{(i')}\left(\lambda_j^{(i')}\right) \le 0$.

Case (3): $a_j^{(i')} = 0$ and $b_j^{(i')} < 0$. In this case, if $\min(\lambda^{(i')}) = 0$, we have $\lambda^+ = 0$ and the feasibility of λ^+. If $\min(\lambda^{(i')}) > 0$, it follows from (4.8) that

$$\kappa_j^{(i)} := c_j^{(i)} - b_j^{(i)} \min\left\{\min(\lambda^{(0)}), \min_{k \text{ s.t. } \lambda_k^{(i)}\in(0,1)} \lambda_k^{(i)}\right\} \ge 0 \text{ for } i = i'. \tag{4.17}$$

If $\min(\lambda^{(i')}) = 1$, we get $f_j^{(i')}(\lambda^+) = b_j^{(i')}\min(\lambda^{(0)}) - c_j^{(i')} \le 0$ because of (4.17). If $\min(\lambda^{(i')}) < 1$, there must exist $k \in \mathbb{N}^+$ such that $\min(\lambda^{(i')}) = \lambda_k^{(i')} = c_k^{(i')}/b_k^{(i')}$, and consequently, $f_j^{(i')}(\lambda^+) = b_j^{(i')}\min\left\{\min(\lambda^{(0)}), \lambda_k^{(i')}\right\} - c_j^{(i')} \le 0$ thanks to (4.17).

Case (4): $a_j^{(i')} > 0$ and $\delta_j^{(i')} < 0$. We have $\lambda^+ = 0$ and thus the feasibility of λ^+.

Case (5): $a_j^{(i')} > 0$ and $\delta_j^{(i')} \ge 0$. Similarly to Case (3), we only need to consider the case when

$$\tau_j^{(i)} := 2a_j^{(i')}\min(\lambda^{(0)}) + b_j^{(i)} + \sqrt{\delta_j^{(i)}} \ge 0 \text{ for } i = i', \tag{4.18}$$

because otherwise we have $\lambda^+ = 0$. The condition (4.18) implies that the smaller root of $f_j^{(i')}(\lambda) = 0$ is not larger than $\min(\lambda^{(0)})$, and that the larger root is $\lambda_j^{(i')}$. Note that for ellipsoidal $E_{i'}$, $j = 1$. Therefore, we have $f_j^{(i')}(\lambda^+) = f_j^{(i')}\left(\min\left\{\min(\lambda^{(0)}), \lambda_j^{(i')}\right\}\right) \le 0$.

Combining the above 5 cases, we know that $f_j^{(i')}(\lambda^+) \le 0$, which proves the feasibility of λ^+ w.r.t. (4.5c).

(ii) Optimality of λ^+. If $\lambda^+ = 1$, the optimality directly follows from the constraint $\lambda \le 1$. In the remaining proof, we consider $\lambda^+ < 1$, and prove that $\lambda^+ + \Delta_\lambda$ is infeasible for problem (4.5) for any $\Delta_\lambda \in (0, 1-\lambda^+]$.

Case (1): $\lambda^+ = \min(\lambda^{(0)}) = 0$. In this case, $\exists j$ such that either $a_j = c_j = 0$, $b_j > 0$ or $a_j > 0$, $c_j = 0$. This implies that $\pi_s(x) \in \partial U(x)$ and $\pi_r(x) \notin U(x)$. As $U(x)$ is convex, 0 is the only feasible point and thus optimal for (4.3c).

Case (2): $\lambda^+ = \min(\lambda^{(0)}) \in (0,1)$. In this case, $\pi_s(x)$ is in the interior of $U(x)$, and $\pi_r(x) \notin U(x)$. From (4.7), we see that $\min(\lambda^{(0)})$ represents the distance from $\pi_s(x)$ to $\partial U(x)$. As $U(x)$ is convex, λ^+ is the optimal solution to (4.5).

Case (3): $\lambda^+ = \max_{i\in\mathbb{N}_w^+}\{\min(\lambda^{(i)})\} \in (0,1)$. In this case, $\forall i \in \mathbb{N}_w^+$, there exists j such that $\lambda_j^{(i)} \le \lambda^+$. Then, we can distinguish five sub-cases based on (4.8):

- If $a_j^{(i)} = 0$, $b_j^{(i)} > 0$, then $f_j^{(i)}(\lambda^+ + \Delta_\lambda) \ge f_j^{(i)}\left(\lambda_j^{(i)} + \Delta_\lambda\right) \ge b_j^{(i)}\Delta_\lambda > 0$. This means $\lambda^+ + \Delta_\lambda$ violates the jth inequality of (4.5c).
- If $a_j^{(i)} = b_j^{(i)} = 0$, $c_j^{(i)} < 0$ or if $a_j^{(i)} > 0$, $\delta_j^{(i)} < 0$, we have $f_j^{(i)}(\lambda) = a_j^{(i)}\lambda^2 + b_j^{(i)}\lambda - c_j^{(i)} > 0$ holds for any $\lambda \in \mathbb{R}$, which means $\lambda^+ + \Delta_\lambda$ violates the jth inequality of (4.5c).
- If $a_j^{(i)} = 0$, $b_j^{(i)} < 0$ and $\kappa_j^{(i)} < 0$, then, for any $\Delta_\lambda \in \left(0, \min\left\{\min(\lambda^{(0)}), \min_{k \text{ s.t. } \lambda_k^{(i)}\in(0,1)} \lambda_k^{(i)}\right\} - \lambda^+\right]$, we have $f_j^{(i)}(\lambda^+ + \Delta_\lambda) = b_j^{(i)}(\lambda^+ + \Delta_\lambda) - c_j^{(i)} \ge -\kappa_j^{(i)} > 0$. For any $\Delta_\lambda > \min\left\{\min(\lambda^{(0)}), \min_{k \text{ s.t. } \lambda_k^{(i)}\in(0,1)} \lambda_k^{(i)}\right\} - \lambda^+$, we have either $\lambda^+ + \Delta_\lambda > \min(\lambda^{(0)})$ or $\lambda^+ + \Delta_\lambda > c_k^{(i)}/b_k^{(i)}$ for a k s.t. $\lambda_k^{(i)} \in (0,1)$. If $\lambda^+ + \Delta_\lambda > \lambda^{(0)}$, it is clear that $\lambda^+ + \Delta_\lambda$ violates the constraint (4.5b). If $\lambda^+ + \Delta_\lambda > c_k^{(i)}/b_k^{(i)}$, we have $f_k^{(i)}(\lambda^+ + \Delta_\lambda) = b_k^{(i)}(\lambda^+ + \Delta_\lambda) - c_k^{(i)} > 0$, which means $\lambda^+ + \Delta_\lambda$ violates the kth inequality of (4.5c). Therefore, any increase $\Delta_\lambda \in (0, 1-\lambda^+]$ for λ^+ is infeasible for (4.5).
- If $a_j^{(i)} > 0$, $\delta_j^{(i)} \ge 0$, $\tau_j^{(i)} < 0$, then the smaller root of the equation $f_j^{(i)}(\lambda) = 0$, denoted by λ^-, is strictly larger than $\min\left(\lambda^{(0)}\right)$. As a result, if $\lambda^+ + \Delta_\lambda < \lambda^-$, we have $f_j^{(i)}(\lambda^+ + \Delta_\lambda) > 0$, i.e., $\lambda^+ + \Delta_\lambda$ is infeasible for (4.5c). Otherwise, we have $\lambda^+ + \Delta_\lambda \ge \lambda^- > \min\left(\lambda^{(0)}\right)$, which means $\lambda^+ + \Delta_\lambda$ is infeasible for (4.5b).
- If $a_j^{(i)} > 0$, $\delta_j^{(i)} \ge 0$, $\tau_j^{(i)} \ge 0$, then, $\lambda_j^{(i)}$ is the larger root of the equation $f_j^{(i)}(\lambda) = 0$, so we have $f_j^{(i)}(\lambda^+ + \Delta_\lambda) > f_j^{(i)}\left(\lambda_j^{(i)}\right) > 0$ holds for any $\Delta_\lambda \in (0, 1-\lambda^+]$.

Combining the above five sub-cases and noticing the arbitrariness of i and the existence of j and k, we can conclude that $\lambda^+ + \Delta_\lambda$ is infeasible for problem (4.5) in case (3).

5

PREDICTIVE CONTROL BARRIER FUNCTIONS FOR PIECEWISE AFFINE SYSTEMS WITH NON-SMOOTH CONSTRAINTS

Obtaining control barrier functions (CBFs) with large safe sets for complex nonlinear systems and constraints is a challenging task. Predictive CBFs address this issue by using an online finite-horizon optimal control problem that implicitly defines a large safe set. The optimal control problem, also known as the predictive safety filter (PSF), involves predicting the system's flow under a given backup control policy. However, for non-smooth systems and constraints, some key elements, such as CBF gradients and the sensitivity of the flow, are not well-defined, making the current methods inadequate for ensuring safety. Additionally, for control-non-affine systems, the PSF is generally nonlinear and non-convex, posing challenges for real-time computation. This chapter considers piecewise affine systems, which are usually control-non-affine, under nonlinear state and polyhedral input constraints. We solve the safety issue by incorporating set-valued generalized Clarke derivatives in the PSF design. We show that enforcing CBF constraints across all elements of the generalized Clarke derivatives suffices to guarantee safety. Moreover, to lighten the computational overhead, we propose an explicit approximation of the PSF. The resulting control methods are demonstrated through numerical examples .

This chapter is based on the paper [96].

5.1. INTRODUCTION

5.1.1. BACKGROUND

Motivated by emerging safety-critical applications such as autonomous driving [191] and automatic therapeutic regimens [110], control of nonlinear systems subject to state and input constraints has been receiving a growing attention in the academic community. Various safe control methods have been developed, such as model predictive control (MPC) [35], control barrier functions (CBFs)-based safety filters [8], solving Hamilton-Jacobi (HJ) reachability problems [14], and constrained reinforcement learning [102]. All methods have distinct advantages but also inherent limitations that hinder their practical effectiveness. In particular, MPC can be computationally expensive for hybrid and nonlinear systems; HJ reachability suffers from the curse of dimensionality; and constrained reinforcement learning often lacks rigorous safety guarantees.

CBFs provide a modular framework with formal safety guarantees, called the CBF-based safety filter, to monitor and modify the potential unsafe control actions online. CBFs use their super-level sets to define control invariant sets, and the safety filter further involves a constrained optimization problem with CBF-based safety constraints to produce a safe control policy that keeps the system within the control invariant set. However, computing a permissive CBF, i.e., one associated with a non-conservative control invariant set, can be particularly challenging for nonlinear systems with both state and input constraints. As a result, CBF-based safety filters may be overly conservative.

Drawing inspiration from MPC, researchers have proposed to use model predictions to reduce the conservatism of CBFs both in discrete-time [197], [198] and in continuous-time domains [38], [52], [93], [195]. In particular, they use a finite-horizon constrained optimal control problem with a terminal CBF constraint to implicitly represent a larger control invariant set, resulting in a predictive safety filter (PSF). The terminal CBF is referred to as the backup CBF in [93]. For continuous-time control-affine systems, the PSF can be cast as convex quadratic programs, enabling efficient real-time implementation.

It should be emphasized that existing predictive CBF approaches for continuous-time systems are inherently limited to smooth systems and smooth safety constraints [38], [52], [93], [195]. Furthermore, to allow for fast computation, these approaches concentrate on control-affine systems. For control-non-affine systems, the resulting safety filter usually involves solving a time-consuming non-convex nonlinear optimization problem with the number of constraints scaling linearly with the prediction horizon.

5.1.2. CONTRIBUTIONS

In this chapter, we address the challenge of designing PSFs for continuous-time piecewise affine (PWA) systems subject to arbitrary, possibly non-smooth, continuous state constraints[1] and polyhedral input constraints. PWA systems are generally control-non-affine and non-smooth, yet they can approximate general nonlinear systems to arbitrary precision and naturally capture hybrid behaviors. The resulting PSF is formulated as a mixed-integer quadratic programming (MIQP) problem. The main contributions of this

[1] We say that a constraint $x \in X$ is continuous if X can be represented by the level set of a continuous function, i.e., $X = \{x | f(x) \leq 0\}$, where f is continuous.

chapter, in comparison to the state of the art, are:

1. We propose a novel PSF that uses generalized Clarke derivatives to enforce safety constraints. This framework is applicable to any Lipschitz continuous but non-smooth CBF and any continuous PWA system with Lipschitz continuous but non-smooth state constraints. We rigorously analyze the safety guarantees of the resulting control policy (Lemma 5.4.1 and Theorem 5.4.2).

2. We present a tractable approach for computing the generalized Clarke derivative (Clarke sensitivity) of the system's flow. In particular, we introduce an alternative Aumann sensitivity and show their equivalence. Notably, the Aumann sensitivity can be efficiently computed through numerical integration of the system.

3. To reduce the computational complexity of solving the MIQP corresponding to the PSF online, we introduce an explicit approximation of the PSF. This approximation is provably feasible and significantly lowers the online computational burden.

4. Additionally, the theoretical safety result (Lemma 2) generalizes classical CBF conditions that require continuous differentiability to encompass CBFs that are merely Lipschitz continuous. This extension applies to general nonlinear systems, not just PWA ones.

5.1.3. RELATED WORK

(Approximate) MPC for PWA systems. Hybrid MPC is widely used to control PWA systems under linear state and input constraints [35]. In the hybrid MPC formulation, the PWA system is equivalently transformed into a mixed-logical dynamical (MLD) system that includes both continuous and discrete state and input variables [21]. This transformation allows the MPC problem to be formulated as a mixed-integer convex quadratic or linear optimization problem, which is solved online to generate safe control inputs. Explicit MPC is an offline version of hybrid MPC, which offline computes and stores the map from the state to the solution of the MPC problem by multi-parametric programming [20]. However, MPC methods struggle with significant computational challenges in high-dimensional systems, either due to solving complex mixed-integer problems or evaluating intricate explicit control laws.

This limitation of MPC motivates the research of approximating MPC control laws, by using, e.g., PWA simplicial functions [160], neural networks [102], sampling-based polytopic trees [175], and minimum-time objective functions [88]. Another promising approach to simplify the MPC problem is to employ reinforcement learning [182] or supervised learning [140], [174] to determine the values of the discrete decision variables, effectively reducing the mixed-integer problem to a convex optimization problem. However, while exact MPC offers formal performance guarantees, it leads to a high computational burden, especially for nonlinear, non-convex constraints. Approximate MPC, on the other hand, still has open challenges regarding stability, robustness, and sample complexity.

Predictive safety filters. Research in both the continuous-time and discrete-time domains has contributed to the development of PSFs, which aim to reduce the conser-

vatism of standard safety filters. Representative work includes [197], [198] in the discrete-time domain and [38], [52], [93], [195] (sometimes also called backup CBFs) in the continuous-time domain. Based on [197], [76] and [100] approximate discrete-time predictive CBFs using neural networks to reduce the computational burden of safety filters. The work in [151] extends the backup CBFs of [93], [195] to guaranteeing safe behavior for complex systems by designing safety filters for reduced-order models. However, continuous-time predictive CBF methods are fundamentally limited to smooth systems and smooth safety constraints. As mentioned before, this requirement arises from the need to differentiate both the safety constraints and the system's flow in the predictive CBF formulation, which requires computing the Jacobian of the closed-loop dynamics with a smooth backup controller. Moreover, to date, there is no related work addressing the design of PSFs for PWA systems.

5

Non-smooth CBFs. [85] proposes non-smooth CBFs for deterministic multi-agent systems, leveraging generalized gradients to ensure forward invariance. This result has been employed in applications where non-smooth safety constraints naturally arise, such as obstacle avoidance involving ellipsoids and polytopes [191] and network connectivity maintenance [155]. Besides, formulating multiple constraints with "min" and "max" functions will generally result in a non-smooth CBF. Building on [85], [191] extends non-smooth CBFs to stochastic systems. Related to these techniques, piecewise continuous CBFs are introduced in [63], where the authors of [63] argue that, in addition to satisfying the CBF inequality, a tangent cone condition at the boundary of the safe set should also be enforced to ensure forward invariance. Meanwhile, Cohen et al. [61] take a different perspective by smoothing the solution to CBF-based safety filters using the Implicit Function Theorem, rather than focusing on the smoothness of the CBFs themselves. All of the above approaches focus on classical CBFs with no predictions. This means that the analysis primarily addresses the non-smoothness of the CBF itself. In contrast, the non-smoothness in our setting arises not only from the structure of the CBF but also from state constraints and the PWA dynamics, making it more challenging to derive safety guarantees.

5.2. NOTATIONS

The set of real vectors with the dimension n is denoted by $\mathbb{R}^n$. The sets $\mathbb{N}$ and $\mathbb{N}^+$ represent the set of non-negative integers and the set of positive integers, respectively. Furthermore, $\mathbb{N}_a = \{0, 1, ..., a\}$ and $\mathbb{N}_a^+ = \{1, 2, ..., a\}$. The norm $||x||$ is the Euclidean norm of the vector x, and $||A||$ is the induced norm of the Euclidean norm for the matrix A. The notation $\text{int}(S)$ refers to the interior of the set S, and $\text{clo}(S)$ is the closure of the set S. Besides, 1_n is the vector of ones with n elements, and I_n is the identity matrix with dimension $n \times n$. A continuous function $\alpha : [0, \infty) \to [0, \infty]$ is said to belong to class $\mathcal{K}_\infty$ if it is strictly increasing and if $\alpha(0) = 0$ and $\lim_{r \to \infty} \alpha(r) = \infty$. The step function $\text{step}(\cdot)$ returns 1 if the input is non-negative, and 0 otherwise. The notation $Df(x_0) \in \mathbb{R}^{m \times n}$ represents the derivative of the differentiable function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ at x_0.

5.3. PRELIMINARIES AND BACKUP CBFS

Consider a general continuous-time nonlinear system of the form

$$\dot{x} = f(x,u), \tag{5.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, and $f(\cdot) : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is locally Lipschitz continuous. The system must always satisfy the given state constraint $x \in X := \{x \in \mathbb{R}^n | h_X(x) \geq 0\}$ and input constraint $u \in U$. Here the function $h_X(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous, and U is a polyhedron[2].

5.3.1. CONTROL BARRIER FUNCTIONS

CBFs offer an effective approach for designing safe control systems.

Definition 5.3.1 (Control barrier function[7])**.** A continuously differentiable function $h(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a *control barrier function* for (5.1) if there exists an $\alpha \in \mathcal{K}_\infty$ such that

$$\sup_{u \in U} \dot{h}(x,u) = \sup_{u \in U} \{\nabla h(x) \cdot f(x,u)\} \geq -\alpha(h(x)) \tag{5.2}$$

holds for any $x \in S := \{x \in \mathbb{R}^n | h(x) \geq 0\}$.

With a CBF available, there always exists a controller $\pi(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ satisfying $\pi(x) \in U$ and $\nabla h(x) \cdot f(x,\pi(x)) \geq -\alpha(h(x))$ for all $x \in S$. This controller will render S forward invariant for (5.1), and S is called the safe set [7]. Under the supplemental condition $S \subseteq X$, the controller π makes the system always satisfy the constraints if the initial state is within S.

5.3.2. PREDICTIVE SAFETY FILTERS FOR SMOOTH SYSTEMS

For nonlinear systems, it is nontrivial to get a permissive CBF, i.e., a CBF with a large safe set S [196]. Drawing ideas from model predictive control, PSFs address this issue using model predictions [52], [151]. We recall this method.

First, a CBF $h_\mathrm{b} : \mathbb{R}^n \to \mathbb{R}$ and an associated safe controller π_b, satisfying (i) $\pi_\mathrm{b}(x) \in U$ for all $x \in X$, and (ii) $\nabla h_\mathrm{b}(x) \cdot f(x,\pi_\mathrm{b}(x)) \geq -\alpha_\mathrm{b}(h_\mathrm{b}(x))$ for all $x \in S_\mathrm{b} := \{x \in \mathbb{R}^n | h_\mathrm{b}(x) \geq 0\} \subseteq X$, are obtained based on existing methods such as LQR or sum-of-squares programming [199], depending on the form of the system. Here, the subscript "b" refers to "backup".

The closed-loop system with the backup controller can be written as

$$\dot{x} = f_\mathrm{b}(x) := f(x,\pi_\mathrm{b}(x)) \tag{5.3}$$

If f and π_b are locally Lipschitz continuous, which is commonly the case in many practical scenarios, including linear, PWA, and polynomial systems and policies, then for any $x(0) = x_0 \in \mathbb{R}^n$, according to [118, Theorem 3.2], the system (5.3) has a unique solution $x(t) = \phi_\mathrm{b}(x_0,t)$, $\forall\, t \in [0,\infty)$.

Definition 5.3.2 (Constrained reachable set)**.** Given the system (5.3) and the set X, for any set S and any time $T > 0$, the T-step *constrained reachable set* for (5.3) is defined as $\Phi_\mathrm{const}(S,T) := \{x \in \mathbb{R}^n | \phi_\mathrm{b}(x,\tau) \in X,\ \forall \tau \in [0,T] \wedge \phi_\mathrm{b}(x,T) \in S\}$.

[2] While the results of this chapter can be extended to cases where U is a union of polyhedra, we assume for simplicity that U is a single polyhedron

The PSF method uses an important property related to constrained reachable sets, stating that the constrained reachable sets of any invariant set S are always controlled-invariant and are supersets of S. See [151, Lemma 1] and [93, Proposition 2] for a more precise statement. This invariance property allows us to synthesize a safe controller that has a larger safe region.

Lemma 5.3.1 (Safety of backup CBFs [93], [151])**.** Consider the system (5.1) with the state and input constraints $x \in X$ and $u \in U$, the backup controller π_b, and the backup CBF h_b. Suppose that f, h_X, and h_b and π_b are continuously differentiable. There exist a controller $\pi_\mathrm{safe}(\cdot): \mathbb{R}^n \to \mathbb{R}^m$ and $\alpha, \alpha_\mathrm{b} \in \mathcal{K}_\infty$ such that the following conditions hold $\forall T \geq 0$ and $\forall x \in \Phi_\mathrm{const}(S_\mathrm{b}, T)$:

$$\begin{aligned} \dot{h}_X\left(\phi_\mathrm{b}(x,\tau), \pi_\mathrm{safe}(x)\right) &\geq -\alpha\left(h_X\left(\phi_\mathrm{b}(x,\tau)\right)\right), \forall \tau \in [0,T] \\ \dot{h}_\mathrm{b}\left(\phi_\mathrm{b}(x,T), \pi_\mathrm{safe}(x)\right) &\geq -\alpha_\mathrm{b}\left(h_\mathrm{b}\left(\phi_\mathrm{b}(x,T)\right)\right) \\ \pi_\mathrm{safe}(\phi_\mathrm{b}(x,\tau)) &\in U, \forall \tau \in [0,T]. \end{aligned} \tag{5.4}$$

Moreover, any locally Lipschitz continuous controller π_safe that satisfies (5.4) renders $\Phi_\mathrm{const}(S_\mathrm{b}, T)$ forward invariant for (5.1).

The proof of Lemma 5.3.1 relies on verifying that

$$h(x) := \min\left\{\min_{\tau\in[0,T]} h_X(\phi_\mathrm{b}(x,\tau)), h_\mathrm{b}(\phi_\mathrm{b}(x,T))\right\} \tag{5.5}$$

is a valid CBF. The function h in (5.5) is called the *predictive CBF*.

Lemma 5.3.1 can be directly employed for safe controller synthesis by designing the following optimization-based PSF[3]:

$$\begin{aligned} \pi_\mathrm{safe}(x) = \underset{u\in U}{\operatorname{argmin}} \ & \|u - \pi_\mathrm{r}(x)\|^2 \\ \text{s.t. } & \dot{h}_X\left(\phi_\mathrm{b}(x,\tau), u\right) \geq -\alpha\left(h_X\left(\phi_\mathrm{b}(x,\tau)\right)\right), \forall \tau \in [0,T] \\ & \dot{h}_\mathrm{b}\left(\phi_\mathrm{b}(x,T), u\right) \geq -\alpha_\mathrm{b}\left(h_\mathrm{b}\left(\phi_\mathrm{b}(x,T)\right)\right), \end{aligned} \tag{5.6}$$

where $T \geq 0$ is the prediction horizon and $\pi_\mathrm{r}(\cdot): \mathbb{R}^n \to \mathbb{R}^m$ is a reference controller. Problem (5.6) is, in general, nonlinear and non-convex. In existing literature, a control-affine system and a polyhedral U are usually considered, which makes (5.6) a convex quadratic program (QP) (after discretizing $[0,\ T]$) [52], [151], [195]. Besides, thanks to the existence of π_b, problem (5.6) is guaranteed to be feasible $\forall x \in \Phi_\mathrm{const}(S_\mathrm{b}, T)$.

For the continuously differentiable system (5.1), the derivatives $\dot{h}_X$ and $\dot{h}_\mathrm{b}$ in (5.6) can be expressed as

$$\dot{h}_X\left(\phi_\mathrm{b}(x,\tau), u\right) = Dh_X\left(\phi_\mathrm{b}(x,\tau)\right) \frac{\partial \phi_\mathrm{b}(x,\tau)}{\partial x} f(x,u) \tag{5.7}$$

$$\dot{h}_\mathrm{b}\left(\phi_\mathrm{b}(x,T), u\right) = Dh_\mathrm{b}\left(\phi_\mathrm{b}(x,T)\right) \frac{\partial \phi_\mathrm{b}(x,T)}{\partial x} f(x,u), \tag{5.8}$$

where $\frac{\partial \phi_\mathrm{b}(x,\tau)}{\partial x} \in \mathbb{R}^{n\times n}$ is called the sensitivity of the flow to its initial condition[4].

[3] The conditions under which π_safe is locally Lipschitz, which is required in Lemma 5.3.1, can be found in [148].
[4] In (5.6), x refers to the initial condition of the safety filter, which is also the current state of the system (5.1).

Remark 5.3.1. Note that the backup controller π_b is a virtual controller that is never applied. It is only used to derive the CBF constraints in (5.6). The applied controller is π_{safe}.

5.3.3. PROBLEM STATEMENT

In this chapter, we focus on a class of continuous PWA systems:

$$\dot{x} = f_{\text{PWA}}(x,u) := A_i x + B_i u + c_i, \text{ for } \begin{bmatrix} x \\ u \end{bmatrix} \in \mathcal{P}_i, \tag{5.9}$$

where $f_{\text{PWA}} : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is a continuous PWA function, $\{\mathcal{P}_i\}_{i=1}^p$ constitutes a strict polyhedral partition [35, Definition 4.5] of the state-input space. The union of all boundaries of all $\mathcal{P}_i$, $i \in \mathbb{N}_p^+$ is called the boundary of the PWA system.

Computing the time derivatives in (5.7) is an essential step for designing the safety filter (5.6). It should be noted that existing literature usually assumes that the system, the CBF, and the constraint X are continuously differentiable [52], [151], [195]. In this case, the terms on the right-hand side of (5.7) are well-defined and safety performance is guaranteed (Lemma 5.3.1). However, the PWA system is usually not continuously differentiable at the boundary of each $\mathcal{P}_i$, and many widely existing constraints, such as polytopic constraints, are not differentiable everywhere. This means that the time derivatives in (5.7) may have multiple possible values and that the results in Lemma 5.3.1 may not hold.

Secondly, for control-affine systems, problem (5.6) can be formulated as a convex QP, which is computationally inexpensive to solve in real time. However, as observed in (5.9), $f_{\text{PWA}}(x,u)$ does not depend linearly on u. In this case, problem (5.6) becomes an MIQP, which needs much more computational effort to solve than a convex QP.

In the remainder, the following problems will be addressed.

Problem 1 (addressed in Section 5.4): How to design a PSF with safety guarantees for a given PWA system, which should address the non-smoothness of the PWA dynamics, the state constraint, and the CBF h_b?

Problem 2 (addressed in Section 5.5): How to overcome the computational intractability of the critical component of the safety filter, in particular, the not well-defined sensitivity for PWA systems?

Problem 3 (addressed in Section 5.6): How to approximate the solution to the safety filter in a computationally efficient way to enable swift online implementation?

5.3.4. GENERALIZED SENSITIVITY

For non-smooth functions, the concept of the Jacobian matrix from smooth functions can be generalized using several approaches, such as the generalized Clarke derivative [57] and Mordukhovich subdifferential [153]. In this chapter, we advocate for the use of the generalized Clarke derivative due to its convenience in performance analysis. The generalized Clarke derivative extends the idea of the Jacobian matrix to Lipschitz continuous functions that may not be differentiable everywhere.

Definition 5.3.3 (Generalized Clarke derivative [57])**.** The *generalized Clarke derivative* of a locally Lipschitz continuous function $\phi(\cdot):\mathbb{R}^n \to \mathbb{R}^{n_\phi}$, denoted by $\partial_C \phi$, is defined as

$$\partial_C \phi(x) = \operatorname{conv}\left\{ \lim_{i\to\infty} D\phi(x_i) \,\middle|\, x_i \to x,\ \phi \text{ is differentiable at } x_i \right\},$$

where $D\phi(x_i)$ is the classical Jacobian at points x_i where ϕ is differentiable. The notation n_ϕ refers to the image dimension of ϕ.

Note that for a locally Lipschitz function, the generalized Clarke derivative is nonempty, compact, convex, and set-valued [57]. It is the convex hull of the limits of classical Jacobian sequences at nearby points where the function is differentiable. If a function is differentiable at a point, the generalized Clarke derivative reduces to the classical Jacobian at that point.

5.4. SAFE CONTROLLER SYNTHESIS

5

For PWA systems, piecewise linear feedback controllers are often designed in the literature [124]. The PWA system (5.9) controlled by a piecewise linear backup controller can therefore be written as

$$\dot{x} = f_{\text{PWA,b}}(x) := D_i x + d_i, \text{ for } x \in \mathcal{R}_i. \tag{5.10}$$

In (5.10), $f_{\text{PWA,b}}:\mathbb{R}^n \to \mathbb{R}^n$ is a globally continuous PWA function, and $\{\mathcal{R}_i\}_{i=1}^r$ constitutes a strict polyhedral partition of the state space. Since (5.10) has a global Lipschitz constant $L = \max_{i\in\mathbb{N}_r^+} ||D_i||$, we can denote the unique solution with any initial condition $x(0) = x_0 \in \mathbb{R}^n$ by $x(\tau) = \phi_b(x_0,\tau)$, $\forall \tau \in [0,\infty)$. Furthermore, according to [118, Theorem 3.4], for any two initial states x_0 and y_0, the solutions of (5.10) satisfy $||\phi_b(x_0,\tau) - \phi_b(y_0,\tau)|| \le ||x_0 - y_0||\exp(L\tau)$, $\forall \tau > 0$. This proves the Lipschitz continuous dependence of $\phi_b(x_0,\tau)$ on the initial state x_0 in $\mathbb{R}^n$ for any finite τ.

Considering the case when ϕ_b, h_b, and h_X is not necessarily differentiable, and based on the design of PSFs for smooth systems in (5.6), we design the following PSF utilizing the generalized Clarke derivatives of ϕ_b, h_b, and h_X:

All-elements PSF:

$$\pi_{\text{safe}}(x) = \underset{u\in U}{\operatorname{argmin}} \ \|u - \pi_r(x)\|^2 \tag{5.11a}$$

$$\text{s.t. } \partial h_X Q f_{\text{PWA}}(x,u) \ge -\alpha\left(h_X\left(\phi_b(x,\tau)\right)\right)$$
$$\forall \partial h_X \in \partial_C h_X(\phi_b(x,\tau)),\ \forall Q \in \partial_C \phi_b(x,\tau),\ \forall \tau \in [0,T] \tag{5.11b}$$

$$\partial h_b Q f_{\text{PWA}}(x,u) \ge -\alpha_b\left(h_b\left(\phi_b(x,T)\right)\right)$$
$$\forall \partial h_b \in \partial_C h_b(\phi_b(x,T)),\ \forall Q \in \partial_C \phi_b(x,T), \tag{5.11c}$$

where $\partial_C \phi_b(x,\tau)$ is called the *Clarke sensitivity*. Different from (5.6) in which the CBF constraints are uniquely determined by the unique sensitivity matrix $\frac{\partial \phi_b}{\partial x}$ and the unique derivatives Dh_X and Dh_b, we require that the CBF constraints should be satisfied for *all* elements contained in $\partial_C h_X$, $\partial_C h_b$, and $\partial_C \phi_b$.

5.4.1. PERFORMANCE ANALYSIS

Before deriving a computationally tractable way to compute the Clarke sensitivity and to solve (5.11), we first analyze the safety performance of π_{safe}.

Assumption 5.4.1. (i) $\pi_{\text{b}}(x) \in U$, for all $x \in X$;

(ii) $\partial h_{\text{b}} f_{\text{PWA,b}}(x) \geq -\alpha_{\text{b}}(h_{\text{b}}(x))$, for all $x \in S_{\text{b}} := \{x \in \mathbb{R}^n | h_{\text{b}}(x) \geq 0\} \subseteq X$ and for all $\partial h_{\text{b}} \in \partial_{\text{C}} h_{\text{b}}(x)$.

Assumption 5.4.1 is reasonable and aligns with similar assumptions commonly found in the literature on predictive CBFs [52], [151] and MPC frameworks that incorporate CBFs or control Lyapunov functions for designing terminal constraints [93]. The key distinction in our assumption is that it requires the CBF constraint to hold for all elements in the Clarke generalized gradient of h_b, rather than relying solely on the unique classical gradient as typically assumed in existing works.

Theorem 5.4.1 (Feasibility)**.** Consider the continuous PWA system (5.9), the piecewise linear backup controller π_{b}, the Lipschitz continuous h_X and h_{b}, and the proposed PSF (5.11). Under Assumption 5.4.1, there exist α, $\alpha_{\text{b}} \in \mathcal{K}_\infty$ such that problem (5.11) is feasible for any $x \in \Phi_{\text{const}}(S_{\text{b}}, T)$.

Proof. We will prove that the backup solution $\pi_{\text{b}}(x)$ is a feasible solution to (5.11). To do so, for any $\tau \in [0, T]$, we need to consider several cases depending on the differentiability of the functions ϕ_{b}, h_{b}, and h_X. For clarity and without loss of generality, we consider the following two cases: (i) $\phi_{\text{b}}(x, \tau)$ is differentiable, and (ii) $\phi_{\text{b}}(x, \tau)$ is not differentiable. The same reasoning applies analogously when considering the differentiability or non-differentiability of h_{b} and h_X, and thus we omit the cases for these two functions for brevity.

(i) If $\phi_{\text{b}}(x, \tau)$ is differentiable w.r.t. x at $x = \bar{x}$, then $\partial_{\text{C}} \phi_{\text{b}}(x, \tau)$ reduces to the classical sensitivity $\frac{\partial \phi_{\text{b}}(x,\tau)}{\partial x}$ and therefore the feasibility of $\pi_{\text{b}}(x)$ follows from Lemma 5.3.1.

(ii) If $\phi_{\text{b}}(x, \tau)$ is not differentiable w.r.t. x at $x = \bar{x}$, noting Definition 5.3.3, consider any sequence $\{x_i\}_{i=0}^{\infty}$ converging to $\bar{x}$ and satisfying that $\phi_{\text{b}}(x, \tau)$ is differentiable w.r.t. x at x_i. It follows from the analysis in the first case that by letting $u_i = \pi_{\text{b}}(x_i)$, we have that

$$Dh_X\big(\phi_{\text{b}}(x_i, \tau)\big) \frac{\partial \phi_{\text{b}}(x_i, \tau)}{\partial x_i} f_{\text{PWA}}(x_i, u_i) \geq -\alpha\big(h_X\left(\phi_{\text{b}}(x_i, \tau)\right)\big)$$

holds. By taking the limit $i \to \infty$ and noting that $\lim_{i\to\infty} \frac{\partial \phi_{\text{b}}(x_i,\tau)}{\partial x_i} \in \partial_{\text{C}} \phi_{\text{b}}(\bar{x}, \tau)$ by Definition 5.3.3, we get the feasibility of (5.11b). An analogous argument can be applied to prove the feasibility of (5.11c). The proof of Theorem 5.4.1 is completed. □

To establish the forward invariance of $\Phi_{\text{const}}(S_{\text{b}}, T)$, i.e., the infinite-horizon safety of π_{safe}, we introduce the following key lemma, which is applicable to invariance analysis for general nonlinear systems involving any non-smooth CBF.

Lemma 5.4.1. Consider the system (5.1). Suppose that there exist a Lipschitz continuous controller $\pi(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$, a Lipschitz continuous $h(\cdot) : \mathbb{R}^n \to \mathbb{R}$, and a Lipschitz

continuous function $\alpha \in \mathcal{K}_\infty$ such that

$$\partial h(x) f(x, \pi(x)) \geq -\alpha(h(x)),\ \forall x \in \{x \in \mathbb{R}^n | h(x) \geq 0\} \text{ and } \forall \partial h(x) \in \partial_C h(x). \tag{5.12}$$

As a result, if $h(x_0) \geq 0$, we have $h(\phi(x_0, t)) \geq 0,\ \forall t \geq 0$, where $\phi(x_0, t)$ is the solution of the system (5.1) controlled by π, starting from x_0.

Proof. See Appendix 5.A.1. □

Remark 5.4.1. The result in Lemma 5.4.1 establishes the forward invariance of $\{x \in \mathbb{R}^n | h(x) \geq 0\}$. This result extends safety guarantees from continuously differentiable CBFs to CBFs that are merely Lipschitz continuous. More importantly, in contrast to the non-smooth CBF analysis in [85], we do not require h to be differentiable almost everywhere.

With the safety guarantees of non-smooth CBFs given in Lemma 5.4.1, we are ready to present the main result on the safety of π_{safe}.

Theorem 5.4.2 (Forward invariance)**.** Consider the continuous PWA system (5.9), the piecewise linear backup controller π_b, the Lipschitz continuous backup CBF h_b, and the proposed all-elements PSF (5.11). If Assumption 5.4.1 holds and the policy π_{safe} is locally Lipschitz continuous, then π_{safe} renders $\Phi_{\text{const}}(S_b, T)$ forward invariant.

Proof. Like [52], we consider the CBF candidate in (5.5), which is Lipschitz continuous and satisfies $\Phi_{\text{const}}(S_b, T) = \{x \in \mathbb{R}^n | h(x) \geq 0\}$ according to [52, Lemma 2].

For each $x \in \Phi_{\text{const}}(S_b, T)$, based on (5.5), we can distinguish two cases: (i) $h(x) = h_X\left(\phi_b(x(t), \tau')\right)$ with a certain $\tau' \in [0, T]$ and (ii) $h(x) = h_b\left(\phi_b(x(t), T)\right)$.

For the first case, we define the composite function $h_\phi = h_X \circ \phi_b$. Given $\tau' \in [0, T]$, we have the following expression for the Clarke derivative of the composite function [59, Theorem 2.3.9]:

$$\partial_C h_\phi(x, \tau') \subset \text{conv}\left\{\partial h_X\, Q,\ \partial h_X \in \partial_C h_X(\phi_b(x, \tau')),\ Q \in \partial_C \phi_b(x, \tau')\right\}$$

Then, we can apply the chain rule to obtain the Clarke derivative of the composite function h, that is, for any $\partial h \in \partial_C h(x)$,

$$\begin{aligned}
&\partial h f_{\text{PWA}}(x, \pi_{\text{safe}}(x)) + \alpha(h(x)) \\
=&\partial h_\phi f_{\text{PWA}}(x, \pi_{\text{safe}}(x)) + \alpha(h_X(x)),\ \exists \partial h_\phi \in \partial_C h_\phi(x, \tau') \\
=&\partial h_X Q f_{\text{PWA}}(x, \pi_{\text{safe}}(x)) + \alpha\left(h_X\left(\phi_b(x, \tau')\right)\right),\ \exists \partial h_X \in \partial_C h_X(\phi_b(x, \tau')) \text{ and } Q \in \partial_C \phi_b(x, \tau') \\
\geq& 0,
\end{aligned} \tag{5.13}$$

where the second equality is obtained by applying the chain rule to $\partial_C h_\phi$, and the last line follows from the fact that π_{safe} satisfies the constraint (5.11b).

