Title
Challenges of Control Barrier Functions: Optimization, Control, Planning, and Navigation

Author
Zeng, Jun

Publication Date
2022

Peer reviewed|Thesis/dissertation

Challenges of Control Barrier Functions: Optimization, Control, Planning, and Navigation

by

Jun Zeng

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Koushil Sreenath, Chair
Professor Shankar S. Sastry
Professor Pieter Abbeel

Spring 2022

Challenges of Control Barrier Functions: Optimization, Control, Planning, and Navigation

Abstract

Challenges of Control Barrier Functions: Optimization, Control, Planning, and Navigation

by

Jun Zeng

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Associate Professor Koushil Sreenath, Chair

Control barrier functions (CBFs) are one of the many used approaches for achieving safety in robot autonomy. This thesis tackles several challenges present in control barrier functions in different aspects, including optimization, control, planning and navigation.

This thesis is composed of three parts. In Part I, we point out the optimization infeasibility between CBF constraint and input constraint, and address the feasibility problem in optimal control for quadratic programming using control barrier functions under input constraints. We also notice that the potential conflict between input constraints and safety constraints also exists in the discrete-time domain together with model predictive control. This conflict is formally identified in reachability analysis and reachability is later enhanced in proposed formulations.

In the Part II, we focus on obstacle avoidance for control and trajectory generation in a tight environment, which requires polytopic obstacle avoidance. We analyze the optimization problem for obstacle avoidance between polytopes. The novel optimizations for obstacle avoidance in continuous domain and discrete-time domain are proposed to solve this challenge.

In Part III of the thesis, we discuss several applications of motion planning and navigation using control barrier functions. We firstly address the motion planning problem within a finite state machine which unifies a mid-level planner and a low-level safety-critical controller with application to autonomous driving. Next, we propose parallelism for motion planning with control barrier functions with application to autonomous racing. Finally, we explore the possibility of safety-critical motion planning for high dimensional systems such as Cassie, a life-size bipedal robot.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

The past five years of research life during the Ph.D. study at UC Berkeley have been an invaluable journey for me. I would not have reached so far without the tremendous and generous help from many people.

First and foremost, my deep and sincere gratitude goes to my Ph.D. advisor Professor Koushil Sreenath, who has been a great mentor to me during the past five years for his profound knowledge, insightful perspectives, patience and enthusiasm on work and research, and positiveness towards life. Without his guidance and persistent help, this dissertation would not have been possible. I sincerely wish I could be a great professional person, an extraordinary mentor as he is in the future. I hope to continue to uphold the same level of excellence you instilled in me throughout the rest of my career.

I am very grateful Professor Shankar Sastry and Professor Pieter Abbeel for serving on my dissertation committee as well as in my qualifying exam committee. I would also want to say thanks to Professor Mark Mueller for serving as a committee member in my qualifying exam. They provided me with great support for my major milestones in the Ph.D. study.

Being a member in the Hybrid Robotics Group has been wonderful in the past five years. I sincerely thank my major collaborators at my lab for all the inspiring discussions: Zhongyu Li, Akshay Thirugnanam, Bike Zhang Prasanth Kotaru, Ayush Agrawal. I also like to express my thanks to the other former and current lab members: Dr. Katherine Poggenssee, Dr. Avinash Sirvaru, Dr. Guofan Wu, Shuxiao Chen, Matthew Wen, Fernando Castaneda, Jason Choi and Johnathon Li, for their help during my graduate study and their efforts to maintain a good research ambiance in the group. Meanwhile, I would like to thank the visiting scholars and students in Hybrid Robotics Group: Suiyi He, Lizhi Yang, Chenyu Yang, Anxing Xiao, Wenzhe Tong, Scott Gilroy, Derek Lau, Mengti Sun, Guo Ning Sue, Haotian Shen, Yufeng Chi, Massimiliano de Sa and Han Hoang Nguyen.

The whole Ph.D. studies would not have been possible without funding from the following sources: National Science Foundation Grant (CMMI-1840219, CMMI-1944722, CMMI-1931853), Berkeley Deep Drive, the UC Berkeley Graduate Division Block Grant Award, Army Research Laboratory, Chin L. S. Chun Fellowship. Great thanks also go to all my external collaborators and the valuable discussions with them enhanced this dissertation and related research through my path of Ph.D studies. In particular, I would like to thank Xiangyu Wu, Ashish Kumar, Professor Mark Mueller, Professor Jitendra Malik from UC Berkeley, Professor Deepak Pathak from Carnegie Mellon University. Moreover, I would like to say great thanks to Mingxiang Fan and Dr. Qianli Xing for their mentorship during the summer internships.

Last but not least, my deepest love would be reserved to my parents for their unconditioned love, encouragement and enlightenment.

Chapter 1

Introduction

Control barrier functions (CBFs) have recently emerged as one of the common approaches for safety-critical control. Although the theory of control barrier functions has been well established, there exist several limitations. Some of these arise due to the mathematical challenges in optimization (e.g., infeasibility due to input constraints and multiple control barrier functions). Other challenges arise for the applications such as the following: limited performance and deadlock issues for trajectory generation with finite state machines; real-time obstacle avoidance in tight environments; and computationally-fast trajectory generation for high-dimensional nonlinear systems.

The remainder of this chapter seeks to (1) provide clarifications of existing challenges of control barrier functions in different domains, including optimization, control, planning and navigation (2) present the thesis contributions in each domain. Finally, an outline of this dissertation is presented at the end of this chapter, highlighting how the challenges are addressed.

1.1 Stability & Safety

The theory and applications of safety-critical control and planning are emerging topics for autonomous systems. "Safety" requires that "bad" things do not happen while liveness requires that "good" things eventually happen. For example, asymptotic stability can be seen as an example of a liveness property in the sense that an asymptotically stable equilibrium point is eventually reached. Similarly, invariance can be seen as an example of a safety property in the sense that any trajectory starting inside an invariant set will never reach the complement of the set. For example, the robot is required to staying outside the obstacle invariantly for safety, i.e., without any physical contact with the obstacle. These asymptotic stability and invariance are later formalized as control Lyapunov functions (CLFs) and control barrier functions (CBFs), respectively.

1.2 Control Barrier Functions

Designing controllers to ensure provable safety guarantees for autonomous systems is vital. The choice of the term "barrier" was motivated by optimization literature where barrier functions are added to the cost functions to avoid undesirable regions through repelling term in the cost function [102]. This collision avoidance behavior, i.e., staying collision-free, is denoted as system safety. In recent years, researchers have formalized this obstacle avoidance in an optimization-based manner with obstacle avoidance constraints. These barrier constraints are required to enforce the system trajectory to stay outside the obstacle region. For systems with a control input, the notion of a barrier was extended to a "control" version to yield a "control barrier function" [205]. A control barrier function is positive within the safe set, zero on the boundary of the safe set, and negative outside the safe set. Moreover, at each point inside the safe set, there always exists a choice of control that can keep the closed-loop system trajectory within the safe set. Furthermore, this control input satisfies a particular inequality on the time-derivative of the control barrier function.

The studies of control barrier functions have been a variety of approaches that can be best described as "Lyapunov-like". In other words, these functions yield invariant level sets such that one can guarantee safety if these level sets are contained in the safe set [12].. A type of quadratic programming, named as CBF-QP in [13], permits us to find the minimum perturbation for a given feedback controller to guarantee safety. Control Lyapunonv functions (CLFs) [58, 158] can be applied to stabilize the closed-loop dynamics of both linear and nonlinear dynamical systems [61]. Together with CBFs, the framework CLF-CBF-QP [11] incorporates control Lyapunov function based quadratic programs, which enables handling safety-critical constraints effectively in real-time with low complexity of solving QPs. These methods are generally applicable for different continuous nonlinear systems such as autonomous cars [129, 40], legged robots [135, 81], and aerial systems [209]. CBFs could also be generalized for high-order dynamics [133, 214] and discrete-time systems [2, 224]. Recently, CBF constraints were also applied in control problems using a data-driven approach [40, 41, 184, 116]. In this thesis, we focus on resolving the following challenges about control barrier functions.

1.3 Challenges

1.3.1 Optimization Feasibility & Safety

For real robot applications, besides the performance (CLF) and safety criteria (CBF) constraints, an input constraint is also usually applied to ensure the optimized input from QP is admissible. The input constraint was not considered in the optimization in some of the previous work, such as [12], which means the optimized control input might not be executable due to the physical limitations of the system. Other work, such as [209, 217, 138, 137, 2, 41], did consider the input constraint, but the potential conflict between input constraint

and CBF constraint was not addressed, potentially resulting in an unsolvable optimization problem. In [11], a valid CBF is specifically designed for an adaptive cruise control scenario, where the valid CBF is calculated with explicit integration of partial differential equations with the system dynamics under input constraint to ensure the feasibility in the optimization. Sufficient conditions about feasibility in CLF-CBF based methods are also discussed in [215]. It has been pointed that there is a potential conflict between safety constraints and CBF constraints within the CLF-CBF-QP formulation.

In this dissertation, we address the feasibility problem in optimal control for CLF-CBF-QP under input constraints in Chapter 3. We also notice that the potential conflict between input constraints and safety constraints also exists in discrete-time domain together with model predictive control, presented in Chapter 4. This conflict is later formally identified in reachability analysis and enhanced in proposed formulations discussed in Chapter 5.

1.3.2 Obstacle Avoidance for Control and Trajectory Generation

CBFs are also widely used for obstacle avoidance [222, 179, 83, 126, 74, 127]. The shapes of robots and obstacles are usually approximated as points [38], paraboloids [52] or hyperspheres [224], where the distance function can be calculated explicitly as an analytic expression from their geometric configuration. The distance functions for these shapes are differentiable and can be used as control barrier functions to construct a safety-critical optimal control problem. However, these approximations usually over-estimate the dimensions of the robot and obstacles, e.g., a rectangle is approximated as the smallest circle that contains it. When a tight-fitting obstacle avoidance motion is expected, robots and obstacles are usually approximated as polytopes. While this makes maneuvers less conservative for obstacle avoidance, computing the distance between two polytopes requires additional effort [63].

In this thesis, the polytopic obstacle avoidance with control barrier functions is discussed. The optimization for obstacle avoidance in continuous domain and discrete-time domain are discussed in Chapter 6 and Chapter 7, respectively.

1.3.3 Applications about Motion planning & Navigation

We are interested in applications about motion planning and navigation by using control barrier functions. Many existing applications that use control barrier functions [66, 2, 12] are within the control domain, e.g. one-step optimization which is relatively too greedy to be applied for motion planning and navigation, which might result in deadlock.

Toward resolving deadlock and local greediness, we firstly address this challenge within a finite state machine which unifies mid-level planner and a low-level safety-critical controller, shown in Chapter 8. Another approach to reduce the formation of a deadlock would be the introduction of parallelism, presented in Chapter 9. We also remark again that theoretical analysis for navigation between polytopic obstacles is shown in Chapter 6 and Chapter 7. Finally, we explore the possibility of safety-critical motion planning in high dimensional systems, e.g., a life-like bipedal robot Cassie, presented in Chapter 10. This highly nonlinear

hybrid system is able to be decoupled as a linear model within a RL (reinforcement learning) controller which is shown to be able to achieve model-based nonlinear model predictive control with CBFs as the local planner in a 3D navigation task.

1.4 Thesis Outline

The contributions of this thesis can be summarized as follows,

- Address feasibility problems between control barrier functions and input constraints in both continuous-time and discrete-time domains and propose formulations to resolve them.
- Propose model predictive control with control barrier functions and apply it to different robotic systems.
- Propose obstacle avoidance algorithms with control barrier functions for both continuous-time and discrete-time domains and apply them for control and trajectory generation problems.
- Discuss various applications on agile, nonlinear and high-dimensional systems for motion planning and navigation using control barrier functions.

We will first revisit an introductory chapter about control barrier functions in Chapter 2. After that, the thesis is divided into three parts, including theory and applications for the above challenges. The first part including Chapter 3, 4, 5 is about the optimization feasibility and the safety using control barrier functions. The second part including Chapter 6, 5 is about the obstacle avoidance for control and trajectory generation using control barrier functions. The third part including Chapter 8, 9, 10 focus on the applications about motion planning and navigation using control barrier functions.

The overall organization of the next eight chapters as well as their logical dependency is illustrated in Figure 1.1.