An analogous argument to that in (5.13) can be applied for the second case, yielding $\partial h f_{\text{PWA}}(x, \pi_{\text{safe}}(x)) + \alpha_b(h(x)) \geq 0$ for any $\partial h \in \partial_C h(x)$.

As a consequence, the CBF candidate h satisfies (5.12) in Lemma 5.4.1. If π_{safe} is locally Lipschitz continuous, applying Lemma 5.4.1 yields $h(\phi_{\text{safe}}(x_0, t)) \geq 0,\ \forall t \geq 0$, if

$x_0 \in \Phi_{\text{const}}(S_b, T)$. Here, $\phi_{\text{safe}}(x_0, t)$ is the solution of the system (5.9) starting from x_0 controlled by π_{safe}. This proves the forward invariance of $\Phi_{\text{const}}(S_b, T)$ under π_{safe}. □

Theorems 5.4.1 and 5.4.2 state that when there are non-smooth CBF constraints, the all-elements PSF can ensure that the system always satisfies the constraints, provided that π_{safe} is locally Lipschitz continuous. Analyzing the conditions under which π_{safe} is Lipschitz continuous is left for future work.

5.5. NON-SMOOTH ANALYSIS OF PWA SYSTEMS

In the previous section, we have shown sufficient conditions to ensure formal guarantees for feasibility and safety for the proposed all-element PSF (5.11). However, to construct the proposed PSF, we need to compute the generalized Clarke derivative for ϕ_b (Clarke sensitivity) and the generalized Clarke derivatives $\partial_C h_X$ and $\partial_C h_b$. Computing $\partial_C h_X$ and $\partial_C h_b$ is generally feasible, since their explicit forms are usually known. Finding a generalized Clarke derivative for ϕ_b requires computing classical Jacobians for all converging sequences x_i where ϕ_b is differentiable, and it is generally intractable. This section addresses this challenge.

The following definition is useful for characterizing the behavior of PWA systems:
Definition 5.5.1 (Time-stamped switching sequence)**.** Consider the PWA system (5.10). Given the initial state x_0, the *time-stamped switching sequence* of the solution $\phi_b(x_0, \tau)$, denoted as

$$\Gamma(x_0) = \{(\tau_0, I_0), (\tau_1, I_1), ..., (\tau_M, I_M)\}, \text{ with } I_k \subseteq \mathbb{N}_r^+, \forall k \in \mathbb{N}_M, \tag{5.14}$$

is the ordered sequence of time instants and the indices of regions such that

- $\tau_0 = 0$ and $x_0 \in \text{clo}(\mathcal{R}_{I_0})$;
- $\phi_b(x_0, \tau) \in \text{clo}(\mathcal{R}_i)$, $\forall i \in I_k$ and $\forall \tau \in [\tau_k, \tau_{k+1})$;
- $I_k \neq I_{k+1}$, $\forall k \in \mathbb{N}_{M-1}$ and $\tau_0 < \tau_1 < ... < \tau_M$.

In (5.14), r is the number of polyhedral regions of (5.10) and M is the number of switching times.

The switching sequence can be infinite, i.e., $M = \infty$, meaning that the system exhibits zeno behavior if τ_M is finite or chattering if $\tau_M = \infty$. Besides, each I_k can contain more than one element. In this case, the system operates on the shared boundary of multiple regions of (5.10) during $[\tau_k, \tau_{k+1})$. If I_k consists of multiple elements, then we define $\mathcal{R}_{I_k} := \bigcup_{i \in I_k} \mathcal{R}_i$.

5.5.1. AUMANN SENSITIVITY FOR PWA SYSTEMS

We propose an alternative and computable generalized sensitivity $Q_A(x_0, \tau)$ using the Aumann integral [11]. In subsequent subsections, we will show that, under certain assumptions, this formulation is equivalent to the Clarke sensitivity.
Definition 5.5.2 (Aumann sensitivity)**.** Consider the PWA system (5.10) and its solution ϕ_b. The *Aumann sensitivity* for any initial state x_0 at time τ, denoted by $Q_A(x_0, \tau)$, is

implicitly defined by

$$Q_{\mathrm{A}}(x_0,\tau) := \Big\{ Q(x_0,\tau) \Big| Q(x_0,\tau) = I_n + \int_0^\tau \partial f_{\mathrm{PWA,b}}(\phi_{\mathrm{b}}(x_0,s)) Q(x_0,s)\,ds, \\ \partial f_{\mathrm{PWA,b}}(x) \in \{D_i | i \text{ s.t. } x \in \mathrm{clo}(\mathcal{R}_i)\} \Big\}. \tag{5.15}$$

The Aumann integral is introduced to handle cases where the integration is carried out on all possible values of the integrand. As a result, the Aumann sensitivity naturally becomes a set-valued function. When this sensitivity takes a single value at a point, it coincides with the classical Jacobian of ϕ_{b} at that point.

Given the initial state x_0, denote by $I(x_0)$ the Cartesian product of the switching sequence $\{I_0, I_1, ..., I_M\}$. With this definition, the Aumann sensitivity is expressed as:

$$Q_{\mathrm{A}}(x_0,\tau) = \left\{ Q(x_0,\tau) \middle| Q(x_0,\tau) = I + \sum_{k=0}^{\bar{k}} \int_{\tau_k}^{\min\{\tau,\tau_{k+1}\}} D_{i_k} Q(x_0,s)\,ds, (i_0,i_1,...,i_M) \in I(x_0) \right\} \\ \tau \in [\tau_{\bar{k}}, \tau_{\bar{k}+1}), \bar{k} \le M-1. \tag{5.16}$$

Each Q is single-valued and can be obtained by the standard sensitivity computation [118, Chapter 3.3]. The cardinality of $Q_{\mathrm{A}}(x_0,\tau)$ is less than or equal to $\Pi_{k\in\mathbb{N}_M}|I_k|$.

5.5.2. PROPERTIES OF THE AUMANN SENSITIVITY

The ultimate objective of this section is to show the equivalence between the Aumann sensitivity and the Clarke sensitivity. Before doing that, we first study the properties of the Aumann sensitivity.

Without loss of generality, suppose that the closure of each $\mathcal{R}_i$ admits the hyperplane representation $\{x \in \mathbb{R}^n | H_i x + h_i \le 0\}$, where $H_i \in \mathbb{R}^{m_i \times n}$, $h_i \in \mathbb{R}^{m_i}$, and m_i is the number of inequalities that constitute $\mathcal{R}_i$. Denote by

$$\Gamma_I := \{x \in \mathbb{R}^n | \bar{g}_I(x) := g_I x + b_I = 0\}$$

the common boundary of two or more adjacent regions $\mathcal{R}_I$, where I is a set of indices of the adjacent regions. Here, $g_I \in \mathbb{R}^{n_{g_I} \times n}$, $b_I \in \mathbb{R}^{n_{g_I}}$, and $\bar{g}_I(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_{g_I}}$.

From (5.15) we know that Q_{A} could take multiple values if the solution ϕ_{b} stays on the common boundary of some regions of (5.10) during a period. This is because during that period, the integral term can take multiple values. This motivates us to check if there is a non-empty set on the boundary such that the solution ϕ_{b} belongs to this set during a certain period. We consider the following *critical set* (could be empty):

$$\mathcal{C} := \bigcup_{I\in\mathscr{I}} \mathcal{C}_I, \text{ where } \mathcal{C}_I = \bigcap_{i\in I} \mathrm{clo}(\mathcal{R}_i) \text{ and} \\ \mathscr{I} := \{I \subseteq \mathbb{N}_r^+ | \{\mathcal{R}_i\}_{i\in I} \text{ are adjacent}, \bar{g}_I^{(q)} = 0,\ \forall q \in \mathbb{N}_n, \text{ and } \exists i,j \in I \text{ s.t. } D_i \ne D_j\}. \tag{5.17}$$

In (5.17), $\bar{g}_I^{(q)}$ is the q-th order element-wise time derivative of $\bar{g}_I$ along the PWA system (5.10). Because the first q order time derivatives of $\bar{g}_I$ are zero, $\bar{g}_I$ remains constant

according to the Cayley–Hamilton Theorem [74]. Therefore, we can interpret $\mathcal{C}$ as the set where the trajectory is possible to stay on a boundary $\bar{g}_I(x) = 0$ of some polyhedra $\mathcal{R}_i$, $i \in I$ for some periods.

For any initial state x_0, define the *forward reachable set* as $\Phi_{\text{for}}(x_0) := \bigcup_{\tau\in[0,\infty)}\{\phi_{\text{b}}(x_0,\tau)\}$. Similarly, for any set S of states, define the *backward reachable set* $\Phi_{\text{back}}(S) := \{x \in \mathbb{R}^n | \exists \tau \in [0,\infty) \text{ s.t. } \phi_{\text{b}}(x,\tau) \in S\}$. We have the following result on the relation among $\mathcal{C}$, Q_{A}, and the classical Jacobian (if it exists).

Proposition 5.5.1. The following statements are equivalent:

(1) The classical Jacobian $\frac{\partial \phi_{\text{b}}(x_0,\tau)}{\partial x_0}$ of $\phi_{\text{b}}(x_0,\tau)$ exists at $x_0 = \bar{x}_0$ for all $\tau \geq 0$.

(2) The Aumann sensitivity $Q_{\text{A}}(x_0,\tau)$ is single-valued at $x_0 = \bar{x}_0$ for all $\tau \geq 0$.

(3) $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} = \emptyset$.

(4) $\bar{x}_0 \notin \Phi_{\text{back}}(\mathcal{C})$.

As a consequence, if $\mathcal{C}$ is empty, the Aumann sensitivity Q_{A} defined in (5.15) is single-valued for any x_0, which implies the global existence of the classical Jacobian $\frac{\partial \phi_{\text{b}}(x_0,\tau)}{\partial x_0}$.

5

Proof. See Appendix 5.A.2. □

The critical set $\mathcal{C}$ is a union of some polyhedra $\mathcal{C}_I$, $I \in \mathscr{I}$, with the dimension strictly lower than n. By using the Lie derivative for (5.17), each $\mathcal{C}_I$, $I \in \mathscr{I}$ can be expressed as

$$\mathcal{C}_I = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} g_I x + b_I = 0 \\ g_I(D_i x + d_i) = 0 \\ g_I D_i (D_i x + d_i) = 0 \\ \vdots \\ g_I D_i^{n-1}(D_i x + d_i) = 0 \\ H_i x + h_i \leq 0 \end{array} \right\} \forall i \in I \quad \text{and } \exists i, j \in I \text{ s.t. } D_i \neq D_j \right\} \tag{5.18}$$

The non-emptiness of each $\mathcal{C}_I$ can be determined by checking the feasibility of a linear program.

Remark 5.5.1. Proposition 5.5.1 implies that the sensitivity Q_{A} is uniquely defined as long as the state does not remain on a cell boundary for any time interval, but always "passes through". An important fact indicated by Proposition 5.5.1 is that the non-existence of $\frac{\partial \phi_{\text{b}}(x_0,\tau)}{\partial x_0}$ does not necessarily occur on the boundary of the PWA system; rather, it can also take place in the interior of any polyhedron $\mathcal{R}_i$. The following example illustrates the set $\mathcal{C}$ and also the set of initial states at which $\frac{\partial \phi_{\text{b}}(x_0,\tau)}{\partial x_0}$ fails to exist.

Example 1. Consider a two-dimensional PWA system illustrated in Fig. 5.1. The system is partitioned by three lines: $x_1 = 0$, $x_1 = 2$, and $x_2 = 0$, resulting in six polyhedra. The critical set $\mathcal{C}$, according to (5.18), consists of the positive segment of the x_1-axis. For

every initial point along the red and blue lines, when the trajectory ϕ_b reaches the right-half plane, it lacks a well-defined Jacobian. To prove this, we first note that for any initial state $\bar{x}_0$ starting from the red line ($\mathcal{C}$), the solution will always remain in $\mathcal{C}$. Therefore, $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} \neq \emptyset$, and $\frac{\partial \phi_b(x_0,\tau)}{\partial x_0}$ does not exist, according to Proposition 5.5.1. Then, we analyze the initial states in the first quadrant. By solving the affine system in this quadrant, we get the evolution: $x_1(\tau) = x_1(0) + \tau$, $x_2(\tau) = (x_1(0) + x_2(0) + 1)e^{\tau} - \tau - x_1(0) - 1$, for $\tau \geq 0$ such that $x_1(\tau) \leq 0$ and $x_2(\tau) \geq 0$. If $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} \neq \emptyset$, at a certain time instant τ_{k_1} it must hold that $x_1(\tau_{k_1}) = x_2(\tau_{k_1}) = 0$. Otherwise, the trajectory can only converge to the red line asymptotically. From $x_1(\tau_{k_1}) = x_2(\tau_{k_1}) = 0$, we can derive the relation:

$$x_2(0) = e^{x_1(0)} - x_1(0) - 1.$$

The blue line corresponds to the above function in the first quadrant.

By simulating the system starting from the points near the blue line, we can find that even a small perturbation on the initial state will result in a large gap when the trajectory goes into the right-half plane. This validates that the classical Jacobian does not exist at the initial states on the blue curve after the critical time when the trajectory enters the red line.

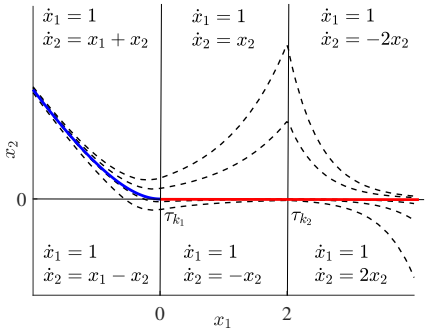


Figure 5.1: The evolution of the PWA system given in Example 1. The black solid lines refer to the boundaries of the polyhedra. The critical set $\mathcal{C}$ is the positive part of the x_1-axis, represented by the red line. The backward reachable set $\Phi_{\text{back}}(\mathcal{C})$ is the union of the red and blue lines. According to Proposition 5.5.1, the solution ϕ_b is not differentiable at all points on the red and blue lines. The black dashed curves represent some trajectories starting near the blue line.

The sensitivity branches at the time τ when the solution $\phi_b(x_0,\tau)$ enters the critical set $\mathcal{C}$. Fig. 5.2 gives a geometric interpretation of $Q_A(x_0,\tau)$.

5.5.3. EQUIVALENCE BETWEEN conv(Q_A) AND $\partial_C \phi_b$

In this subsection, we show that $\text{conv}(Q_A(x_0,\tau)) = \partial_C \phi_b(x_0,\tau)$, $\forall x_0 \in \mathbb{R}^n$ and $\forall \tau \in [0,\infty)$. Toward this end, the following lemma and assumption are useful.

Lemma 5.5.1. Consider the PWA system (5.10) and its Aumann sensitivity Q_A defined in (5.15). For any $x_0 \in \mathbb{R}^n$, any $\tau \in [0,\infty)$, and any $Q(x_0,\tau) \in Q_A(x_0,\tau)$, $Q(x_0,\tau)$ is invertible.

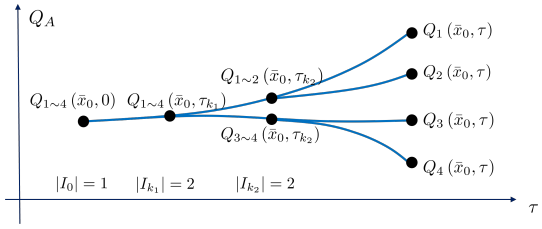


Figure 5.2: Evolution of the sensitivity function Q_{A} over time for any initial state at the blue line in Fig. 5.1. The components Q_1 to Q_4 represent individual elements of Q_{A}. The sensitivity branches when the solution $\phi_{\mathrm{b}}(\bar{x}_0,\tau)$ enters each critical set. In Example 1, this occurs at τ_{k_1} when the trajectory reaches $(0,0)$ and at τ_{k_2} when the trajectory reaches $(2,0)$.

Proof. According to (5.16), any $Q(x_0,\tau)$ can be recursively computed by

$$Q(x_0,t)=\begin{cases} e^{D_{i_0}\tau},\tau\in[\tau_0,\tau_1) \\ e^{D_{i_1}(\tau-\tau_1)}Q(x_0,\tau_1),t\in[\tau_1,\tau_2) \\ \dots \\ e^{D_{i_{M-1}}(\tau-\tau_{M-1})}Q(x_0,\tau_{M-1}),\tau\in[\tau_{M-1},\tau_M), \end{cases} \tag{5.19}$$

where $\{i_0,i_1,\dots,i_M\}\in I(x_0)$ and $I(x_0)$ is the Cartesian product of the switching sequence $\{I_0,I_1,\dots,I_M\}$ for given x_0.

As $e^{D_{i_0}\tau}$ is always invertible, and AB is invertible for any two invertible matrices A and B, it is straightforward to prove by induction that $Q(x_0,\tau)$ is invertible for any $x_0\in\mathbb{R}^n$ and $\tau\in[0,\infty)$. □

Assumption 5.5.1. For any $x_0\in\Phi_{\mathrm{back}}(\mathcal{C})$, any $\tau\in[0,\infty)$, and any $Q(x_0,\tau)\in \mathrm{conv}(Q_{\mathrm{A}}(x_0,\tau))$, $Q(x_0,\tau)$ is invertible.

Remark 5.5.2. Since any $Q\in Q_{\mathrm{A}}$ is invertible according to Lemma 5.5.1, we further assume that any element in the convex hull of Q_{A} is invertible. Although convex combinations may not preserve invertibility, Assumption 5.5.1 is typically satisfied in practical scenarios such as adaptive cruise control with PWA models in [62] and the inverted pendulum system considered in the case study of this chapter. This assumption is crucial in the subsequent analysis on the equivalence between $\mathrm{conv}(Q_{\mathrm{A}})$ and $\partial_{\mathrm{C}}\phi_{\mathrm{b}}$, as it enables the application of the inverse function theorem [57, Theorem 1] for the non-smooth solution ϕ_{b}.

Proposition 5.5.2. Consider the PWA system (5.10) and its two generalized sensitivities Q_{A} and $\partial_{\mathrm{C}}\phi_{\mathrm{b}}$. Suppose Assumption 5.5.1 holds. We have $\mathrm{conv}(Q_{\mathrm{A}}(x_0,\tau))=\partial_{\mathrm{C}}\phi_{\mathrm{b}}(x_0,\tau)$, $\forall x_0\in\mathbb{R}^n$ and $\forall\tau\in[0,\infty)$.

Proof. See Appendix 5.A.3. □

Proposition 5.5.2 indicates that the convex hull of the Aumann sensitivity, computed via (5.16) can replace the Clarke sensitivity in the formulation of the proposed PSF (5.11).

5.5.4. IMPLEMENTATION OF THE ALL-ELEMENTS PSF

Problem (5.11) is a semi-infinite programming problem [31, Chapter 5.4], i.e., it contains infinitely many constraints. To solve it, we adopt the following three steps.

Step 1: Transforming the constraints. Under the equivalence between $\partial_C \phi_b$ and $\mathrm{conv}(Q_A)$, we replace $\partial_C \phi_b$ with $\mathrm{conv}(Q_A)$ in the formulation (5.11) of the proposed all-elements PSF. In other words, in (5.11b) and (5.11c), the CBF constraints should be satisfied for any $Q \in \mathrm{conv}(Q_A)$.

We first eliminate the infinite number of constraints stemming from the set-valued generalized Clarke derivatives. Since (i) the CBF constraints are linear in h_X, h_b, and Q, and (ii) the uncertainty sets $\partial_C h_X$, $\partial_C h_b$, and $\mathrm{conv}(Q_A)$ are convex, according to [23, Proposition 2.1], (5.11b) and (5.11c) are equivalent to

$$\begin{aligned} &\partial h_X Q f_{\mathrm{PWA}}(x,u) \geq -\alpha\left(h_X\left(\phi_b(x,\tau)\right)\right),\ \forall \partial h_X \in J_{h_X}(\phi_b(x,\tau)),\ \forall Q \in Q_A(x,\tau),\ \forall \tau \in [0,T], \\ &\partial h_b Q f_{\mathrm{PWA}}(x,u) \geq -\alpha_b\left(h_b\left(\phi_b(x,T)\right)\right),\ \forall \partial h_b \in J_{h_b}(\phi_b(x,T)),\ \forall Q \in Q_A(x,T), \end{aligned} \tag{5.20}$$

where

$$J_{h_X}(\phi_b(x,\tau)) = \left\{ \lim_{i\to\infty} Dh_X(y_i) \,\middle|\, y_i \to \phi_b(x,\tau),\ h_X \text{ is differentiable at } y_i \right\}$$

and J_{h_b} is defined similarly to J_{h_X}. According to Definition 5.3.3, the set-valued functions J_{h_b} and J_{h_X} denote the collections of limiting derivatives that generate $\partial_C h_b$ and $\partial_C h_X$, respectively, i.e., $\partial_C h_b(\phi_b(x,T)) = \mathrm{conv}(J_{h_b}(\phi_b(x,T)))$ and $\partial_C h_X(\phi_b(x,\tau)) = \mathrm{conv}(J_{h_X}(\phi_b(x,\tau)))$.

Note that J_{h_b} and J_{h_X} are in general computable, as the explicit expressions of h_b and h_X are usually known. Besides, the Aumann sensitivity Q_A is computed through (5.16). Note that $|J_{h_b}|$, $|J_{h_X}|$, and $|Q_A|$ are finite.

Step 2: Mixed-integer encoding of f_{PWA}. Mixed-integer encoding is commonly used in hybrid MPC in which the PWA prediction model is usually converted into an equivalent mixed-logical dynamical form [35].

If the partition of the PWA system (5.9) is only in the state space, i.e., $f_{\mathrm{PWA}}(x,u) := A_i x + B_i u + c_i$, for $x \in \mathcal{P}_i$, the inequalities in (5.20) are naturally linear in u and hence there is no need to do mixed-integer encoding.

If the partition is in both state and input spaces, to make the inequalities in (5.20) linear in u, we follow [21] to use the Big-M approach to get the equivalent mixed-integer

formulation of $f_{\text{PWA}}(x,u)$, which is given by

$$f_{\text{PWA}}(x,u) = \sum_{i=1}^{p} f_i,\ \sum_{i=1}^{p} \Delta_i = 1$$
$$\text{for } i = 1,2,...,p: \begin{cases} f_i \le \overline{M}\Delta_i \\ f_i \ge \underline{M}\Delta_i \\ f_i \le A_i x + B_i u + c_i - \underline{M}(1-\Delta_i) \\ f_i \ge A_i x + B_i u + c_i - \overline{M}(1-\Delta_i) \\ \Psi_i^{(x)} x + \Psi_i^{(u)} u \le \psi_i + \overline{M}_i^*(1-\Delta_i), \end{cases} \tag{5.21}$$

where the matrices $\Psi_i^{(x)}$, $\Psi_i^{(u)}$, and ψ_i constitute the hyperplane representation $\{[x^T,\ u^T]^T \in \mathbb{R}^{n+m} | \Psi_i^{(x)} x + \Psi_i^{(u)} u \le \psi_i\}$ of the polyhedral region $\mathcal{P}_i$ in (5.9), $\overline{M}$, $\underline{M}$, and $\overline{M}_i^*$ are some bounds of affine functions $A_i x + B_i u + c_i$ and $\Psi_i^{(x)} x + \Psi_i^{(u)} u - \psi_i$ over $X \times U$. The detailed expressions of these bounds are provided in [21]. Besides, $\Delta_i \in \{0,1\}, i = 1,2,\ldots,p$ are binary variables.

Step 3: Discretizing the time τ. After Steps 1 and 2, the infinite number of constraints in (5.20) comes only from the continuous time set $[0,T]$. In practice, to obtain computational tractability, the constraints are usually relaxed into a finite number of constraints that should be satisfied at some time instants [151]. In particular, given the prediction horizon $T > 0$ and a number $N \in \mathbb{N}^+$, the time set $[0,T]$ is discretized into $\{\tau_{(l)}\}_{l=0}^{N}$, where each $\tau_{(l)} = lT/N$. As a result, (5.20) is relaxed into an inequality constraint that should be satisfied at finitely many $\tau_{(l)}$.

After applying the three steps above, problem (5.11) becomes a convex QP if the PWA system (5.9) is partitioned only over the state space. If the partition is defined over the joint state-input space, the problem (5.11) typically becomes an MIQP.

5.6. EXPLICIT APPROXIMATION OF THE SAFE CONTROLLER

In the previous section, we have shown that the intractable constraints in (5.11) can be reformulated using Aumann sensitivity. However, when the partition of the PWA system (5.9) is defined over the joint state-input space, the computation time for solving the MIQP derived from (5.11) in practice grows dramatically with the number of binary variables in the worst case [169]. The number of binary variables itself scales linearly with the number p of regions. Consequently, the proposed all-elements PSF becomes impractical for real-time control systems with limited computational resources. To address this challenge (Problem 3), we further derive an explicit approximation for the solution to (5.11). This explicit solution is guaranteed to be feasible w.r.t. (5.11) and can be computed efficiently.

Our design is based on the optimization-free safety filter proposed in Chapter 4. An important property of (5.11) we use is that the backup solution $\pi_b(x)$ is a feasible solution to (5.11) for any $x \in \Phi_{\text{const}}(S_b, T)$, as has been proved in Theorem 5.4.1.

For each x, define by $U_{(5.11)}(x)$ the feasible set of (5.11). First, we restrict the constraint

on u from u belonging to the whole feasible set $U_{(5.11)}(x)$ to

$$u \in U_{(5.11)}(x) \cap U_{\text{line}}(x), \tag{5.22}$$

where $U_{\text{line}}(x)$ is the line segment with the endpoints $\pi_{\text{b}}(x)$ and $\pi_{\text{r}}(x)$. Mathematically, $U_{\text{line}}(x) = \{u \in \mathbb{R}^m | u = \pi_{\text{b}}(x) + \lambda(\pi_{\text{r}}(x) - \pi_{\text{b}}(x)),\ \lambda \in [0,1]\}$.

By adding the constraint $u \in U_{\text{line}}(x)$ into the optimization problem (5.11), we obtain that the optimizer $\hat{\pi}_{\text{safe}}(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ of the modified problem takes the following form:

$$\hat{\pi}_{\text{safe}}(x) = \pi_{\text{b}}(x) + \lambda^*(x)(\pi_{\text{r}}(x) - \pi_{\text{b}}(x)), \tag{5.23}$$

where

$$\begin{aligned} \lambda^*(x) = \max_{\lambda \in [0,1]}\ & \lambda \\ \text{s.t. } & u = \pi_{\text{b}}(x) + \lambda(\pi_{\text{r}}(x) - \pi_{\text{b}}(x)),\ u \in U_{(5.11)}(x). \end{aligned} \tag{5.24}$$

Since each polyhedral region $\mathcal{P}_i$ has the hyperplane representation $\{[x^T,\ u^T]^T \in \mathbb{R}^{n+m} | \Psi_i^{(x)} x + \Psi_i^{(u)} u \le \psi_i\}$, for any state x, we can determine the slice $U_i(x)$ of $\mathcal{P}_i$, defined by

$$U_i(x) := \left\{ u \in \mathbb{R}^m | \Psi_i^{(u)} u \le \psi_i - \Psi_i^{(x)} x \right\}. \tag{5.25}$$

Let $I_u(x)$ denote the set of indices i for which $U_i(x)$ is not empty. By substituting the expression of f_{PWA} into (5.11) and applying Step 1 of Section 5.5.4, we can rewrite the optimizer λ^* in a more compact form:

$$\begin{aligned} \lambda^*(x) = \max_{\lambda}\ & \lambda \\ \text{s.t. } & \exists i \in I_u(x) \text{ s.t. } \eta_i(x,\tau)\lambda \le \omega_i(x,\tau),\ \forall \tau \in [0,T], \end{aligned} \tag{5.26}$$

where η_i and ω_i are vector-valued functions with proper dimensions. The expressions of η_i and ω_i can be obtained based on (5.9), (5.20), and (5.24). To derive the explicit form of λ^*, the constraint $\lambda \in [0,1]$ is incorporated in (5.26) with the form of $[1\ \ -1]^T \lambda \le [1\ \ 0]^T$.

The following proposition gives the explicit form of the optimizer λ^*.

Proposition 5.6.1. Consider the optimization problem (5.24) or (5.26), the optimizer λ^* has the expression:

$$\lambda^*(x) = \max_{i \in I_u(x)} \min_{\tau \in [0,T]} \min_{j \in \mathbb{N}^+_{\dim(\eta_i)}} \lambda_{i,j}(x,\tau), \tag{5.27}$$

with

$$\lambda_{i,j}(x,\tau) = \begin{cases} \dfrac{\omega_{i,j}(x,\tau)}{\eta_{i,j}(x,\tau)}, & \text{if } \eta_{i,j} > 0 \\ \text{step}(\omega_{i,j}(x,\tau)), & \text{if } \eta_{i,j} = 0 \\ \text{step}\left(\omega_{i,j}(x,\tau) - \eta_{i,j}(x,\tau) \min\limits_{\lambda_{i,j'}(x,\tau') \in (0,1)} \lambda_{i,j'}(x,\tau')\right), & \text{if } \eta_{i,j} < 0, \end{cases} \tag{5.28}$$

where $\omega_{i,j}$ and $\eta_{i,j}$ represent the j-th element of ω_i and η_i, respectively.

Proof. See Appendix 5.A.4. □

Since $\hat{\pi}_{\text{safe}}$ is a feasible solution to the proposed all-elements PSF (5.11), we immediately have the following corollary of Theorem 5.4.2 and Proposition 5.6.1 on the safety performance of $\hat{\pi}_{\text{safe}}$:

Corollary 5.6.1. Consider the continuous PWA system (5.9) and the proposed all-elements PSF (5.11). Suppose that Assumptions 5.4.1 and 5.5.1 hold and that the approximate policy $\hat{\pi}_{\text{safe}}$ in (5.23) with λ^* computed from (5.27) is locally Lipschitz continuous. Then, the approximate policy $\hat{\pi}_{\text{safe}}$ renders $\Phi_{\text{const}}(S_{\text{b}}, T)$ forward invariant.

Akin to Step 3 of Section 5.5.4, implementing the approximate policy (5.23) needs the discretization of the time set $[0, T]$ appearing in (5.27).

Remark 5.6.1. Unlike [194], which computes explicit approximations for convex optimization problems, our approach accommodates non-convex MIQPs. Compared to the optimization-free safety filter in Chapter 4, which also finds explicit approximations for MIQP problems, we extend the framework to problems with infinitely many constraints.

5.7. CASE STUDIES

5.7.1. INVERTED PENDULUM

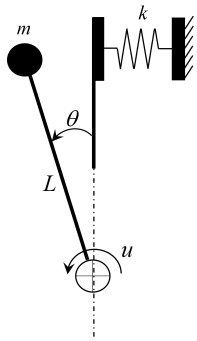


Figure 5.3: Diagram of the inverted pendulum interacting with an elastic wall.

To demonstrate the theoretical results, control of a simple inverted pendulum that interacts with an elastic wall (illustrated in Fig. 5.3) is considered. The system is linearized around the upright position, resulting in the following PWA system:

$$\ddot{\theta} = \begin{cases} \frac{g}{L}\theta + \frac{1}{mL^2}u, & \text{if } \theta \geq 0 \\ (\frac{g}{L} - kL)\theta + \frac{1}{mL^2}u, & \text{if } \theta < 0, \end{cases}$$

where θ is the pendulum angle, u is the input torque, the state variable $x = [\theta \ \dot{\theta}]^T$, $m = 1$ kg, $L = 1$ m, $g = 10$ m/s^2, and $k = 2$ N/m is the spring of the wall. The state constraint is $x \in X := \{x \mid |\theta| \leq 0.5, \ |\dot{\theta}| \leq 2\}$, and the input constraint is $|u| \leq 10$. In the simulation, the system is discretized by forward Euler with a sampling time of 0.001 seconds.

The backup policy is designed as a linear feedback controller, given by $\pi_{\mathrm{b}}(x) = -12\theta - 3\dot{\theta}$. Under this policy, the resulting PWA system takes the form:

$$\dot{x} = f_{\mathrm{PWA,b}}(x) = \begin{cases} \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} x, & \text{if } x_1 \geq 0 \\ \begin{bmatrix} 0 & 1 \\ -4 & -3 \end{bmatrix} x, & \text{if } x_1 < 0. \end{cases} \tag{5.29}$$

The feedback gain $[-12 \ -3]$ is selected such that both linear modes of (5.29) are Hurwitz stable.

According to Proposition 5.5.1, $x = [0 \ 0]^T$ is the only initial point at which the Aumann sensitivity contains multiple (two in this case) elements. Under the backup policy, a quadratic CBF $h_{\mathrm{b}}(x) = \gamma - x^T R x$ is synthesized using the LMI approach presented in [197]. The reference control policy is chosen as $\pi_{\mathrm{r}}(x) = -12(\theta - \theta_{\mathrm{r}}) - 3(\dot{\theta} - \dot{\theta}_{\mathrm{r}}) + mL^2\ddot{\theta}_{\mathrm{r}} - mgL\theta_{\mathrm{r}}$, which is a proportional-derivative plus feedforward controller. The reference angle $\theta_{\mathrm{r}}(t) = 0.8\cos(t)$. This means that during some periods, the reference state violates the state constraint. For the PSF in (5.11), the horizon T is set to 1 second and $[0, \ T]$ is further discretized into $N = 50$ intervals, resulting in the time grid $\{lT/N\}_{l=0}^{N}$.

Fig. 5.4 compares the time responses of the closed-loop system under the PSF (5.11) and the standard safety filter using the CBF h_{b} without predictions. The initial state is $[0.1 \ 0.1]^T$. From Fig. 5.4, it is observed that even though the reference state violates the state constraint, the trajectories controlled by these two safety filters always satisfy the state constraint. As evidenced by the first and third sub-figures, the PSF is less conservative because the corresponding trajectory stays closer to the reference and the value of h_{b} is negative during some periods. Furthermore, Fig. 5.5 displays the trajectories and also the safe set of h_{b}. It is found that the feasible region of the PSF exceeds the boundary of the safe set of h_{b} (i.e., the feasible region of the standard safety filter), highlighting the superiority of the predictive approach over the standard method.

Next, we examine a scenario in which the PSF based on the classical gradient (5.7) fails to ensure safety. Consider the initial state $x(0) = [0.5 \ \ -2]^T$, which lies on the boundary of X. The function h_X, defined as $h_X(x) := \min\{0.5 - \theta, \ \theta + 0.5, \ 2 - \dot{\theta}, \ \dot{\theta} + 2\}$, is not differentiable at $x(0)$. We design two PSFs: the first is based on (5.11), and the second is a modified version of (5.11) that includes only a single, randomly selected gradient from $\partial_{\mathrm{C}} h_X$. The second one corresponds to the classical PSF, which implicitly assumes that h_X is differentiable everywhere. To highlight the difference between the two safety filters, a constant reference policy is chosen as $\pi_{\mathrm{r}}(x) = -10$.

Fig. 5.6 illustrates the closed-loop trajectory of $\dot{\theta}$ under both safety filters. It is evident in the zoomed figure that the proposed all-elements PSF using the generalized Clarke derivative (blue curve) drives the system to remain within the safe region, while the safety filter using the single gradient (purple curve), proposed in [52], [93], experiences constraint violations during the initial phase[5].

[5] When the filter becomes infeasible, we switch to applying the reference π_{r} through a saturation unit $\max\{\min\{\cdot, \ 10\}, \ -10\}$, which enforces only the input constraint.

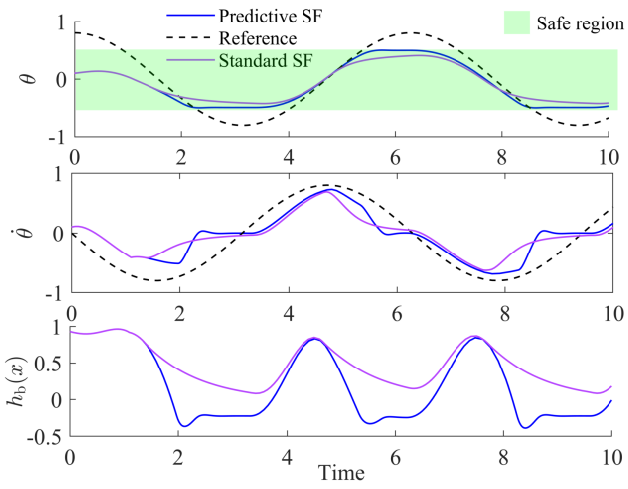


Figure 5.4: Time responses of the closed-loop system with the PSF (5.11) and the standard safety filter. The initial state is $[0.1\ \ 0.1]^T$. "SF" means "safety filter". Both SFs successfully control the system without recording any constraint violations. The predictive SF is less conservative than the standard SF because the trajectory regulated by the predictive SF is closer to the reference.

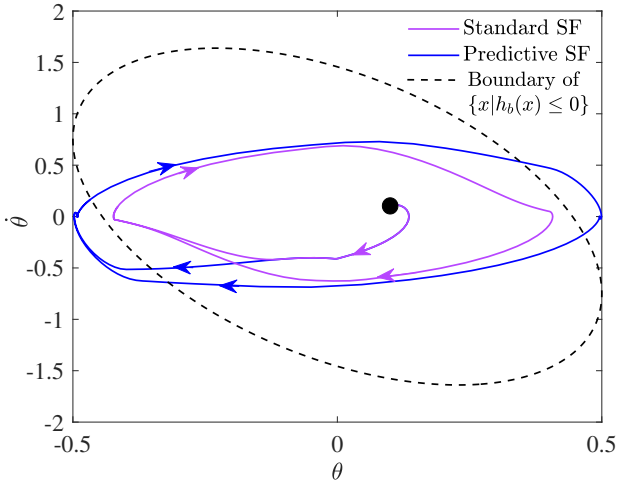


Figure 5.5: Trajectories of the closed-loop system (solid lines) and the safe set of h_b (black dashed line). The black hole represents the initial state. By virtue of predictions, the predictive SF enlarges the original feasible region $\{x|h_b(x) \leq 0\}$.

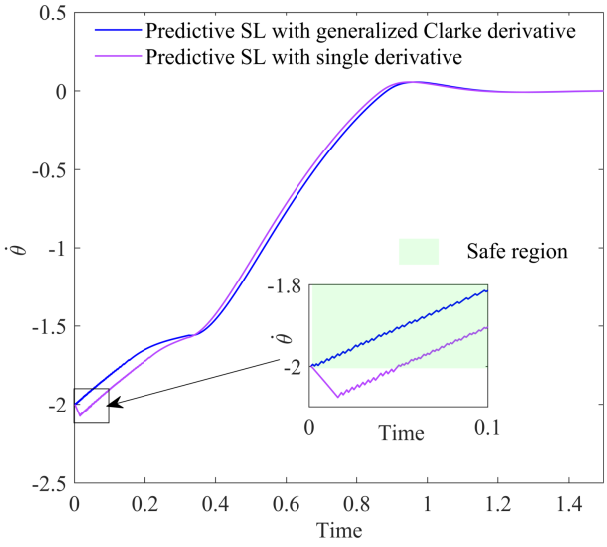


Figure 5.6: Comparison of PSFs using generalized Clarke derivative and single derivative methods. The highlighted inset zooms into the initial phase of the trajectory, illustrating the difference in early behavior between the two methods. When the initial state is at the boundary of the state constraint set, the proposed all-elements PSF using generalized Clarke derivative successfully avoids the unsafe region, while the classical PSF proposed in [52], [93] fails.