Chapter 3

Safety is one of the fundamental problems in robotics. Recently, a quadratic program based control barrier functions (CBFs) method has emerged as a way to enforce safety-critical constraints. Together with control Lyapunov functions (CLFs), it forms a safety-critical control strategy, named CLF-CBF-QP, which can mediate between achieving the control objective and ensuring safety, while being executable in real-time. However, once additional constraints such as input constraints are introduced, the CLF-CBF-QP may encounter infeasibility. In order to address the challenge that arises due to the infeasibility, we propose an optimal-decay form for safety-critical control wherein the decay rate of the CBF is optimized point-wise in time so as to guarantee point-wise feasibility when the state lies inside the safe

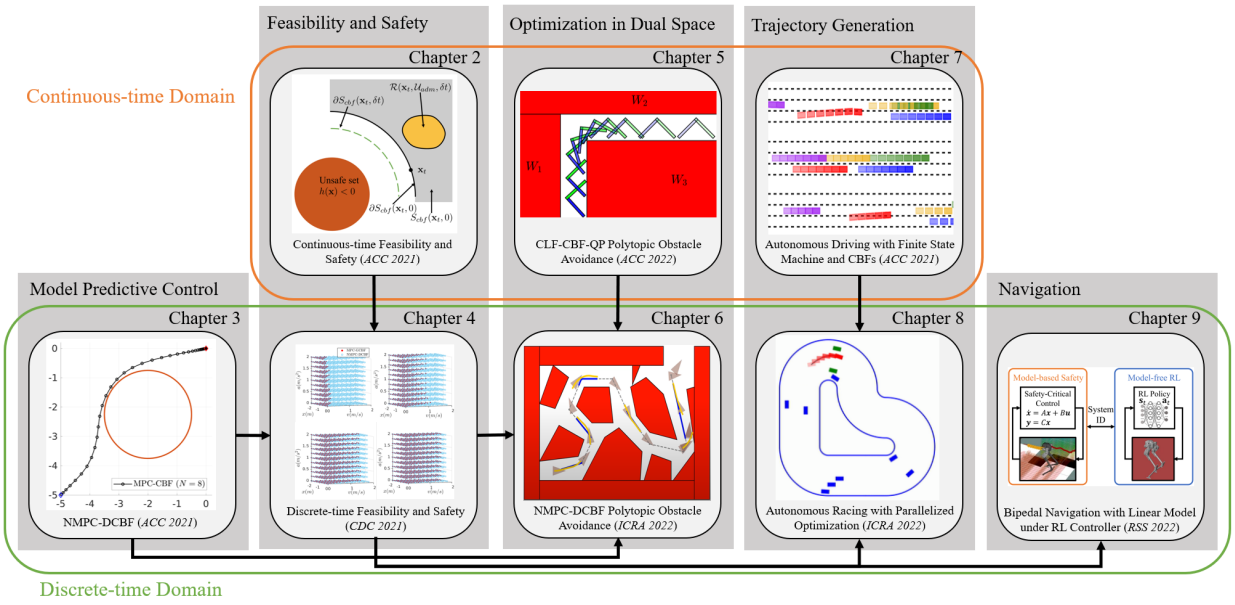


Figure 1.1: Thesis organization of chapters and their logic dependency.

set. The proposed control design is numerically validated using an adaptive cruise control example.

Chapter 4

The optimal performance of robotic systems is usually achieved near the limit of state and input bounds. Model predictive control (MPC) is a prevalent strategy to handle these operational constraints, however, safety still remains an open challenge for MPC as it needs to guarantee that the system stays within an invariant set. In order to obtain safe optimal performance in the context of set invariance, we present a safety-critical model predictive control strategy utilizing discrete-time control barrier functions (CBFs), which guarantees system safety and accomplishes optimal performance via model predictive control. We analyze the feasibility and the stability properties of our control design. We verify the properties of our method on a 2D double integrator model for obstacle avoidance. We also validate the algorithm numerically using a competitive car racing example, where the ego car is able to overtake other racing cars.

Chapter 5

Safety is one of the fundamental problems in robotics. Recently, one-step or multi-step optimal control problems for discrete-time nonlinear dynamical system were formulated to offer tracking stability using control Lyapunov functions (CLFs) while subject to input con-

straints as well as safety-critical constraints using control barrier functions (CBFs). The limitations of these existing approaches are mainly about feasibility and safety. In the existing approaches, the feasibility of the optimization and the system safety cannot be enhanced at the same time theoretically. In this chapter, we propose two formulations that unify CLFs and CBFs under the framework of nonlinear model predictive control (NMPC). In the proposed formulations, safety criteria is commonly formulated as CBF constraints and stability performance is ensured with either a terminal cost function or CLF constraints. Slack variables with relaxing technique are introduced on the CBF constraints to resolve the tradeoff between feasibility and safety so that they can be enhanced at the same. The advantages about feasibility and safety of proposed formulations compared with existing methods are analyzed theoretically and validated with numerical results.

Chapter 6

Developing controllers for obstacle avoidance between polytopes is a challenging and necessary problem for navigation in tight spaces. Traditional approaches can only formulate the obstacle avoidance problem as an offline optimization problem. To address these challenges, we propose a duality-based safety-critical optimal control using nonsmooth control barrier functions for obstacle avoidance between polytopes, which can be solved in real-time with a QP-based optimization problem. A dual optimization problem is introduced to represent the minimum distance between polytopes and the Lagrangian function for the dual form is applied to construct a control barrier function. We validate the obstacle avoidance with the proposed dual formulation for L-shaped (sofa-shaped) controlled robot in a corridor environment. We demonstrate real-time tight obstacle avoidance with non-conservative maneuvers on a moving sofa (piano) problem with nonlinear dynamics.

Chapter 7

Obstacle avoidance between polytopes is a challenging topic for optimal control and optimization-based trajectory planning problems. Existing work either solves this problem through mixed-integer optimization, relying on simplification of system dynamics, or through model predictive control with dual variables using distance constraints, requiring long horizons for obstacle avoidance. In either case, the solution can only be applied as an offline planning algorithm. In this chapter, we exploit the property that a smaller horizon is sufficient for obstacle avoidance by using discrete-time control barrier function (DCBF) constraints and we propose a novel optimization formulation with dual variables based on DCBFs to generate a collision-free dynamically-feasible trajectory. The proposed optimization formulation has lower computational complexity compared to existing work and can be used as a fast online algorithm for control and planning for general nonlinear dynamical systems. We validate our algorithm on different robot shapes using numerical simulations with a kinematic bicycle model, resulting in successful navigation through maze environments with polytopic obstacles.

Chapter 8

A new control design is developed for guaranteeing a vehicle's safety during lane change maneuvers in a complex traffic environment. The proposed method uses a finite state machine (FSM), where a quadratic program based optimization problem using control Lyapunov functions and control barrier functions (CLF-CBF-QP) is used to calculate the system's optimal inputs via rule-based control strategies. The FSM can make switches between different states automatically according to the command of driver and traffic environment, which makes the ego vehicle find a safe opportunity to do a collision-free lane change maneuver. By using a convex quadratic program, the controller can guarantee the system's safety at a high update frequency. A set of pre-designed typical lane change scenarios as well as randomly generated driving scenarios are simulated to show the performance of our controller.

Chapter 9

This chapter presents a novel planning and control strategy for competing with multiple vehicles in a car racing scenario. The proposed racing strategy switches between two modes. When there are no surrounding vehicles, a learning-based model predictive control (MPC) trajectory planner is used to guarantee that the ego vehicle achieves better lap timing performance. When the ego vehicle is competing with other surrounding vehicles to overtake, an optimization-based planner generates multiple dynamically-feasible trajectories through parallel computation. Each trajectory is optimized under a MPC formulation with different homotopic Bezier-curve reference paths lying laterally between surrounding vehicles. The time-optimal trajectory among these different homotopic trajectories is selected and a low-level MPC controller with control barrier function constraints for obstacle avoidance is used to guarantee system's safety-critical performance. The proposed algorithm has the capability to generate collision-free trajectories and track them while enhancing the lap timing performance with steady low computational complexity, outperforming existing approaches in both timing and performance for a autonomous racing environment. To demonstrate the performance of our racing strategy, we simulate with multiple randomly generated moving vehicles on the track and test the ego vehicle's overtake maneuvers.

Chapter 10

Bridging model-based safety and model-free reinforcement learning (RL) for dynamic robots is appealing, since model-based methods are able to provide formal safety guarantees while RL-based methods are able to exploit the robot agility by learning from the full-order system dynamics. However, current model-based and RL approaches to tackle this problem are mostly restricted to simple systems. In this chapter, we propose a new method to combine model-based safety with reinforcement learning by finding a low-dimensional representation of the system controlled by a RL policy and applying formal stability and safety guarantees on that simple model. We use an underactuated bipedal robot Cassie, which is a high

dimensional nonlinear system with hybrid dynamics, and its RL-based walking controller as an example. We show that a low-dimensional dynamical model is sufficient to capture the dynamics of the closed-loop system. We demonstrate that this model is linear and asymptotically stable, and is decoupled across control input in all dimensions. We further exemplify that such linearity exists even when using different RL control policies. Such results point out an interesting direction to understand the relationship between RL and optimal control: whether RL tends to linearize the nonlinear system during training. Furthermore, we illustrate that the found linear model is able to provide formal guarantees by safety-critical optimal control framework with an example of autonomous navigation using Cassie while taking advantage of the agility provided by the RL-based controller.

1.5 Chapter Summary

In this chapter, we presented the background of control barrier functions and point out some existing challenges in this field. The challenges can be found in three domains 1) optimization feasibility and safety, 2) obstacle avoidance and trajectory generation 3) applications on motion planning and navigation. After that, we presented the core contributions of the thesis together with the thesis outline. In the following chapters, we will go through all the mentioned challenges and present our solutions with respect to them.

Chapter 2

Background on Control Barrier Functions

In this brief chapter, we will describe existing work on using control barrier functions for continuous-time and discrete-time systems.

2.1 Continuous-time System & Safety

We firstly consider a nonlinear affine control system in this form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipchitz. The system is also subject to input constraints

$$\mathbf{u}(t) \in \mathcal{U}_{adm}(\mathbf{x}(t)),\ \forall t \geq 0, \tag{2.2}$$

where $\mathcal{U}_{adm}(\mathbf{x}(t)) \subset \mathbb{R}^m$ denotes the set of admissible inputs, which could be state-dependent.

Definition 2.1 (Safe Set and Safe State)**.** *We consider a set $\mathcal{C} \subset \mathbb{R}^n$ defined as the superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, yielding:*

$$\begin{aligned} \mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}, \\ \partial\mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) = 0\}, \\ Int(\mathcal{C}) &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) > 0\}. \end{aligned} \tag{2.3}$$

We call the set $\mathcal{C}$ as the safe set and h as the safety function.

This function $h(.)$ is usually denoted as a distance function between the robot and the obstacle, for example, the robot is currently in the safe set (staying outside the obstacle) shown in Figure 2.1.

A feedback controller could be denoted as $\pi : (\mathbf{x}, t) \to \mathbf{u} : \mathbf{u}(\mathbf{x}(t)) \in \mathcal{U}_{adm}$, the closed loop system under this controller could be expressed as follow:

$$\dot{\mathbf{x}} = f^{\pi}_{cl}(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\pi(\mathbf{x}, t). \tag{2.4}$$