5.7.2. TEMPERATURE CONTROL

To demonstrate the computational advantage of the proposed explicit solution, we further consider a more complex PWA system with more partitions. In particular, let us consider control of the temperatures $x = [T_1\ T_2\ T_3\ T_4]^T$ of four rooms in a building. The layout of the four rooms and heat flows are shown in Fig. 5.7.

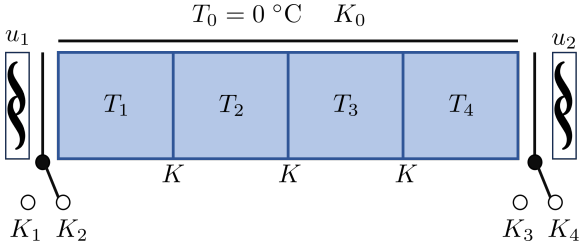


Figure 5.7: Diagram of the building.

The ambient temperature T_0 outside the building is assumed to be constantly zero, i.e., $T_0 = 0\ ^\circ\mathrm{C}$. Heat flows occur between each two adjacent rooms as well as between the external environment and each room. These thermal interactions are modeled using linear equations. For example, the temperature dynamics in Room 2 are given by $\dot{T}_2 = K(T_1 - T_2) + K(T_3 - T2) + K_0(T_0 - T_2)$, where the coefficients K and K_0 are the ratios of the corresponding thermal capacitance to thermal resistance. Room 1 and Room 4 are

each equipped with a temperature controller (u_1 and u_2) to regulate the temperature in the respective room. The heat flow between a room and its controller is modeled by a piecewise linear equation. For instance, when the temperature difference $|u_1 - T_1|$ exceeds a threshold of 5 °C, the system automatically switches to a mode with a larger coefficient $K_2 > K_1$ to facilitate better heat conduction. To summarize, the system admits the following state space equation:

$$\begin{aligned}
\dot{T}_1 &= \begin{cases} K(T_2-T_1)+K_1(u_1-T_1)+K_0(T_0-T_1), \text{if } |u_1-T_1| \le 5 \\ K(T_2-T_1)+K_2(u_1-T_1)+K_0(T_0-T_1), \text{otherwise} \end{cases} \\
\dot{T}_2 &= K(T_1-T_2)+K(T_3-T_2)+K_0(T_0-T_2) \\
\dot{T}_3 &= K(T_2-T_3)+K(T_4-T_3)+K_0(T_0-T_3) \\
\dot{T}_4 &= \begin{cases} K(T_3-T_4)+K_3(u_2-T_4)+K_0(T_0-T_4), \text{if } |u_2-T_4| \le 5 \\ K(T_3-T_4)+K_4(u_2-T_4)+K_0(T_0-T_4), \text{otherwise,} \end{cases}
\end{aligned}$$

where $K = 0.0035\ \text{s}^{-1}$, $K_0 = 0.001\ \text{s}^{-1}$, $K_1 = 0.01\ \text{s}^{-1}$, $K_2 = 0.02\ \text{s}^{-1}$, $K_3 = 0.008\ \text{s}^{-1}$, and $K_4 = 0.016\ \text{s}^{-1}$. This PWA system can be rewritten as the standard form of (5.9), with a total of 9 polyhedral regions.

The control objective is to adjust the temperatures of Room 2 and Room 3 to 20 °C, while keeping the temperatures of Room 1 and Room 4 no higher than 26 °C. The input constraints are given by $u_1, u_2 \in [-10, 35]$. The initial state is $[5\ 20\ 10\ 15]^T$. In the simulation, the system is discretized using the forward Euler method with a sampling time of 1 second.

The reference controller is designed based on backstepping [118, Chapter 14.3]. In particular, to make T_2 and T_3 track 20 °C, a virtual controller v for T_1 and T_4 is designed as

$$v = K_{23}^u \left(\begin{bmatrix} T_2 \\ T_3 \end{bmatrix} - \begin{bmatrix} 20 \\ 20 \end{bmatrix} \right) - B_{23}^{-1} A_{23} \begin{bmatrix} 20 \\ 20 \end{bmatrix},$$

where A_{23} and B_{23} are the system matrices of the linear sub-equations for $\dot{T}_2$ and $\dot{T}_3$, where T_2 and T_3 are the states and T_1 and T_4 are the inputs. The feedback gain K_{23}^u is chosen as the LQR gain for the linear subsystem with $Q = I_2$ and $R = I_2$. The second term of v is a feedforward term. Then, the real reference input is determined by

$$\pi_\text{r}(x) = K_{14}^u \left(\begin{bmatrix} T_1 \\ T_4 \end{bmatrix} - v \right) - B_{14}^{-1} A_{14} v - \begin{bmatrix} T_2 K/K1 \\ T_3 K/K3 \end{bmatrix},$$

where A_{14} and B_{14} are the system matrices of the linear subsystem for T_1 and T_4, corresponding to the region where $|u_1 - T_1| \le 5$ and $|u_2 - T_4| \le 5$, with T_2 and T_3 treated as external disturbances. The feedback gain K_{14}^u is designed similarly to K_{23}^u. The second term of π_r is a feedforward term, while the third term compensates for the external disturbances.

Now, consider the backup CBF candidate $h_b(x) = \min\{24 - T_1,\ 24 - T_4\}$. It can be verified that by choosing the backup controller $\pi_\text{b}(x) = \begin{bmatrix} -KT_2/K_1 \\ -KT_3/K_3 \end{bmatrix}$, h_b satisfies $\partial h_\text{b} f_{\text{PWA,b}}(x) \ge$

$-\alpha_b h_b(x)$, $\forall x \geq [0\ 0\ 0\ 0]^T$, $\forall \partial h_b \in \partial_C h_b(x)$ and $\forall \alpha_b \in (0, K + K_0 + \max\{K_1, K_3\}]$. Here $f_{\text{PWA,b}}$ represents the system (5.30) controlled by π_b. The above analysis indicates that h_b and π_b are valid, i.e., satisfy Assumption[6] 5.4.1.

Fig. 5.8 plots the time responses of the closed-loop system controlled by $\hat{\pi}_{\text{safe}}$ and π_{safe}. The PSF (5.11) is implemented over a prediction horizon of $T = 4$ seconds, which is discretized into $N = 40$ uniformly spaced time steps. This corresponds to a discretization interval of 0.1 seconds. From the top-left and bottom-right sub-figures, it is observed that all the controllers satisfy the safety requirements ($T_1, T_4 \leq 26\,^\circ$C). On the other hand, the top-right and bottom-left sub-figures show that both $\hat{\pi}_{\text{safe}}$ and π_{safe} with $N = 40$ drive the temperatures T_2 and T_3 closer to the target temperature (20 °C), compared to $\hat{\pi}_{\text{safe}}$ and π_{safe} with $N = 0$ (i.e., policies derived from a standard CBF-based safety filter without predictions). Furthermore, consistent with intuition, the approximate explicit policy $\hat{\pi}_{\text{safe}}$ has a slightly worse tracking performance than the exact one π_{safe}, as evidenced by the red and blue curves in the top-right and bottom-left sub-figures.

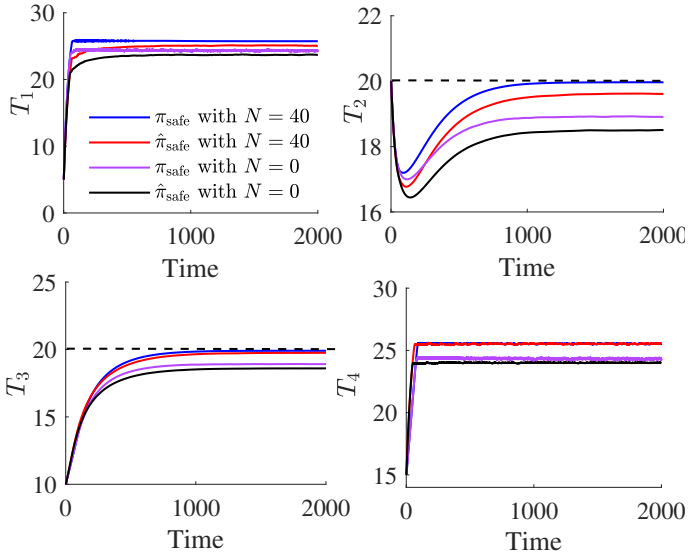
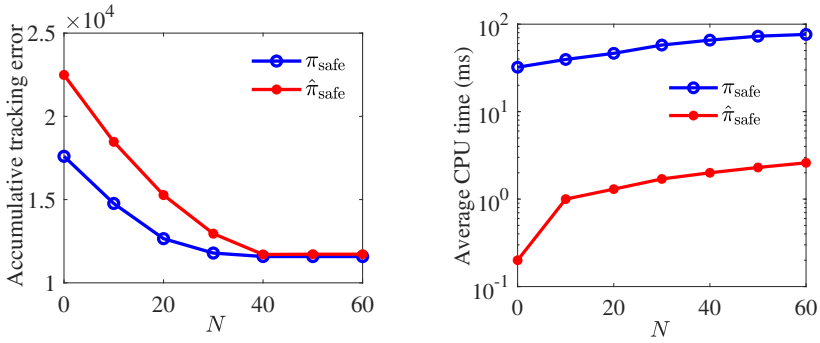


Figure 5.8: Time responses of the closed-loop system controlled by $\hat{\pi}_{\text{safe}}$ and π_{safe}. The black dashed lines represent the target temperature. The policy with $N = 40$ is derived from (5.11) with the prediction horizon $T = 4$ seconds and discretized into $N = 40$ uniformly spaced time steps. The policy with $N = 0$ is derived from a standard CBF-based safety filter without any predictions. Both $\hat{\pi}_{\text{safe}}$ and π_{safe} with $T = 4$ drive the temperatures T_2 and T_3 closer to the target temperature (20 °C), compared to $\hat{\pi}_{\text{safe}}$ and π_{safe} with $N = 0$. As expected, the approximate explicit policy $\hat{\pi}_{\text{safe}}$ exhibits a slightly worse tracking performance compared to the exact policy π_{safe}.

In Fig. 5.9, we compare the tracking performance and average CPU time of the controller π_{safe} and its explicit approximation $\hat{\pi}_{\text{safe}}$ for different prediction horizons. The tracking

[6] Strictly speaking, Assumption 5.4.1 is not completely satisfied because the inequality in condition (ii) holds only for $x \geq [0\ 0\ 0\ 0]^T$ in this example. However, this does not pose a problem as long as the system operates within the positive state region.

performance is defined as the summation of $(T_2-20)^2+(T_3-20)^2$ over 2000 seconds. The approximate controller $\hat{\pi}_{\text{safe}}$ retains a comparable tracking performance to the exact one π_{safe}. On the other hand, the average CPU time for $\hat{\pi}_{\text{safe}}$ (red line) is 1–2 orders of magnitude lower than that for the exact policy π_{safe} (blue line) across all prediction horizons, highlighting the practicality of the approximation in real-time or resource-constrained settings. Meanwhile, increasing the prediction horizon leads to reduced tracking errors for both $\hat{\pi}_{\text{safe}}$ and π_{safe}, while their CPU times grow approximately linearly. This demonstrates that incorporating predictions can effectively reduce the conservatism of the safety filter, compared to a standard CBF-based approach without predictions.



(a) Accumative tracking error.

(b) Average CPU time (log scale).

Figure 5.9: Tracking performance and average CPU time (log scale) for different prediction horizons for the exact policy π_{safe} and the approximate policy $\hat{\pi}_{\text{safe}}$. The tracking errors (defined as the summation of $(T_2-20)^2+(T_3-20)^2$ over 2000 seconds) are comparable, but the approximate policy significantly reduces the computational cost by up to two orders of magnitude across all horizons.

5.8. CONCLUSIONS AND FUTURE WORK

This chapter has developed a novel predictive safety filter (PSF) to achieve less conservative control of constrained piecewise affine (PWA) systems compared to standard safety filters that use control barrier functions (CBFs) or projections onto invariant sets. We have extended the predictive CBF method to handle the non-smoothness occurring in PWA dynamics, in the state constraint, as well as in the CBF. We have further derived an explicit approximation of the solution to the PSF. Such an approximation can be efficiently computed through basic arithmetic and min/max operations. A case study on an inverted pendulum interacting with an elastic wall has shown improved safety performance compared to classical PSFs. Furthermore, simulations on temperature control of a building have demonstrated the computational advantages of the approximate solution.

In the future, we will consider designing PSFs for general Lipschitz continuous nonlinear systems and exploring their explicit approximation. The PSF method can be viewed as an online version of HJ reachability. We believe that unifying these two methods is a meaningful future research direction. Besides, another interesting topic is to consider probabilistic safety constraints under model uncertainties.

5.A. APPENDICES

5.A.1. PROOF OF LEMMA 5.4.1

For the convenience of using the comparison theorem [118, Lemma 3.4], we define $\bar{h} = -h$. Since $\bar{h}$ is not always differentiable, we consider the generalized directional derivative [58]

$$\bar{h}^\circ(x;v) := \limsup_{w\to 0,\delta\downarrow 0} \frac{\bar{h}(x+w+\delta v)-\bar{h}(x+w)}{\delta} \tag{5.30}$$

for any $v \in \mathbb{R}^n$. The generalized directional derivative is well-defined for Lipschitz continuous functions. According to [58, Proposition 1.4], we have

$$\bar{h}^\circ(x;v) = \max_{\partial\bar{h}\in\partial_C\bar{h}(x)} \{\partial\bar{h}v\}.$$

By specifying $v = f(x,\pi(x))$ and noting that $\partial_C\bar{h}(x) = -\partial_C h(x)$ and that (5.12) holds, we have

$$\bar{h}^\circ(x;f(x,\pi(x))) = \max_{\partial\bar{h}\in\partial_C\bar{h}(x)} \{\partial\bar{h}f(x,\pi(x))\} \le -\alpha(\bar{h}(x)),\ \forall x\in\{x\in\mathbb{R}^n|\bar{h}(x)\le 0\}. \tag{5.31}$$

In order to prove the result by the comparison theorem, we consider the upper right-hand time derivative of $\bar{h}$ along $\dot{x} = f(x,\pi(x))$, which is defined by

$$\mathrm{D}^+\bar{h}(x_0,t) := \limsup_{\tau\downarrow 0} \frac{\bar{h}(\phi(x_0,t+\tau))-\bar{h}(\phi(x_0,t))}{\tau}. \tag{5.32}$$

Since ϕ is the solution of the closed-loop system $\dot{x} = f(x,\pi(x))$, we have the relation $\phi(x_0,t)+\limsup_{w\to 0,\delta\downarrow 0} w+\delta f(\phi(x_0,t),\pi(\phi(x_0,t))) = \limsup_{\tau\downarrow 0}\phi(x_0,t+\tau)$. As a result, by comparing (5.30) and (5.32) and noting that (5.31) holds, we obtain

$$\begin{aligned}\mathrm{D}^+\bar{h}(x_0,t) &= \bar{h}^\circ(\phi(x_0,t),f(\phi(x_0,t),\pi(\phi(x_0,t))))\\ &\le -\alpha(\bar{h}(\phi(x_0,t))),\ \forall t\ge 0\end{aligned}$$

if $x_0 \in \{x\in\mathbb{R}^n \mid \bar{h}(x)\le 0\}$. Because α is Lipschitz continuous, directly applying the comparison theorem [118, Lemma 3.4] yields $\bar{h}(\phi(x_0,t)) \le 0,\ \forall t\ge 0$, which consequently proves Lemma 5.4.1.

5.A.2. PROOF OF PROPOSITION 5.5.1

The equivalence between (3) and (4) follows from the definitions of backward and forward reachable sets. The equivalence between (1) and (2) stems from Definition 5.5.2.

If $\Phi_{\text{for}}(\bar{x}_0)\cap\mathcal{C}\neq\emptyset$, $\phi_{\text{b}}(\bar{x}_0,\tau)$ is on the boundary of multiple polyhedra of the PWA system (5.10) during $[\tau_k,\ \tau_{k+1})$, where $\{(\tau_0,I_0),\ (\tau_1,I_1),\ \ldots,\ (\tau_M,I_M)\}$ constitutes the time-stamped switching sequence of the solution $\phi_{\text{b}}(\bar{x}_0,\tau)$. To see why this is true, note that if $\Phi_{\text{for}}(\bar{x}_0)\cap\mathcal{C}\neq\emptyset$, there exists at least one instant $\tau_k\ge 0,\ k\in\mathbb{N}_M$ such that $\phi_{\text{b}}(\bar{x}_0,\tau_k)\in\mathcal{C}$. According to the definition of $\mathcal{C}$, $\bar{g}_{I_k}^{(q)}(x(\tau_k)) = g_{I_k}D_i^{q-1}(D_i x(\tau_k)+d_i) = 0,\ \forall q\in\mathbb{N}_n^+$. Using the Cayley–Hamilton Theorem [74], we know that $\bar{g}_{I_k}^{(q)}(x(\tau_k)) = 0,\ \forall q\in\mathbb{N}$, meaning that

any order time derivative of $\bar{g}_{I_k}$ vanishes when $\tau = \tau_k$. Consequently, $\bar{g}_{I_k}$ remains zero during $[\tau_k, \tau_{k+1})$, $\tau_{k+1} > \tau_k$, i.e., the set Γ_I is an *invariant manifold* [118, Chapter 8.1], so the trajectory must stay in Γ_I over the time interval $[\tau_k, \tau_{k+1})$. We refer to all such intervals as *critical periods*. During these periods, ϕ_b resides at the intersection of the closure of multiple regions.

(3) ⇒ (2): If $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} = \emptyset$, we can distinguish two cases. In the first case, there is no critical period. This means that for any switching instant τ_k, the trajectory crosses the hyperplane $\{x \in \mathbb{R}^n | \bar{g}_{I_k \cup I_{k+1}}(x) = 0\}$ from $\mathcal{R}_{I_k}$ and immediately enters the interior of one of the intersecting polyhedra $\mathcal{R}_{I_{k+1}}$. Here, all I_k have only one element. When $\tau \in (\tau_k, \tau_{k+1})$, we know that $\partial f_{\text{PWA,b}}(\phi_b(\bar{x}_0, \tau))$ is uniquely determined by D_{I_k}. As a result, according to (5.16), $Q_A(\bar{x}_0, \tau)$ is single-valued.

In the second case of $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} = \emptyset$, there still exists one or more critical periods during which $\bar{g}_{I_k}^{(q)} = 0$, $\forall q \in \mathbb{N}_n^+$. However, it must hold that $D_{i_k} = D_{j_k}$ for every two adjacent polyhedra $\mathcal{R}_{i_k}$ and $\mathcal{R}_{j_k}$, $i_k, j_k \in I_k$. In this case $Q_A(\bar{x}_0, \tau)$ is still single-valued.

(2) ⇒ (3): We prove this implication by showing ¬ (3) ⇒ ¬ (2). If $\Phi_{\text{for}}(\bar{x}_0) \cap \mathcal{C} \neq \emptyset$, according to the definition of $\mathcal{C}$, during each critical period $[\tau_k, \tau_{k+1})$, $\phi_b(\bar{x}_0, \tau)$ lies at the intersection of two or more adjacent polyhedra. As a result, the integral term of (5.16) admits at least two different forms $D_{i_k} Q(\bar{x}_0, s)$ and $D_{j_k} Q(\bar{x}_0, s)$, $i_k, j_k \in I_k$ from $s = \tau_k$ to $s = \tau_{k+1}$. By holding the integral term during the other periods $[\tau_0, \tau_1), ..., [\tau_{k-1}, \tau_k)$, $[\tau_{k+1}, \tau_{k+2}), ...$ fixed, we know that $Q_A(\bar{x}_0, \tau)$ must take at least two different values for all $\tau > \tau_k$.

In addition, if $\mathcal{C}$ is empty, then it follows from the above discussions that $\phi_b(x_0, \tau)$ has a global classical Jacobian w.r.t. x_0.

5.A.3. PROOF OF PROPOSITION 5.5.2

If $\phi_b(x_0, \tau)$ is differentiable w.r.t. x_0 at $x_0 = \bar{x}_0$, it is obvious that $\partial_C \phi_b(\bar{x}_0, \tau) = Q_A(\bar{x}_0, \tau) = \frac{\partial \phi_b(x_0, \tau)}{\partial x_0}\Big|_{x_0 = \bar{x}_0}$. In the remainder of the proof, we consider the case when $Q_A(\bar{x}_0, \tau)$ has multiple values.

First, we prove $\partial_C \phi_b \subseteq \text{conv}(Q_A)$. For any $\bar{x}_0$ and τ such that $\phi_b(x_0, \tau)$ is not differentiable w.r.t. x_0 at $x_0 = \bar{x}_0$, consider any sequence $\{y_j\}_{j=0}^{\infty}$ satisfying $\lim_{j \to \infty} y_j = \bar{x}_0$ and that $\phi_b(x_0, \tau)$ is differentiable w.r.t. x_0 at y_j. Since y_j converges to $\bar{x}_0$, there exists a sufficiently large $\bar{j} \in \mathbb{N}^+$ such that for any $j \geq \bar{j}$, $\phi_b(y_j, \tau)$ admits the same switching sequence $\{i_0, i_1, ..., i_M\}$ satisfying $\{i_0, i_1, ..., i_M\} \in I(\bar{x}_0)$, where $I(\bar{x}_0)$ is the Cartesian product of the switching sequence of $\phi_b(\bar{x}_0, \tau)$.

According to Definition 5.3.3, $\lim_{j \to \infty} J_{\phi_b}(y_j, \tau) \in \partial_C \phi_b(\bar{x}_0, \tau)$, where J_{ϕ_b} refers to the classical Jacobian and has the expression

$$J_{\phi_b}(y_j, \tau) = \begin{cases} e^{D_{i_0}\tau}, \tau \in [\tau_0, \tau_1) \\ e^{D_{i_1}(\tau - \tau_1)} J_{\phi_b}(y_j, \tau_1), \tau \in [\tau_1, \tau_2) \\ ... \\ e^{D_{i_{M-1}}(\tau - \tau_{M-1})} J_{\phi_b}(y_j, \tau_{M-1}), \tau \in [\tau_{M-1}, \tau_M). \end{cases} \tag{5.33}$$

By comparing (5.19) and (5.33) and noting that the switching time τ_k is continuous to y_j, we can prove by induction that $\lim_{j\to\infty} J_{\phi_b}(y_j,\tau) = Q_{(i_0,i_1,\ldots,i_M)}(\bar{x}_0,\tau) \in Q_A(\bar{x}_0,\tau)$. Here, the subscript $(i_0,i_1,\ldots,i_M)$ means that $Q_{(i_0,i_1,\ldots,i_M)}$ corresponds to the element of Q_A that is determined by the switching sequence $(i_0,i_1,\ldots,i_M)$. The containment of $\partial_C\phi_b$ in $\mathrm{conv}(Q_A)$ follows from the arbitrariness of the sequence $\{y_j\}_{j=0}^{\infty}$.

Conversely, we prove $\mathrm{conv}(Q_A) \subseteq \partial_C\phi_b$. Since we have proved that $\partial_C\phi_b \subseteq \mathrm{conv}(Q_A)$, according to Assumption 5.5.1, any element of $\partial_C\phi_b$ is invertible. As a result, by virtue of the inverse function theorem for generalized Clarke derivatives [57, Theorem 1], we know that there exist a Lipschitz continuous function $\phi_b^{-1}(\cdot,\tau): \mathbb{R}^n \to \mathbb{R}^n$, a neighborhood Y of $\bar{x}_0$, and a neighborhood Z of $\phi_b(\bar{x}_0,\tau)$ such that $\phi_b^{-1}(\phi_b(y,\tau)) = y$, $\forall\, y \in Y$ and $\phi_b(\phi_b^{-1}(z,\tau)) = z$, $\forall\, z \in Z$.

Then, for any $\bar{x}_0$ and any τ such that $\phi_b(x_0,\tau)$ is not differentiable w.r.t. x_0 at $x_0 = \bar{x}_0$, consider any element $Q_{(i_0,i_1,\ldots,i_M)}(\bar{x}_0,\tau)$ in $Q_A(\bar{x}_0,\tau)$. Note that $Q_{(i_0,i_1,\ldots,i_M)}(\bar{x}_0,\tau)$ can be expressed by (5.19). Since ϕ_b is differentiable almost everywhere, we immediately know that $\Phi_{\mathrm{back}}(\mathcal{C})$ has the dimension strictly lower than n. Therefore, we can always find a sequence $\{z_j\}_{j=1}^{\infty}$ satisfying $z_j \in Z$, $\forall\, j \in \mathbb{N}^+$, $z_j \notin \Phi_{\mathrm{back}}(\mathcal{C})$, $\forall\, j \in \mathbb{N}^+$, and $\lim_{j\to\infty} z_j = \phi_b(\bar{x}_0,\tau)$. By substituting z_j to the inverse function ϕ_b^{-1}, we get a sequence $y_j = \phi_b^{-1}(z_j,\tau)$, $j = 1,2,\ldots$ Thanks to the Lipschitz continuity of the inverse function, the sequence $\{y_j\}_{j=0}^{\infty}$ satisfies $\phi_b(y_j,\tau) = z_j$, $\lim_{j\to\infty} y_j = \lim_{j\to\infty} \phi_b^{-1}(z_j,\tau) = \bar{x}_0$, and $y_j \notin \Phi_{\mathrm{back}}(\mathcal{C})$. As y_j converges to $\bar{x}_0$, there exists a $\bar{j} \in \mathbb{N}^+$ such that for any $j \geq \bar{j}$, $\phi_b(y_j,\tau)$ admits the same switching sequence $\{i_0,\ i_1,\ \ldots,\ i_M\} \in I(\bar{x}_0)$. As a result, $\lim_{j\to\infty} J_{\phi_b}(y_j,\tau) = Q_{(i_0,i_1,\ldots,i_M)}(\bar{x}_0,\tau)$. It follows from Definition 5.3.3 that $Q_{(i_0,i_1,\ldots,i_M)}(\bar{x}_0,\tau) \in \partial_C\phi_b(\bar{x}_0,\tau)$. This completes the proof of Proposition 5.5.2.

5.A.4. PROOF OF PROPOSITION 5.6.1

Let $\lambda^+(x)$ denote the right-hand side of (5.28). The proof is divided into two parts: proving the feasibility of $\lambda^+(x)$ and the optimality of $\lambda^+(x)$.

(i) *Feasibility of* $\lambda^+(x)$. For any $x \in \Phi_{\mathrm{const}}(S_b,T)$, since $\pi_b(x)$ is feasible for (5.11), 0 is feasible for (5.26) as well. Therefore, we know that $\exists i \in I_u(x)$ such that $\omega_{i,j}(x,\tau) \geq 0$, $\forall \tau \in [0,T]$ and $\forall\, j \in \mathbb{N}^+_{\dim(\eta_i)}$. Due to the maximization operator in (5.27), it further holds that $\lambda^+(x) \geq 0$.

Suppose without loss of generality that $\lambda^+(x) = \lambda_{i^+,j^+}(x,\tau^+)$ for a certain $i^+ \in I_u(x)$, $j^+ \in \mathbb{N}^+_{\dim(\eta_{i^+})}$, and $\tau^+ \in [0,T]$. It follows from the two minimization operators in (5.27) that

$$\lambda^+(x) \leq \lambda_{i^+,j}(x,\tau),\ \forall\, j \in \mathbb{N}^+_{\dim(\eta_{i^+})} \text{ and } \forall \tau \in [0,T]. \tag{5.34}$$

Then, for any $j \in \mathbb{N}^+_{\dim(\eta_{i^+})}$ and any $\tau \in [0,T]$, based on (5.28) we can distinguish the following four cases:

Case (1): $\eta_{i^+,j}(x,\tau) > 0$. In this case, $\lambda_{i^+,j}(x,\tau) = \frac{\omega_{i^+,j}(x,\tau)}{\eta_{i^+,j}(x,\tau)}$. Directly applying (5.34) yields

$$\eta_{i^+,j}(x,\tau)\lambda^+(x) \leq \omega_{i^+,j}(x,\tau). \tag{5.35}$$

Case (2): $\eta_{i^+,j}(x,\tau)=0$ and $\omega_{i^+,j}(x,\tau)\geq 0$. In this case, (5.35) obviously holds.

Case (3): $\eta_{i^+,j}(x,\tau)=0$ and $\omega_{i^+,j}(x,\tau)<0$. From (5.28), we know that $\lambda_{i^+,j}(x,\tau)=0$, which further implies $\lambda^+(x)=0$ because of (5.34) and $\lambda^+(x)\geq 0$. Consequently, the feasibility of $\lambda^+(x)=0$ results from the feasibility of $\pi_{\mathrm{b}}(x)$.

Case (4): $\eta_{i^+,j}(x,\tau)<0$. In this case, we only need to focus on the situation when $\lambda^+(x)>0$ because if $\lambda^+(x)=0$ then the feasibility of $\lambda^+(x)$ follows from the feasibility of $\pi_{\mathrm{b}}(x)$. If $\lambda^+(x)>0$, then by defining $\lambda_{i^+}(x):=\min_{\tau'\in[0,T],j'\in\mathbb{N}^+_{\dim(\eta_{i^+})},\lambda_{i^+,j'}(x,\tau')\in(0,1)}\lambda_{i^+,j'}(x,\tau')$, we have

$$\omega_{i^+,j}(x,\tau)-\eta_{i^+,j}(x,\tau)\lambda_{i^+}(x)\geq 0. \tag{5.36}$$

Noticing the relation $\lambda^+(x)=\lambda_{i^+,j^+}(x,\tau^+)\geq\lambda_{i^+}(x)$, we derive (5.35) from (5.36).

Combining the four cases above and noticing the arbitrariness of j and τ, we can conclude that $\lambda^+(x)$ satisfies the constraint of (5.26) for $i=i^+$. This proves the feasibility of $\lambda^+(x)$.

(ii) *Optimality of* $\lambda^+(x)$. If $\lambda^+(x)=1$, the optimality follows directly from the constraint $\lambda\leq 1$. In the remainder of the proof, we consider $\lambda^+(x)<1$ and prove that $\lambda^+(x)+\Delta_\lambda$ is infeasible for problem (5.26) for any $\Delta_\lambda\in\left(0,1-\lambda^+(x)\right]$.

For any $i\in I_u(x)$, let $\lambda_i(x)=\lambda_{i,j^*}(x,\tau^*):=\min_{\tau\in[0,T]}\min_{j\in\mathbb{N}^+_{\dim(\eta_i)}}\lambda_{i,j}(x,\tau)$, and then we get the relation

$$\max_{i\in I_u(x)}\lambda_i(x)=\lambda^+(x)<1. \tag{5.37}$$

Analogous to the proof of the feasibility, for any $i\in I_u(x)$, we consider the following four cases:

Case (1): $\eta_{i,j^*}(x,\tau^*)>0$. In this case, $\lambda_{i,j^*}(x,\tau^*)=\frac{\omega_{i,j^*}(x,\tau^*)}{\eta_{i,j^*}(x,\tau^*)}$. Therefore, by (5.37) we have

$$\eta_{i,j^*}(x,\tau^*)(\lambda^+(x)+\Delta_\lambda)\geq\underbrace{\eta_{i,j^*}(x,\tau^*)\lambda_{i,j^*}(x,\tau^*)}_{=\omega_{i,j^*}(x,\tau^*)}+\underbrace{\eta_{i,j^*}(x,\tau^*)\Delta_\lambda}_{>0}>\omega_{i,j^*}(x,\tau^*),$$

which shows the infeasibility of $\lambda^+(x)+\Delta_\lambda$ regarding the j^*-th component of the constraint of (5.26).

Case (2): $\eta_{i,j^*}(x,\tau^*)=0$ and $\omega_{i,j^*}(x,\tau^*)\geq 0$. In this case, it follows from (5.28) that $\lambda^+(x)=\lambda_{i,j^*}(x,\tau^*)=1$, which contradicts (5.37). Therefore, this case cannot occur.

Case (3): $\eta_{i,j^*}(x,\tau^*)=0$ and $\omega_{i,j^*}(x,\tau^*)<0$. It is straightforward to get that $\eta_{i,j^*}(x,\tau^*)(\lambda^+(x)+\Delta_\lambda)=0>\omega_{i,j^*}(x,\tau^*)$. Similar to case (1), it can be proved that $\lambda^+(x)+\Delta_\lambda$ is infeasible.

Case (4): $\eta_{i,j^*}(x,\tau^*)<0$. Since $\lambda_{i,j^*}(x,\tau^*)$ can take either 0 or 1 in this case and recalling that $\lambda_{i,j^*}(x,\tau^*)<1$, it must hold that $\lambda_{i,j^*}(x,\tau^*)=0$ and that $\omega_{i,j^*}(x,\tau^*)-\eta_{i,j^*}(x,\tau^*)\lambda_{i,j''}(x,\tau'')<0$, where $\lambda_{i,j''}(x,\tau'')=$

$\min_{\tau' \in [0,\, T],\, j' \in \mathbb{N}^+_{\dim(\eta_i)},\, \lambda_{i,j'}(x,\tau') \in (0,1)} \lambda_{i,j'}(x,\tau')$. As a consequence, if $\lambda^+(x) + \Delta_\lambda \leq \lambda_{i,j''}(x,\tau'')$, we have

$$\eta_{i,j^*}(x,\tau^*)(\lambda^+(x) + \Delta_\lambda) \geq \eta_{i,j^*}(x,\tau^*)\lambda_{i,j''}(x,\tau'') > \omega_{i,j^*}(x,\tau^*). \tag{5.38}$$

If $\lambda^+(x) + \Delta_\lambda > \lambda_{i,j''}(x,\tau'')$, consider the j''-th component of the constraint of (5.26). In particular, since $\lambda_{i,j''}(x,\tau'') \in (0,1)$, by (5.28) we know that $\omega_{i,j''}(x,\tau'') > 0$, $\eta_{i,j''}(x,\tau'') > 0$, and $\lambda_{i,j''}(x,\tau'') = \frac{\omega_{i,j''}(x,\tau'')}{\eta_{i,j''}(x,\tau'')}$. Then we have

$$\eta_{i,j''}(x,\tau'')(\lambda^+(x) + \Delta_\lambda) > \eta_{i,j''}(x,\tau'')\lambda_{i,j''}(x,\tau'') > \omega_{i,j''}(x,\tau''). \tag{5.39}$$

Combining (5.38) and (5.39), we observe that either the j^*-th row or the j''-th row of the constraint of (5.26) is violated for $\lambda^+(x) + \Delta_\lambda$ in Case (4).

Integrating the discussions in the four cases above and noticing the arbitrariness of i, we can conclude that $\lambda^+(x) + \Delta_\lambda$ is infeasible for problem (5.26).

6

INTEGRATED LEARNING AND OPTIMIZATION FOR CONTROL OF CONSTRAINED NONLINEAR SYSTEMS: A MODEL-BASED APPROACH

Learning-based control with safety guarantees usually requires real-time safety certification and modifications of possibly unsafe learning-based policies. The control barrier function (CBF) method uses a safety filter (SF) containing a constrained optimization problem to produce safe policies. However, finding a valid CBF for a general nonlinear system requires a complex function parameterization, which in general makes the policy optimization problem difficult to solve in real time. For nonlinear systems with nonlinear state constraints, this chapter proposes the novel concept of state-action CBFs (SACBFs), which do not only characterize the safety at each state but also evaluate the control inputs taken at each state. SACBFs, in contrast to CBFs, enable a flexible parameterization, resulting in an SF that involves a convex quadratic optimization problem, which significantly alleviates the online computational burden. We propose a learning-based approach to synthesize SACBFs. The effect of learning errors on the effectiveness of SACBFs is addressed by constraint tightening and introducing a new concept called contractive-set CBFs. This ensures formal safety guarantees for the learned CBFs and control policies. Simulation results on an inverted pendulum with elastic walls validate the proposed CBFs in terms of constraint satisfaction and CPU time.

This chapter is based on the paper [100].

6.1. INTRODUCTION

Learning-based control methods have demonstrated extensive success across many applications, such as autonomous vehicles [81] and robotics [69]. Both supervised learning and reinforcement learning (RL) have been widely used for controller synthesis. However, learning-based controllers may provide unsafe control actions that result in undesirable or even destructive effects on the system. In control systems, safety means that the trajectories of the closed-loop system should satisfy state and input constraints for the entirety of the system's evolution. The lack of safety guarantees limits the ability of learning-based control to achieve safety-critical tasks.

There has been an increasing interest in designing controllers for safety-critical systems. One notable approach is using supervised learning to approximate model predictive control (MPC) policies. While traditional online MPC can be computationally intensive for nonlinear constrained systems [178], recent work has demonstrated that neural networks trained to mimic MPC behavior can achieve satisfactory performance in both safety and real-time efficiency [106]. However, approximate MPC typically relies on expert data and lacks adaptability to new scenarios, whereas reinforcement learning (RL) allows agents to explore and learn optimal control strategies through interaction with the environment, making it more flexible and scalable for complex tasks.

Safe reinforcement learning (safe RL) is a subfield of RL focusing on ensuring safety during or after the learning phase. Techniques in safe RL generally fall into two main categories: constrained policy optimization during learning and policy refinement after learning. Constrained policy optimization during learning uses penalties [102], multiplier methods [156], or constraint elimination [215] when training the RL policy. These approaches are often straightforward to implement and compatible with standard RL algorithms. However, they typically involve a trade-off between safety and optimal performance. Policy refinement after learning uses a safety filter (SF) accompanied by a safety certificate to modify unsafe control actions. Examples include invariant sets [130] and control barrier functions [69]. While policy refinement can provide formal safety guarantees, it usually requires prior knowledge on safety certificates, which can be difficult to obtain [196], due to the non-convex and nonlinear nature of the underlying problem.

Among policy refinement methods, CBFs offer a particularly attractive solution for enforcing safety for nonlinear systems. Similar to energy-based functions, CBFs define sublevel sets that represent safe regions of the state space and they can be used to construct SFs. CBFs provide a theoretically sound way to ensure that long-term safety constraints are respected by a single CBF constraint. SFs adjust control inputs in real time to enforce compliance with the CBF constraint. In general, SFs are more online computationally efficient than some other safe controllers (such as MPC) that need long-term prediction. In addition to CBF-based SFs, there are also predictive SFs [196] and SFs that involve projection onto invariant sets [114]. In this chapter, we will use the term "CBF-SF" specifically to refer to a "CBF-based SF".

In the realm of CBF-based methodologies, even though the state and input constraints are known, there is a lack of universally applicable methods for generating valid CBFs, necessitating reliance on manually designed or problem-specific CBFs. For some par-

ticular kinds of systems, such as linear, piecewise affine [124], or polynomial systems [163], CBFs can be computed by solving convex optimization problems. For nonlinear systems with general constraints, using learning-based algorithms accompanied by advanced function approximators to estimate CBFs has been explored in several contexts. Roughly, there are four kinds of methods to learn CBFs: the optimization-based method [172], the learner-verifier method [66], the Hamilton-Jacobi (HJ) reachability method [55], and the predictive SF method [76], [197]. The first two methods can be conservative, i.e., the learned safe set can be a small subset of the maximal controlled invariant set. In comparison, the HJ reachability method relies on the dynamic programming principle to approximate the maximal controlled invariant set iteratively, while the predictive SF method is based on a receding-horizon open-loop optimal control problem to implicitly determine the safe set, which also converges to the maximal controlled invariant set as the horizon goes to infinity [121]. A comprehensive comparison of HJ reachability, predictive SFs, and CBFs can be found in [196].