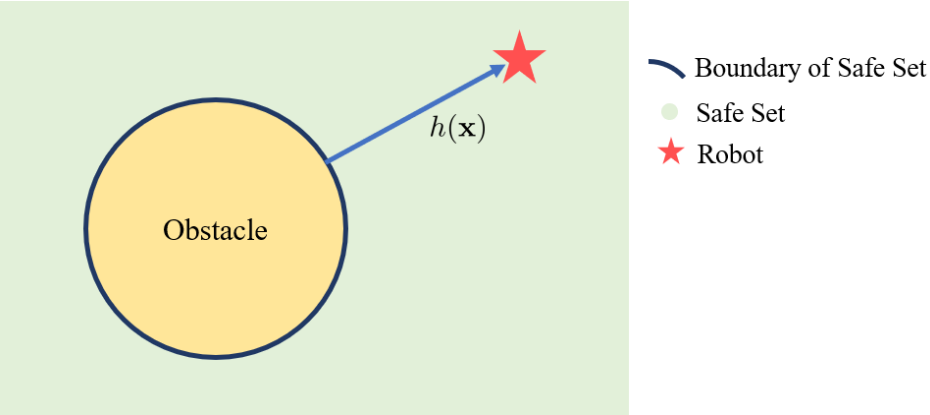


Figure 2.1: Safe Set and Distance function

Definition 2.2 (Forward Invariant)**.** *A set $\mathcal{C}^\pi$ is forward invariant with controller π if for every $\mathbf{x}_0 \in \mathcal{C}^\pi$, $\xi^\pi_{\mathbf{x}_0,t_0} \in \mathcal{C}^\pi$ for $\mathbf{x}(t_0) = \mathbf{x}_0$ and $t \in I(\mathbf{x}_0) = [t_0, t_0 + \tau)$, where $I(\mathbf{x}_0)$ represents a maximum interval of existence. When the system is forward complete, $\tau = \infty$. Note that $\xi^\pi_{\mathbf{x}_0,t_0}(t)$ denotes the solution of the system (2.4) at time t by starting at initial state $\mathbf{x}_0$ and initial time t_0, and applying controller $\pi(\mathbf{x}, t) \in \mathcal{U}_{adm}$ over $[t_0, t]$.*

The definition of forward invariance allows us to define the safety of the closed loop system under the controller π as follows,

Definition 2.3 (Safety)**.** *The system (2.4) is safe with respect to a set $\mathcal{C}$ if the set $\mathcal{C}$ is forward invariant.*

2.2 CLFs & CBFs for Continuous-time System

Here are definitions of class $\mathcal{K}$ and $\mathcal{K}_\infty$ functions which will be used for describing CLFs and CBFs in this section.

Definition 2.4 (Class $\mathcal{K}$ and $\mathcal{K}_\infty$ Functions)**.** *A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if 1) it is strictly increasing 2) it is satisfied $\alpha(0) = 0$.*

Moreover, this function is said to be long to class $\mathcal{K}_\infty$ if 1) it belongs to class $\mathcal{K}$ 2) it is satisfied $a = \infty$ 3) it is satisfied $\lim_{r\to\infty} \alpha(r) = \infty$.

Based on the definition of the safety function h and the nonlinear affine control system in (2.1), we define a control barrier function as follows,

Definition 2.5 (CBF)**.** *Let $\mathcal{C} \subset \mathcal{D} \subset \mathbb{R}^n$ be the superlevel set of a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$, then h is a control barrier function (CBF) if there exists an extended*

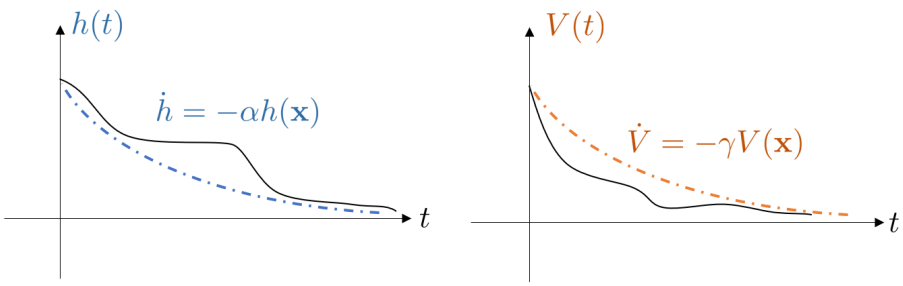


Figure 2.2: Intuitions of control Lyapunov and barrier functions.

class $\mathcal{K}_\infty$ function α such that for the control system (2.1)*:*

$$\sup_{u \in \mathcal{U}_{adm}} \dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x})), \tag{2.5}$$

for all $\mathbf{x} \in \mathcal{D}$.

A control input satisfying (2.5) ensures invariance of the set $\mathcal{C}$ and thus safety [9]. In practice, unless a CBF is handcrafted with consideration with set calculation or some other approximations, it's usually not a valid one. In other words, (2.5) is not valid for all $\mathbf{x} \in \mathcal{D}$ while only for a subset of $\mathcal{D}$ [226].

Definition 2.6 (Valid CBF)**.** *We denote the h represents a valid CBF when the set $\mathcal{C}$ is forward invariant under CBF constraints.*

Besides the system safety, we are also interested in stabilizing the system (2.1) with a feedback control law $\mathbf{u}$ to a given state. In a nonlinear context, this can be achieved by equivalently finding a feedback control law that drives a positive definite function V to zero, see [177]. Concretely, the control Lyapunov function is defined as follows,

Definition 2.7 (CLF)**.** *V is a control Lyapunov Function (CLF) if it is positive definite and satisfies:*

$$\inf_{\mathbf{u} \in \mathcal{U}_{adm}} \dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma(V(\mathbf{x})), \tag{2.6}$$

for all $\mathbf{x} \in \mathcal{D}$.

To understand the CLF and CBF constraints, if we regard the $\alpha(.)$ and $\gamma(.)$ functions as positive scalars instead of class $\mathcal{K}$, then the CBF constraint provides an exponential-decay lower bound which is always positive and the CLF constraint provides an exponential-decay upper bound which will converge to zero, illustrated in Figure 2.2. These two constraints enforce the system safety and stability together.

2.3 Continuous-time CBF-based Controllers

Having established control barrier functions and control Lyapunov functions, we will present how to synthesize controllers to guarantee safety and stability for them.

Suppose the control barrier function is of relative-degree one, where the first time-derivative of the CBF has to depend on the control input, i.e., $L_g h(\mathbf{x}) \neq 0$, where L_g represents the lie derivative with respect to g. Hence in this case, we have

$$\dot{h}(\mathbf{x}, \mathbf{u}) = L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}. \tag{2.7}$$

This allows us to define safety-critical optimal control for a nonlinear affine system (2.1). Suppose we are given a feedback controller $\mathbf{u} = k(\mathbf{x})$ for the control system (2.1) and we wish to guarantee safety. we can consider the following Quadratic Program (QP) based controller that finds the minimum perturbation on $\mathbf{u}$:

CBF-QP [12]:

$$\mathbf{u}(\mathbf{x}) = \underset{\mathbf{u}\in\mathbf{R}^m}{\operatorname{argmin}} \frac{1}{2}||\mathbf{u} - k(\mathbf{x})||^2 \tag{2.8a}$$

$$\text{s.t. } L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})), \tag{2.8b}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}). \tag{2.8c}$$

The above QP based formulation of safety-critical controllers suggests a means in which to unify safety and stability. Concretely, we consider the following QP based controller:

CLF-CBF-QP [12]:

$$\mathbf{u}(\mathbf{x}) = \underset{(\mathbf{u},\delta)\in\mathbf{R}^{m+1}}{\operatorname{argmin}} \frac{1}{2}\mathbf{u}^T H(\mathbf{x})\mathbf{u} + \phi(\delta) \tag{2.9a}$$

$$\text{s.t. } L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} \leq -\gamma(V(\mathbf{x})) + \delta, \tag{2.9b}$$

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})), \tag{2.9c}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}), \tag{2.9d}$$

---

where $H(\mathbf{x})$ is any positive definite matrix (pointwise in $\mathbf{x}$) and $\phi(\delta)$ is chosen to be quadratic and positive.

2.4 Generalization of CBFs for Discrete-time Systems

Similarly, we can describe stability and safety in the discrete-time system. In this chapter, we consider a discrete-time control system as follows,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{2.10}$$

with $\mathbf{x} \in \mathcal{X}$ representing the system state with the control input $\mathbf{u}$ confined by admissible input set $\mathcal{U}$.

Based on the safety set defined in (2.3), it's usually regarded as the ensemble of states satisfying the distance constraints

$$h(\mathbf{x}) \geq 0. \tag{2.11}$$

In a stricter manner, the function h becomes a control barrier function in the discrete-time domain if it satisfies the following relation,

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) \geq -\alpha_k h(\mathbf{x}_k),\ 0 < \alpha_k \leq 1, \tag{2.12}$$

where $\Delta h(\mathbf{x}_k, \mathbf{u}_k) := h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k)$. Satisfying constraint (2.12), we have $h(\mathbf{x}_{k+1}) \geq (1-\alpha_k)h(\mathbf{x}_k)$, i.e, the lower bound of the control barrier function $h(\mathbf{x})$ decreases exponentially at time k with the rate $1 - \alpha_k$.

Besides the system safety, we are also interested in stabilizing the system with a feedback control law $\mathbf{u}$ under a control Lyapunov function V in the discrete-time domain,

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) \leq -\gamma_k V(\mathbf{x}_k),\ 0 < \gamma_k \leq 1, \tag{2.13}$$

where $\Delta V(\mathbf{x}_k, \mathbf{u}_k) := V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k)$. Similarly as above, the upper bound of control Lyapunov function decreases exponentially at time k with the rate $1 - \gamma_k$.

The discrete-time control Lyapunov function and control barrier function can be unified into one optimization program, which achieves the control objective and guarantees system safety. This formulation was introduced in [2] and is presented as follows,

---

DCLF-DCBF:

$$\mathbf{u}_k^* = \underset{(\mathbf{u}_k, s) \in \mathbb{R}^{m+1}}{\operatorname{argmin}} \quad \mathbf{u}_k^T H(\mathbf{x}) \mathbf{u}_k + \phi(s) \tag{2.14a}$$

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) + \gamma_k V(\mathbf{x}_k) \leq s \tag{2.14b}$$

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k h(\mathbf{x}_k) \geq 0 \tag{2.14c}$$

$$\mathbf{u}_k \in \mathcal{U}, \tag{2.14d}$$

---

where $H(\mathbf{x})$ is any positive definite matrix and $s \geq 0$ is a slack variable together with an additional cost term $\phi(s) \geq 0$ that allows the Lyapunov function to grow when the CLF and CBF constraints are conflicting. The safe set $\mathcal{C}$ in (4.4) is invariant along the trajectories of the discrete-time system with controller (2.14) if $h(\mathbf{x}_0) \geq 0$ and $0 < \alpha_k \leq 1$.

2.5 Chapter Summary

In this chapter, we provided a brief introduction about mathematical background of control Lyapunov and barrier functions for both continuous-time and discrete-time systems. We will discuss the thesis research in following chapters.

Part I

Optimization Feasibility and Safety

Chapter 3

Guaranteed Point-wise Feasibility with Input Constraints [1]

3.1 Introduction

3.1.1 Motivation

Safety-critical optimal control and planning is one of the fundamental problems in robotics, *e.g.*, robots need to be able to safely avoid obstacles while using minimal energy. In order to ensure the safety of robotic systems while achieving optimal performance, the tight coupling between potentially conflicting control objectives and safety criteria is usually formulated as an optimization problem. Recent work [9, 11] formulates this problem using control barrier functions as an optimization problem, where the safety criteria is formulated as constraints. Additionally, a robotic system usually has a constrained set for the admissible inputs due to physical limitations, which can be taken as additional constraints in the optimization problem [137, 2, 41]. However, the input constraint and control barrier function constraint might be in conflict in the optimization and make the optimization infeasible. In this chapter, we study this feasibility problem and illustrate a form of safety-critical control with control barrier functions which satisfies the input constraint and guarantees point-wise feasibility along the trajectory.

3.1.2 Related Work

CBFs have recently been introduced as a promising way to ensure set invariance by considering the system dynamics. Furthermore, a safety-critical control design for continuous-time systems was proposed by unifying a control Lyapunov function (CLF) and a control barrier function (CBF) through a quadratic program (CLF-CBF-QP) [11, 12]. This method could be

[1] The material of this chapter is from "Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility" published in 2021 American Control Conference (ACC) [226].

deployed as a real-time optimization-based controller with safety-critical constraints, shown in [9, 209]. The adaptive, robust, and stochastic cases of safety-critical control with CBF have been considered in [138, 217, 185]. CBFs have also been used for high relative degree safety constraints for nonlinear systems [133, 214]. Besides the continuous-time domain, the formulation of CBF was generalized into discrete-time systems in [2], and systems evolving on manifolds in [209]. Recently, CBF constraints were also applied in control problems using a data-driven approach [40, 184, 156, 116, 157] and optimal control design [224, 44].