No matter how the CBF is learned, to control a given system with safety guarantees, it is inevitable to solve an online optimization problem with CBF-based constraints, which is usually non-convex for discrete-time nonlinear systems [3]. To obtain a satisfactory approximation accuracy, sophisticated function approximators such as deep neural networks (NNs) are commonly used to parameterize the CBF [69], [76]. However, this will inherently cause non-convexity and increase the complexity of the optimization problem. As a result, the increased online computational load makes the CBF-based approach unsuitable for situations where fast computation of control inputs is required. Besides, the effect of approximation errors on the validity of the learned CBFs and the resulting control policies has not yet been addressed, as highlighted in [69]. Note that [55] introduces control barrier-value functions by unifying Hamilton-Jacobi (HJ) reachability and CBFs to address issues of finding valid CBFs. However, [55] focuses on theoretical guarantees of robust safety to disturbances but does not address learning errors.

The chapter has the following main contributions:

1. **Computationally efficient safety enforcement using state-action CBFs:** We introduce the novel concept of state-action CBFs (SACBFs). Unlike standard CBFs [3], [10], [55], which are often computationally intensive when ensuring safety in nonlinear systems, SACBFs provide a flexible parametrization. The flexibility lies in the ability to enforce safety constraints through a convex quadratic optimization problem, significantly reducing the computational burden during online operations.

2. **Handling approximation errors in learned CBFs:** We propose a constraint-tightening approach alongside the novel concept of contractive-set CBFs to address approximation errors in learned CBFs. This ensures that the invariance property of CBFs is maintained even when approximation errors occur, which is not yet achieved in existing work on learning CBFs [55], [76], [172], [185]. Additionally, we examine the relationship between SACBFs and contractive-set CBFs, and develop a new learning-based method to approximate SACBFs. This ensures the safety of the policy filtered by SACBFs, provided the approximation error is suf-

ficiently small. Besides, we discuss the trade-off between online computational efficiency and safety when learning the proposed SACBFs.

6.2. PRELIMINARIES AND PROBLEM FORMULATION

6.2.1. PRELIMINARIES

We consider a deterministic, discrete-time, and nonlinear system

$$x_{t+1} = f(x_t, u_t), \quad t = 0, 1, \ldots, \tag{6.1}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ are the state and the input at time step t, and $f(\cdot,\cdot) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a continuous function satisfying $f(0,0) = 0$. We consider a constrained optimal control problem in which the states and inputs should satisfy time-invariant constraints: $x_t \in X := \{x \in \mathcal{X} | h(x) \leq 0\}$ and $u_t \in U \subseteq \mathcal{U}$. Here, $h(\cdot) : \mathcal{X} \to \mathbb{R}$ is a continuous function that defines the state constraint[1]. In most parts of the chapter, we assume that f is fully known. However, in Section 6.5.3, we briefly discuss the applicability of our method when there is system uncertainty. Besides, we assume that X is compact and that U is a polytope.

The control objective is to regulate a predefined control policy, which could be a policy learned through various methods such as RL [28], learning-based MPC [114], or any other suitable control technique that needs safety guarantees. The primary concern is ensuring constraint satisfaction *after* learning, rather than safe exploration [60], which is not the focus here.

6.2.2. CONTROL BARRIER FUNCTIONS

To achieve the objective, we need to design a control policy that can ensure constraint satisfaction at all time steps. A set of states for which such a policy exists needs to be defined. Usually, this class of sets is called controlled-invariant sets or safe sets. For high-dimensional systems, however, some controlled-invariant sets could have complex representations, making the controller synthesis difficult. A CBF uses the sub-level set of a scalar function to conveniently define the safe set.

Definition 6.2.1 (Control barrier function)**.** A continuous function $B(\cdot) : \mathcal{X} \to \mathbb{R}$ is called a *control barrier function* (CBF) with a corresponding safe set $\mathcal{S}_B := \{x \in \mathbb{R}^{n_x} | B(x) \leq 0\} \subseteq \mathcal{X}$, if $\mathcal{S}_B$ is non-empty,

$$h(x) \leq 0, \ \forall x \in \mathcal{S}_B, \tag{6.2}$$

and

$$\forall x \in \mathcal{S}_B, \ \exists u \in U \text{ s.t. } B(f(x,u)) \leq 0. \tag{6.3}$$

Furthermore, a CBF is called an exponential CBF if $h(x) \leq B(x), \ \forall x \in \mathcal{S}_B$ and if there exists a $\beta \in (0,1)$ such that

$$\forall x \in \mathcal{S}_B, \ \exists u \in U \text{ s.t. } B(f(x,u)) \leq \beta B(x). \tag{6.4}$$

[1] For the constraint defined by multiple inequalities $h_i(x) \leq 0, \ i = 1, 2, ..., I$, we can let $h(x) = \max_{i \in \{1,...,I\}} h_i(x)$. The set $\{x | h(x) \leq 0\}$ is then identical to $\{x | h_i(x) \leq 0, \ i = 1, 2, ..., I\}$, and h will be continuous if each h_i is continuous.

The discrete-time CBF introduced in this chapter has similar properties to the continuous-time CBF in Chapter 5. In particular, with a CBF available, one can generate a safe control policy in $\mathcal{S}_B$ by using the following optimization-based approach:

$$\begin{aligned}\pi_{\text{safe}}(x) = \arg\min_{u\in U} \; & ||u-\pi_0(x)||_2 \\ \text{s.t. } & B(f(x,u)) \le 0, \text{if } B \text{ is a CBF (not exponential)} \\ & \text{or } B(f(x,u)) \le \beta B(x), \text{if } B \text{ is an exponential CBF,}\end{aligned} \tag{6.5}$$

which serves as a CBF-SF [130] for any continuous unsafe policy π_0. The definition of a safe policy is formally given as follows.

Definition 6.2.2 (Safe policy)**.** A policy $\pi(\cdot): \mathcal{X} \to \mathcal{U}$ is safe in $\mathcal{S} \subseteq X$ for system (6.1) under the state and input constraints $x \in X$, $u \in U$, if $\pi(x) \in U$, $\forall x \in \mathcal{S}$, and for any initial state in $\mathcal{S}$, the state trajectory of (6.1) steered by π will always stay in $\mathcal{S}$.

For general nonlinear systems with state and input constraints, synthesizing a non-conservative CBF is a difficult task. Hand-crafted or application-specific heuristics are mostly used in the literature to design candidate CBFs, which can be either unsafe or overly conservative (see [193, Figure 1] for an illustrative example). To deal with this issue, using advanced function approximators to learn a CBF certificate has received much attention (see [69] for a comprehensive survey).

6.2.3. LIMITATIONS OF EXISTING CBFS

Problem P1: high online computational complexity. One main limitation of (6.5) is that it needs to solve a usually non-convex optimization problem in real-time. If a CBF is represented by a complex function approximator $B_\theta(\cdot)$ with the parameter θ, solving (6.5) may take much online computation time and result in very sub-optimal solutions. Even if a convex B_θ is formed, the constraint in (6.5) could be non-convex due to the nonlinearity of f.

Problem P2: effects of approximation errors. Another limitation of (6.5) is that the approximation error of B may affect the safety of the system controlled by the optimizer of (6.5) with B replaced by B_θ. Besides, the recursive feasibility of (6.5) is also not guaranteed with B_θ.

6.3. STATE-ACTION CONTROL BARRIER FUNCTIONS

To deal with Problem P1, motivated by Q-learning in RL [28], we propose a novel safety filter (SACBF-SF) in the following form:

$$\pi_{\text{safe}}(x) = \arg\min_{u} \{||u-\pi_0(x)||_2, \text{ s.t. } u \in U,\ Q(x,u) \le 0\}, \tag{6.6}$$

where $Q(\cdot,\cdot): \mathcal{X} \times U \to \mathbb{R}$ is a function of states and actions. We will analyze how to enforce the safety of π_{safe} by imposing conditions on Q. To achieve this, we introduce the definition of SACBFs as follows.

Definition 6.3.1 (State-action control barrier function (SACBF))**.** A function $Q^B(\cdot,\cdot): \mathcal{X} \times U \to \mathbb{R}$ is called a *state-action control barrier function* (SACBF) with a corresponding safe set $\mathcal{S}_Q$ of states, if the pair $(Q^B, \mathcal{S}_Q)$ satisfies the following conditions:

(i) $\mathcal{S}_Q$ is non-empty, and $h(x) \leq 0,\ \forall x \in \mathcal{S}_Q$.

(ii) $\min_{u\in U} Q^B(x,u) \leq 0,\ \forall x \in \mathcal{S}_Q$.

(iii) For any $x \in \mathcal{S}_Q$, any $u \in U$ satisfying $Q^B(x,u) \leq 0$ ensures that $f(x,u) \in \mathcal{S}_Q$.

Unlike the definition of standard CBFs, we do not prescribe the form of $\mathcal{S}_Q$ based solely on Q. Besides, condition (i) imposes $\mathcal{S}_Q \subseteq X$. The following lemma builds the connection between standard CBFs and SACBFs and provides an explicit form of $\mathcal{S}_Q$ when the SACBF is derived from a standard CBF.

Lemma 6.3.1. (i) For the SACBF-SF (6.6), any SACBF Q will render π_{safe} safe in $\mathcal{S}_Q$. In other words, $\mathcal{S}_Q$ is control-invariant.

(ii) If B is a CBF, $Q(\cdot,\cdot) = B(f(\cdot,\cdot))$ will be an SACBF with the safe set $\mathcal{S}_B$, i.e., $\mathcal{S}_Q = \mathcal{S}_B$.

Proof. (i) Based on the condition (ii) of Definition 6.3.1, for any $x_0 \in \mathcal{S}_Q$, there exists a $u_0 \in U$ such that $Q(x,u) \leq 0$. This means that problem (6.6) is feasible when $x = x_0$. As $Q(x_0,u_0) \leq 0$ implies $f(x_0,u_0) \in \mathcal{S}_Q$, the control-invariance of $\mathcal{S}_Q$ follows. Consequently, problem (6.6) is recursively feasible for the initial state x_0. Due to the arbitrariness of x_0, π_{safe} is safe according to Definition 6.3.1.

(ii) By specifying $Q(\cdot,\cdot) = B(f(\cdot,\cdot))$ and $\mathcal{S}_Q = \mathcal{S}_B$, we have that $\mathcal{S}_Q$ and Q satisfy the conditions (i) and (ii) of Definition 6.3.1. Furthermore, for any $x_0 \in \mathcal{S}_Q$, consider any u_0 such that $Q(x_0,u_0) \leq 0$. We have $B(f(x_0,u_0)) \leq 0$, which further implies $f(x_0,u_0) \in \mathcal{S}_Q$. In addition, as B and f are continuous, Q is also continuous. □

Similarly to designing CBF, the challenge of using (6.6) is that the explicit form of an SACBF cannot be directly obtained based on Definition 6.3.1. We will present a learning-based approach to synthesize SACBFs in Section 6.5.

The advantage of directly approximating Q over approximating B is that we can design a specific structure for Q_θ to simplify the constraint in (6.6), so that the online computational cost of solving (6.6) is reduced. To achieve this, we can specify the following parameterization:

$$Q_\theta(x,u) = q_{1,\theta}(x) + q_{2,\theta}(x)u + u^T Q_{3,\theta}(x)u, \text{ with } Q_{3,\theta} \succeq 0, \tag{6.7}$$

which will make problem (6.6) a convex QP with linear and convex quadratic constraints. In (6.7), $q_{1,\theta}(\cdot): \mathbb{R}^{n_x} \to \mathbb{R}$, $q_{2,\theta}(\cdot): \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ and $Q_{3,\theta}(\cdot): \mathbb{R}^{n_x} \to \mathbb{R}^{n_u \times n_u}$ are parameterized functions with all parameters condensed in θ.

Our proposed concept of SACBFs is inspired by the definition of state-action (Q) value functions in RL [28]. Both kinds of functions not only characterize the energy of each state but also quantify the quality of taking each action in each state. In Section 6.5, we will demonstrate that the Q value function for a specific finite-horizon optimal control problem is, in fact, an SACBF.

6.4. CONSTRAINT TIGHTENING AND CONTRACTIVE-SET CBFS

To cope with Problem P2, we propose a method to compute conservative CBFs based on state constraint tightening. In particular, we consider two kinds of conservative CBFs. The first kind includes the CBFs of system (6.1) under the tightened state constraint $X_\lambda := \{x \in \mathcal{X} | h(x) + \lambda \leq 0\}$ where λ is a positive constraint backoff. In this situation, the CBFs still satisfy (6.3) or (6.4), while condition (6.2) is tightened to $h(x) + \lambda \leq B(x), \forall x \in \mathcal{S}_B$.

The second one, called the contractive-set CBF, is defined as follows

Definition 6.4.1 (Contractive-set CBF)**.** A CBF $B(\cdot) : \mathcal{X} \to \mathbb{R}$ is called a *λ-contractive-set CBF*[2] if there exists a $\lambda > 0$ such that

$$\min_{u \in U} B(f(x,u)) \leq -\lambda, \ \forall x \in \mathcal{S}_B. \tag{6.8}$$

Similarly, an exponential CBF $B(\cdot) : \mathcal{X} \to \mathbb{R}$ is called a *λ-contractive-set exponential CBF* if there exist $\lambda > 0$ and $\beta \in (0,1)$ such that

$$\min_{u \in U} B(f(x,u)) \leq \beta B(x) - \lambda, \ \forall x \in \mathcal{S}_B. \tag{6.9}$$

The definition of contractive-set CBFs is introduced to endow the safe set $\mathcal{S}_B$ with a contractive property. It enforces an energy decay when the state is on the boundary of the safe set. In particular, if B is a λ-contractive-set CBF, for any $x \in \partial\mathcal{S}_B$, there exists an input $u \in U$ such that $B(f(x,u)) \leq -\lambda$, which means that $x^+ = f(x,u)$ is in the interior of $\mathcal{S}_B$. Such a property can guarantee that any approximation of B is still a valid CBF if the approximation error is sufficiently small.

6.5. APPROXIMATING SACBFS

In this section, we propose a method for approximating the SACBF. Inspired by HJ reachability [55] and predictive CBF [76], we propose a comprehensive framework that uses the optimal value functions of a sequence of optimization problems to implicitly represent the CBF sequence $\{B_k\}_{k=1}^{\infty}$. By tuning some parameters, this framework can compute standard CBFs, exponential CBFs, as well as contractive-set CBFs. Although the exact expression of each B_k is, in general, intractable to compute, we can use learning-based approaches such as supervised learning and RL [28] and then use the results of Lemma 6.3.1 to approximate SACBFs.

[2] Note that in fact the definition of contractive-set CBFs given here is not consistent with the common definition of contraction mappings [28]. Formally speaking, the CBF we define should be called a CBF with a contractive safe set [6]. However, for compactness, we adopt "contractive-set CBF" in the chapter.

6.5.1. CBF GENERATOR BASED ON REACHABILITY ANALYSIS

We consider each B_k as the value function of the following optimization problem:

$$\begin{aligned} B_k(x) := \min_{\{x_t,u_t\}_{t=0}^{k}} \max\left\{ \max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t)+\lambda_t), \alpha^k B_0(x_k)\right\} \\ \text{s.t. } x_{t+1} = f(x_t,u_t),\ u_t\in U,\ t\in\mathcal{I}(k-1) \\ x_0 = x. \end{aligned} \tag{6.10}$$

In (6.10), $\mathcal{I}(k-1)=\{0,1,...,k-1\}$, $\alpha\geq 1$, and B_0 is a CBF, which, however, could have a very small safe set. We take $\alpha=1$ if the CBF B_0 is not an exponential CBF, or $\alpha=1/\beta$ if B_0 is an exponential CBF. To solve (6.10), we need to know the explicit formulation of B_0. An LMI-based method that can compute a local quadratic B_0 for the nonlinear system (6.1) is reported in [197]. In Appendix 6.A.2, this method is extended to our situations where additional conditions such as (6.8), (6.9), and (6.11)-(6.12) in the subsequent Theorem 6.5.1 are required. The LMI method to obtain B_0 relies on the linearization of both the system and the constraints. The resulting B_0 is, in general, very conservative. In contrast, by introducing the reachability problem (6.10), we get less conservative CBFs B_k, as stated in Theorem 6.5.1 below.

In (6.10), $\lambda_t\geq 0,\ t\in\mathcal{I}(k-1)$ are tuning parameters. We consider the following three options for choosing λ_t:

Option 1: $\lambda_t=0$. **Option 2:** $\lambda_t=\lambda$, where $\lambda>0$ is a constant. **Option 3:** $\lambda_t=t\lambda$, where $\lambda>0$ is a constant.

Option 1 means that we are constructing CBFs for the system (6.1) under the original state constraints. If Option 2 is chosen, it is seen from (6.10) that we increase the function h to $h+\lambda$, i.e., we tighten the original state constraints to $x\in X_\lambda=\{x\in\mathcal{X}|h(x)+\lambda\leq 0\}$. More conservatively, in Option 3, we require a linear decrease rate for h w.r.t. the time step to construct contractive-set CBFs.

In the original HJ reachability analysis [55], B_0 is chosen as h, which is not a CBF. It has been proven in [55] that only when $k=\infty$, B_∞ is a CBF. Although this result shows a strong connection between CBFs and HJ reachability, it cannot be used in practice since we cannot solve (6.10) with $k=\infty$. In our case, we require the initial function B_0 to be a CBF. As a result, any $B_k,\ k=1,2,...$ will become a CBF, with a non-shrinking safe set as k increases. The following theorem formally states this property, and the proof is given in Appendix 6.A.1.

Theorem 6.5.1. Consider B_k from (6.10). Suppose that B_0 is a (exponential) CBF for (6.1) with the state constraint $x\in X$.

1. Let $\lambda_t=0,\ t\in\mathcal{I}(k-1)$. Then $B_k,\ k=1,2,...$ is a (exponential) CBF for system (6.1) with the state constraint $x\in X$.
2. Let $\lambda_t=\lambda>0,\ t\in\mathcal{I}(k-1)$. If λ, B_0 satisfy

$$h(x)+\lambda\leq B_0(x), \forall x\in\mathcal{S}_{B_0}, \tag{6.11}$$

then $B_k,\ k=1,2,...$ is a (exponential) CBF for system (6.1) with the tightened state constraint $x\in X_\lambda=\{x|h(x)+\lambda\leq 0\}$.

3. Let $\lambda_t = t\lambda,\ t \in \mathcal{I}(k-1),\ \lambda > 0$. If B_0 is a λ-contractive (exponential) CBF and k, λ, B_0 satisfy
$$h(x) + k\lambda \leq B_0(x), \forall x \in \mathcal{S}_{B_0}, \tag{6.12}$$
then $B_k,\ k = 1,2,...$ is a λ-contractive (exponential) CBF for system (6.1) with the state constraint $x \in X$.

4. In the statements (i)-(iii), B_k is a continuous function in $\mathcal{X}$, and the safe sets satisfy $\mathcal{S}_{B_0} \subseteq \mathcal{S}_{B_1} \subseteq ... \subseteq \mathcal{S}_{B_\infty}$. Furthermore, if f, h, and B_0 are Lipschitz continuous, B_k is Lipschitz continuous.

If B_k and f are (Lipschitz) continuous in their domains, then the function Q defined by
$$Q(x,u) = B_k(f(x,u)) \tag{6.13}$$
is also (Lipschitz) continuous in $\mathcal{X} \times U$.

The condition (6.12) indicates that the horizon k should not be selected too large when using (6.10) to generate a contractive-set CBF.

The structure of (6.10) is similar to that in [76], [121]. The main difference is that we use the maximum of $h + \lambda_t$ and B_0 over state trajectories in a finite horizon, while [76], [121] uses the summation of $\max\{h, 0\}$ and $\max\{B_0, 0\}$ over state trajectories in a finite horizon. This difference means that the safe set $\mathcal{S}_{B_k}$ is the zero-sublevel set of B_k in our situation, while the safe set in [76], [121] is the zero-level set of B_k, i.e., the set $\{x \in \mathbb{R}^{n_x} | B_k(x) = 0\}$. Besides, in [76], [121] the weight on the terminal CBF B_0 needs to be carefully selected.

6.5.2. LEARNING SACBFS FROM CBF SAMPLES

We assume that the prediction model in the reachability problem (6.10) is known. To approximate SACBFs, we opt for supervised learning, which is more computationally efficient than RL because it can solve (6.10) exactly and efficiently for each state sample.

After the CBF generator is provided with a fixed k, a regression model can be trained to obtain an approximation of an SACBF.

First, state and input samples are collected in a compact region $\Omega \times U$ of interest. The region Ω is task-specified and should be contained in the space where the system is physically realistic. If the state space $\mathcal{X}$ of the system is compact and small, letting $\Omega = \mathcal{X}$ is an ideal choice. If $\mathcal{X}$ is unbounded or too large, one possible choice for Ω is $\Omega = \{x \in \mathcal{X} | B_k(x) \leq \bar{B}\}$ [76], where $\bar{B} > 0$ is a positive constant. In the case of $\alpha > 1$, the values of B_k and Q are likely to grow exponentially as k increases, so it is necessary to specify Ω as $\{x \in \mathcal{X} | B_k(x) \leq \bar{B}\}$ to avoid approximating probably unbounded Q. Besides, various sampling methods [147], such as (quasi) random sampling and sampling from a uniform grid, can be applied.

After N samples $\left\{x_i^{(s)}, u_i^{(s)}\right\}_{i=1}^N$ are collected, problem (6.10) is solved with x specified as each $f\left(x_i^{(s)}, u_i^{(s)}\right)$. As a result, N data tuples $\left\{\left(x_i^{(s)},\ u_i^{(s)},\ Q(x_i^{(s)}, u_i^{(s)})\right)\right\}_{i=1}^N$ are obtained after substituting $x = x_i^{(s)},\ u = u_i^{(s)}$ into (6.13). For general nonlinear systems, (6.10)

is usually a nonlinear non-convex optimization problem, which requires a multi-start strategy [171] to find a sufficiently good optimum. For linear systems with linear or ellipsoidal constraints, (6.10) is a convex quadratically constrained quadratic program (QCQP), and the global optimum can be conveniently obtained via gradient-based optimization methods. For piecewise affine systems with piecewise affine constraints, (6.10) can be regarded as a mixed-integer QCQP, which can be globally solved by the branch-and-bound approach [206].

In this work, we use NNs to approximate the SACBF. For the parameterization (6.7), we use three NNs to represent $q_{1,\theta}$, $q_{2,\theta}$, and $Q_{3,\theta}$ in (6.7), respectively. There are several ways to guarantee the positive-semidefiniteness of $Q_{3,\theta}$. For example, we can further parameterize $Q_{3,\theta}$ by $Q_{3,\theta} = L_\theta L_\theta^T$, where $L_\theta \in \mathbb{R}^{n_u \times n_u}$ is a lower triangular matrix with non-negative diagonal entries. Alternatively, we can parameterize $Q_{3,\theta}$ by $Q_{3,\theta} = P_0 \text{diag}(r_\theta)\text{diag}(r_\theta)P_0^T$, where $r_\theta \in \mathbb{R}^{n_u}$ is the output of an NN and $P_0 \in \mathbb{R}^{n_u \times n_u}$ is a given matrix.

Finally, θ is optimized to minimize the mean square error between $B_k(f)$ and Q_θ over all state-action samples, using classical NN training algorithms such as stochastic gradient descent [86].

6.5.3. PERFORMANCE ANALYSIS

After the approximation Q_θ has been obtained, it can be integrated into the SACBF-SF (6.6), i.e., Q in (6.6) is replaced by Q_θ. In general, the safety of the policy generated from (6.6) will not be ensured due to the approximation error. However, we will now demonstrate that we can guarantee safety in the presence of a small approximation error. To achieve this, we need an assumption on the boundedness of this error.

Assumption 6.5.1. The approximation error is uniformly bounded in $\Omega \times U$. In other words, there exists a non-negative constant $\Delta < \infty$ such that

$$|Q_\theta(x,u) - Q(x,u)| \leq \Delta, \ \forall (x,u) \in \Omega \times U,$$

where Ω is a compact set and satisfies $\mathcal{S}_Q \subseteq \Omega$.

Assumption 6.5.1 is common in the literature studying performance guarantees of learning-based control [76], [101], [106]. If the chosen function approximator represents a continuous function, since Q is also continuous, the approximation error is always upper-bounded in any compact region. To obtain the upper bound, we can first get the error bound on a finite number of samples, and then extract a statistical estimate [106] or compute a deterministic bound in the whole region by using the Lipschitz property of Q and Q_θ [101].

With Assumption 6.5.1, we modify the SACBF-SF (6.6) to

$$\begin{aligned} \pi_\theta(x) = \arg\min_{u \in U} \ & ||u - \pi_0(x)||_2 \\ \text{s.t. } & Q_\theta(x,u) \leq 0 \text{ for Options 1 and 2} \\ & Q_\theta(x,u) \leq -\lambda + \Delta \text{ for Option 3.} \end{aligned} \tag{6.14}$$

The following theorem characterizes the safety performance of the policy π_θ.

Theorem 6.5.2 (Safety and recursive feasibility)**.** Consider the system (6.1) controlled by π_θ, the SACBF Q from (6.13), and the SACBF-SF (6.14). Suppose that Assumption 6.5.1 holds.

(i) If B_k is a CBF for the original state constraint, for any initial state $x_0 \in \mathcal{S}_{B_k}$ that makes problem (6.14) recursively feasible, the closed-loop system $x_{t+1} = f(x_t, \pi_\theta(x_t))$, $t = 0, 1, \ldots$ has the maximum constraint violation Δ, i.e., $\max_{t\in\mathcal{I}(\infty)}\{h(x_t)\} \le \Delta$.

(ii) If B_k is a CBF for the tightened state constraint $x \in X_\lambda$ and $\Delta \le \lambda$, for any initial state $x_0 \in \mathcal{S}_{B_k}$ that makes problem (6.14) recursively feasible, the closed-loop system $x_{t+1} = f(x_t, \pi_\theta(x_t))$ always satisfies the state constraint $x \in X$, i.e., $\max_{t\in\mathcal{I}(\infty)}\{h(x_t)\} \le 0$.

(iii) If B_k is a λ-contractive-set CBF for the original state constraint and $\Delta \le \lambda/2$, problem (6.14) will be recursively feasible for all initial states in $\mathcal{S}_{B_k}$ and the closed-loop system $x_{t+1} = f(x_t, \pi_\theta(x_t))$ always satisfies the state constraint $x \in X$, i.e., $\max_{t\in\mathcal{I}(\infty)}\{h(x_t)\} \le 0$.

Proof. The proofs of (i) and (ii) can be combined. From Assumption 6.5.1 and (6.14), we know that if (6.14) is recursively feasible for the initial state x_0, we have $Q(x_t, \pi_\theta(x_t)) \le \Delta$, $\forall\, t \in \mathcal{I}(\infty)$, which further implies $B_k(x_t) \le \Delta$, $\forall\, t \in \mathcal{I}(\infty)$. If B_k is a CBF for (6.1) with the original state constraint $x \in X$, we get $h(x_t) \le B_k(x_t) \le \Delta$. Similarly, if B_k is a CBF for (6.1) with the tightened state constraint $x \in X_\lambda$ and $\lambda \ge \Delta$, we get $h(x_t) \le B_k(x_t) - \lambda \le 0$. This completes the proofs of the statements (i) and (ii).

Next, we consider the last statement of Theorem 6.5.2. For any initial state $x_0 \in \mathcal{S}_{B_k}$, there exists a u_0 such that $Q(x_0, u_0) \le -\lambda$ according to (6.8) and (6.9). Since Q_θ satisfies Assumption 6.5.1, u_0 makes $Q_\theta(x_0, u_0) \le -\lambda + \Delta$, which means problem (6.14) is feasible at the initial state x_0. Then, we consider any feasible solution u_0' to (6.14) when $x = x_0$, i.e., any $u_0' \in U$ such that $Q_\theta(x_0, u_0') \le -\lambda + \Delta$. It follows from Assumption 6.5.1 and $\Delta \le \lambda/2$ that $Q(x_0, u_0') \le -\lambda + 2\Delta \le 0$. This means that $B_k(f(x_0, u_0')) \le 0$, i.e., the next state $x_1' = f(x_0, u_0')$ is in $\mathcal{S}_{B_k}$. As we have proved that problem (6.14) is feasible for any $x_0 \in \mathcal{S}_{B_k}$, the recursive feasibility of (6.14) is thus proved. A direct consequence of the recursive feasibility is that $B_k(x_t) \le 0$, $\forall\, t \in \mathcal{I}(\infty)$, where x_t is the state trajectory of the system $x_{t+1} = f(x_t, \pi_\theta(x_t))$. Moreover, since B_k is a CBF, we have $h(x_t) \le 0$, $\forall\, t \in \mathcal{I}(\infty)$. This finishes the proof of (iii). □

In other words, in Option 2 of (6.10), we get a CBF B_k for the tightened constraint $x \in X_\lambda$. The policy π_θ will always make the system satisfy the original state constraint if (6.14) is always feasible. However, the feasibility of problem (6.14) is only guaranteed for the initial state, i.e., problem (6.14) can be infeasible for some subsequent states. Practically, one needs to perform offline some statistical or deterministic verification [106] to analyze the recursive feasibility of (6.14). Otherwise, as the recursive feasibility of (6.14) cannot be predicted in advance, one needs to do some relaxation for the constraint in (6.14) once infeasibility is observed. In comparison, in Option 3 of (6.10), we get a λ-contractive-set CBF B_k. An induced feature is the recursive feasibility of problem (6.14). As a result, under a sufficiently small approximation error, the SACBF approximation Q_θ will render π_θ a safe policy according to Definition 6.2.2.

The proposed constraint tightening approach naturally has robustness to model uncertainty. In particular, suppose that (6.1) is a nominal system, and the real system is $x_{t+1} = f_r(x_t, u_t)$. The mismatch between f and f_r can be handled similarly to the learning error of Q. Formally, we have the following corollary:

Corollary 6.5.1 (Robustness to model uncertainty)**.** Suppose that Assumption 6.5.1 is fulfilled and the nominal system satisfies $\left|f(x,u) - f_r(x,u)\right| \leq \kappa, \forall (x,u) \in \Omega \times U$, where $\kappa \geq 0$. Then, if B_k is a λ-contractive-set CBF for the nominal system under the original state constraint and $\lambda \geq 2\Delta + L_B\kappa$ with L_B the Lipschitz constant of B_k, problem (6.14) will be recursively feasible for the real system with all initial states in $\mathcal{S}_{B_k}$ and the closed-loop system $x_{t+1} = f_r(x_t, \pi_\theta(x_t))$ always satisfies the state constraint $x \in X$, i.e., $\max_{t \in \mathcal{J}(\infty)}\{h(x_t)\} \leq 0$.

The proof is similar to the proof of (iii) of Theorem 6.5.2 and is thus omitted here.

6.5.4. DISCUSSION ON SACBFS

One limitation of the parameterization (6.7) is that it sacrifices the universal approximation property [94] of NNs, since it restricts Q_θ to be quadratic on u. This may result in a large error bound Δ and cause constraint violations according to Theorem 6.5.2. As our main motivation for introducing state-action CBFs is to improve online computational efficiency, such a sacrifice is acceptable. Conversely, fully parameterizing Q_θ with a continuous NN, possessing universal approximation capabilities, increases the online computational complexity and introduces challenges in finding the global optimum of (6.14). In conclusion, when applying an approximation to the SACBF, there is typically a natural trade-off between online computational efficiency and safety.

Note that a parametrization of an SACBF Q, instead of parameterizing a standard CBF B, may increase the approximation error on $B(f)$, especially when Q is parameterized in a quadratic form. This error can result in the learned Q failing to satisfy the conditions of a valid SACBF. To address this issue, we have proposed the constraint-tightening approach (Section 6.4) to ensure that the learning-based approach can still generate a valid SACBF under mild approximation errors. However, a quadratic parametrization of Q typically requires more stringent constraint tightening, which can lead to increased conservatism in the learned barrier certificates.

The conservatism of the learned SACBF Q_θ as well as the policy π_θ is strongly influenced by Δ. According to Theorem 6.5.2, to guarantee the recursive feasibility of (6.14) and the safety of π_θ, a larger Δ requires a larger λ in (6.10). This, in turn, increases the conservatism of the target Q. In practice, to minimize the conservatism, one can start with a small λ and iteratively increase it until Assumption 6.5.1 and $\Delta \leq \lambda/2$ are satisfied. Increasing the approximation capability of the chosen function approximator, as discussed in [18], can also be beneficial.

We focus on discrete-time nonlinear systems because our work targets safety in learning-based control methods, like RL and supervised learning of MPC policies, which typically use discrete-time models [28], [166]. Additionally, SFs for discrete-time systems are computationally more challenging, leading to nonlinear programs, whereas continuous-time systems involve simpler convex quadratic programs [3]. Therefore, the computa-

tional benefits of introducing SACBFs are more evident in the context of discrete-time nonlinear systems.

Compared to learning standard CBFs [172], the sampling complexity of learning the proposed SACBFs increases mildly because the domain of the function to be approximated only changes from $\mathbb{R}^{n_x}$ to $\mathbb{R}^{n_x+n_u}$. More importantly, advanced sampling strategies, such as active learning [32] and counterexample-guided inductive synthesis [1], can be utilized to improve the scalability of approximating Q.

6.6. CASE STUDY

Table 6.1: Performance of different controllers. "SR" means "Safety Rate", defined as the number of initial states leading to a safe closed-loop trajectory divided by the total number of initial states.

Safety filters	Basic control policies								
	Learning MPC			ADP			LQR		
	SR (%)	CPU (ms)	Cost	SR (%)	CPU (ms)	Cost	SR (%)	CPU (ms)	Cost
No filter	63.52	0.024	99.69	36.66	0.024	125.11	56.99	0.023	122.10
Standard CBF	63.52	2.3	126.18	39.93	2.4	161.82	57.17	2.3	120.00
Quad. SACBF	70.05	1	131.76	47.19	1.1	161.83	65.52	0.9	115.80
Nonlin. SACBF	65.15	2.5	132.22	40.19	3.0	168.19	60.80	2.9	137.30
Quad tightened SACBF	98.73	1.0	97.29	98.73	1.0	99.44	92.92	1.0	91.47
Nonlin. tightened SACBF	82.40	2.7	117.26	82.03	2.7	116.56	90.56	3.0	116.92
Quad. contr. SACBF	100	0.9	98.84	100	1.0	101.91	100	0.9	93.10
Nonlin. contr. SACBF	96.91	2.1	110.44	92.20	2.0	128.99	94.57	3.0	109.97
MPC	Horizon=5			Horizon=6			Horizon=7		
	70.24	38.2	98.55	95.83	37.6	97.43	100	38.5	96.86

We validate the proposed methods for the pendulum system considered in Chapter 3, which is a piecewise affine system. The simulations are conducted in MATLAB R2021a on an AMD Core R7-5800H CPU @3.20GHz machine. All optimization problems involved in the policy filters (6.5) and (6.14) are solved using the Matlab function "fmincon" and the active-set optimization algorithm. The problem (6.10) and the MPC problem are transformed equivalently into mixed-integer quadratic programming problems and are solved using Gurobi [92].

The system contains the state $x = [\theta\ \dot{\theta}]^T$ where θ is the pendulum angle, and the input u, which is the torque acting on the bottom end of the pendulum. The state and input constraints are given by $|x_1| \le 0.15$, $|x_2| \le 1$, and $|u| \le 4$. The system is discretized using the explicit Euler method with a time step of 0.05 s.

Using the approach in Appendix 6.A.2 and letting $\lambda = 0.2$ for Option 2 and $\lambda = 0.05$ for Option 3, we obtain the initial CBF $B_0(x) = x^T P x - 1$ with $P = \begin{bmatrix} 124.74 & 7.44 \\ 7.44 & 2.24 \end{bmatrix}$, $\begin{bmatrix} 283.40 & 20.25 \\ 20.25 & 4.69 \end{bmatrix}$, and $\begin{bmatrix} 522.39 & 25.69 \\ 25.69 & 5.39 \end{bmatrix}$ for Options 1, 2, and 3, respectively. We sample from the state-input space $\{(x,u) \big| |x_1| \le 0.16, |x_2| \le 1.1, |u| \le 4\}$ on a uniform grid and we obtain 40^3 samples $\{(x_i^{(s)}, u_i^{(s)})\}_{i=1}^{40^3}$. The successor state $f\left(x_i^{(s)}, u_i^{(s)}\right)$ of each sample $\left(x_i^{(s)}, u_i^{(s)}\right)$ is fed to the CBF generator (6.10) with $k = 7$. Only those samples satisfying $B_k(f(x_i^{(s)}, u_i^{(s)})) \le 10$ are collected. After this procedure, 42109, 38609 and 37027 samples

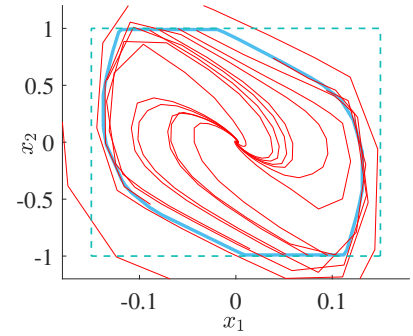
are collected in each option.

We compare SACBFs and standard CBFs. We consider 4 different kinds of safety certificates: (i) SACBF for the original state constraint; (ii) SACBF for the tightened state constraint $|x_1| \leq 0.14$, $|x_2| \leq 0.9$; (iii) Contractive-set SACBF; (iv) Standard CBF for the original state constraint. For the learning, we use NNs with three hidden hyperbolic tangent layers containing 16, 64, and 8 neurons. We parameterize each standard CBF by an NN. For the parameterization of each SACBF, we use (6.7) or fully parameterize it by an NN. Besides, we also consider 4 different control policies. The first three include (i) supervised learning of the MPC policy [114], (ii) the approximate dynamic programming (ADP) policy [102], which is a kind of RL policy, and (iii) the LQR policy for the linearized system around the origin. These policies may be unsafe and are thus taken as π_0 in (6.5) and (6.14). The learning-based control policies (i) and (ii) are also represented by NNs. All NNs are trained via the stochastic gradient descent algorithm [86]. The main reason for choosing these three particular policies is that they can achieve sub-optimal control objectives using very limited computational resources. However, they generally cannot guarantee constraint satisfaction. Therefore, we use these three policies as the baseline to demonstrate that the proposed SACBF-SF can enhance the rate of constraint satisfaction for these policies. The fourth control policy is implicit MPC. One can refer to Appendix 6.A.3 for a detailed description of these policies.

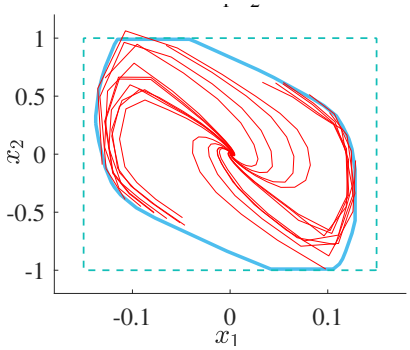
To test all the control methods, we uniformly randomly sample initial states from the state sample set $\{x_i^{(s)}\}_{i=1}^{40^3}$ and select a total of 551 initial states x near the boundary of the safe set. Therefore, they are in general more difficult to stabilize than any other states in the safe set. For each initial state, we simulate the closed-loop systems for 50 time steps.