However, the input constraint was not considered in the optimization in some of the previous work, such as [12], which means the optimized control input might not be executable due to the physical limitations of the system. Other work, such as [209, 217, 138, 137, 2, 41], did consider the input constraint, but the potential conflict between input constraint and CBF constraint was not addressed, potentially resulting in an unsolvable optimization problem. In [11], a valid CBF is specifically designed for adaptive cruise control scenario with explicit integration problem over the system dynamics under input constraint, which could ensure the feasibility of the optimization. However, the feasibility problem for general nonlinear systems remains an unsolved topic for the safety-critical optimal control.

3.1.3 Contribution

The contributions of this chapter are as follows.

- We present the reasons of potential infeasibility in the CBF-QP and CLF-CBF-QP.
- With analysis in both input-space and state-space, we reveal quantitatively and qualitatively that the infeasibility appears potentially due to a small decay rate of the control barrier function constraint.
- We propose an optimal-decay form of CBF-QP and CLF-CBF-QP where the decay rate of the control barrier function is optimized.
- We prove that our optimal-decay formulation is point-wise feasible for any state lying strictly inside the safe set. An adaptive cruise control example is used to numerically verify this.

3.2 Background

We consider a nonlinear affine system of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) + g(\mathbf{x}(t))\mathbf{u}, \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, with $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ being locally Lipschitz. The system is subject to input constraints

$$\mathbf{u}(t) \in \mathcal{U}_{adm}(\mathbf{x}(t)),\ \forall t \geq 0, \tag{3.2}$$

where $\mathcal{U}_{adm}(\mathbf{x}(t)) \subset \mathbb{R}^m$ denotes the set of admissible inputs, which could be state dependent.

We consider a set $\mathcal{C} \subset \mathbb{R}^n$ defined as the zero-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, yielding:

$$\begin{aligned} \mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}, \\ \partial\mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) = 0\}, \\ \text{Int}(\mathcal{C}) &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) > 0\}. \end{aligned} \tag{3.3}$$

Throughout this chapter, we refer to $\mathcal{C}$ as a safe set.

The definitions of control barrier functions and control Lyapunov functions are summarized as follow, see [12] for detailed explanations. The function h becomes a control barrier function if $\dfrac{\partial h}{\partial \mathbf{x}} \neq 0$ for all $\mathbf{x} \in \partial\mathcal{C}$, and there exists an extended class $\mathcal{K}_\infty$ function α such that for the control system (3.1), h satisfies

$$\exists\ \mathbf{u} \text{ s.t. } \dot{h}(\mathbf{x}, \mathbf{u}) \geq -\alpha(h(\mathbf{x})),\ \alpha \in \mathcal{K}_\infty. \tag{3.4}$$

Besides the system safety, we are also interested in stabilizing the system with a feedback control law $\mathbf{u}$ under a control Lyapunov function V with a class $\mathcal{K}$ function γ, *i.e.*,

$$\exists\ \mathbf{u} \text{ s.t. } \dot{V}(\mathbf{x}, \mathbf{u}) \leq -\gamma(V(\mathbf{x})),\ \gamma \in \mathcal{K}. \tag{3.5}$$

Note that we can write down

$$\dot{h}(\mathbf{x}, \mathbf{u}) = L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}, \tag{3.6}$$

where $L_f h(\mathbf{x})$ and $L_g h(\mathbf{x})$ are Lie-derivatives of $h(\mathbf{x})$ along $f(\mathbf{x})$ and $g(\mathbf{x})$, respectively. We can also write down

$$\dot{V}(\mathbf{x}, \mathbf{u}) = L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u}, \tag{3.7}$$

where $L_f V(\mathbf{x})$ and $L_g V(\mathbf{x})$ are Lie-derivatives of $V(\mathbf{x})$. The above construction of the CLF and CBF allows us to define safety-critical control for a nonlinear affine system (3.1).

Given a feedback controller $\mathbf{u} = k(\mathbf{x})$ for the control system (3.1), we wish to guarantee safety. We consider the following Quadratic Program (QP) based controller that finds the optimal $\mathbf{u}$ in the optimization as follows:

CBF-QP:

$$\mathbf{u}(\mathbf{x}) = \underset{\mathbf{u} \in \mathbf{R}^m}{\text{argmin}} \frac{1}{2}||\mathbf{u} - k(\mathbf{x})||^2 \tag{3.8a}$$

$$\text{s.t. } L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})), \tag{3.8b}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}). \tag{3.8c}$$

When the input constraint (3.8c) is excluded, we have a single inequality constraint, thus the CBF-QP has a closed-form solution per the KKT conditions, and this method was used in [58, 13]. However, when the input constraint is considered, there might not exist any $\mathbf{u}$ satisfying both input constraint and CBF constraint simultaneously. This could lead to a potential infeasible optimization problem.

We could also use a QP based formulation of safety-critical control which unifies safety and stability. Concretely, we consider the following QP based controller:

CLF-CBF-QP:

$$\mathbf{u}(\mathbf{x}) = \underset{(\mathbf{u},\delta)\in\mathbf{R}^{m+1}}{\operatorname{argmin}} \frac{1}{2}\mathbf{u}^T H(\mathbf{x})\mathbf{u} + p\delta^2 \tag{3.9a}$$

$$\text{s.t. } L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} \leq -\gamma(V(\mathbf{x})) + \delta, \tag{3.9b}$$

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})), \tag{3.9c}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}). \tag{3.9d}$$

---

where $H(\mathbf{x})$ is any positive definite matrix (point-wise in $\mathbf{x}$), and we have a relaxation variable δ on the CLF constraint (3.9b) with additional quadratic cost in (3.9a). When we exclude the input constraint (3.9d) out of the optimization, the solvability can be guaranteed since the CLF constraint is relaxed and the CBF constraint (3.9c) is the only hard constraint. This method was applied in [10, 61]. However, when the input constraint (3.9d) is also considered, we might again encounter an infeasible optimization problem.

3.3 Point-wise Feasibility in Input-Space

Having presented the background of safety-critical control, we will now show how to pick an appropriate α in the CLF-CBF-QP/CBF-QP to guarantee point-wise feasibility from the perspective of input-space, *i.e.*, how to guarantee feasibility of the optimization problem at a given state $\mathbf{x}(t) = \mathbf{x}_t$.