The simulation results are shown in Table 6.1. The performance metrics include the rate of safety, the average CPU time for computing the control input per time step, and the total cost $\sum_{t=0}^{50} C(x_t, u_t)$ with the stage cost the same as that in the considered basic controllers. The safety rate is defined as the number of initial states leading to a safe closed-loop trajectory divided by the total number of initial states. Overall, the table indicates that incorporating appropriate safety filters, especially quadratic and nonlinear SACBFs, can significantly improve the safety performance of control policies without excessively compromising on CPU time and total cost. The online computational benefit of using the SACBF with the quadratic parametrization (6.7) primarily stems from the fact that solving a convex quadratic program is typically faster than solving a nonlinear program with the same number of constraints and decision variables, especially when the nonlinear constraints involve an NN. The maximum performance loss is about 34%, which occurs when applying the nonlinear SACBF to the ADP controller. Considering that the main objective of our work is to impose safety using low computational resources, such a performance degradation is acceptable. The inferior performance of the nonlinear contractive-set SACBF compared to its quadratic counterpart with respect to constraint satisfaction is primarily due to the suboptimal solutions when solving (6.14). In other words, the assumption of perfect nonlinear optimization does not hold in practical scenarios, leading to the rare instances of constraint violations observed in Table 6.1. It should be noticed that the computational advantage of the quadratic SACBF will be larger if multi-start nonlinear optimization is used to solve (6.5) or (6.14) with a fully

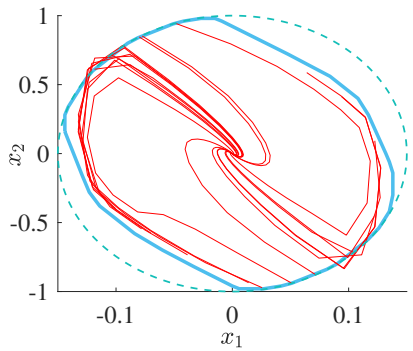
parameterized CBF.



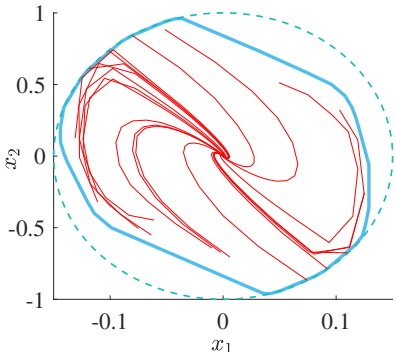
(a) State trajectories and the safe set for the SACBF under the linear state constraint



(b) State trajectories and the safe set for the contractive-set SACBF under the linear state constraint



(c) State trajectories and the safe set for the SACBF under the nonlinear state constraint



(d) State trajectories and the safe set for the contractive-set SACBF under the nonlinear state constraint

Figure 6.1: Closed-loop trajectories of the learning MPC controller under the SACBF-SF. The red curves represent the trajectories starting from different initial states in the safe set. The blue curves refer to the boundaries of the safe set. The green dashed curves represent the boundaries of the state constraint set.

Fig. 6.1(a) and Fig. 6.1(b) show the state trajectories of the closed-loop system with the policy π_θ from (6.14), with π_0 being the learning MPC controller and Q_θ being the nonlinear SACBF or the nonlinear contractive-set SACBF. Without constraint tightening, the system evolution records several constraint violations, as evidenced by Table 6.1. Conversely, the contractiveness property of SACBFs ensures that almost all state trajectories remain within its safe set. To further demonstrate the proposed method's effectiveness under nonlinear constraints, we consider a quadratic state constraint given by $x \in X := \{x \in \mathbb{R}^2 | \frac{\theta^2}{0.15^2} + \frac{\dot{\theta}^2}{1^2} \leq 1\}$. The resulting state trajectories of the closed-loop system under the policy π_θ are displayed in Fig. 6.1(c) and Fig. 6.1(d), from which we observe that the contractive-set SACBF guarantees safety for all simulated trajectories under this

quadratic constraint.

6.7. CONCLUSIONS AND FUTURE WORK

This chapter has presented a new type of CBFs, called SACBFs, that can synthesize safe controllers for general constrained nonlinear systems with a very small online computational overhead. We have also developed a new approach to learn the proposed CBF from data, and proposed a constraint tightening approach to improve the robustness of the learned CBFs to learning errors.

In the future, we will first focus on learning the proposed CBFs for large-scale problems in a computationally efficient manner. We will also investigate how the proposed CBFs can be exploited to guide the learning of the basic policy. This will help to reduce the potential degradation of other control performance measures caused by the SF. We will investigate whether undesired equilibria might arise when applying the proposed SACBF-SF, as observed for continuous-time systems.

6.A. APPENDICES

6.A.1. PROOF OF THEOREM 6.5.1

In the following, we will consider the case when B_0 is an exponential CBF, i.e., CBF satisfying the additional condition (6.4). The case when B_0 is a general CBF satisfying (6.3) can be analyzed similarly to the case when B_0 is an exponential CBF.

The proofs of the first and second statements can be combined by considering $\lambda_t = \lambda$ and $\lambda \geq 0$. For any x_0 such that $B_k(x_0) \leq 0$, letting $(x^*_{t+1}, u^*_t),\ t \in \mathcal{I}(k-1)$ be any one of the optimal solutions to (6.10) when $x = x_0$, we have

$$h(x^*_t) + \lambda \leq B_k(x_0) \leq 0,\ t = 1, 2, ..., k-1 \text{ and } B_0(x^*_k) \leq 0. \tag{6.15}$$

As B_0 is an exponential CBF and $B_0(x^*_k) \leq 0$, we have (i) $h(x^*_k)+\lambda \leq B_0(x^*_k)$ based on (6.11), and (ii) there exists an input $u^*_k \in U$, such that $x^*_{k+1} = f(x^*_k, u^*_k)$ will make $B_0(x^*_{k+1}) \leq \beta B_0(x^*_k)$. Now, we consider the value of $B_k(x^*_1)$. Based on the above discussion, it is clear that the trajectory $(x^*_{t+1}, u^*_t),\ t = 1, 2, ..., k$ is feasible for problem (6.10) when $x = x^*_1$. Due to the optimality of $B_k(x^*_1)$, we have

$$\begin{aligned}
B_k(x^*_1) &\leq \max\left\{ \max_{t\in\mathcal{I}(k-1)} \alpha^t (h(x^*_{t+1}) + \lambda), \alpha^k B_0(x^*_{k+1}) \right\} \\
&\leq \frac{1}{\alpha} \max\left\{ \max_{t\in\mathcal{I}(k)} \alpha^t (h(x^*_t) + \lambda), \alpha^{k+1} B_0(x^*_{k+1}) \right\} \\
&\leq \frac{1}{\alpha} \max\left\{ \max_{t\in\mathcal{I}(k-1)} \alpha^t (h(x^*_t) + \lambda), \alpha^k B_0(x^*_k), \underbrace{\alpha^k (h(x^*_k) + \lambda)}_{\leq \alpha^k B_0(x^*_k)} \right\} \\
&\leq \frac{1}{\alpha} \max\left\{ B_k(x_0), \alpha^k B_0(x^*_k) \right\} \\
&\leq \beta B_k(x_0).
\end{aligned} \tag{6.16}$$

In (6.16), the second inequality is true because we add the term $h(x_0^*)+\lambda$ to the inner max block. The third inequality follows from $B_0(x_{k+1}^*) \le \frac{1}{\alpha} B_0(x_k^*)$. The fourth inequality holds because the maximization of the first and second terms in the outer max operation equals $B_k(x_0)$. The last inequality holds because $\alpha^k B_0(x_k^*) \le B_k(x_0)$, according to (6.10). From (6.16), we can conclude that $\min_{u\in U} B_k(f(x_0,u)) \le \beta B_k(x_0)$. This, together with (6.15), shows that B_k is an exponential CBF for system (6.1) under the state constraint $x \in X_\lambda = \{x | h(x)+\lambda \le 0\}$, since x_0 is selected arbitrarily in $\mathcal{S}_{B_k}$. Therefore, by specifying $\lambda = 0$, we prove the first statement, and by letting $\lambda > 0$ we prove the second statement.

Next, we will prove the third statement of Theorem 6.5.1. Similarly to the proof of the first two statements, we consider any x_0 such that $B_k(x_0) \le 0$. Denoting by $(x_{t+1}^*, u_t^*),\ t \in \mathcal{I}(k-1)$ any one of the optimal solutions to (6.10) when $x = x_0$, we have

$$h(x_t^*) + \lambda_t \le B_k(x_0) \le 0,\ t = 1,2,...,k-1 \text{ and } B_0(x_k^*) \le 0. \tag{6.17}$$

Since B_0 is a λ-contractive-set exponential CBF and $B_0(x_k^*) \le 0$, we have (i) $h(x_k^*) + k\lambda \le B_0(x_k^*)$, and (ii) there is a control input $u_k^* \in U$ such that the successor state $x_{k+1}^* = f(x_k^*, u_k^*)$ ensures that $B_0(x_{k+1}^*) \le \beta B_0(x_k^*) - \lambda$. Then, owing to the optimality of $B_k(x_1^*)$, we have

$$\begin{aligned}
B_k(x_1^*) &\le \max\left\{ \max_{t\in\mathcal{I}(k-1)} \alpha^t (h(x_{t+1}^*) + t\lambda), \alpha^k B_0(x_{k+1}^*) \right\} \\
&\le \frac{1}{\alpha} \max\left\{ \max_{t\in\mathcal{I}(k)} \alpha^t (h(x_t^*) + (t-1)\lambda), \alpha^{k+1} B_0(x_{k+1}^*) \right\} \\
&\le -\lambda + \frac{1}{\alpha} \max\left\{ \max_{t\in\mathcal{I}(k)} \alpha^t (h(x_t^*) + t\lambda), \alpha^{k+1} (B_0(x_{k+1}^*) + \lambda) \right\} \\
&\le -\lambda + \frac{1}{\alpha} \max\left\{ \max_{t\in\mathcal{I}(k-1)} \alpha^t (h(x_t^*) + t\lambda), \alpha^k B_0(x_k^*), \alpha^k (h(x_k^*) + k\lambda) \right\} \\
&\le -\lambda + \frac{1}{\alpha} \max\left\{ B_k(x_0), \alpha^k B_0(x_k^*) \right\} \\
&\le -\lambda + \beta B_k(x_0).
\end{aligned} \tag{6.18}$$

In (6.18), the second line is true because of the additional introduced item $h(x_0^*)-\lambda$ in the inner max block. The forth inequality holds because of $B_0(x_{k+1}^*) \le \beta B_0(x_k^*) - \lambda$. The fifth inequality follows from $\max\{\max_{t\in\mathcal{I}(k-1)} \alpha^t (h(x_t^*) + t\lambda), \alpha^k B_0(x_k^*)\} = B_k(x_0)$. The last inequality holds because $\alpha^k B_0(x_k^*) \le B_k(x_0)$, according to (6.10). Together with (6.17), (6.18) implies that B_k is a λ-contractive-set exponential CBF for system (6.1) with the state constraint $x \in X$.

Finally, we prove the last statement of Theorem 6.5.1. In all the cases of $\lambda_t = 0$, $\lambda_t = \lambda$, and $\lambda_t = t\lambda$, we consider the state and input sequences $\{x_t^*\}_{t=0}^{k+1}$, $\{u_t^*\}_{t=0}^{k}$, where $x_0^* = x_0$. The sequences satisfy

$$B_k(x_0^*) \le 0,\ h(x_t^*) + \lambda_t \le 0,\ t \in \mathcal{I}(k),\ B_0(x_{k+1}^*) \le 0. \tag{6.19}$$

From the last inequality in (6.19) and Definitions 6.2.1 and 6.4.1, we know that $h(x_{k+1}^*) + \lambda_{k+1} \le 0$ and that there exists a $u_{k+1}^* \in U$ such that the value of B_0 at $x_{k+2}^* = f(x_{k+1}^*, u_{k+1}^*)$

is smaller than or equal to zero. Meanwhile, note that the sequences $\{x_t^*\}_{t=0}^{k+2}$ and $\{u_t^*\}_{t=0}^{k+1}$ constitute a feasible solution to problem (6.10) with $k+1$, so we get an upper bound on $B_{k+1}(x_0)$ as $B_{k+1}(x_0) \leq 0$. Due to the arbitrariness of x_0 in $\mathcal{S}_{B_k}$, we can conclude that $\mathcal{S}_{B_k} \subseteq \mathcal{S}_{B_{k+1}}$, which further by recursion proves that $\mathcal{S}_{B_0} \subseteq \mathcal{S}_{B_1} \subseteq ... \subseteq \mathcal{S}_{B_\infty}$.

For the continuity of the value function, we use a similar analysis structure as in [121]. For any $x_0, y_0 \in \mathcal{X}$, let $(x_{t+1}^*, u_t^{x*}), (y_{t+1}^*, u_t^{y*}),\ t \in \mathcal{I}(k-1)$ be any one of the optimal solutions to (6.10), with $x = x_0$ and $x = y_0$ respectively. We consider the difference between $B_k(x_0)$ and $B_k(y_0)$:

$$
\begin{aligned}
&B_k(x_0) - B_k(y_0) \\
=&\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t^*)+\lambda_t), \alpha^k B_0(x_k^*)\right\} - \max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(y_t^*)+\lambda_t), \alpha^k B_0(y_k^*)\right\} \\
=&\underbrace{\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t^*)+\lambda_t), \alpha^k B_0(x_k^*)\right\} - \max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t')+\lambda_t), \alpha^k B_0(x_k')\right\}}_{\leq 0 \text{ due to optimality of } x_t^*} \\
&+\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t')+\lambda_t), \alpha^k B_0(x_k')\right\} - \max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(y_t^*)+\lambda_t), \alpha^k B_0(y_k^*)\right\} \\
\leq&\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(x_t')+\lambda_t), \alpha^k B_0(x_k')\right\} - \max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t(h(y_t^*)+\lambda_t), \alpha^k B_0(y_k^*)\right\} \\
\leq&\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t h(x_t') - (h(y_t^*)), \alpha^k(B_0(x_k') - B_0(y_k^*))\right\},
\end{aligned} \tag{6.20}
$$

where $x_t',\ t = 0, 1, ..., k$ is the state trajectory starting from x_0 and applying u_t^{y*}, and the last inequality holds because of the triangle inequality for the maximum norm. Since (i) the trajectories x_t' and $y_t^*,\ t = 0, 1, ..., k$ are obtained by applying the same control inputs, (ii) f and h are continuous in their domains, and (iii) the maximum of continuous functions of x yields a continuous function of x, there exists a $\delta > 0$ such that $B_k(x_0) - B_k(y_0) < \epsilon$ whenever $||x_0 - y_0||_2 < \delta$. A mirror argument proves that $B_k(y_0) - B_k(x_0) < \epsilon$ whenever $||x_0 - y_0||_2 < \delta$. The continuity of B_k w.r.t. x thus follows.

Furthermore, if f, h, and B_0 are Lipschitz continuous, with Lipschitz constants L_f, L_h, and L_{B_0}, respectively, from (6.20) we find that for any $x_0,\ y_0$ such that $||x_0 - y_0||_2 \leq \delta$, we have

$$
B_k(x_0) - B_k(y_0) \leq \underbrace{\max\left\{\max_{t\in\mathcal{I}(k-1)} \alpha^t L_h L_f^t, \alpha^k L_{B_0} L_f^k\right\}}_{:=L_{B_k}} \delta. \tag{6.21}
$$

Following a similar argument as for (6.21), we can get that $B_k(y_0) - B_k(x_0) \leq L_{B_k}\delta$ whenever $||x_0 - y_0||_2 \leq \delta$. Thus, we conclude the Lipschitz continuity of B_k w.r.t. x.

6.A.2. COMPUTING B_0

In this subsection, we provide a detailed procedure for synthesizing the initial CBFs in Options 1-3. In Option 1, a CBF B_0 for the system (6.1) with the original state constraint needs to be obtained. In Option 2, a CBF for (6.1) with the tightened constraint $x \in X_\lambda$ is required. In Option 3, a contractive-set CBF is needed.

We follow the similar design steps as presented in [124], [197]. Firstly, we linearize the system (6.1) around the origin by $x_{k+1} = Ax_k + Bu_k + r(x_k, u_k)$, with the matrices $A := (\partial/\partial x) f(x,u)\big|_{(0,0)}$, $B := (\partial/\partial u) f(x,u)\big|_{(0,0)}$, and the high-order error term $r(x(k), u(k)) = f(x,u) - Ax - Bu$. Since U is a polytope, we can compute its half-space representation $\{u \in \mathbb{R}^{n_u} | H_u x \le h_u\}$. If the state constraint is linear, i,e., h is the maximum of some affine functions, we compute the half-space representation of X as $\{x \in \mathbb{R}^{n_x} | H_x x \le h_x\}$. Otherwise, we compute a polyhedron $\{x \in \mathbb{R}^{n_x} | H_x x \le h_x\}$ that is an inner approximation of X. To this end, we first specify H_x, which determines the shape of the polyhedron. The choice of H_x is task-specific and the simplest choice is $H_x = [\mathbb{I}_{n_x} \ -\mathbb{I}_{n_x}]^T$, which will make the polyhedron a hyper-rectangle. Here, $\mathbb{I}_{n_x}$ denotes the $n_x \times n_x$ identity matrix. Then, we decrease h_x until the following condition is verified:

$$\max_{x \in \mathbb{R}^{n_x}} \{h(x), \text{ s.t. } H_x x \le h_x\} \le \begin{cases} 0 \text{ for Options } 1,3 \\ -\lambda \text{ for Option } 2. \end{cases}$$

A positive h_x always exists if the origin is contained in X for Options 1 and 3 or if the origin is contained in X_λ for Option 2.

With the linearized system and constraints, we focus on finding a quadratic CBF of the form $B_0(x) = x^T E^{-1} x - \gamma$, where $E \in \mathbb{R}^{n_x \times n_x}$ is positive-definite and γ can be taken as any positive value. To allow for solving convex optimization problems to obtain E, we parameterize a linear control law $u = YE^{-1}x$, where $Y \in \mathbb{R}^{n_u \times n_x}$. As a result, the following lemma shows that one can make B_0 a (contractive) CBF for the linearized system by solving LMI inequalities. In the lemma, repeated blocks within symmetric matrices are replaced by $*$ for brevity and clarity.

Lemma 6.A.1. Consider the CBF and control law parameterizations $B_0(x) = x^T E^{-1} x - \gamma$, $u(x) = YE^{-1}x$. For the linearized system $x_{k+1} = Ax_k + Bu_k$ with the state and input constraints: $x \in \{x \in \mathbb{R}^{n_x} | H_x x \le h_x\}$, $u \in U := \{u \in \mathbb{R}^{n_u} | H_u u \le h_u\}$, we have the following results:

(i) To make B_0 a CBF, it is sufficient to find E and Y such that

$$\begin{bmatrix} pE & EA^T + Y^T B^T \\ * & E \end{bmatrix} \succeq 0 \tag{6.22}$$

$$\begin{bmatrix} h_{u,i}^2 & H_{u,i} Y \\ * & E/\gamma \end{bmatrix} \succeq 0, \ i = 1, \ldots, \dim(h_u) \tag{6.23}$$

$$\begin{bmatrix} (h_{x,i} - q)^2 & H_{x,i} E \\ * & E/\gamma \end{bmatrix} \succeq 0, \ i = 1, \ldots, \dim(h_x) \tag{6.24}$$

hold with $p = 1$ and $q = 0$.

(ii) To make B_0 a λ-contractive-set CBF satisfying (6.12) for a given positive integer k, it is sufficient to find E and Y such that (6.22), (6.23), and (6.24) hold with $p = 1 - \lambda/\gamma$ and $q = k\lambda \in (0, \min_i h_{x,i})$.

Proof: By substituting the expression of B_0 and applying the Schur complement to (6.22), we get that B_0 satisfies $B_0(Ax + BYE^{-1}x) \le pB_0(x) + p\gamma - \gamma$, $\forall x \in \mathcal{S}_{B_0}$. Applying the Schur

complement to (6.23), we observe that (6.23) is equivalent to $YE^{-1}x \in U$, $\forall x \in \mathcal{S}_{B_0}$. Applying the Schur complement to (6.24) yields that (6.24) is equivalent to $H_{x,i}x - h_{x,i} + q \leq 0$, $\forall x \in \mathcal{S}_{B_0}$ and $\forall i \in \{1,...,\dim(h_x)\}$. By letting $p = 1$ and $q = 0$, or letting $p = 1 - \lambda/\gamma$ and $q = k\lambda \in (0, \min_i h_{x,i})$, we prove the statements in Lemma 6.A.1. □

The condition $k\lambda \in (0, \min_i h_{x,i})$ indicates that the horizon k should not be selected too large when using (6.10) to generate a contractive-set CBF.

Maximizing the volume of the safe set $\mathcal{S}_{B_0}$ is equivalent to maximizing the determinant of E. Therefore, we solve the following convex optimization problem:

$$\begin{aligned} \min_{E,Y} \ & -\log\det(E) \\ \text{s.t. } & (6.22),\ (6.23),\ (6.24), \text{ and } E \succeq 0. \end{aligned} \tag{6.25}$$

After problem (6.25) is solved, the validity of the obtained CBF B_0 for the nonlinear system (6.1) is verified via

$$\begin{aligned} r \geq \max_{x \in \mathbb{R}^{n_x}} \ & f^T(x, YE^{-1}x)E^{-1}f(x, YE^{-1}x) - \gamma \\ \text{s.t. } & x^T E^{-1} x - \gamma \leq 0, \end{aligned} \tag{6.26}$$

where $r = 0$ for normal CBF verification and $r = -\lambda$ for λ-contractive-set CBF verification. In (6.26), if the condition of (6.26) does not hold, we decrease γ and repeat checking (6.26). As shown in [166, Section 2.5.5], there always exists a positive γ such that (6.26) holds.

6.A.3. DETAILED DESCRIPTION OF THE CONTROLLERS IN THE CASE STUDY

We use four different controllers in the simulation.

(i) MPC: We take the horizon $N = 5$, 6, or 7. For the stage cost $C_{\text{MPC}} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$, we let $C(x,u) = x^T Q x + Ru^2$ with $Q = \text{diag}([20\ 1])$ and $R = 1$. The terminal stage cost is $x_N^T Q_T x_N$ with Q_T obtained by solving the discrete algebraic Riccati equation for the subsystem $x_{t+1} = A_3 x_t + B u_t$ whose region contains the origin. The terminal constraint set is the maximal positively invariant set for the autonomous system $x_{t+1} = (A_3 - BK)x_t$ with K the solution to the above-mentioned discrete algebraic Riccati equation. Owing to the quadratic nature of the cost function and the PWA property of the predictive model, the MPC problem can be equivalently converted into a mixed-integer convex quadratic programming problem [35], and consequently, globally optimal solutions can be found efficiently by using the branch-and-bound approach [206]. We use Gurobi [92] to solve the MPC problem online.

(ii) Supervised learning MPC: Following the approach in [114], we use an explicit controller, more precisely, an NN with three hidden ReLU layers containing 16, 32, and 8 neurons, to approximate the implicit MPC control law u_{MPC}. In particular, we uniformly randomly sample the states from the state constraint set and select 4000 states that make the above-mentioned MPC problem with $N = 7$ feasible. Next, 4000 state-input pairs $\{x_i,\ u_{\text{MPC}}(x_i)\}_{i=1}^{4000}$ are obtained by solving the MPC problems with different initial conditions, and the NN is trained on these data pairs to approximate the map u_{MPC}.

(iii) Approximate dynamic programming: As demonstrated in [102], ADP can produce a safe control policy by adding penalty terms in the cost function. In the ADP scheme, the objective is to find a control policy $\pi : \mathcal{X} \to \mathcal{U}$ to minimize the infinite-horizon cost

$$J_\pi(x_0) = \sum_{t=0}^{\infty} C_{\mathrm{RL}}(x_t, \pi(x_t))$$

for any initial state $x_0 \in \mathcal{X}$. Following the existing literature [29], [102], the state cost is specified as $C_{\mathrm{RL}}(x,u) = C_{\mathrm{MPC}}(x,u) + 30\max\{0,|x_1|-0.15\} + 3\max\{0,|x_2|-1\}$, where the second and third terms penalize the state constraint violation. The ADP algorithm uses a policy NN and a value NN, which evaluates J_π, to find the optimal policy iteratively, and terminates when the parameters of the value NN undergo sufficiently slight changes in two consecutive iterations. One can see Algorithm 3 for details. Although the training of the ADP policy accommodates state constraints, the ADP policy can still cause constraint violation because (i) the penalty terms in C_{RL} are external penalties and thus do not strictly guarantee constraint satisfaction, and (ii) there are approximation errors in the policy and value NNs. It can be observed from Table 6.1 that the proposed safe filters improve the safety of the ADP policy.

(iv) LQR: the LQR feedback control law $u_{\mathrm{LQR}} = -Kx$, which is obtained when computing MPC in (i), can also be applied as the basic policy π_0 in the SFs (6.5) and (6.14). When testing u_{LQR} without any SF, we simply project the resulting input value onto the input constraint set, which is a convex quadratic program.

7

INTEGRATED LEARNING AND OPTIMIZATION FOR CONTROL OF CONSTRAINED NONLINEAR SYSTEMS: A MODEL-FREE APPROACH

Ensuring safety in the sense of constraint satisfaction for learning-based control is a critical challenge, especially in the model-free case. While safety filters address this challenge in the model-based setting by modifying unsafe control inputs, they typically rely on predictive models derived from physics or data. This reliance limits their applicability for advanced model-free learning control methods. To address this gap, we propose a new optimization-based control framework that determines safe control inputs directly from data. The benefit of the framework is that it can be updated through arbitrary model-free learning algorithms to pursue optimal performance. As a key component, the concept of direct data-driven safety filters (3DSF) is first proposed. The framework employs a novel safety certificate, called the state-action control barrier function (SACBF). We present three different schemes to learn the SACBF. Furthermore, based on input-to-state safety analysis, we present the error-to-state safety analysis framework, which provides formal guarantees on safety and recursive feasibility even in the presence of learning inaccuracies. The proposed control framework bridges the gap between model-free learning-based control and constrained control, by decoupling performance optimization from safety enforcement. Simulations on vehicle control illustrate the superior performance regarding constraint satisfaction and task achievement compared to model-based methods and reward shap-

ing.

This chapter is based on the paper [104].

7.1. INTRODUCTION

7.1.1. BACKGROUND

Learning-based control has achieved state-of-the-art performance in addressing complex problems in the presence of uncertainty, including applications in transportation systems [95] and robotics [98]. However, ensuring safety is still a challenging problem, particularly when an explicit model of the system is unavailable. Traditional model-based approaches to safety-critical control, such as model predictive control (MPC) [166], struggle with online computational efficiency and rely on the model, while emerging data-driven methods often lack well-understood safety guarantees.

In control problems, safety is typically defined as maintaining state and input variables within given constraints *throughout* the system's evolution. The difficulty lies in the fact that unsafe control policies do not necessarily immediately violate constraints but will lead to constraint violation in the future. As a fundamental principle in safe control, control invariance ensures that a system remains within a safe operating set, contained in the state constraint set. This property is crucial for guaranteeing long-term safety.

A widely adopted approach to enforcing long-term safety is the use of safety filters, which provide a modular framework that can be applied to any control policy, even those without explicit safety considerations [8], [55], [77], [81], [102], [149], [198]. The basic principle of safety filters is to post-process the input of a given control policy such that the resulting closed-loop system remains forward invariant w.r.t. the specified state and input constraints. The design of safety filters typically consists of two phases: an offline phase, where a safety certificate characterizing safe states is computed, and an online phase, where this certificate is incorporated as a constraint to modify potentially unsafe control actions from the reference controller. With the development of different kinds safety certificates, various kinds of safety filters have been proposed, such as control barrier function (CBF)-based safety filters [8], [77], [149], invariant set-based safety filters [102], Hamilton-Jacobi reachability-based safety filters [55], [81], and predictive safety filters [198]. Learning-based approaches have increasingly been used to synthesize safety certificates, particularly for complex, nonlinear systems with non-convex state and input constraints. For a detailed overview, we refer the reader to relevant work in the literature [47], [77], [172], [188], [205] as well as comprehensive surveys [69], [196].

Despite the differences and connections, almost all safety filters rely on a mathematical model, which is either exactly derived from physical principles or approximately estimated. In most formulations, model information is required in both the offline and online phases. Specifically, enforcing invariance conditions, first in the safety certificate and then in the control policy, requires explicit knowledge of the system dynamics. Recently, there has been a growing number of approaches focusing on offline construction of safety certificates using only state transition data [187]. However, these approaches cannot abandon the reliance on an explicit model during the online phase. This limitation is mainly due to the inherent property of existing safety certificates, which fully work on the state space. In particular, when using an existing safety certificate, safety filters need a prediction model to enforce safety conditions on the successor states. A detailed explanation is provided at the end of Section 7.2.

To overcome the above limitation, data-driven safety filters have received significant attention. Almost all existing data-driven safety filters belong to *indirect* data-driven methods [53], [75], [81], [188], [196], [198], [205]. The difference between direct and indirect methods was made in the context of the adaptive control community. In indirect approaches, first, system identification or disturbance estimation is performed, and then a controller is learned based on the obtained model. In direct methods, the controller is learned directly from data, bypassing the need for system identification. Indirect data-driven safety filters have one main issue: The errors arising from both model identification and certificate learning will compound, leading to a degradation in the safety performance of the filtered controller. Among all data-driven safety filters, there is only one *direct* data-driven formulation, which learns discriminating hyper-planes to directly regulate the control inputs [123]. However, this method is limited to linear safety constraints on inputs, potentially leading to conservative control actions if nonlinear constraints are considered, and lacks formal guarantees regarding constraint satisfaction.

An alternative approach bypasses model identification by jointly learning a CBF and an explicit policy that enforces the CBF constraint, but this often results in overly conservative policies focused solely on safety [69], [214].

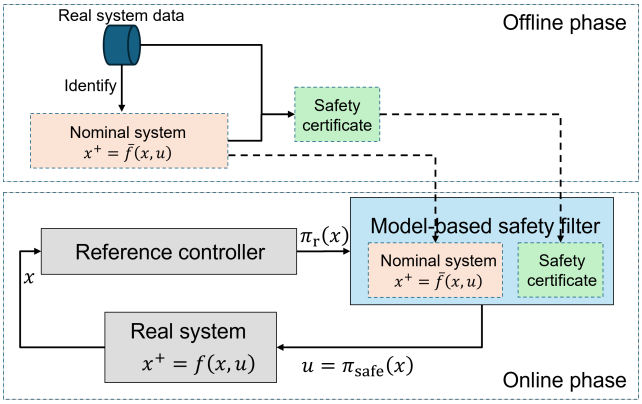
Similar to the distinction between indirect and direct data-driven control, learning-based control can also be categorized into model-based learning and model-free learning. Learning-based control, encompassing supervised learning (SL) and reinforcement learning (RL) [40], iteratively finds an optimal control policy that minimizes a predefined cost. Due to the stochastic nature of learning algorithms, learning-based control, especially in the absence of an explicit model, cannot fully guarantee safety without using safety filters to regulate policy execution. However, as almost all data-driven safety filters still rely on an underlying model, there still remains a gap when applying data-driven safety filters to learning-based control approaches that do not use an explicit model.

In this chapter, we focus on designing a direct data-driven safety filter (3DSF) based on our previously proposed concept of *state-action control barrier functions* (SACBFs) in Chapter 6. Unlike classical safety certificates, which only evaluate safety in the state space, the SACBF framework enables safety evaluation for each state-action pair. This key feature eliminates the need for explicit system dynamics during policy modification, making it particularly suitable for model-free learning-based control. The comparison between the proposed 3DSF and the existing indirect data-driven counterparts is illustrated in Fig. 7.1. For the indirect data-driven methods, both system identification and certificate learning should be performed one by one or simultaneously. The identified model and learned certificate are both used in the safety filter design. In comparison, the proposed 3DSF only uses an SACBF.

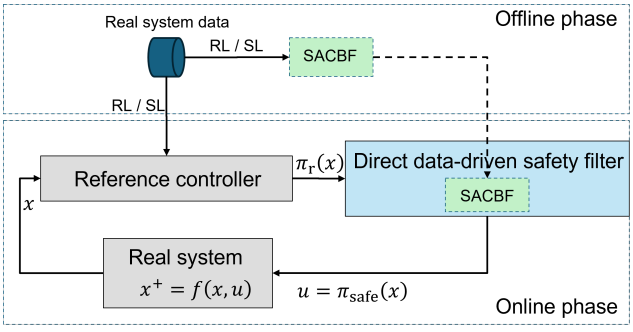
7.1.2. CONTRIBUTIONS

The chapter contributes to the state of the art in the following aspects:

1. **Optimization-based direct data-driven safe control:** We propose a novel safe



(a) The flow chart of indirect data-driven safe control using safety filters involving classical safety certificates.



(b) The flow chart of the proposed 3DSF, a special form of the optimization-based control framework we propose.

Figure 7.1: The comparison between the proposed 3DSF and the existing indirect data-driven counterparts.

control framework for general nonlinear systems with nonlinear constraints. The main advantage is that the safe controller can be *trained and implemented using state transition data only*, without the need for system identification. The control framework incorporates a novel safety filter, which we call the direct data-driven safety filter (3DSF). This framework, which can be integrated with arbitrary learning-based controller synthesis methods, also provides formal guarantees on constraint satisfaction.

2. **Learning-based synthesis of SACBFs:** We develop three distinct learning-based methods for synthesizing SACBFs directly from data: including SL from a known CBF, learning from an expert safe controller, and RL (self-learning).

3. **Robustness analysis via error-to-state safety (ESSf):** We propose a systematic framework, called Error-to-State Safety (ESSf), to analyze how learning-induced errors in SACBFs affect overall safety performance. The framework motivates the approach of state constraint tightening followed by SACBF constraint relaxation to ensure that the learned SACBF-regulated controller meets safety requirements.

4. **A unified framework extending RL to constrained control:** Building on the proposed optimization-based safe control, we present a general strategy to extend classical unconstrained value-based RL algorithms to constrained ones so that suboptimal performance regarding other tasks is achieved. This reversely shows another benefit of our proposed optimization-based approach: it separates the learning of the controller into two components: optimizing performance through learning the objective function, and ensuring safety through learning the constraints.

Additionally, for contribution (2), if system nonlinearities are known, we formulate the synthesis of a quadratic SACBF as solving a convex optimization problem with linear matrix inequality (LMI) constraints.

7.2. PRELIMINARIES AND PROBLEM FORMULATION

7.2.1. NOTATIONS

The set $\mathbb{R}$ ($\mathbb{R}_{\geq 0}$) is the set of (nonnegative) real numbers, and the set of real vectors with dimension n is denoted by $\mathbb{R}^n$. The sets $\mathbb{N}$ and $\mathbb{N}^+$ represent the set of non-negative integers and the set of positive integers, respectively. Besides, $\mathbb{N}_a = \{0, 1, ..., a\}$, and $\mathbb{N}_a^+ = \{1, ..., a\}$ for any positive integer a. The matrix I_n is the identity matrix with dimension $n \times n$. The notation $A^\dagger$ represents the right inverse of the matrix A. The relation $A \succeq 0$ means that the matrix A is positive semi-definite. The determinant of a matrix A is denoted by $\det(A)$. The dimension of a vector x is denoted by $\dim(x)$. The norm $||x||_2$ is the Euclidean norm of the vector x. The number of elements in a finite set S is called its cardinality, denoted by $|S|$. The set $\mathcal{B}_\epsilon(\bar{z}) := \{z \in \mathbb{R}^{n_z} \mid ||z - \bar{z}||_2 \leq \epsilon\}$ denotes the closed ball around $\bar{z}$ with radius ϵ. The unit step function $\text{step}(\cdot)$ returns 1 if the input is larger than 0 and returns 0 otherwise. A continuous function $\alpha(\cdot) : [0, a) \to [0, \infty)$ for some $a \in (0, \infty]$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. A class $\mathcal{K}$ function α is said to belong to class $\mathcal{K}_\infty$ if it further satisfies $a = \infty$ and $\alpha(r) \to \infty$ as $r \to \infty$.

7.2.2. PRELIMINARIES

We consider a deterministic discrete-time nonlinear system

$$x_{t+1} = f(x_t, u_t), \quad t = 0, 1, \ldots, \tag{7.1}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ are the state and the input at time step t, and $f(\cdot,\cdot) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a continuous function. We consider a constrained optimal control problem, in which the states and inputs should satisfy time-invariant constraints: $x_t \in X := \{x \in \mathcal{X} | h(x) \leq 0\}$ and $u_t \in U \subseteq \mathcal{U}$ for all time steps. Here, $h(\cdot) : \mathcal{X} \to \mathbb{R}$ is a scalar-valued continuous function that defines the state constraint[1]. The set U is compact. For the convenience of performance analysis and sampling, we require the compactness of $\mathcal{X}$ and X. In our framework, we do not assume the knowledge of the explicit form of the system dynamics $f(\cdot,\cdot)$. Instead, we require the availability of transition samples (x_t, u_t, x_{t+1}), achieved through simulation or experimental methods. Various sampling strategies, including random sampling and grid-based sampling, may be employed to obtain the transition triples.

Given an initial state x_0, we are interested in designing a deterministic control policy $\pi : \mathcal{X} \to \mathcal{U}$ to steer the trajectory of the system to a non-empty target region $X_{\text{tar}} \subseteq X$. To achieve the control objective, for any state x and input u, a stage cost is defined by $g(x,u)$, which is non-negative and upper-bounded. For any policy π, the value function $J^\pi(\cdot) : \mathcal{X} \to [0, \infty)$ is defined by

$$J^\pi(x) = \lim_{k\to\infty} \sum_{t=0}^{k} \gamma^t g(x_t, \pi(x_t)) \text{ s.t. (7.1) and } x_0 = x, \tag{7.2}$$

where $\gamma \in (0,1)$ is a discount factor. The objective is to find an optimal deterministic policy $\pi^*(\cdot) : \mathcal{X} \to \mathcal{U}$ that solves the following infinite-horizon optimal control problem:

$$J^*(x) := \inf_\pi J^\pi(x) \text{ s.t. } h(x_t) \leq 0, \pi(x_t) \in U, t = 0, 1, \ldots, \tag{7.3}$$

where $J^*(\cdot) : \mathcal{X} \to [0, \infty)$ is the optimal value function.

The exact form of the optimal policy π^* is difficult to compute due to the following reasons: (i) The number of constraints in (7.3) is infinite; (ii) The closed-form expressions of the objective function are unknown since the system is unknown and the horizon is infinite. To get an approximation of $\pi^*(\cdot)$, a parameterized policy is usually preferred in the literature and then the parameters are updated using RL or SL [40], [41].

7.2.3. CONTROL BARRIER FUNCTIONS FOR SAFETY

To ensure that the learned policy satisfies the constraints in (7.3), two different classes of methods, including cost shaping [143] and using safety certificates [3], [8], [77], have been developed in recent papers. In contrast to the cost-shaping method, which usually does not provide strict safety guarantees, using safety certificates can impose a constraint on the control input to ensure safety. The CBF is one of the most popular safety certificates.

[1] For the constraint defined by multiple inequalities $h_i(x) \leq 0,\ i = 1, 2, ..., I$, we can let $h(x) = \max_{i \in \mathbb{N}_I^+} h_i(x)$. The set $\{x | h(x) \leq 0\}$ is identical to $\{x | h_i(x) \leq 0,\ \forall i \in \mathbb{N}_I^+\}$, and h will be continuous if each h_i is continuous.

Definition 7.2.1 (Control barrier function [3])**.** A function $B(\cdot): \mathcal{X} \to \mathbb{R}$ is called a *control barrier function* (CBF) with a corresponding safe set $\mathcal{S}_B := \{x \in \mathbb{R}^{n_x} | B(x) \leq 0\} \subseteq \mathcal{X}$, if $\mathcal{S}_B$ is non-empty, $h(x) \leq B(x),\ \forall x \in \mathcal{X}$, and if there exists a $\beta_B \in [0,1]$ such that

$$\min_{u \in U} B(f(x,u)) \leq \beta_B B(x),\ \forall x \in \mathcal{X}. \tag{7.4}$$

With a CBF available, one can implicitly construct a policy $\pi(\cdot)$ as an optimizer of the following nonlinear optimization problem:

$$\begin{aligned} \pi(x) := \arg\min_{u \in U}\ & Q(x,u) \\ \text{s.t. } & B(f(x,u)) \leq \beta_B B(x), \end{aligned} \tag{7.5}$$

where $Q: \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a problem-dependent objective function. For instance, one can specify Q as $||u - \pi^{\text{unsafe}}(x)||_2$ where π^{unsafe} is a reference policy that may have some other acceptable performance. With this specification, (7.5) works as a safety filter to refine the potentially unsafe policy π^{unsafe}. One can also let Q be an approximation of the state-action optimal value function (Q function) of (7.3), that is,

$$Q^*(x,u) := g(x,u) + \gamma J^*(f(x,u)), \tag{7.6}$$

which is commonly used in RL. In this situation, π becomes the greedy policy w.r.t. the optimal value function under a given CBF constraint.

A valid CBF is sufficient to guarantee the safety of π [8]. However, the main limitation of (7.5) is that the knowledge of f is required. Even though some data-driven methods exist for learning CBFs from black-box models [187], system identification is necessary to implement (7.5). This limitation significantly restricts the application of safe filters and CBFs in reinforcement learning algorithms that do not require an explicit model. Moreover, the mismatch between the identified model and the real system, along with inaccuracies in learning CBFs, jointly affect the safety performance of π.

7.3. STATE-ACTION CBFS AND CONTROL PARAMETERIZATION

Inspired by the Q function in RL, we propose a new optimization-based control framework that does not contain f:

$$\begin{aligned} \pi(x) := \arg\min_{u \in U}\ & Q(x,u) \\ \text{s.t. } & Q^B(x,u) \leq 0, \end{aligned} \tag{7.7}$$

where Q can be designed similarly to that in (7.5), and another function $Q^B(\cdot,\cdot): \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is used to add a constraint to regulate the behavior of π. To make π satisfy the safety requirements $h(x_t) \leq 0,\ \forall t \in \mathbb{N}$, we consider the definition of state-action CBFs, which we have introduced in Chapter 6 and [100].

Definition 7.3.1 (State-action control barrier function (SACBF))**.** A function $Q^B(\cdot,\cdot): \mathcal{X} \times U \to \mathbb{R}$ is called a *state-action control barrier function* (SACBF) with a corresponding safe set $\mathcal{S}_Q$ of states, if the pair $(Q^B, \mathcal{S}_Q)$ satisfies the following conditions:

(i) $\mathcal{S}_Q$ is non-empty, and $h(x) \leq 0,\ \forall x \in \mathcal{S}_Q$.

(ii) $\min_{u \in U} Q^B(x,u) \leq 0,\ \forall x \in \mathcal{S}_Q$.

(iii) For any $x \in \mathcal{S}_Q$, any $u \in U$ satisfying $Q^B(x,u) \leq 0$ ensures that $f(x,u) \in \mathcal{S}_Q$.

According to the definition of SACBFs, we have the following properties for (7.7), which we have introduced in Chapter 6.

Lemma 7.3.1 (Safety of SACBFs [100])**.** Considering the policy π in (7.7), if Q^B is an SACBF with the safe set $\mathcal{S}_Q$, π will render (7.1) positively invariant in $\mathcal{S}_Q$. As a result, the trajectories of (7.1), starting from $x_0 \in \mathcal{S}_Q$, controlled by π, satisfy $h(x_t) \leq 0$ and $u_t \in U,\ \forall t \in \mathbb{N}$.

Remark 7.3.1. Analogous to CBFs, SACBFs are difficult to compute exactly for nonlinear problems. In the remainder of this chapter, we will explore using learning-based methods to synthesize the optimization-based controller, i.e., to learn the functions Q and Q^B. A tractable way is to parameterize them by Q_θ and Q^B_ω with the parameters θ and ω, and then learn these parameters. Instead of simultaneously updating Q and Q^B, we prefer to first learn an SACBF Q^B_ω (Section 7.4) and subsequently to integrate it into the learning of the objective function Q_θ to achieve the optimal control objective (Section 7.6). This strategy is motivated by the observation that safety is inherently independent of other performance metrics, whereas achieving optimality regarding task performance should be addressed under the premise of ensuring safety.

The policy π in (7.7) with the parameterization Q_θ and Q^B_ω is denoted by $\pi_{\theta,\omega}$.

7.4. LEARNING STATE-ACTION CONTROL BARRIER FUNCTIONS

In this section, we propose three learning-based approaches to approximate an SACBF Q^B_ω. For the first one, we assume the knowledge of a valid CBF and use SL to obtain an SACBF. The second approach, inspired by [172], employs sampling-based methods to approximate the solution of a semi-infinite optimization problem. Such a solution is guaranteed to lead to a valid SACBF. This approach requires prior knowledge of a safety control policy (usually conservative w.r.t. task performance). For the third approach, we connect the synthesis of SACBFs with Hamilton-Jacobi reachability [82], and thereby propose an RL-based approach to obtain Q^B_ω. The third approach only relies on the availability of state transition data. Besides, when the nonlinearity of the system is known, we also find that the synthesis of SACBFs can be achieved by solving a convex optimization problem with LMI conditions.

7.4.1. SUPERVISED LEARNING SACBFS FROM CBFS

We first present our simplest learning-based method for obtaining an SACBF, based on a rather restrictive assumption that we have a CBF at hand. In certain applications, such as adaptive cruise control [8], CBFs can be manually designed based on state constraints, such as the distance between adjacent vehicles. However, as noted in the introduction, such CBFs cannot be used to design a safety filter (7.5) when the model is not fully known. In this subsection, we explore connections between SACBFs and CBFs and demonstrate that one can easily learn an SACBF from a given CBF.

Assumption 7.4.1. A CBF $B(\cdot): \mathcal{X} \to \mathbb{R}$ is known.

Proposition 7.4.1. Under the constraints $h(x) \le 0$ and $u \in U$, suppose that B is a CBF satisfying (7.4) with $\beta_B \in [0,1]$, and that the safe set is $\mathcal{S}_B$. Then, Q^B defined by

$$Q^B(x,u) := \max\{h(x), \frac{1}{\beta_B} B(f(x,u))\} \tag{7.8}$$

is an SACBF with the safe set $\mathcal{S}_Q = \mathcal{S}_B$.

Proof. From the relation between Q^B and B, we can get that $h(x) \le \min_{u\in U} Q^B(x,u) \le 0$, $\forall x \in \mathcal{S}_B$. The above arguments prove that Q^B obeys (i) and (ii) of Definition 7.3.1. Furthermore, for any $x \in \mathcal{S}_B$, if $Q^B(x,u) \le 0$, we have $B(f(x,u)) \le 0$ and thus $f(x,u) \in \mathcal{S}_B$. This finishes the proof. □

Proposition 7.4.1 implies a straightforward way to learn an SACBF. Initially, a collection of transition triples $\{(x^{(i)}, u^{(i)}, f(x^{(i)}, u^{(i)}))\}_{i=1}^N$, is generated from the state space $\mathcal{X}$ and the control space U. Subsequently, the labels $Q^B(x^{(i)}, u^{(i)})$ can be computed utilizing (7.8). Following this, a regression model Q^B_ω is constructed and trained to minimize the empirical loss, specifically the mean squared error between Q^B_ω and the computed labels.

7.4.2. LEARNING SACBFS FROM AN EXPERT CONTROLLER

In the presence of both state and input constraints, manually crafted CBFs often fail. Towards the goal of learning a valid SACBF, like in [172] where CBFs are synthesized from expert demonstrations, we assume the availability of an expert safe controller $\pi_s(\cdot): \mathcal{X} \to \mathcal{U}$ in this subsection. This assumption is reasonable in certain scenarios, such as human-controlled systems (e.g., driving), where safety is achieved implicitly without an explicit model.

Formally, we have the following assumption in this subsection.

Assumption 7.4.2. There exists a continuous policy π_s and a compact set $\mathcal{S}_0 \subseteq X$ ($\mathcal{S}_0$ can be unknown) such that with the initial condition $x_0 \in \mathcal{S}_0$, the state-input trajectories of the system (7.1) with π_s always stay in $\mathcal{S}_0 \times U$ and such that the states reach X_{tar} in a finite number of time steps T.

Under Assumption 7.4.2, we formulate the synthesis of Q^B as the following optimization problem:

$$\min_{Q^B, q} \int_{x\in\mathcal{X}} q(x)dx \tag{7.9a}$$

$$\text{s.t. } 0 \le q(x),\ \forall x \in \mathcal{X} \setminus \mathcal{S}_0 \tag{7.9b}$$

$$Q^B(x, \pi_s(x)) \le \beta q(x), \forall x \in \mathcal{S}_0 \tag{7.9c}$$

$$q(f(x,u)) \le Q^B(x,u), \forall (x,u) \in \mathcal{S}_0 \times U, \tag{7.9d}$$

where Q^B and $q(\cdot): \mathcal{X} \to \mathbb{R}$ are continuous in their domain, and $\beta \in [0,1)$ is a tuning parameter.

Proposition 7.4.2 (Converse SACBFs)**.** Under Assumption 7.4.2, there exists a $\beta \in [0,1)$ such that the problem (7.9) is feasible and such that any optimal solution pair (Q^{B*}, q^*) ensures that Q^{B*} is an SACBF with the safe set $\mathcal{S}_Q^* = \{x \in \mathcal{X} | q^*(x) \leq 0\}$.

Proof. The proof consists of two parts. In the first part, we will prove the feasibility of (7.9). By fixing $Q(x,u) = q(f(x,u))$, which satisfies (7.9d), the constraint (7.9c) becomes $q(f(x,\pi_s(x))) \leq \beta q(x),\ \forall x \in \mathcal{S}_0$.

Since $\mathcal{S}_0$ is compact, there exists a continuous function q such that (7.9b) and

$$\begin{aligned} \mathcal{S}_0 &= \{x \in \mathcal{X} | q(x) \leq 0\}, \\ \partial\mathcal{S}_0 &= \{x \in \mathcal{X} | q(x) = 0\}, \\ \text{Int}(\mathcal{S}_0) &= \{x \in \mathcal{X} | q(x) < 0\} \end{aligned} \tag{7.10}$$

hold. Actually, the above statement can be proved by considering $q(x) = \max\{h(x), \bar{q}(x)\}$, where $\bar{q}$ is a distance function defined by

$$\bar{q}(x) = \begin{cases} -\inf_{y \in \partial\mathcal{S}_0} \|x - y\| \text{ if } x \in \mathcal{S}_0 \\ \inf_{y \in \partial\mathcal{S}_0} \|x - y\| \text{ otherwise.} \end{cases} \tag{7.11}$$

With the above definitions, we follow the proof of the Converse CBF Theorem in the continuous time domain [8, Proposition 3] to prove the feasibility of (7.9). In particular, for any continuous q such that (7.9b) and (7.10) hold, define a function $\alpha(\cdot) : [0,\infty) \to \mathbb{R}$ by

$$\alpha(r) = \sup_{\{x \in \mathcal{X} | -r \leq q(x) \leq 0\}} q(f(x,\pi_s(x))) - q(x). \tag{7.12}$$

Since $\{x \in \mathcal{X} | -r \leq q(x) \leq 0\}$ is compact and q, f, and π_s are continuous, α is well-defined and non-decreasing, and satisfies $q(f(x,\pi_s(x))) - q(x) \leq \alpha(-q(x)),\ \forall x \in \mathcal{S}_0$.

When x is such that $q(x) = 0$, i.e., $x \in \partial\mathcal{S}_0$, by the invariance of $\mathcal{S}_0$, we have $q(f(x,\pi_s(x))) \leq 0$, and consequently $\alpha(0) \leq 0$. Meanwhile, by taking the extreme values 0 and $-r$ of $q(f(x,\pi_s(x)))$ and $q(x)$ respectively, we have $\min_{r>0} \frac{\alpha(r)}{r} \leq \min_{r>0} \frac{0-(-r)}{r} = 1$, which, combined with $\alpha(0) \leq 0$, implies that there exists a class $\mathcal{K}$ function $\bar{\alpha}$ upper-bounding α and satisfying

$$\begin{aligned} &q(f(x,\pi_s(x))) - q(x) \leq \bar{\alpha}(-q(x)), \forall x \in \mathcal{S}_0 \\ &\min_{r>0} \frac{\bar{\alpha}(r)}{r} \leq 1. \end{aligned}$$

Letting

$$\beta = 1 - \min_{r>0} \frac{\bar{\alpha}(r)}{r} \in [0,1)$$

proves the feasibility of (7.9).

In the second part of the proof, we will show that the optimal solution to (7.9) yields an SACBF. To this end, we first establish the non-emptiness of $\mathcal{S}_Q^*$ by contradiction. Suppose that $\mathcal{S}_Q^*$ is empty, i.e., $q^*(x) > 0,\ \forall x \in \mathcal{X}$. Consider a new function $q'(\cdot) : \mathcal{X} \to \mathbb{R}$ defined by

$q'(x) = \min\{q^*(x), \max\{h(x), \bar{q}(x)\}\}$, where $\bar{q}$ is defined by (7.11). Based on the proof of the feasibility of (7.9), $\max\{h(x), \bar{q}(x)\}$ satisfies all the constraints in (7.9). Therefore, we have

$$q'(x^+) \leq \min\{\beta q^*(x), \beta \max\{h(x), \bar{q}(x)\}\} \leq \beta q'(x),\ \forall x \in \mathcal{S}_0$$

with $x^+ = f(x, \pi_s(x))$. This implies that the new function q' satisfies all the constraints in (7.9). Moreover, since the zero sub-level set of $\max\{h, \bar{q}\}$ is non-empty, the zero sub-level set of q' is non-empty as well. Furthermore, it strictly holds that $\int_{x\in\mathcal{X}} q'(x)dx < \int_{x\in\mathcal{X}} q^*(x)dx$ because (i) $q'(x) \leq q^*(x)$, $\forall x \in \mathcal{X}$, and (ii) $q^* > 0$ and $q' \leq 0$ in the zero sub-level set of q'. These arguments prove that the new function q' provides a better solution than q^*, which contradicts the optimality of q^*.

The condition (i) of Definition 7.3.1 holds because of (7.9b) and the non-emptiness of $\mathcal{S}_Q^*$. Moreover, $\forall x \in \mathcal{S}_Q^*$, it follows from (7.9c) that $\min_{u\in U} Q^{B*}(x,u) \leq Q^{B*}(x, \pi_s(x)) \leq \beta q(x) \leq 0$, meaning that the condition (ii) holds. Furthermore, (7.9d) enforces the condition (iii). These complete the proof that Q^{B*} is an SACBF. □

For computational tractability, Q^B and q in (7.9) are parameterized as Q_ω^B and q_ω, with all the parameters condensed in ω. As a result, the search space becomes the parameter space of ω. To solve (7.9), we use state transition samples to relax the constraints that should hold in the continuous state(-input) space to constraints that only need to hold for the samples.

7

In particular, let $\left\{x_0^{(i)}\right\}_{i=1}^N$ denote the set of initial states. Here, N is the number of samples. For each $x_0^{(i)}$, we apply π_s to (7.1) for T time steps and get the state-input trajectory $\left\{\left(x_t^{(i)}, \pi_s(x_t^{(i)})\right)\right\}_{t=0}^T$. If this trajectory obeys $x_t^{(i)} \in X$, $\pi_s\left(x_t^{(i)}\right) \in U$, $\forall t \in \mathbb{N}_T$, and $x_T^{(i)} \in X_{\text{tar}}$, then $x_t^{(i)} \in \mathcal{S}_0$, $\forall t \in \mathbb{N}_T$. Otherwise, $x_t^{(i)} \notin \mathcal{S}_0$, $\forall t \in \mathbb{N}_T$. The above checking procedure allows us to separate the state trajectories according to whether or not they are in $\mathcal{S}_0$, without knowing $\mathcal{S}_0$. Then, we define the set $\mathcal{S}_s := \left\{x_t^{(i)} | x_t^{(i)} \in X, \pi_s\left(x_t^{(i)}\right) \in U, \forall t \in \mathbb{N}_T, x_T^{(i)} \in X_{\text{tar}}, \forall i \in \mathbb{N}_N^+\right\}$, which contains the safe state trajectories. Meanwhile, we define $\mathcal{X}_s := \left\{x_t^{(i)}\right\}_{i=1\ t=0}^{N\ \ T}$.

For the optimization problem (7.9), we replace the infinite sets $\mathcal{S}_0$, $\mathcal{X}$, and $\mathcal{S}_0 \times U$ by the finite sets $\mathcal{S}_s$, $\mathcal{X}_s$, and $\mathcal{S}_s \times U_s$, respectively. Here U_s is the set of inputs sampled in U through some sampling strategy. As a result, (7.9) is simplified to a constrained optimization problem with a finite number of constraints. If Q_ω^B and q are linear in ω, the problem (7.9) becomes linear and can be solved efficiently. In a more general case, such as when these functions are represented by deep neural networks, a tractable solution involves transforming the constraints into soft penalties and adding them to the objective.

Remark 7.4.1. Due to the above constraint relaxation, the validity of the approximated solution is, however, not guaranteed. To get a valid SACBF, we can follow the approach of [172] to tighten the constraints in (7.9). Assuming Q_ω^B, q_ω, π_s, and f to be Lipschitz as

well as having sufficient sampling, Q^B_ω will be a valid SACBF[2]. Although this constraint tightening approach will guarantee that $Q^B_{\omega^*}$ is a valid SACBF, it may be conservative under insufficient sampling. Furthermore, estimating the Lipschitz constants is a non-trivial task. To address this limitation, we will propose a new tightening approach that exploits the inherent robustness of the SACBF. This will be elaborated on in Section 7.5.

7.4.3. LEARNING SACBFS VIA RL

In the previous subsection, we have proposed an optimization-based approach to learn SACBFs from a known safe controller. This approach has two main limitations. First, a safe controller is sometimes hard to obtain as a prior. Second, the safe set of the learned SACBF cannot be larger than the safe region $\mathcal{S}_0$ of the safe controller. On the other hand, it is desirable to learn an SACBF with the largest possible safe set directly from randomly generated transition data, without any expert supervision. For this purpose, we observe that the proposed SACBFs show some similarities with state-action value functions (Q functions) in RL. These similarities motivate us to use RL methods to synthesize SACBFs.

We consider the optimal value function of the following optimal control problem:

$$\begin{aligned} B^*(x) := \min_{\{(x_t,u_t)\}_{t=0}^{\infty}} \ & \max_{t\in\mathbb{N}} h(x_t) \\ & \text{s.t. (1)},\ u_t \in U,\ t\in\mathbb{N}, \text{ and } x_0 = x, \end{aligned} \tag{7.13}$$

which is a typical reachability problem [82] in the discrete-time domain.

Remark 7.4.2. The optimal value function B^*, which can be obtained by dynamic programming, is a CBF with the safe set being the maximal control-invariant set [55]. Using Proposition 7.4.1 and letting $\beta_B = 1$, we know that the state-action optimal value function of (7.13), defined by

$$Q^{B*}(x,u) := \max\{h(x),\ B^*(f(x,u))\}, \tag{7.14}$$

is an SACBF, of which the safe set is also the maximal control-invariant set.

Combining (7.13) and (7.14), it is observed that Q^{B*} is one solution of the Bellman optimality equation [40]:

$$Q^B(x,u) = \max\left\{h(x), \min_{u^+\in U} Q^B(f(x,u),u^+)\right\}. \tag{7.15}$$

For general nonlinear systems, accurately computing Q^{B*} from (7.13) and (7.14) is nearly impossible. Similar to the proposed SL method and the proposed expert-guided learning method, an approximation of Q^{B*} is necessary. Since the form of f is unknown, it is not possible to design query-based algorithms and use SL methods to get the approximation. However, by employing (7.15) and the theoretical results in Proposition 7.4.1, we can adapt off-the-shelf value-based RL methods such as temporal difference learning methods [15], [203] and neural fitted Q iteration (FQI) [170] to approximate Q^{B*}.

[2] "Sufficient sampling" means that $\mathcal{X}_s \times U_s$ constitutes an ϵ-net of $\mathcal{X}\times U$ with $\epsilon > 0$ [172]. Namely, $\forall x \in \mathcal{X}$ and $\forall u \in U$, $\exists x' \in \mathcal{X}_s$ and $\exists u' \in U_s$ such that $||x-x'||_2^2 + ||u-u'||_2^2 \le \epsilon^2$.

Before learning Q^{B*}, it is crucial to recognize that (7.15) may have multiple solutions and that most RL methods based on Bellman iteration could approach any solution of (7.15). The following result eliminates our concern by stating that any solution of (7.15), satisfying $\min_{x\in\mathcal{X},u\in U} Q^B(x,u)\le 0$, is an SACBF.

Proposition 7.4.3. Consider the Bellman optimality equation (7.15). Any solution Q^B satisfying $\min_{x\in\mathcal{X},u\in U} Q^B(x,u)\le 0$ is an SACBF with the safe set $\mathcal{S}_Q=\{x\in\mathbb{R}^{n_x}|\min_{u\in U} Q^B(x,u)\le 0\}$.

Proof. It follows from $\min_{x\in\mathcal{X},u\in U} Q^B(x,u)\le 0$ that $\mathcal{S}_Q$ is non-empty. From (7.15), we deduce that $h(x)\le \min_{u\in U} Q^B(x,u)\le 0,\ \forall x\in\mathcal{S}_Q$. The above arguments prove that Q^B satisfies (i) of Definition 7.3.1. The second condition of Definition 7.3.1 directly follows from $\mathcal{S}_Q=\{x\in\mathbb{R}^{n_x}|\min_{u\in U} Q^B(x,u)\le 0\}$. Finally, if $Q(x,u)\le 0$, by (7.15) we have $\min_{u^+\in U} Q^B(f(x,u),u^+)\le 0$, which means that the successor state $f(x,u)\in\mathcal{S}_Q$. □

Following the temporal difference learning method in [15], the training loss for the candidate SACBF Q^B_ω is designed as

$$\begin{aligned} l(\mathcal{D},\omega) &= l_1(\mathcal{D},\omega)+\rho\, l_2(\mathcal{D},\omega), \text{ with}\\ l_1(\mathcal{D},\omega) &= \sum_{(x,u,x^+)\in\mathcal{D}}\left(\max\left\{h(x),\min_{u^+\in U} Q^B_\omega(x^+,u^+)\right\}-Q^B_\omega(x,u)\right)^2\\ l_2(\mathcal{D},\omega) &= \sum_{(x,u,x^+)\in\mathcal{D}} Q^B_\omega(x,u), \end{aligned} \tag{7.16}$$

where $\mathcal{D}=\left\{\left(x^{(i)},u^{(i)},f\left(x^{(i)},u^{(i)}\right)\right)\right\}_{i=1}^N$ represents the collection of state transition triples. Intuitively, minimizing the loss l_1 encourages Q^B_ω to reduce the temporal difference, thereby approaching the solution of (7.15). Meanwhile, as the Bellman equation (7.4.3) may have multiple solutions, the loss l_2 is introduced to guide Q^B_ω toward the smallest solution of (7.15), which has the largest safe set. The positive parameter ρ balances the relative importance of the two loss functions.

7.4.4. SPECIAL CASE WHEN NONLINEARITY IS KNOWN

The methods for synthesizing SACBFs proposed above are aimed at general nonlinear systems. When the nonlinearity of the system is known, we can leverage formulas from data-driven control [71], [72] and compute SACBFs through convex optimization. Formally, in this subsection, we require the system to satisfy the following conditions:

Assumption 7.4.3.

- The system is described by the following equation:

$$x_{t+1}=A\bar{v}(x_t)+Bu_t, \tag{7.17}$$

where $\bar{v}(x)=\begin{bmatrix} x\\ v(x)\end{bmatrix}$ includes a *known* nonlinear function $v(\cdot):\mathbb{R}^{n_x}\to\mathbb{R}^{n_v}$. Besides, the linear system obtained by linearizing (7.17) at the origin is controllable in the absence of the constraints.

- The constraints are linear, i.e., $X=\{x\in\mathbb{R}^{n_x}|H_x x\le h_x\}$, $U=\{u\in\mathbb{R}^{n_u}|H_u u\le h_u\}$. Here, H_x and H_u are matrices with appropriate dimensions. Besides, h_x and h_u

are two vectors with strictly positive elements. This indicates that the origin is contained in the interior of $X \times U$.

- The target region $X_{\text{tar}} = \{0\}$.

To begin with, we recall the definition of Hankel matrix [71] associated to any time-varying signal $\{z_t\}_{t\in\mathbb{N}}$ with $z_t \in \mathbb{R}^{n_z}$, $t \in \mathbb{N}$:

$$\mathbf{z}_{i,j,k} = \begin{bmatrix} z_i & z_{i+1} & \cdots & z_{i+k-j} \\ z_{i+1} & z_{i+2} & \cdots & z_{i+k-j+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{j+i-1} & z_{j-i} & \cdots & z_{k+i-1} \end{bmatrix}.$$

Then, the Hankel matrices $\mathbf{x}_{i,j,k}$ and $\mathbf{u}_{i,j,k}$ contain the state and input data collected from a simulation of the system in (7.17). The following lemma shows that (7.17) has equivalent data-based representations.

Lemma 7.4.1 ([72])**.** Let $\mathbf{D} := \begin{bmatrix} \mathbf{u}_{0,1,k} \\ \bar{\mathbf{v}}_{0,1,k} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{0,1,k} \\ \mathbf{x}_{0,1,k} \\ \mathbf{v}_{0,1,k} \end{bmatrix}$. Suppose that $\text{rank}(\mathbf{D}) = n_x + n_u + n_v$ and that Assumption 7.4.3 holds. Then, the system in (7.17) has the following equivalent representation:

$$x_{t+1} = \mathbf{x}_{1,1,k}\mathbf{D}^{\dagger} \begin{bmatrix} u_t \\ \bar{v}(x_t) \end{bmatrix}. \tag{7.18}$$

Furthermore, the system in closed loop with $u = K\bar{v}(x)$ has the following equivalent representation:

$$x_{t+1} = \mathbf{x}_{1,1,k} G \bar{v}(x_t) \tag{7.19}$$

with $G = [G_1\ G_2]$, $G_1 \in \mathbb{R}^{k\times n_x}$, and $G_2 \in \mathbb{R}^{k\times n_v}$ satisfying

$$\begin{bmatrix} K \\ I_{n_x+n_v} \end{bmatrix} = \mathbf{D}G. \tag{7.20}$$

In Lemma 7.4.1, the rank condition on $\mathbf{D}$ is referred to as the condition of *persistency of excitation*, which is commonly assumed among direct data-driven control approaches [71]. This condition implies that the collected data is sufficiently informative to represent the system behavior. Next, we show that under persistency of excitation and a mild assumption on the nonlinear term v (See Assumption 7.4.4 below), we can formulate a data-driven computation of an SACBF in terms of a convex optimization problem subject to LMI constraints.

Assumption 7.4.4. There exist a constant $\eta > 0$ and a matrix $M \in \mathbb{R}^{n_x\times n_x}$ such that $v^T(x)v(x) \le \eta x^T M^T M x$, $\forall x \in X$.

This assumption limits the magnitude of the nonlinear term $v(x)$ by a quadratic function $\eta x^T M^T M x$. Such an assumption is commonly encountered in the literature addressing the stabilization of nonlinear systems using LMIs [181].

Consider

$$Q_P^B(x,u) := \max\left\{ h(x), [u^T \ \bar{v}^T(x)]\mathbf{D}^{\dagger T}\mathbf{x}_{1,1,k}^T P^{-1}\mathbf{x}_{1,1,k}\mathbf{D}^{\dagger}\begin{bmatrix} u \\ \bar{v}(x)\end{bmatrix} - 1 \right\}, \tag{7.21}$$

where $P \in \mathbb{R}^{n_x \times n_x}$ is a positive-definite matrix. The consideration of the parameterization (7.21) is inspired by Proposition 7.4.1 and (7.18), in which we specify $h(x) = \max_i\{H_{x,i}x - h_{x,i}\}$ so that $\{x \in \mathbb{R}^{n_x} | h(x) \le 0\}$ is identical to $\{x \in \mathbb{R}^{n_x} | H_x x \le h_x\}$. The second argument of the max function indicates that we firstly consider a quadratic CBF candidate $x^T P^{-1} x - 1$, and then use the open-loop representation (7.18) and the result in Proposition 7.4.1 to get an SACBF candidate.

Now, we consider the following convex optimization problem:

$$\min_{P,R_1,G_2} \ -\log\det(P) \tag{7.22a}$$

$$\text{s.t.} \begin{bmatrix} P & 0 & R_1^T\mathbf{x}_{1,1,k}^T & PM^T \\ * & I_{n_v} & G_2^T\mathbf{x}_{1,1,k}^T & 0 \\ * & * & P & 0 \\ * & * & * & I_{n_x}/\eta \end{bmatrix} \ge 0 \tag{7.22b}$$

$$\begin{bmatrix} h_{u,i}^2 P & 0 & R_1^T\mathbf{u}_{0,1,k}^T H_{u,i}^T & PM^T \\ * & I_{n_v} & G_2^T\mathbf{u}_{0,1,k}^T H_{u,i}^T & 0 \\ * & * & 1 & 0 \\ * & * & * & I_{n_x}/\eta \end{bmatrix} \ge 0, \ i = 1,...,\dim(h_u) \tag{7.22c}$$

$$\begin{bmatrix} h_{x,i}^2 & H_{x,i}P \\ * & P \end{bmatrix} \ge 0, \ i = 1,...,\dim(h_x) \tag{7.22d}$$

$$\begin{bmatrix} \mathbf{x}_{0,1,k} \\ \mathbf{v}_{0,1,k} \end{bmatrix} R_1 = \begin{bmatrix} P \\ 0 \end{bmatrix} \tag{7.22e}$$

$$\begin{bmatrix} \mathbf{x}_{0,1,k} \\ \mathbf{v}_{0,1,k} \end{bmatrix} G_2 = \begin{bmatrix} 0 \\ I_{n_v} \end{bmatrix} \tag{7.22f}$$

$$P \ge 0, \ R_1 \in \mathbb{R}^{k\times n_x}, \ G_2 \in \mathbb{R}^{k\times n_v}, \tag{7.22g}$$

where "*" indicates that the lower triangular part of the matrix is the symmetric counterpart of the upper triangular part, and $H_{x,i}$ and $h_{x,i}$ refer to the ith row of H_x and h_x, respectively. Based on the optimization problem (7.22), we can construct an SACBF Q_P^B, as stated in the next result.

Proposition 7.4.4. Consider the system (7.17), the SACBF parameterization (7.21), and the optimization problem (7.22). Suppose that $\text{rank}(\mathbf{D}) = n_x + n_u + n_v$ and that Assumptions 7.4.3-7.4.4 hold. Then, problem (7.22) is feasible, and its optimal solution P^* makes $Q_{P^*}^B$ an SACBF with the corresponding safe set $\{x \in \mathbb{R}^{n_x} | x^T P^{*-1} x \le 1\}$.

Proof. The proof is based on the standard argument of using LMIs to synthesize CBFs [100]. By letting $R_1 = G_1 P$ and $K = \mathbf{u}_{0,1,k} G$, (7.22e) and (7.22f) are equivalent to (7.20).

We consider the condition

$$x^{+T} P^{-1} x^+ \le 1, \ \forall x^T P^{-1} x \le 1, \ \forall v^T(x) v(x) \le \eta x^T M^T M x, \tag{7.23}$$

where x^+ is the successor state of the closed-loop system (7.19). By sequentially using the S-procedure [36], utilizing the Schur complement [36], multiplying both sides by $\mathrm{diag}([P\ I_{n_v}\ I_{n_x}])$, and applying the Schur complement once more, we get that (7.22b) is sufficient to ensure (7.23). The condition (7.23) further implies the invariance of the safe set for the closed-loop system (7.19).

Then, by a similar argument, we can prove that (7.22c) is a sufficient condition for

$$H_u K\bar{v}(x) \leq h_u,\ \forall x \in \{x \in \mathbb{R}^{n_x} | x^T P^{-1} x \leq 1\} \tag{7.24}$$

to hold. The condition (7.24) means that the controller $u = K\bar{v}(x)$ satisfies the input constraint for all states in the safe set. Besides, (7.22d) is equivalent to the condition

$$\{x \in \mathbb{R}^{n_x} | x^T P^{-1} x \leq 1\} \subseteq X. \tag{7.25}$$

The above arguments prove that $B := x^T P^{*-1} x - 1$ is a CBF. Meanwhile, by comparing the right-side of (7.21) and the form of B, and noting that the open loop system (7.17) can be represented by (7.18), we have the relation $Q^B_{P^*}(x,u) = \max\{h(x), B(f(x,u))$. Finally, according to Proposition 7.4.1, we can conclude that $Q^B_{P^*}$ is an SACBF.

We now prove the feasibility of (7.22). The feasibility of (7.22e) and (7.22f) follows from (7.20). Regarding the other constraints, since (i) the linearized system of (7.17) is controllable, (ii) $\lim_{x\to 0} \frac{||v(x)||_2}{||x||_2} = 0$ from Assumption 7.4.4, and (iii) the origin is within the interior of X, we can always replace P by $\alpha_P P$ with $\alpha_P \in (0,1)$ to make the conditions (7.23)-(7.25) satisfied. This proves the feasibility of (7.22b)-(7.22d). □

7.5. PERFORMANCE ANALYSIS AND GUARANTEE UNDER LEARNING ERRORS

In the previous section, we have introduced three different learning-based methods for computing an SACBF. However, learning errors are unavoidable and can arise from a combination of factors, including insufficient data, loss function mismatch, and suboptimal optimization. These errors have the potential to invalidate the learned SACBF. To address this problem, we propose a systematic analysis to evaluate the robust safety performance of (7.7) when it includes an inaccurate approximation Q^B_ω. This analysis further leads to a practical approach for handling learning errors through state constraint tightening followed by SACBF constraint relaxation.

Our analysis is inspired by the concept of input-to-state safety (ISSf) [120], which was originally developed for studying set invariance in the presence of disturbances. Before introducing the framework, we point out that the analysis of safety performance can be applied to the SL method in Section 7.4.1 and the expert-guided learning method in Section 7.4.2, while the RL method in Section 7.4.3 is excluded. We will explain the reason for this exclusion at the end of this section.

First, we relax the constraint in the original optimization-based controller (7.7), based on the intuition that it may not longer be possible to render (7.7) recursively feasible in

the presence of learning errors. Define a control policy with a relaxed SACBF constraint as

$$\pi^{\mathrm{r}}_{\theta,\omega}(x) := \arg\min_{u\in U} Q_\theta(x,u) \quad \text{s.t. } Q^B_\omega(x,u) \le \kappa(\varepsilon), \tag{7.26}$$

where $\kappa(\cdot)$ is a $\mathcal{K}_\infty$ function and $\varepsilon \ge 0$. The notation ε will be used to quantify the learning error, and κ will be elucidated later.

Then, similar to how CBFs are extended to ISSf CBFs [120], we introduce error-to-state safety SACBFs (ESSf SACBFs), capturing the set invariance when Q^B_ω is different from Q^B.

Definition 7.5.1 (ε-ESSf SACBF)**.** A function $\hat{Q}^B(\cdot,\cdot) : \mathcal{X}\times U \to \mathbb{R}$ is called an *ε-error-to-state safe SACBF* (ε-ESSf SACBF) with a corresponding safe set $\hat{\mathcal{S}}_Q$, if there exists a $\mathcal{K}_\infty$ function $\kappa(\cdot)$ such that the pair $(\hat{Q}^B, \hat{\mathcal{S}}_Q)$ satisfies the following conditions:

(i) $\hat{\mathcal{S}}_Q$ is non-empty.

(ii) $\min_{u\in U} \hat{Q}^B(x,u) \le \kappa(\varepsilon),\ \forall x \in \hat{\mathcal{S}}_Q$.

(iii) For any $x \in \hat{\mathcal{S}}_Q$, any $u \in U$ satisfying $\hat{Q}^B(x,u) \le \kappa(\varepsilon)$ ensures that $f(x,u) \in \hat{\mathcal{S}}_Q$.

Now, we are ready to give our main results on the ESSf property of Q^B_ω learned by the two methods presented in Sections 7.4.1 and 7.4.2.

Theorem 7.5.1 (ESSf for the SL method)**.** Under Assumption 7.4.1, consider the controlled system $x_{t+1} = f(x_t, \pi^{\mathrm{r}}_{\theta,\omega}(x_t))$, an SACBF Q^B, which is induced by a CBF B from (7.8) with $\beta_B < 1$, and an approximation Q^B_ω of Q^B. Suppose that

$$\left|Q^B(x,u) - Q^B_\omega(x,u)\right| \le \varepsilon,\ \forall (x,u) \in \mathcal{X}\times U \tag{7.27}$$

holds. Letting $\kappa(\varepsilon) = \frac{1+\beta_B}{1-\beta_B}\varepsilon$, we have:

(i) The approximation Q^B_ω is an ε-ESSf SACBF. The corresponding safe set is $\mathcal{S}_\omega = \{x \in \mathcal{X} | B(x) \le \frac{2\beta_B}{1-\beta_B}\varepsilon\}$.

(ii) The optimization-based controller (7.26) is recursively feasible with the initial condition $x \in \mathcal{S}_\omega$, i.e., $\mathcal{S}_\omega$ is forward invariant for the controlled system $x_{t+1} = f(x_t, \pi^{\mathrm{r}}_{\theta,\omega}(x_t))$.

(iii) Furthermore, if B is the CBF for (7.1) under the *tightened* state constraint $x \in X_\varepsilon := \{x \in \mathbb{R}^n | h(x) + \frac{2\beta_B}{1-\beta_B}\varepsilon \le 0\}$, the controlled system satisfies the original constraints $x_t \in X$ and $u_t \in U$, $\forall t \in \mathbb{N}$.

Proof. According to Lemma 7.3.1, $\mathcal{S}_Q = \mathcal{S}_B$. The non-emptiness of $\mathcal{S}_\omega$ follows directly from the non-emptiness of $\mathcal{S}_B$. Then, we will prove the feasibility of (7.26) for any $x \in \mathcal{S}_\omega$ (Condition (ii) of Definition 7.5.1). From (7.8) and using the properties of CBFs, we have $\min_{u\in U} Q(x,u) \le \max\{h(x), B(x)\} \le B(x),\ \forall x \in \mathcal{X}$. For all $x \in \mathcal{S}_\omega$, it follows from (7.27) that

$$\min_{u\in U} Q^B_\omega(x,u) \le \min_{u\in U} Q^B(x,u) + \varepsilon \le B(x) + \varepsilon \le \underbrace{\frac{1+\beta_B}{1-\beta_B}\varepsilon}_{\kappa(\varepsilon)}.$$

Next, let $x \in \mathcal{S}_\omega$ and u be any input satisfying $u \in U$ and $Q_\omega^B(x,u) \le \kappa(\varepsilon)$. We have

$$B(f(x,u)) \le \beta_B Q^B(x,u) \le \beta_B Q_\omega^B(x,u) + \beta_B \varepsilon \le \frac{2\beta_B}{1-\beta_B}\varepsilon.$$

The above inequalities prove the forward invariance of the set $\mathcal{S}_\omega$ and the satisfaction of Condition (iii) of Definition 7.5.1. The above arguments prove the statements (i) and (ii).

Furthermore, if B is a CBF under the *tightened* constraint $x \in X_\varepsilon$, we derive that $B(x) - \frac{2\beta_B}{1-\beta_B}\varepsilon \ge h(x)$, which further implies that the safe set $\mathcal{S}_\omega$ of the learned SACBF Q_ω^B is contained in the original state constraint set X. As $\mathcal{S}_\omega$ is forward invariant, the infinite-time safety of the controlled system follows. □

Theorem 7.5.2 (ESSf for the expert-guided learning method)**.** Under Assumption 7.4.2, consider the controlled system $x_{t+1} = f(x_t, \pi^{\mathrm{r}}_{\theta,\omega}(x_t))$ and approximations Q_ω^B and q_ω of Q^B and q in (7.9). Suppose that

$$Q_\omega^B(x,\pi_{\mathrm{s}}(x)) \le \beta q_\omega(x) + \varepsilon,\ \forall x \in \mathcal{S}_0 \tag{7.28a}$$

$$q_\omega(f(x,u)) \le Q_\omega^B(x,u) + \varepsilon,\ \forall (x,u) \in \mathcal{S}_0 \times U \tag{7.28b}$$

$$q_\omega(x) \ge \frac{2}{1-\beta}\varepsilon,\ \forall x \in \mathcal{X} \setminus \mathcal{S}_0 \tag{7.28c}$$

holds. Letting $\kappa(\varepsilon) = \frac{1+\beta}{1-\beta}\varepsilon$, we have:

(i) The approximation Q_ω^B is an ε-ESSf SACBF. The corresponding safe set is $\mathcal{S}_\omega = \{x \in \mathbb{R}^n | q_\omega(x) \le \frac{2}{1-\beta}\varepsilon\}$.

(ii) The optimization-based controller (7.26) is recursively feasible with the initial condition $x \in \mathcal{S}_\omega$, i.e., $\mathcal{S}_\omega$ is forward invariant for the controlled system $x_{t+1} = f(x_t, \pi^{\mathrm{r}}_{\theta,\omega}(x_t))$.

(iii) Furthermore, if $h(x) + \frac{2}{1-\beta}\varepsilon \le q_\omega(x),\ \forall x \in \mathcal{X}$, the controlled system satisfies the original constraints $x_t \in X$ and $u_t \in U,\ \forall t \in \mathbb{N}$.

Proof. (7.28c) leads to $\mathcal{S}_\omega \subseteq \mathcal{S}_0$. Consider any $x \in \mathcal{S}_\omega$, we have

$$\min_{u \in U} Q_\omega^B(x,u) \le Q_\omega^B(x,\pi_{\mathrm{s}}(x)) \le \beta q_\omega(x) + \varepsilon \le \underbrace{\frac{1+\beta}{1-\beta}\varepsilon}_{\kappa(\varepsilon)},$$

which proves the feasibility of (7.26).

Moreover, consider any $x \in \mathcal{S}_\omega$ and any $u \in U$ satisfying $Q_\omega^B(x,u) \le \kappa(\varepsilon)$. We have

$$q_\omega(f(x,u)) \le Q_\omega^B(x,u) + \varepsilon \le \frac{2}{1-\beta}\varepsilon.$$

The rest of the proof can be completed using the same reasoning as that of the proof of Theorem 7.5.1. □

The condition in (7.27) quantifies the local approximation quality of the regression model Q_ω^B, which is often assumed in SL literature [100], [106], [154]. The two conditions in (7.28a) and (7.28b) are motivated by (7.9c) and (7.9d), where we assume that the constraints in (7.9c) and (7.9d) are violated and the degree of violation is limited by ε. Besides, the condition (7.28c) is also justifiable, as it is introduced to ensure that the safe set of Q_ω^B is a subset of $\mathcal{S}_0$.

Remark 7.5.1. Different from existing work that primarily addresses the robustness and ISSf for learning-based control under external disturbances [77], input disturbances [106], or inaccurate observations [200], our findings offer a systematic way for designers to guarantee the safety of the relaxed safety-filtered policy $\pi^{\mathrm{r}}_{\theta,\omega}$ when there are perturbations on the safety constraints, by appropriately tightening the state constraint. More importantly, we explicitly quantify the degrees of tightening and relaxation, based on the parameters β and β_B of the given CBF and the safe policy π_{s} as well as the bound ε on the learning error. In particular, to ensure safety for the SL method, the state constraint must be tightened to $h(x)+\frac{2\beta_B}{1-\beta_B}\varepsilon\le 0$, while the safety constraint in the safety filter should be relaxed by adjusting $Q_\omega^B(x,u)\leftarrow Q_\omega^B(x,u)-\frac{1+\beta_B}{1-\beta_B}\varepsilon$. Similarly, in the expert-guided learning method, the state constraint should be tightened to $h(x)+\frac{2}{1-\beta}\varepsilon\le 0$, and (7.9b) should be adjusted to (7.28c), with the relaxed SACBF $Q_\omega^B(x,u)\leftarrow Q_\omega^B(x,u)-\frac{1+\beta}{1-\beta}\varepsilon$ in the safety filter.

Remark 7.5.2. The proposed constraint tightening approach is inspired by robust MPC [106], which ensures both recursive feasibility and safety. In contrast, our optimization-based control framework enforces safety through an SACBF constraint, rather than imposing state constraints over a finite prediction horizon. As a result, recursive feasibility alone does not guarantee safety, highlighting the need for constraint relaxation. Broadly speaking, state constraint tightening contributes to safety, whereas SACBF constraint relaxation addresses feasibility concerns.

At the end of this section, we explain why the RL method cannot be included in the proposed ESSf framework. To make Q^{B*} in (7.14) an SACBF, the optimal control problem (7.13) should be undiscounted, which leads to a non-contractive Bellman equation (7.15) [25]. As a consequence, the learning error of Q^{B*} could become unbounded. Besides, in Theorem 7.5.1 we require $\beta_B<1$ while in the reachability formulation (7.13), $\beta_B=1$. The above two factors make the ESSf analysis inapplicable to the RL method.

7.6. REFINING POLICIES WITH SACBF CONSTRAINTS

Given a valid SACBF obtained by the methods in the previous section, in this section, we update θ to refine the feasible policy $\pi_{\theta,\omega}$, bringing it closer to the optimal policy that solves (7.3).

A convenient approach for designing Q_θ, allowing θ to be updated using most existing unconstrained policy-based RL algorithms is to consider a Euclidean distance objective function $Q_\theta(x,u):=||u-\pi_\theta(x)||_2$, where $\pi_\theta:\mathcal{X}\to\mathcal{U}$ is an explicit controller (e.g., a neural network controller) with the parameter θ [102]. This approach, however, could significantly compromise the optimality of the projected policy $\pi_{\theta,\omega}$. A less conservative ap-

proach is to make Q_θ approximate the constrained optimal value function Q^* defined in (7.6), so that the policy derived from (7.7) more closely approximates the optimal constrained policy. To this end, we present a unified approach to transform unconstrained value-based RL algorithms into constrained ones, utilizing the obtained SACBF.

Normally, to obtain Q^*, as is shown in [35, Chapter 7.4.1], one typically needs to recursively compute the backward reachable sets X_k, $k = 0, 1, \ldots$ and enforce the constraint $f(x,u) \in X_k$ during the Bellman iteration. This procedure is, in general, intractable for nonlinear systems. Instead, we replace the state constraints $h(x_t) \le 0$, $\forall t \in \mathbb{N}$ in (7.3) with the obtained SACBF constraints $Q^B_\omega(x_t, \pi(x_t)) \le 0$, $\forall t \in \mathbb{N}$. Note that this will be an equivalent transformation if $Q^B_\omega = Q^{B*}$. The benefit of this transformation is that the constraints during the Bellman iteration become fixed to $Q^B_\omega(x_t, \pi(x_t)) \le 0$. This is a result of the inherent invariance of the safe set of Q^B_ω.

By combining the following expression of $\bar{J}^*$:

$$\bar{J}^*(x) := \inf_\pi J^\pi(x) \text{ s.t. } Q^B_\omega(x_t, \pi(x_t)) \le 0, \pi(x_t) \in U, t \in \mathbb{N}, \tag{7.29}$$

and the expression (7.7) of $\pi_{\theta,\omega}$, we know that $\pi_{\theta,\omega}$ is optimal for the (7.29) if Q_θ equals $\bar{Q}^*(x,u) := g(x,u) + \gamma \bar{J}^*(f(x,u))$.

The constrained optimal state-action value function $\bar{Q}^*$ satisfies the following constrained Bellman equation:

$$\begin{aligned} \bar{Q}^*(x,u) = \Gamma \bar{Q}^* := g(x,u) + \gamma \min_{u^+ \in U} \bar{Q}^*(f(x,u), u^+) \\ \text{s.t. } Q^B_\omega(f(x,u), u^+) \le 0, \end{aligned} \tag{7.30}$$

where Γ is the Bellman operator. It is easy to show that Γ is a monotonous contraction mapping. As a consequence, the uniqueness of the solution to (7.30) holds and the constrained Bellman iteration $\bar{Q}_{k+1} = \Gamma \bar{Q}_k$ converges to $\bar{Q}^*$ for any real-valued and bounded $\bar{Q}_0 : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ [25, Proposition 2.1.1].

With this property, we can update θ to minimize the average Bellman residual over $\mathcal{S}_\omega \times U$. Formally speaking, we find

$$\theta^* := \arg\min_\theta \int_{\mathcal{S}_\omega \times U} (q_\theta(x,u) - Q_\theta(x,u))^2 dx du, \tag{7.31a}$$

$$\begin{aligned} \text{where } q_\theta(x,u) = g(x,u) + \gamma \min_{u^+ \in U} Q_\theta\left(f(x,u), u^+\right) \\ \text{s.t. } Q^B_\omega\left(f(x,u), u^+\right) \le 0. \end{aligned} \tag{7.31b}$$

Problem (7.31a) is bi-level and intractable to solve exactly. In practice, following standard offline value iteration algorithms [40], we iteratively update θ. In each iteration, q_θ is treated as the target value, and θ is updated such that the difference between q_θ and Q_θ is minimized. This ultimately forms the proposed constrained fitted Q iteration (constrained FQI) algorithm (Algorithm 4). Furthermore, just as standard FQI can be adapted to constrained FQI, other online value-based RL algorithms, such as constrained deep Q-learning, constrained approximate SARSA, and Lagrangian RL, can likewise be designed accordingly.

Algorithm 4 Constrained fitted Q iteration

1: **Input:** The SACBF Q_ω^B, the state and input constraint sets X and U, $\mathcal{X}$, the sample set $\mathcal{D} \subseteq \mathcal{S}_\omega \times U \times \mathcal{S}_\omega$, the learning rate $\zeta > 0$, the error threshold $\varepsilon_{\text{QI}} > 0$, and the maximum number of updates $K \in \mathbb{N}^+$
2: **Initialize** θ_0 randomly in $[-1, 1]$, and $k \leftarrow 0$
3: **Repeat** at each iteration k
$q_s \leftarrow \Gamma Q_{\theta_k}(x_s, u_s)$ for each $(x_s, u_s, x_s^+) \in \mathcal{D}$
$\theta_{k+1} \leftarrow \arg\min_\theta \sum_{s=1}^{|\mathcal{D}|} \left(q_s - Q_{\theta_k}(x_s, u_s)\right)^2$
$k \leftarrow k+1$
4: **Until** $|Q_{\theta_{k+1}}(x,u) - Q_{\theta_k}(x,u)| \le \varepsilon_{\text{QI}}$, $\forall (x,u) \in \mathcal{X} \times U$, or $k > K$
5: **Output** Q_{θ_k}

7.7. COMPARISON WITH OTHER APPROACHES AND LIMITATIONS

Comparison with learning CBFs: Machine learning has recently been developed for synthesizing CBFs (safety filters) with or without model uncertainties. A common methodology to synthesizing permissive CBFs is to link CBFs with value functions of optimal control problems, such as the approaches in [47], [77], [81], [187] and the current chapter. The value function can then be approximated by RL or SL. Some authors assume the knowledge of a nominal model and a valid CBF for that model [188], [205]. Then, RL is used to learn the discrepancies between the CBF constraint associated with the nominal model and that related to the real model. However, all the above methods can only enable the design of model-based safety filters. This limitation is due to the inherent nature of CBFs, which need the model information to enforce the invariance condition. In contrast, in the current chapter, we fundamentally propose a new direct data-driven safe control framework, in which an SACBF is employed to evaluate the feasibility of input signals without using model information.

The paper [123] uses discriminating hyperplanes (a series of linear constraints) to represent safe constraints in safety filters. This approach eliminates the dependence on any specific safety certificate, and consequently, on the model. The hyperplanes can be updated by RL, in which the reward is designed such that unsafe policies are highly penalized. However, the RL method of [123] can still be understood as a reward shaping method for managing safety, which lacks formal guarantees regarding constraint satisfaction. Besides, approximating nonlinear constraints by linear constraints may lead to a rather conservative policy. In contrast, the proposed optimization-based control framework, which involves a nonlinear program, provides a formally sound and more general method to enhance safety and to eliminate dependence on an explicit model.

Comparison with integrating RL and MPC: The combination of MPC and RL can have various kinds of forms. [90] uses RL algorithms to update a parameterized nonlinear MPC scheme. It is shown in [90] that the parameterized MPC scheme can produce safe and stabilizing policies as long as the RL algorithm updates parameters in a personally-defined safe and stable set. Such a set, however, is usually problem-dependent. It is pos-

sible to use LMIs to determine such a set for linear systems. Different from [90], which parameterizes all terms of MPC, including the terminal cost, the prediction model, and the constraints, [154] only parameterizes the terminal cost and uses approximate value iteration to learn it offline. Similarly, [132] adopts approximate policy iteration to learn the terminal cost offline. It is proven in [132], [154] that the resulting MPC controller makes the system safe and asymptotically stable if the approximation error is bounded and the MPC horizon is large enough. The above combinations, however, fail to solve the online computational problem faced by MPC, since their policies are still determined by online optimization over a long prediction horizon. In comparison, the proposed optimization-based control approach yields significant online computational benefits by using two state-action value functions Q_θ and Q_ω^B to approximate the value function and safety constraints implicitly defined by the parameterized MPC scheme.

Moreover, we acknowledge that, unlike parametrized MPC, which employs an explicit prediction model to derive optimal and safe policies for long-term goals, our optimization-based control framework lacks explainability in the resulting policy. However, our method offers greater flexibility, as it allows the integration of any RL algorithm into the control synthesis.

Comparison with safe RL: Existing methods for safe RL include using reward shaping [102], [143], Lagrangian methods combined with policy gradient or actor-critic algorithms [211], interior-point optimization [135], and safety filtering [53], [68]. Additionally, a small body of work explores direct stochastic optimization of neural network controllers over finite horizons, improving safety at sampled states but demanding significant computational resources [129]. However, as discussed in the introduction, because learning algorithms operate stochastically, learning-based control—particularly when an explicit model is unavailable—requires safety filters to regulate policy execution and to ensure reliable safety. The proposed optimization-based control framework is compatible with all the safe RL methods mentioned above and provides formal safety assurance under learning errors.

Comparison with the model-based method in Chapter 6: In Chapter 6, the definition of SACBFs is proposed, which contributes to a safety enhancement framework analogous to (7.7). The current chapter expands Chapter 6 in the following two aspects. First, all the learning-based methods for SACBFs proposed in Section 7.4 totally remove the assumption on the availability of a nominal model, which is required in Chapter 6. Second, in Chapter 6, the robustness to learning errors is addressed by a new CBF with a contractive safe set, which needs a stronger invariance condition than invariant sets. In contrast, the current chapter establishes an ESSf analysis framework using the fundamental robustness property of the SACBF, resulting in significantly less conservative constraint tightening than that in Chapter 6.

Limitations: One downside is that safety constraints will inevitably be violated when collecting samples of state transitions to train the controller. This may be inappropriate for real-world applications where maintaining safety is essential throughout the learning process. Using simulators to generate sample data during learning can solve this issue, although the gap from sim to real needs further robustness analysis.

We extend the barrier certificate, originally used to characterize state safety, to the SACBF, which captures both state-input safety. However, this extension introduces a drawback: it necessitates sampling and learning in the state-input space rather than just the state space, thereby increasing computational and sample complexity.

7.8. CASE STUDY

In this section, we verify the proposed data-driven approaches for synthesizing safety filters and their application to safe learning-based control for an autonomous vehicle moving in a 2-D space containing obstacles.

7.8.1. MODEL

We consider the kinematic vehicle model [77]:

$$\begin{aligned} \dot{p}_x &= v\cos(\Psi) \\ \dot{p}_y &= v\sin(\Psi) \\ \dot{v} &= a \\ \dot{\Psi} &= v\tan(\delta)/L, \end{aligned} \tag{7.32}$$

where $L = 0.1$. The state vector x includes the position p_x, p_y, the speed v, and the yaw angle Ψ. The acceleration a and the steering angle δ are the inputs. The input constraints are given by $-5 \le a \le 2$ and $|\delta| \le \pi/4$. The state constraints are specified by $|p_x| \le 2.6$, $|p_y| \le 2.6$, $0 \le v \le 1$, and $|\Psi| \le \pi$, as well as the requirement of avoiding some obstacles shown in Fig. 7.2. According to the considered state constraints, the function h is as follows:

$$h(x) = \max\left\{|p_x| - 2.6, |p_y| - 2.6, -v, v-1, |\Psi| - \pi, \max_{i=1,2,3,4}\{r_i^2 - (p_x - c_{x,i})^2 - (p_y - c_{y,i})^2\}\right\}.$$

Here, r_i denotes the radius of each obstacle, while $c_{x,i}$ and $c_{y,i}$ represent the x- and y-coordinates of the center of each obstacle. The state constraint is tightened to $\{x|h(x) + 0.2 \le 0\}$ when learning SACBFs. The target set is $X_{\text{tar}} = \{x \in \mathbb{R}^4 | p_x^2 + p_y^2 \le 0.1^2\}$. The system is discretized using the Euler method with a sampling time of 0.05 s.

7.8.2. SYNTHESIZING SACBFS

The synthesis of the SACBF is performed using three approaches, including the expert-guided learning approach presented in Section 7.4.2, the RL approach described in Section 7.4.3, and the LMI approach of Section 7.4.4. The SL approach in Section 7.4.1 is not considered because it is unrealistic to assume knowing a CBF without knowing the model in this example. For the expert-guided approach, we adopt artificial potential fields (APF) [202] to design the expert controller. The APF-based controller has a certain capability for obstacle avoidance but is prone to falling into local optima, which can either reside within obstacle regions or be located far from the target.

To get the training data sets $\mathcal{S}_s$, $\mathcal{X}_s$ for the expert-guided approach, and U_s, we start the system from 10000 randomly generated initial conditions inside $\mathcal{X} = \{x \in \mathbb{R}^4 \mid |p_x| \le 3,\ |p_y| \le 3,\ -0.2 \le v \le 1.2,\ |\Psi| \le \pi\}$ and get the trajectories over 200 time steps. For

the remaining learning models, including using RL to learn the SACBF Q^B, learning the optimal value function Q, learning the reference control policies, and identifying the model in the subsequent statistical tests, we use uniformly random sampling to get 10^6 state-input samples from $\mathcal{X}$.

We use neural networks with [128 128 32] "tansig" layers to represent all the neural SACBF[3]. The training algorithm is stochastic gradient descent with momentum [165], with a learning rate of 0.001. For the expert-guided learning approach, q_ω is represented by a neural network with [64 64 32] "tansig" layers, and the constraints in (7.9b)-(7.9d) are penalized in the loss with $\beta = 0.1$ and the penalty weights $\lambda_{(7.9b)} = 1$ and $\lambda_{(7.9c)} = \lambda_{(7.9d)} = 10$. After 10 epochs of training, it is verified that the three inequalities in (7.28a)-(7.28c) hold for $\varepsilon = 0.083$ at all the training samples. Therefore, we know that these three inequalities hold over the continuous sets $\mathcal{S}_0$, $\mathcal{S}_0 \times U$, and $\mathcal{X} \setminus \mathcal{S}_0$ with high probability by employing probabilistic verification methods [108]. According to Theorem 7.5.2, under the tightened constraint $h(x)+0.2 \le 0$, the safe set $\{x | q_\omega(x) \le 2\varepsilon/(1-\beta)\}$ is contained in the original state constraint with high probability.

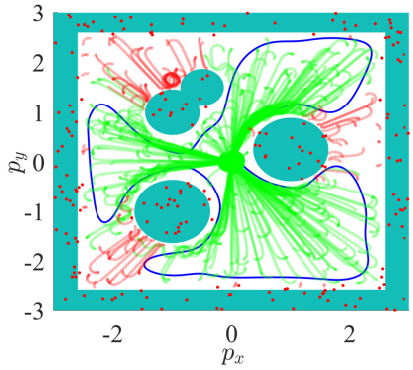
In Figs. 7.2(a)-(d), we illustrate the neural SACBFs obtained by the expert-guided learning approach and the RL approach, and their corresponding safe sets. To visualize the 4D sets, we display their shapes on the p_x-p_y plane with fixed $v = 0.5$ and $\Phi = 0$. Both safe sets avoid intersections with obstacles and wall areas. The safe set derived from the expert controller is smaller than that learned through the RL approach. This difference arises because the RL method theoretically approximates the maximal safe set, whereas the expert controller (APF) is not always feasible within the maximal safe set, which is illustrated by the red curves in Fig. 7.2(a).

For the LMI approach, since Assumption 7.4.3 cannot be satisfied for the original vehicle model in (7.32), we make some simplifications. In particular, (i) only the subsystems of p_y and Ψ are considered, (ii) the speed is fixed to a constant 0.5, (iii) the obstacles are removed, and (iv) the approximation $\tan\delta \approx \delta$ is used. Then, the resulting simplified system is of the form (7.17), with the nonlinear term $\sin\Psi$. After using the input signal $\delta(t) = 0.1\sin t$ to excite the system, we get the state-input data $\mathbf{D}$ (defined in Lemma 7.4.1) with the time horizon 20. The persistency of excitation condition is satisfied. By solving the LMI problem (7.22), we obtain Q_P^B with $P = \begin{bmatrix} 6.25 & -0.2996 \\ -0.2996 & 0.1176 \end{bmatrix}$. The corresponding safe set is visualized in Fig. 7.3. As observed, the LMI approach is conservative, as it restricts the SACBF candidate to quadratic forms.

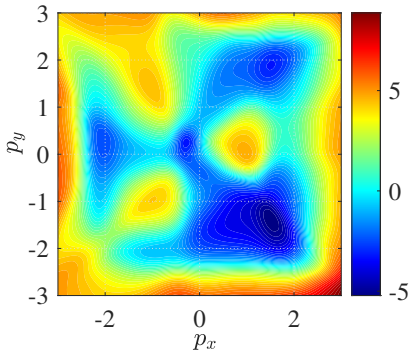
7.8.3. CLOSED-LOOP SIMULATION

Hereafter, we evaluate the performance of the proposed optimization-based controller (7.7). We construct two reference control policies obtained using: (i) APF and (ii) deep deterministic policy gradient (DDPG) [131]. The APF contains two elements: attractive potential fields, which guide the vehicle toward the target set, and repulsive potential fields, which prevent the vehicle from hitting the obstacles. To highlight the performance of the safety filter, we only apply the attractive potential field to get the reference

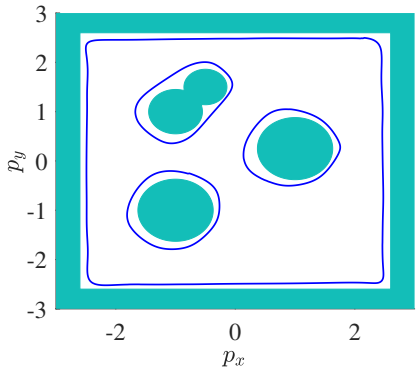
[3]The vector $[a_1\ a_2\ \ldots\ a_s]$ means the neural network contains s hidden layers with a_i units in each layer.



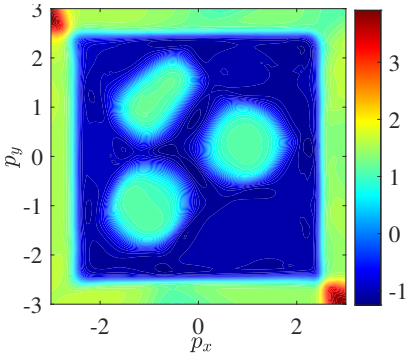
(a) The boundaries (blue curves) of the safe set $\{x|q_\omega(x) \leq 2\varepsilon/(1-\beta)\}$ learned from the expert controller. The green curves represent the safe trajectories of the APF controller, while the red curves and points represent the unsafe trajectories and initial states.



(b) The value of q_ω learned from the expert controller.



(c) The boundaries (blue curves) of the safe set $\{x|\min_{u\in U} Q^B_\omega(x,u) \leq 0\}$ learned by RL



(d) The value of $\min_{u\in U} Q^B_\omega(\cdot,u)$ learned by RL.

Figure 7.2: The SACBFs and their corresponding safe sets in the p_x-p_y plane with $v = 0.5$ and $\Psi = 0$. The green area represents the obstacles and walls. In the contour figures, values below zero mean that the positions are feasible.

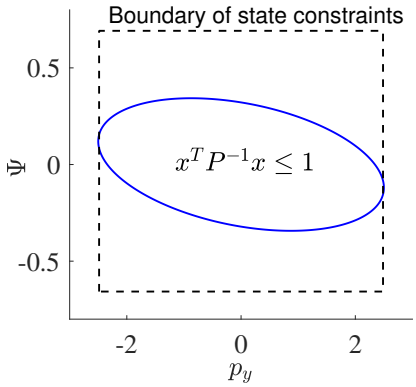


Figure 7.3: The boundaries (blue curves) of the safe set $\{x|x^T P^{-1} x \leq 1\}$ of the quadratic SACBF (7.21) obtained by solving LMIs

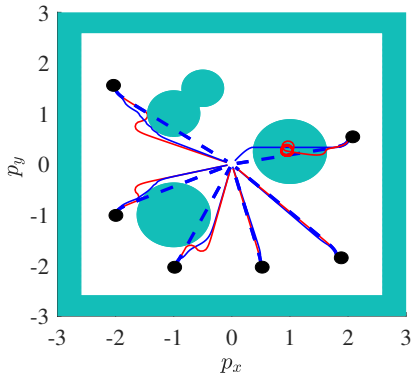
controller (referred to as unsafe APF). Similarly, in the case of DDPG, we do not incorporate safety constraints during the training process (referred to as unsafe DDPG). For DDPG, the stage cost is designed as $g(x,u) = \max\{0, p_x^2 + p_y^2 - 0.1^2\}$, with a discount factor $\gamma = 0.99$. For comparison, we evaluate the filtered policies against: (i) the APF method incorporating both attractive and repulsive potential fields (referred to as the safe APF); (ii) the DDPG policy trained with a penalty on constraint violations in the cost function (referred to as safe DDPG) [91].

Fig. 7.4(a) demonstrates the performance of the 3DSF learned using the expert-guided learning approach. When applying the unsafe policy, the trajectories exhibit some constraint violations. In contrast, the learned SACBF enables the vehicle to smoothly avoid obstacles, except for the initial condition $p_x = 2,\ p_y = 0.5$ (the rightmost black dot). To explain this constraint violation, we note that the expert controller (safe APF) also exhibits constraint violations when starting from this initial state. Consequently, the 3DSF, which is learned based on this expert controller, recognizes this initial state as unsafe and therefore fails to refine the reference policy.

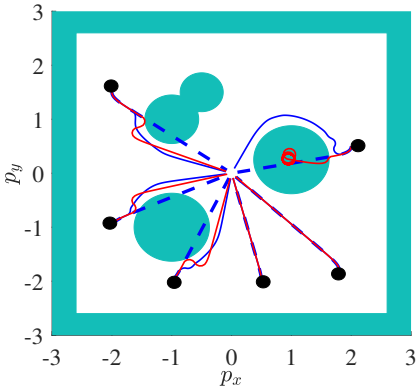
In comparison, the 3DSF learned by RL refines the reference policy for all the initial states. The trajectory starting from the rightmost initial state in Fig. 7.4(b) demonstrates the superiority of the proposed 3DSF learned by RL. In particular, the safe APF policy makes the trajectory fail into a local optimum inside the right obstacle, while the unsafe APF policy with the 3DSF circumvents the local optimum and steers the system to the target.

In Figs. 7.4(c-d), we show the trajectories of the vehicle controlled by the DDPG policies. It is found that although for some initial states safe DDPG successfully plans a safe trajectory, some trajectories fail to converge to the target set. This illustrates that it is hard to balance the task performance and constraint satisfaction by naively adding penalties to the stage cost during training. In contrast, the safety filter significantly enhances the safety performance of the unsafe DDPG policy while ensuring that the policy reaches the

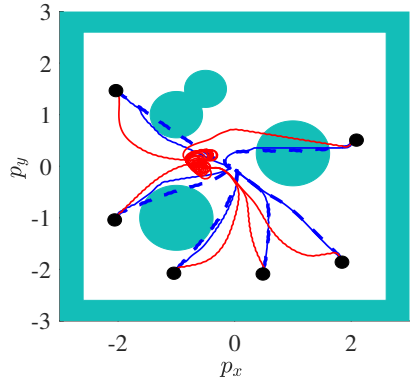
target.



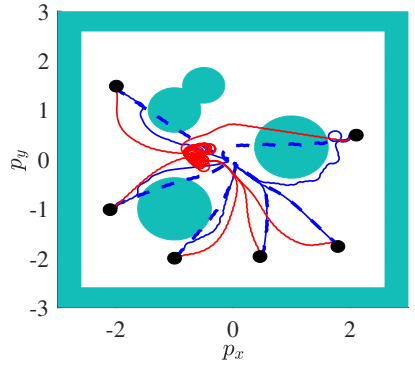
(a) Closed-loop trajectories. Dashed blue curve: unsafe APF; Solid blue curve: unsafe APF with 3DSF learned from the expert controller; Red curve: safe APF; Black dots: initial states.



(b) Closed-loop trajectories. Dashed blue curve: unsafe APF; Solid blue curve: unsafe APF with 3DSF learned by RL; Red curve: safe APF; Black dots: initial states.



(c) Closed-loop trajectories. Dashed blue curve: unsafe DDPG; Solid blue curve: unsafe DDPG with 3DSF learned from the expert controller; Red curve: safe DDPG; Black dots: initial states.

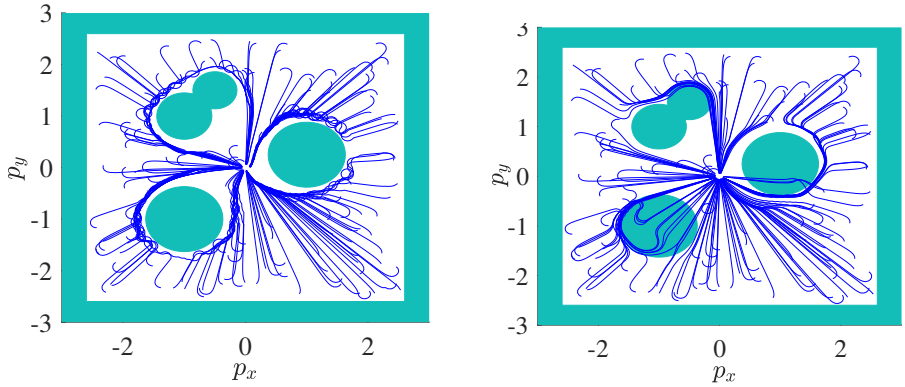


(d) Closed-loop trajectories. Dashed blue curve: unsafe DDPG; Solid blue curve: unsafe DDPG with 3DSF learned by RL; Red curve: safe DDPG; Black dots: initial states.

Figure 7.4: Comparison of closed-loop vehicle trajectories under different controllers and different safety filters.

7.8.4. STATISTICAL EVALUATION AND COMPARISON WITH INDIRECT DATA-DRIVEN SAFETY FILTERS

Finally, we statistically compare the proposed 3DSF against an indirect data-driven safety filter. In this subsection, the SACBF is learned by the RL approach because it has been illustrated in the previous subsection that the SACBF learned by the RL approach results in a larger safe set. To design the indirect data-driven safety filter, we intend to



(a) Trajectories of the vehicle controlled by the unsafe APF policy filtered by the 3DSF.

(b) Trajectories of the vehicle controlled by the unsafe APF policy filtered by the indirect safety filter.

Figure 7.5: Comparison of closed-loop vehicle trajectories under the 3DSF and the indirect safety filter.

learn a standard CBF based on an approximate model. We first use a neural network with [128 128 128] "elu" hidden layers to identify the kinematic model. After 30 epochs of training, we get an approximate model with a sufficiently small ($2.38 \cdot 10^{-4}$) mean square error. Then, following the approaches of [47], [81], [100], the approximate model is used as a prediction model in the reachability problem[4] (7.13). We solve (7.13) with the initial state x_0 being each state sample obtained in Section 7.8.2. Then, we get the target value for the neural CBF, which is parameterized by a neural network with [128 128 32] "tansig" hidden layers.

We randomly sample 141 initial states that lie in the intersection of the safe set of the SACBF and the safe set of the CBF. Fig. 7.5 shows the trajectories controlled by the unsafe APF controller with the 3DSF and the indirect counterpart, respectively. Notably, the results demonstrate the absence of undesired equilibria, limit cycles, or unbounded trajectories.

As has been mentioned in Section 7.2, (7.7) can be utilized either as a safety filter (3DSF) for a pre-obtained reference control policy or as a policy generator to determine suboptimal control inputs greedy to Q_θ. Therefore, we apply the constrained FQI (Algorithm 4) to get the approximation Q_θ of the constrained optimal value function $\bar{Q}^*$ defined in (7.30). This makes (7.5) a greedy policy optimization problem.

The results in Table 7.1 provide a detailed comparison of the safety rate, total cost, and average CPU time for those trajectories starting from the sampled initial states. The total cost reflects the ability to govern the vehicle to the target set. The key findings are summarized below:

[4] The reachability problem (7.13) is defined over an infinite horizon. To make it solvable for each sampled state, we truncate the horizon at 20. While a longer horizon provides a better approximation of the maximal safe set, a horizon of 20 balances between sample complexity and safety performance.

Table 7.1: Comparison of safety rate, total cost, and average online CPU time for different control policies and safety filters. The total cost is defined as the sum of $p_x^2 + p_y^2$ over 1000 time steps. The safety rate is defined as the ratio between the number of successful trajectory plans (without constraint violations) and the number of all trajectory plans. "SF" means "safety filter".

Controller	Performance	No SF	Indirect SF	3DSF
Unsafe APF	Safety Rate	58.16%	79.43%	100%
	Total Cost	114.30	136.59	154.25
	CPU Time	0.0043 ms	1.60 ms	2.19 ms
Safe APF	Safety Rate	81.56%	80.85%	100%
	Total Cost	373.39	521.46	155.87
	CPU Time	0.0060 ms	1.51 ms	2.21 ms
Unsafe DDPG	Safety Rate	70.92%	79.43 %	96.45%
	Total Cost	120.33	141.22	187.54
	CPU Time	0.39 ms	2.21 ms	2.49 ms
Safe DDPG	Safety Rate	82.27%	89.36%	100%
	Total Cost	560.60	571.45	611.90
	CPU Time	0.36 ms	2.49 ms	2.61 ms
Constrained FQI	Safety Rate			100%
	Total Cost	–	–	173.85
	CPU Time			4.95 ms

- Both the 3DSF and the indirect data-driven safety filter can in general reduce the rate of constraint violation while not significantly degrading the performance regarding the total cost. When combined with safety filters, even unsafe policies (e.g., Unsafe APF and Unsafe DDPG) achieve safety rates comparable to their safe counterparts.
- Most importantly, the 3DSF significantly improves the safety rate compared to the indirect counterpart. The worse performance of the indirect counterpart is likely due to the superposition effect of the model error and the CBF learning error.
- Constrained FQI achieves high safety rates (100%) and a lower total cost (173.85) compared to safe and unsafe DDPG with the 3DSF. This validates that including the SACBF constraint in the training process of RL could improve the task performance.

7.9. CONCLUSIONS AND FUTURE WORK

We have proposed an optimization-based control framework that contains an SACBF constraint to ensure safety for learning-based control methods. The main advantages lie in its universal applicability to RL and SL methods for designing controllers, as well as its online computational efficiency. Three learning algorithms have been developed within the optimization-based control framework, making the controller strive for opti-

mal performance while ensuring safety to the greatest extent possible. We have analyzed the theoretical properties regarding the robustness of SACBF and translated the results to error-to-state safety (ESSf) of the proposed control framework w.r.t. learning errors. Simulations conducted on an obstacle avoidance problem demonstrate the aforementioned advantages.

Future work will focus on (i) increasing the scalability to higher-dimensional systems, (ii) examining the effect of external uncertainties, and (iii) designing distributed optimization-based control for multi-agent systems subject to unknown dynamics and nonlinear constraints.

8

CONCLUSIONS, IMPACTS, AND FUTURE RESEARCH

In this final chapter, the main conclusions, the social and scientific impacts, and recommendations for future research are provided.

8.1. CONCLUSIONS

This PhD thesis answers the central research question (How to effectively integrate optimization and learning-based control, in such a way that performance optimality, safety, and computational tractability are preserved?) by developing a suite of control strategies. These include combining ADP and MPC, constrained ADP with PWA penalties, optimization-free safety filters, predictive safety mechanisms, and ultimately, a unified framework that integrates learning and optimization in control design. The key conclusions and findings can be summarized as follows:

Learning-based control for linear and PWA systems under constraints.

- (Chapters 2-3) For linear systems and PWA systems, explicit MPC based on multi-parametric optimization yields theoretical insights into the structure of the value function and the optimal policy. These structural properties can be exploited to enhance learning efficiency and performance during value function and policy approximation.
- (Chapter 3) For systems governed by PWA dynamics and subject to PWA constraints, we demonstrate that incorporating PWA penalty terms into the objective function preserves the PWA nature of both the value function and the resulting policy. This observation motivates the use of PWA neural networks within the proposed ADP scheme and supports the application of mixed-integer linear programming for performance validation.

- (Chapters 4-5) There exists a trade-off between safety guarantees and online computational efficiency. Using PWA approximation for value functions, constraints, and dynamics can speed up solving the online safety-constrained control problem. Explicit, though possibly sub-optimal solutions can be derived, further reducing computational overhead.
- (Chapter 5) Despite the computational benefits, PWA approximations introduce non-smoothness into the policy optimization problem, potentially compromising safety guarantees. This challenge can be addressed by leveraging generalized Clarke derivatives of PWA functions to rigorously enforce safety constraints.

Integrated offline learning and online optimization for control under constraints.

- (Chapters 4-7) Online safety-constrained optimization that explicitly incorporates hard long-term safety constraints defined by a safety certificate can provide long-horizon safety guarantees for learning-based controllers, assuming the certificate is valid. Although exact long-term safety constraints (i.e., safety certificates) are often analytically unavailable, they can be approximated by learning value functions of reachability problems.
- (Chapters 6-7) The proposed integrated optimization and learning framework decomposes the control synthesis task into two parts: learning the objective function to maximize performance, and learning constraint representations to ensure safety, thereby enabling modular and interpretable design. We advocate for first learning the safety constraints and subsequently integrating them into the learning of the objective function. This is because safety is inherently independent of other performance metrics, whereas achieving optimality regarding task performance should be addressed under the premise of ensuring safety.
- (Chapters 6-7) Constraint-tightening is not sufficient to guarantee safety when using learning-based methods to synthesize safety certificates or invariant sets. To have this guarantee, drawing inspiration from input-to-state safety analysis, we find that the safety certificate should be robust to approximation errors, or the invariant set should admit a contractive property.
- (Chapters 6-7) Compared to model-based approaches, model-free integration of learning and optimization is considerably more challenging due to increased sample complexity and the lack of an explicit system model for evaluating critical components such as safety constraints and value functions. While RL can alleviate the model dependence, it remains hindered by substantial data demands, sample inefficiency, and possibly unstable learning.

8.2. IMPACTS OF THE PHD THESIS

8.2.1. SCIENTIFIC IMPACTS

This PhD thesis makes contributions to advancing the state-of-the-art learning-based control with safety considerations.

While the integration of machine learning and MPC has been extensively studied (e.g.,

[147], [168]), this PhD thesis has proposed a fundamental and more general framework that integrates machine learning and general nonlinear optimization, offering a principled framework to design learning-based controllers with formal safety and stability guarantees.

Although online safety-constrained optimization can provide formal performance guarantees for learning-based controllers, it is usually computationally prohibitive for nonlinear or hybrid systems. By proposing optimization-efficient algorithms and tools (e.g., piecewise quadratic neural networks for value function approximation, optimization-free safety filters, SACBF-based safety filters), the thesis significantly reduces the computational requirements of solving the online safety-constrained optimization problem, and consequently, addresses the trade-off between online computational efficiency and performance guarantees.

A central question in integrated learning and optimization for control is how to avoid the degradation of task performance when using online optimization to enforce safety, e.g., using safety filters or MPC. One promising solution is to include the optimization layer in the learning process, either by differentiating through the optimization problem (which can be computationally expensive) [212] or by treating the optimization as part of the environment in RL (which may vastly increase the action space) [167]. The optimization-free safety filters presented in Chapters 4 and 5 find explicit approximations of safe controllers, enabling more efficient gradient computation and policy learning compared to controllers based on implicit optimization solutions. This advancement contributes to a significant reduction in offline computational complexity during the policy training process.

As has been mentioned in the introduction, CBFs and Hamilton-Jacobi (HJ) reachability are two prominent choices for safe controller synthesis and verification. A unified view of them is presented in [55], and their connections with learning-based control and predictive safety filters are pointed out in the recent tutorial paper [196]. A major challenge in this area is that RL methods often fail to approximate solutions to infinite-horizon HJ reachability problems due to the absence of contraction properties, which can lead to divergence of learning algorithms. Conversely, incorporating a discount factor gives contraction properties but makes the gradient of the resulting CBF vanish, and thus unsuitable for controller synthesis. The learning-based approaches for synthesizing CBFs presented in Chapters 4 and 6 overcome this issue by considering an alternative finite-horizon reachability problem with a conservative CBF as the terminal function. Such a design guarantees the convergence of RL algorithms and the validity of the resulting learned CBF. Overall, the developed learning-based approach for synthesizing CBFs expands the direction of combining learning and HJ reachability for safety control synthesis, where a trade-off between the convergence of learning and the validity of safety certificates naturally exists.

Finally, this PhD thesis also provides a comprehensive view on how to mitigate the effects of unavoidable learning errors on safety performance. Note that previous work mostly addresses the effects of system mismatches [164], measurement noise [65], and input disturbances [120]. We extend these robustness results to make the proposed con-

trollers robust to learning errors. This contributes to the foundation for a unified robust control framework that seamlessly incorporates various kinds of uncertainties. This impact is both theoretical and practical, guiding practitioners in deploying learning-based controllers with predictable behavior.

8.2.2. SOCIAL IMPACTS

Enabling safer autonomous systems. Safety remains a fundamental barrier to the widespread adoption of autonomous technologies in real-world applications such as self-driving vehicles, robotics, healthcare devices, and industrial automation. In these applications, unsafe behaviors are likely to cause catastrophic consequences. By developing optimization-free and predictive safety filters, as well as novel safety certificates like state-action control barrier functions, this PhD thesis introduces computationally efficient methods to ensure real-time safety, even under model uncertainties and learning errors. These techniques provide opportunities for the deployment of autonomous systems in unstructured or partially known environments where guaranteeing safety is paramount.

Reducing the resource barriers to intelligent control. The proposed control frameworks avoid the need for computationally intensive online optimization. Instead, they leverage offline learning, PWA approximations, and explicit controller design to reduce the burden on real-time computation. This improves the accessibility of advanced control techniques for resource-constrained settings, such as embedded systems, low-power medical devices, or affordable robotics, facilitating broader societal adoption of automation and intelligent decision-making.

Supporting the responsible use of AI in safety-critical domains. This work contributes to the foundation of responsible AI by proposing control-theoretic guarantees for learning-based control, which are traditionally viewed as black-box or unpredictable. By integrating formal methods like reachability analysis, Lyapunov functions, control barrier functions, data-driven control, and constraint tightening, the proposed approaches provide traceable and verifiable safety assurances, critical for applications in healthcare, transportation, and human-robot interaction, where AI has shown powerful capabilities but failures may have serious consequences.

Contributing to modular and adaptive autonomy. Finally, the proposed control frameworks provide modular strategies for integrating learning and optimization for constrained control, i.e., it can work in conjunction with a wide range of learning-based methods and optimization techniques. These capabilities are essential for the development of modular and adaptive autonomous infrastructure, whether in smart cities, renewable energy systems, or precision agriculture, where each component should function out of the box and adapt dynamically to changes in other components.

In summary, this PhD thesis not only advances the theoretical foundations of safe learning-based control but also paves the way for its responsible and impactful application across a wide range of societal domains.

8.3. RECOMMENDATIONS FOR FUTURE RESEARCH

In this section, we outline some potential future research directions based on the conclusions and findings of this PhD thesis.

A unified framework for robust control under general uncertainties. In most parts of this PhD thesis, we have assumed full knowledge of system dynamics. However, uncertainties arise naturally from various aspects, including external disturbances, inaccurate state measurements, and unmodeled dynamics. Most existing literature has considered a particular type of uncertainty [65], [120], [164], [205]. A comprehensive robustness framework capable of systematically accounting for multiple sources of uncertainty remains underdeveloped. Future research could focus on constructing such a unified framework, with particular attention to evaluating and mitigating its potential conservatism, both theoretically and in practical implementations. To achieve this, a possible solution is to extend the existing input-to-stability/safety analysis to incorporate more kinds of uncertainties.

Infinite-horizon chance-constrained control. One of the main insights of this PhD thesis is the inherent trade-off between strong safety guarantees and online computational efficiency. This observation suggests that pursuing probabilistic safety, instead of absolute guarantees, may result in more online computationally efficient controllers. In some cases, strict guarantees of safety may not be necessary. In other words, sometimes safety constraints can be satisfied with at least some probability. This can be formulated as a chance-constrained control problem. Current research has predominantly focused on finite-horizon settings, with approaches such as chance-constrained MPC and chance-constrained ADP [176]. The extension to the infinite-horizon case is a challenging and interesting future research topic. The difficulty lies in the fact that the optimal policy in this setting may be ill-defined, non-Markovian, or non-stationary, thereby hindering the applicability of traditional dynamic programming techniques. Possible avenues for future work include leveraging state augmentation to restore Markov properties, or adopting conservative approximations via expected value constraints or Conditional Value-at-Risk (CVaR) formulations [183].

Distributionally robust safe learning-based control. In some situations, uncertainties are not random but instead stem from the strategic or adversarial behavior of other agents. Examples include the behavior of pedestrians and other drivers in transportation systems. Addressing such cases necessitates a distributionally robust control approach, where controllers are designed to maintain performance under the worst-case distribution within a prescribed ambiguity set. This paradigm has recently gained growing attention in the MPC community, where distributionally robust MPC has been proposed [127]. However, solely relying on solving online MPC problems could be computationally intractable in the distributionally robust setting or could result in an overly conservative, linearly parameterized control policy. A promising research direction lies in exploring how learning techniques can enhance the performance and tractability of distributionally robust controllers. One specific idea is to use robust or adversarial RL to synthesize safety certificates that are distributionally robust, and then utilize these safety certificates in the optimization-based controller design.

Distributed learning-based control with safety considerations. While the whole PhD thesis has focused on learning-based control of single-agent systems, in many real-world scenarios, multiple agents operate in shared environments, constituting a large-scale system. Examples include power networks and intelligent transportation systems with multiple autonomous vehicles. A natural extension is to develop distributed and safe learning-based control for such large-scale systems. Key challenges include handling inter-agent communication, interaction constraints, as well as the issue of a nonstationary environment in multi-agent learning, while ensuring safety and achieving a cooperative objective. Investigating distributed architectures, scalable policy learning algorithms, and cooperative control strategies under safety-critical constraints will be critical for enabling robust and intelligent behavior in complex, interconnected systems. To overcome these challenges, a naive approach is to treat other agents as external uncertainties and design a robustly safe learning-based controller for each individual agent. However, this strategy often leads to overly conservative behaviors. A more promising direction is to develop stochastic models for the surrounding agents using, e.g., polynomial chaos expansions, and incorporate them into the design of distributionally robust controllers, thereby achieving a better balance between safety and performance.

BIBLIOGRAPHY

[1] A. Abate, A. Edwards, and M. Giacobbe, "Neural abstractions", *Advances in Neural Information Processing Systems*, vol. 35, pp. 26 432–26 447, 2022.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization", in *International Conference on Machine Learning*, PMLR, 2017, pp. 22–31.

[3] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation.", in *Robotics: Science and Systems*, Cambridge, MA, USA, vol. 13, 2017, pp. 1–10.

[4] A. K. Akametalu, S. Ghosh, J. F. Fisac, V. Rubies-Royo, and C. J. Tomlin, "A minimum discounted reward Hamilton–Jacobi formulation for computing reachable sets", *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 1097–1103, 2023.

[5] A. Alessio and A. Bemporad, "A survey on explicit model predictive control", *Nonlinear Model Predictive Control: Towards New Challenging Applications*, pp. 345–369, 2009.

[6] A. Alessio, M. Lazar, A. Bemporad, and M. Heemels, "Squaring the circle: An algorithm for generating polyhedral invariant sets from ellipsoidal ones", *Automatica*, vol. 43, no. 12, pp. 2096–2103, 2007.

[7] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications", in *2019 European Control Conference (ECC)*, 2019, pp. 3420–3431.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems", *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[9] A. A. Amin and K. M. Hasan, "A review of fault tolerant control systems: Advancements and applications", *Measurement*, vol. 143, pp. 58–68, 2019.

[10] A. Anand, K. Seel, V. Gjærum, A. Håkansson, H. Robinson, and A. Saad, "Safe learning for control using control Lyapunov functions and control barrier functions: A review", *Procedia Computer Science*, vol. 192, pp. 3987–3997, 2021.

[11] R. J. Aumann, "Integrals of set-valued functions", *Journal of Mathematical Analysis and Applications*, vol. 12, no. 1, pp. 1–12, 1965.

[12] E. M. Aylward, P. A. Parrilo, and J.-J. E. Slotine, "Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming", *Automatica*, vol. 44, no. 8, pp. 2163–2170, 2008.

[13] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer, *Non-linear Parametric Optimization*. Springer, 1983.

[14] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances", in *the 56th IEEE Conference on Decision and Control (CDC)*, 2017, pp. 2242–2253.

[15] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1817–1824.

[16] M. Baoti, F. J. Christophersen, and M. Morari, "Constrained optimal control of hybrid systems with a linear performance index", *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1903–1919, 2006.

[17] M. Baotić, F. Borrelli, A. Bemporad, and M. Morari, "Efficient on-line computation of constrained optimal control", *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2470–2489, 2008.

[18] A. R. Barron, "Approximation and estimation bounds for artificial neural networks", *Machine Learning*, vol. 14, pp. 115–133, 1994.

[19] L. Beckenbach, P. Osinenko, T. Göhrt, and S. Streif, "Constrained and stabilizing stacked adaptive dynamic programming and a comparison with model predictive control", in *2018 European Control Conference (ECC)*, 2018, pp. 1349–1354.

[20] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems", in *2000 American Control Conference (ACC)*, vol. 2, 2000, pp. 1190–1194.

[21] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints", *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[22] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[23] A. Ben-Tal and A. Nemirovski, "Robust convex optimization", *Mathematics of Operations Research*, vol. 23, no. 4, pp. 769–805, 1998.

[24] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees", *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[25] D. Bertsekas, *Abstract Dynamic Programming*. Athena Scientific, 2022.

[26] D. P. Bertsekas, "Nonlinear programming", *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[27] D. P. Bertsekas, "Value and policy iterations in optimal control and adaptive dynamic programming", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 500–509, 2015.

[28] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[29] P. N. Beuchat and J. Lygeros, "Approximate dynamic programming via penalty functions", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 814–11 821, 2017.

[30] F. Blanchini, "Set invariance in control", *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[31] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*. Springer Science & Business Media, 2013.

[32] A. D. Bonzanini, J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Scalable estimation of invariant sets for mixed-integer nonlinear systems using active deep learning", in *the 61st IEEE Conference on Decision and Control (CDC)*, 2022, pp. 3431–3437.

[33] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, "An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems", in *2003 American Control Conference (ACC)*, vol. 6, 2003, pp. 4717–4722.

[34] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems", *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[35] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[36] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[37] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[38] J. Breeden and D. Panagou, "Predictive control barrier functions for online safety critical control", in *the 61st IEEE Conference on Decision and Control (CDC)*, 2022, pp. 924–931.

[39] L. Brunke, M. Greeff, A. W. Hall, *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[40] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2017.

[41] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators", *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[42] M. Cannon, B. Kouvaritakis, and X. Wu, "Probabilistic constrained MPC for multiplicative and additive stochastic uncertainty", *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1626–1632, 2009.

[43] A. Cetinkaya, H. Ishii, and T. Hayakawa, "An overview on denial-of-service attacks in control systems: Attack models and security analyses", *Entropy*, vol. 21, no. 2, p. 210, 2019.

[44] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Żak, and G. T. Buzzard, "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences", *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 135–148, 2016.

[45] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. G. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 405–419, 2020.

[46] A. Chakrabarty, R. Quirynen, C. Danielson, and W. Gao, "Approximate dynamic programming for linear systems with state and input constraints", in *2019 European Control Conference (ECC)*, 2019, pp. 524–529.

[47] S. Chen and M. Fazlyab, "Learning performance-oriented control barrier functions under complex safety constraints and limited actuation", *arXiv preprint arXiv:2401.05629*, 2024.

[48] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for piecewise affine systems with neural network controllers", *arXiv preprint arXiv:2008.06546*, 2020.

[49] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for hybrid systems", in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[50] S. Chen, K. Saulnier, N. Atanasov, *et al.*, "Approximating explicit model predictive control using constrained neural networks", in *2018 Annual American Control Conference (ACC)*, 2018, pp. 1520–1527.

[51] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets", *Automatica*, vol. 135, p. 109 947, 2022.

[52] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study", in *the 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6835–6841.

[53] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks", in *The AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[54] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control", *Systems & Control Letters*, vol. 29, no. 3, pp. 121–129, 1996.

[55] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier–value functions for safety-critical control", in *the 60th IEEE Conference on Decision and Control*, 2021, pp. 6814–6821.

[56] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Neural networks for continuous online learning and control", *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1511–1531, 2006.

[57] F. Clarke, "On the inverse function theorem", *Pacific Journal of Mathematics*, vol. 64, no. 1, pp. 97–102, 1976.

[58] F. H. Clarke, "Generalized gradients and applications", *Transactions of the American Mathematical Society*, vol. 205, pp. 247–262, 1975.

[59] F. H. Clarke, *Optimization and Nonsmooth Analysis*. SIAM, 1990.

[60] M. H. Cohen and C. Belta, "Safe exploration in model-based reinforcement learning using control barrier functions", *Automatica*, vol. 147, p. 110 684, 2023.

[61] M. H. Cohen, P. Ong, G. Bahati, and A. D. Ames, "Characterizing smooth safety filters via the implicit function theorem", *IEEE Control Systems Letters*, vol. 7, pp. 3890–3895, 2023.

[62] D. Corona and B. De Schutter, "Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods", *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 365–372, 2008.

[63] C. J. Cortes, M. Thitsa, and S. Coogan, "Discontinuous barrier functions for piecewise continuous dynamics", in *2024 American Control Conference (ACC)*, 2024, pp. 3987–3992.

[64] R. K. Cosner, P. Culbertson, A. J. Taylor, and A. D. Ames, "Robust safety under stochastic uncertainty with discrete-time control barrier functions", *arXiv preprint arXiv:2302.07469*, 2023.

[65] R. K. Cosner, A. W. Singletary, A. J. Taylor, T. G. Molnar, K. L. Bouman, and A. D. Ames, "Measurement-robust control barrier functions: Certainty in safety with uncertainty in state", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6286–6291.

[66] H. Dai, B. Landry, M. Pavone, and R. Tedrake, "Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems", in *the 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1274–1281.

[67] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control", *arXiv preprint arXiv:2109.14152*, 2021.

[68] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces", *arXiv preprint arXiv:1801.08757*, 2018.

[69] 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", *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, 2023.

[70] B. De Cooman and J. Suykens, "A dual perspective of reinforcement learning for imposing policy constraints", *IEEE Transactions on Artificial Intelligence*, 2025.

[71] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness", *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2019.

[72] C. De Persis and P. Tesi, "Learning controllers for nonlinear systems from data", *Annual Reviews in Control*, p. 100 915, 2023.

[73] B. De Schutter and T. van den Boom, "MPC for continuous piecewise-affine systems", *Systems & Control Letters*, vol. 52, no. 3-4, pp. 179–192, 2004.

[74] H. P. Decell Jr, "An application of the Cayley-Hamilton theorem to generalized matrix inversion", *SIAM Review*, vol. 7, no. 4, pp. 526–528, 1965.

[75] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, "Control barriers in bayesian learning of system dynamics", *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 214–229, 2021.

[76] A. Didier, R. C. Jacobs, J. Sieber, K. P. Wabersich, and M. N. Zeilinger, "Approximate predictive control barrier functions using neural networks: A computationally cheap and permissive safety filter", in *2023 European Control Conference (ECC)*, 2023, pp. 1–7.

[77] A. Didier and M. N. Zeilinger, "Approximate predictive control barrier function for discrete-time systems", *arXiv preprint arXiv:2411.11610*, 2024.

[78] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints", *Neurocomputing*, vol. 484, pp. 128–141, 2022.

[79] S. Fahandezh-Saadi and M. Tomizuka, "In proximity of ReLU DNN, PWA function, and explicit MPC", *arXiv preprint arXiv:2006.05001*, 2020.

[80] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems", *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[81] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems", *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[82] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning", in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8550–8556.

[83] L. Gharavi, B. De Schutter, and S. Baldi, "Efficient MPC for emergency evasive maneuvers, Part I: Hybridization of the nonlinear problem", *arXiv preprint arXiv:2310.00715*, 2023.

[84] A. Ghezzi, J. Hoffman, J. Frey, J. Boedecker, and M. Diehl, "Imitation learning from nonlinear MPC via the exact Q-loss and its Gauss-Newton approximation", in *the 62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 4766–4771.

[85] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems", *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 310–315, 2017.

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[87] D. Görges, "Relations between model predictive control and reinforcement learning", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4920–4928, 2017.

[88] P. Grieder, M. Kvasnica, M. Baotić, and M. Morari, "Stabilizing low complexity feedback control of constrained piecewise affine systems", *Automatica*, vol. 41, no. 10, pp. 1683–1694, 2005.

[89] N. Groot, B. De Schutter, and H. Hellendoorn, "Integrated model predictive traffic and emission control using a piecewise-affine approach", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 587–598, 2012.

[90] S. Gros and M. Zanon, "Learning for MPC with stability & safety guarantees", *Automatica*, vol. 146, p. 110 598, 2022.

[91] A. Gupta, A. S. Khwaja, A. Anpalagan, L. Guan, and B. Venkatesh, "Policy-gradient and actor-critic based state representation learning for safe driving of autonomous vehicles", *Sensors*, vol. 20, no. 21, p. 5991, 2020.

[92] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2024. [Online]. Available: https://www.gurobi.com.

[93] T. Gurriet, M. Mote, A. Singletary, P. Nilsson, E. Feron, and A. D. Ames, "A scalable safety critical control framework for nonlinear systems", *IEEE Access*, vol. 8, pp. 187 249–187 275, 2020.

[94] B. Hanin, "Universal function approximation by deep neural nets with bounded width and ReLU activations", *Mathematics*, vol. 7, no. 10, p. 992, 2019.

[95] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2020.

[96] K. He, A. Alan, S. Shi, T. v. d. Boom, and B. De Schutter, "Predictive control barrier functions for piecewise affine systems with non-smooth constraints", *arXiv preprint arXiv:2510.21321*, 2025.

[97] K. He, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach", *arXiv preprint arXiv:2205.10065*, 2022.

[98] K. He, C. Dong, A. Yan, Q. Zheng, B. Liang, and Q. Wang, "Composite deep learning control for autonomous bicycles by using deep deterministic policy gradient", in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 2766–2773.

[99] K. He, S. Shi, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained piecewise affine systems with stability and safety guarantees", *arXiv preprint arXiv:2306.15723*, 2023.

[100] K. He, S. Shi, T. van den Boom, and B. De Schutter, "State-action control barrier functions: Imposing safety on learning-based control with low online computational costs", *arXiv preprint arXiv:2312.11255*, 2023.

[101] K. He, S. Shi, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach", *Automatica*, vol. 160, p. 111 456, 2024.

[102] K. He, S. Shi, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained piecewise affine systems with stability and safety guarantees", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 3, pp. 1722–1734, 2024.

[103] K. He, S. Shi, T. van den Boom, and B. De Schutter, "Efficient and safe learning-based control of piecewise affine systems using optimization-free safety filters", in *the 63rd IEEE Conference on Decision and Control (CDC)*, 2024, pp. 5046–5053.

[104] K. He, S. Shi, T. van den Boom, and B. D. Schutter, *From learning to safety: A direct data-driven framework for constrained control*, 2025. arXiv: 2505.15515.

[105] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0", in *2013 European Control Conference (ECC)*, https://www.mpt3.org/, Zürich, Switzerland, 2013, pp. 502–510.

[106] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees", *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[107] A. Heydari, "Stability analysis of optimal adaptive control using value iteration with approximation errors", *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 3119–3126, 2018.

[108] W. Hoeffding, "Probability inequalities for sums of bounded random variables", *The Collected Works of Wassily Hoeffding*, pp. 409–426, 1994.

[109] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[110] A. M. Jarrett, D. Faghihi, D. A. Hormuth, *et al.*, "Optimal control theory for personalized therapeutic regimens in oncology: Background, history, challenges, and opportunities", *Journal of Clinical Medicine*, vol. 9, no. 5, p. 1314, 2020.

[111] H. Jiang, H. Zhang, Y. Luo, and J. Han, "Neural-network-based robust control schemes for nonlinear multiplayer systems with uncertainties via adaptive dynamic programming", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 3, pp. 579–588, 2018.

[112] Z.-P. Jiang, T. Bian, W. Gao, *et al.*, "Learning-based control: A tutorial and some recent results", *Foundations and Trends® in Systems and Control*, vol. 8, no. 3, pp. 176–284, 2020.

[113] M. Johansson and A. Rantzer, "Computation of piecewise quadratic Lyapunov functions for hybrid systems", in *1997 European Control Conference (ECC)*, IEEE, 1997, pp. 2005–2010.

[114] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning", *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020.

[115] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis", in *the 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 4947–4954.

[116] B. Karg and S. Lucia, "Reinforced approximate robust nonlinear model predictive control", in *2021 23rd International Conference on Process Control (PC)*, 2021, pp. 149–156.

[117] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control", in *the 39th IEEE Conference on Decision and Control*, vol. 5, 2000, pp. 4951–4956.

[118] H. K. Khalil, *Nonlinear Systems*, 3rd. Prentice Hall, 2002.

[119] D. Kleinman, "On an iterative technique for riccati equation computations", *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 114–115, 1968.

[120] S. Kolathaya and A. D. Ames, "Input-to-state safety with control barrier functions", *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 108–113, 2018.

[121] M. Korda, "Computing controlled invariant sets from data using convex optimization", *SIAM Journal on Control and Optimization*, vol. 58, no. 5, pp. 2871–2899, 2020.

[122] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit MPC", *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.

[123] W. Lavanakul, J. J. Choi, K. Sreenath, and C. J. Tomlin, "Safety filters for black-box dynamical systems by learning discriminating hyperplanes", *arXiv preprint arXiv:2402.05279*, 2024.

[124] M. Lazar, W. Heemels, S. Weiland, and A. Bemporad, "Stabilizing model predictive control of hybrid systems", *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1813–1818, 2006.

[125] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[126] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems", *arXiv preprint arXiv:2005.01643*, 2020.

[127] B. Li, Y. Tan, A.-G. Wu, and G.-R. Duan, "A distributionally robust optimization based method for stochastic model predictive control", *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 5762–5776, 2021.

[128] J. Li, D. Fridovich-Keil, S. Sojoudi, and C. J. Tomlin, "Augmented Lagrangian method for instantaneously constrained reinforcement learning problems", in *the 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2982–2989.

[129] Y. Li, K. Hua, and Y. Cao, "Using stochastic programming to train neural network approximation of nonlinear MPC laws", *Automatica*, vol. 146, p. 110 665, 2022.

[130] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Robust action governor for uncertain piecewise affine systems with non-convex constraints and safe reinforcement learning", *arXiv preprint arXiv:2207.08240*, 2022.

[131] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning", in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[132] M. Lin, Z. Sun, Y. Xia, and J. Zhang, "Reinforcement learning-based model predictive control for discrete-time systems", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 3312–3324, 2023.

[133] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, "Finite approximation error-based value iteration ADP", *Adaptive Dynamic Programming with Applications in Optimal Control*, pp. 91–149, 2017.

[134] T. Liu, Y. Gao, and M. Buss, "Adaptive output tracking control of piecewise affine systems with prescribed performance", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 5398–5410, 2021.

[135] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-point policy optimization under constraints", in *The AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 4940–4947.

[136] W. Lohmiller, P. Gassert, and J.-J. Slotine, "Deep minmax networks", in *the 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2929–2934.

[137] F. V. Louveaux, "Piecewise convex programs", *Mathematical Programming*, vol. 15, no. 1, pp. 53–62, 1978.

[138] H. Ma, C. Liu, S. E. Li, S. Zheng, and J. Chen, "Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning", in *Learning for Dynamics and Control Conference*, PMLR, 2022, pp. 97–109.

[139] E. T. Maddalena, C. d. S. Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit MPC controllers from data", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 362–11 367, 2020.

[140] S. Mallick, A. Dabiri, and B. De Schutter, "Learning-based model predictive control for piecewise affine systems with feasibility guarantees", *arXiv preprint arXiv:2412.00490*, 2024.

[141] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments", in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 31–38.

[142] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems", *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2433–2448, 2020.

[143] P.-F. Massiani, S. Heim, F. Solowjow, and S. Trimpe, "Safe value functions", *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2743–2757, 2022.

[144] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming", in *2019 European Control Conference (ECC)*, 2019, pp. 1494–1499.

[145] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter, "Learning approximate semi-explicit hybrid MPC with an application to microgrids", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5207–5212, 2020.

[146] N. Mehr, D. Sadigh, R. Horowitz, S. S. Sastry, and S. A. Seshia, "Stochastic predictive freeway ramp metering from signal temporal logic specifications", in *2017 American Control Conference (ACC)*, 2017, pp. 4884–4889.

[147] A. Mesbah, K. P. Wabersich, A. P. Schoellig, *et al.*, "Fusion of machine learning and MPC under uncertainty: What advances are on the horizon?", in *2022 American Control Conference (ACC)*, 2022, pp. 342–357.

[148] P. Mestres, A. Allibhoy, and J. Cortés, "Regularity properties of optimization-based controllers", *European Journal of Control*, vol. 81, p. 101 098, 2025, ISSN: 0947-3580.

[149] P. Mestres, Y. Chen, E. Dall'anese, and J. Cortés, "Control barrier function-based safety filters: Characterization of undesired equilibria, unbounded trajectories, and limit cycles", *arXiv preprint arXiv:2501.09289*, 2025.

[150] F. Micheli, T. Summers, and J. Lygeros, "Data-driven distributionally robust MPC for systems with uncertain dynamics", in *the 61st IEEE Conference on Decision and Control (CDC)*, 2022, pp. 4788–4793.

[151] T. G. Molnar and A. D. Ames, "Safety-critical control with bounded inputs via reduced order models", in *2023 American Control Conference (ACC)*, 2023, pp. 1414–1421.

[152] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks", *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[153] B. S. Mordukhovich and N. M. N. and, "Geometric approach to convex subdifferential calculus", *Optimization*, vol. 66, no. 6, pp. 839–873, 2017. DOI: 10.1080/02331934.2015.1105225.

[154] F. Moreno-Mora, L. Beckenbach, and S. Streif, "Predictive control with learning-based terminal costs using approximate value iteration", *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3874–3879, 2023.

[155] P. Ong, B. Capelli, L. Sabattini, and J. Cortés, "Nonsmooth control barrier function design of continuous constraints for network connectivity maintenance", *Automatica*, vol. 156, p. 111 209, 2023.

[156] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro, "Learning safe policies via primal-dual methods", in *the 58th IEEE Conference on Decision and Control (CDC)*, 2019, pp. 6491–6497.

[157] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods", *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1321–1336, 2022.

[158] A. Perrusquía, "Human-behavior learning: A new complementary learning perspective for optimal decision making controllers", *Neurocomputing*, vol. 489, pp. 157–166, 2022.

[159] T. Pippia, J. Sijs, and B. De Schutter, "A parametrized model predictive control approach for microgrids", in *the 57th IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3171–3176.

[160] T. Poggi, S. Trimboli, A. Bemporad, and M. Storace, "Explicit hybrid model predictive control: Discontinuous piecewise-affine approximation and FPGA implementation", *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1350–1355, 2011.

[161] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost", *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2736–2749, 2016.

[162] R. Postoyan, M. Granzotto, L. Buşoniu, B. Scherrer, D. Nešsić, and J. Daafouz, "Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration", in *the 58th IEEE Conference on Decision and Control (CDC)*, 2019, pp. 487–492.

[163] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates", *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[164] S. V. Rakovic, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen, "Parameterized tube model predictive control", *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2746–2761, 2012.

[165] A. Ramezani-Kebrya, K. Antonakopoulos, V. Cevher, A. Khisti, and B. Liang, "On the generalization of stochastic gradient descent with momentum", *Journal of Machine Learning Research*, vol. 25, no. 22, pp. 1–56, 2024.

[166] J. B. Rawlings, D. Q. Mayne, M. Diehl, *et al.*, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[167] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl, "A hierarchical approach for strategic motion planning in autonomous racing", in *2023 European Control Conference (ECC)*, 2023, pp. 1–8.

[168] R. Reiter, J. Hoffmann, D. Reinhardt, *et al.*, "Synthesis of model predictive control and reinforcement learning: Survey and classification", *arXiv preprint arXiv:2502.02133*, 2025.

[169] A. Richards and J. How, "Mixed-integer programming for control", in *2005 American Control Conference (ACC)*, 2005, pp. 2676–2683.

[170] M. Riedmiller, "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method", in *European Conference on Machine Learning*, 2005, pp. 317–328.

[171] A. Rinnooy Kan and G. Timmer, "Stochastic global optimization methods part II: Multi level methods", *Mathematical Programming*, vol. 39, no. 1, pp. 57–78, 1987.

[172] A. Robey, H. Hu, L. Lindemann, *et al.*, "Learning control barrier functions from expert demonstrations", in *the 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3717–3724.

[173] A. Robey, L. Lindemann, S. Tu, and N. Matni, "Learning robust hybrid control barrier functions for uncertain systems", *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 1–6, 2021.

[174] L. Russo, S. H. Nair, L. Glielmo, and F. Borrelli, "Learning for online mixed-integer model predictive control with parametric optimality certificates", *IEEE Control Systems Letters*, vol. 7, pp. 2215–2220, 2023.

[175] S. Sadraddini and R. Tedrake, "Sampling-based polytopic trees for approximate optimal control of piecewise affine systems", in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7690–7696.

[176] N. Schmid, M. Fochesato, S. H. Li, T. Sutter, and J. Lygeros, "Computing optimal joint chance constrained control policies", *IEEE Transactions on Automatic Control*, 2025.

[177] R. Schwan, C. N. Jones, and D. Kuhn, "Stability verification of neural network controllers using mixed-integer programming", *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 7514–7529, 2023.

[178] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective", *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.

[179] P. O. Scokaert and J. B. Rawlings, "Constrained linear quadratic regulation", *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1163–1169, 1998.

[180] S. Shi, A. Tsiamis, and B. De Schutter, "Suboptimality analysis of receding horizon quadratic control with unknown linear systems and its applications in learning-based control", *arXiv preprint arXiv:2301.07876*, 2023.

[181] D. Šiljak and D. Stipanović, "Robust stabilization of nonlinear systems: The LMI approach", *Mathematical Problems in Engineering*, vol. 6, no. 5, pp. 461–493, 2000.

[182] C. F. O. da Silva, A. Dabiri, and B. De Schutter, "Integrating reinforcement learning and model predictive control with applications to microgrids", *arXiv preprint arXiv:2409.11267*, 2024.

[183] A. Singletary, M. Ahmadi, and A. D. Ames, "Safe control for nonlinear systems with stochastic uncertainty via risk control barrier functions", *IEEE Control Systems Letters*, vol. 7, pp. 349–354, 2022.

[184] Y. Song and C. T. Maravelias, "Constructing optimal piecewise quadratic approximations for enhancing deterministic global optimization", *INFORMS Journal on Computing*, 2025.

[185] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7139–7145.

[186] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 943–949, 2008.

[187] D. C. Tan, F. Acero, R. McCarthy, D. Kanoulas, and Z. Li, "Value functions are control barrier functions: Verification of safe policies using control theory", *arXiv preprint arXiv:2306.04026*, 2023.

[188] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions", in *Learning for Dynamics and Control*, PMLR, 2020, pp. 708–717.

[189] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization", *arXiv preprint arXiv:1805.11074*, 2018.

[190] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions", in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 286–292.

[191] A. Thirugnanam, J. Zeng, and K. Sreenath, "Nonsmooth control barrier functions for obstacle avoidance between convex regions", *arXiv preprint arXiv:2306.13259*, 2023.

[192] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions", *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.

[193] S. Tonkens and S. Herbert, "Refining control barrier functions through hamilton-jacobi reachability", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 355–13 362.

[194] J. Tordesillas, J. P. How, and M. Hutter, "RAYEN: Imposition of hard convex constraints on neural networks", *arXiv preprint arXiv:2307.08336*, 2023.

[195] D. E. Van Wijk, S. Coogan, T. G. Molnar, M. Majji, and K. L. Hobbs, "Disturbance-robust backup control barrier functions: Safety under uncertain dynamics", *IEEE Control Systems Letters*, vol. 8, pp. 2817–2822, 2024.

[196] K. P. Wabersich, A. J. Taylor, J. J. Choi, *et al.*, "Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems", *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[197] K. P. Wabersich and M. N. Zeilinger, "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control", *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2638–2651, 2022.

[198] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems", *Automatica*, vol. 129, p. 109 597, 2021.

[199] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares", in *2018 American Control Conference (ACC)*, 2018, pp. 585–590.

[200] Y. Wang and X. Xu, "Observer-based control barrier functions for safety critical systems", in *2022 American Control Conference (ACC)*, 2022, pp. 709–714.

[201] Z. Wang and R. M. Jungers, "Data-driven computation of invariant sets of discrete time-invariant black-box systems", *arXiv preprint arXiv:1907.12075*, 2019.

[202] C. W. Warren, "Global path planning using artificial potential fields", in *1989 IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 1989, pp. 316–317.

[203] C. J. Watkins and P. Dayan, "Q-learning", *Machine Learning*, vol. 8, pp. 279–292, 1992.

[204] P. Werbos, "Approximate dynamic programming for realtime control and neural modelling", *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pp. 493–525, 1992.

[205] T. Westenbroek, A. Agrawal, F. Castañeda, S. S. Sastry, and K. Sreenath, "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers", *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 19–24, 2021.

[206] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999, vol. 55.

[207] W. Xiao, T.-H. Wang, R. Hasani, *et al.*, "Barriernet: Differentiable control barrier functions for learning of safe robot control", *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2289–2307, 2023.

[208] F. Xiong, B. Sun, X. Yang, *et al.*, "Guided policy search for sequential multitask learning", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 216–226, 2018.

[209] J. Xu, T. van den Boom, L. Buşoniu, and B. De Schutter, "Model predictive control for continuous piecewise affine systems using optimistic optimization", in *2016 American Control Conference (ACC)*, 2016, pp. 4482–4487.

[210] J. Xu, J. Wang, J. Rao, Y. Zhong, and H. Wang, "Adaptive dynamic programming for optimal control of discrete-time nonlinear system with state constraints based on control barrier function", *International Journal of Robust and Nonlinear Control*, vol. 32, no. 6, pp. 3408–3424, 2022.

[211] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning", in *International Conference on Machine Learning*, PMLR, 2022, pp. 25 636–25 655.

[212] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC", *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[213] X. Zhao and Q. Xu, "Explicit reinforcement learning safety layer for computationally efficient inverter-based voltage regulation", in *2023 American Control Conference (ACC)*, 2023, pp. 4501–4506.

[214] J. Zheng, J. Miller, and M. Sznaier, "Data-driven safe control of discrete-time nonlinear systems", *IEEE Control Systems Letters*, vol. 8, pp. 1553–1558, 2024.

[215] L. Zheng, Y. Shi, L. J. Ratliff, and B. Zhang, "Safe reinforcement learning of control-affine systems with vertex networks", in *Learning for Dynamics and Control*, PMLR, 2021, pp. 336–347.

CURRICULUM VITÆ

Kanghui He was born on July 2, 1996, in Xi'an, Shaanxi, China. He is a PhD candidate at the Delft Center for Systems and Control, Delft University of Technology, the Netherlands, under the supervision of Prof. Bart De Schutter, Prof. Ton van den Boom, and Dr. Shengling Shi. He received the B.Sc. degree at the School of Mechanical Engineering and Automation, Beihang University, in 2018, and the M.Sc. degree (with an outstanding graduation thesis and national scholarship) in the Department of Dynamics and Control, Beihang University, in 2021. He was a visiting student in the Automated Driving and Human-Machine System Lab in August 2019, hosted by Prof Chen Lv. He was a research assistant in the School of Automation, Tsinghua University, in 2021. He was a visiting research fellow in the Department of Electrical Engineering, KU Leuven, from February 2025 to April 2025, hosted by Prof. Johan Suykens.

His research interests lie in the intersection of control, learning, and optimization. He has research experience in safe learning-based control, disturbance rejection control, learning-based planning and control of mobile robots, multi-sensor fusion of unmanned aerial vehicles, and deep-learning-assisted radiotherapy. His doctoral research focuses on integrating learning and optimization to develop control methods with safety guarantees and computational efficiency. His research ambition is to develop learning-based decision-making algorithms for large-scale, complex systems under uncertainties and constraints.

List of Publications

Journals

5. **K. He**, S. Shi, T. van den Boom, and B. De Schutter. Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach, *Automatica*, vol. 160, p. 111 456, 2024.

4. **K. He**, S. Shi, T. van den Boom, and B. De Schutter. State-action control barrier functions: Imposing safety on learning-based control with low online computational costs, *conditionally accepted by IEEE Transactions on Automatic Control*, 2025.

3. **K. He**, S. Shi, T. van den Boom, and B. De Schutter. Approximate dynamic programming for constrained piecewise affine systems with stability and safety guarantees, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 3, pp. 1722–1734, 2024.

2. **K. He**, S. Shi, T. van den Boom, and B. De Schutter. From learning to safety: A direct data-driven framework for constrained control, *submitted to Journal*, 2025.

1. **K. He**, A. Alan, S. Shi, T. van den Boom, and B. De Schutter. Predictive control barrier functions for piecewise affine systems with non-smooth constraints, *submitted to Journal*, 2025.

Conferences

1. **K. He**, S. Shi, T. van den Boom, and B. De Schutter. Efficient and safe learning-based control of piecewise affine systems using optimization-free safety filters, *IEEE 63rd Conference on Decision and Control (CDC)*, 2024.