For the state $\mathbf{x}_t$ at time t, we define the feasible superlevel set $\mathcal{U}_{cbf}(\mathbf{x}_t)$ as the region satisfying the CBF constraint in input-space, *i.e.*,

$$\mathcal{U}_{cbf}(\mathbf{x}_t) := \{\mathbf{u} \in \mathbb{R}^m : L_f h(\mathbf{x}_t) + L_g h(\mathbf{x}_t)\mathbf{u} \geq -\alpha(h(\mathbf{x}_t))\}. \tag{3.10}$$

Note that $\mathcal{U}_{cbf}(\mathbf{x}_t)$ is a half-space in the input-space $\mathbb{R}^m$ since the CBF constraint is affine in $\mathbf{u}$. The level set of the CBF constraint in input-space is defined as $\partial\mathcal{U}_{cbf}(\mathbf{x}_t)$,

$$\partial\mathcal{U}_{cbf}(\mathbf{x}_t) = \{\mathbf{u} \in \mathbb{R}^m : L_f h(\mathbf{x}_t) + L_g h(\mathbf{x}_t)\mathbf{u} = -\alpha(h(\mathbf{x}_t))\}. \tag{3.11}$$

Then the feasibility problem becomes whether the intersection between $\mathcal{U}_{cbf}(\mathbf{x}_t)$ and $\mathcal{U}_{adm}(\mathbf{x}_t)$ as defined in (3.2) is empty or not. If the intersection is not empty, the optimization in CBF-QP or CLF-CBF-QP is feasible at $\mathbf{x}_t$.

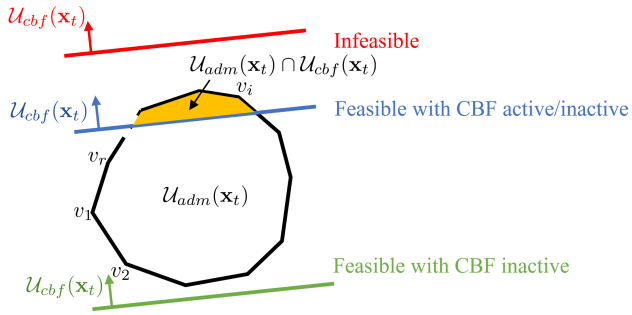


Figure 3.1: Feasibility analysis from intersection between CBF constraints and Input Constraints.

In order to provide a quantitative way of explaining point-wise feasibility from the perspective of input-space, we suppose that the set of admissible inputs $\mathcal{U}_{adm}(\mathbf{x}_t)$ could be described as a convex polytope defined with r vertices, presented in Figure 3.1, where each vertex is noted as $v_i(\mathbf{x}_t) \in \mathbb{R}^m$ and $i \in \{1, 2, ..., r(\mathbf{x}_t)\}$.

Then we have $\mathcal{U}_{adm}(\mathbf{x}_t)$ could be written as

$$\mathcal{U}_{adm}(\mathbf{x}_t) = \{\mathbf{u} \in \mathbb{R}^m : \mathbf{u} = \sum_{i=1}^{r(\mathbf{x}_t)} \lambda_i(\mathbf{x}_t) v_i(\mathbf{x}_t), \sum_{i=1}^{r(\mathbf{x}_t)} \lambda_i(\mathbf{x}_t) = 1, \lambda_i(\mathbf{x}_t) \geq 0\}. \tag{3.12}$$

Then the necessary and sufficient condition of having the intersection between $\mathcal{U}_{cbf}(\mathbf{x}_t)$ defined in (3.10) and $\mathcal{U}_{adm}(\mathbf{x}_t)$ defined in (3.12) being not empty is that, at least one vertex of polytope $\mathcal{U}_{adm}(\mathbf{x}_t)$ lies inside or on the surface of the set $\mathcal{U}_{cbf}(x)$, shown as follows,

$$\mathcal{U}_{adm}(\mathbf{x}_t) \cap \mathcal{U}_{cbf}(\mathbf{x}_t) \neq \emptyset \Leftrightarrow \exists i, v_i \in \mathcal{U}_{cbf}(\mathbf{x}_t). \tag{3.13}$$

Hence, the set of candidate α functions in the CBF constraint (3.4), denoted as $\mathcal{K}^{\alpha}_{fea}(\mathbf{x}_t)$, that guarantees the feasibility of the optimization, becomes the union of the set of functions that guarantees any vertex in $\mathcal{U}_{adm}(\mathbf{x}_t)$ lies inside or on the surface of the set $\mathcal{U}_{cbf}(\mathbf{x}_t)$, which could be written as follows

$$\mathcal{K}^{\alpha}_{fea}(\mathbf{x}_t) = \bigcup_{i=1}^{r(\mathbf{x}_t)} \{\alpha \in \mathcal{K}_\infty : \alpha(h(\mathbf{x}_t)) \geq -L_f h(\mathbf{x}_t) - L_g h(\mathbf{x}_t) v_i(\mathbf{x}_t)\},$$

and it could be reformulated with a minimum operator

$$\mathcal{K}^{\alpha}_{fea}(\mathbf{x}_t) = \{\alpha \in \mathcal{K}_\infty : \alpha(h(\mathbf{x}_t)) \geq \min_i(-L_f h(\mathbf{x}_t) - L_g h(\mathbf{x}_t) v_i(\mathbf{x}_t))\}. \tag{3.14}$$

Therefore, given a function α, the CBF-QP/CLF-CBF-QP in (3.8) and (3.9) are point-wise feasible if $\exists \alpha \in \mathcal{K}_\infty$ s.t.,

$$\alpha(h(\mathbf{x}_t)) \geq \min_i(-L_f h(\mathbf{x}_t) - L_g h(\mathbf{x}_t) v_i(\mathbf{x}_t))$$

is satisfied at state $\mathbf{x}_t$.

Moreover, given a state $\mathbf{x}_t$, from (3.14), we can see when

$$\min_i(-L_f h(\mathbf{x}_t) - L_g h(\mathbf{x}_t) v_i(\mathbf{x}_t)) \leq 0. \tag{3.15}$$

We have $\mathcal{K}^{\alpha}_{fea}(\mathbf{x}_t) = \mathcal{K}_\infty$, *i.e.*, we have the point-wise feasibility, for any $\alpha \in \mathcal{K}_\infty$. However, when (3.15) is not satisfied, $\mathcal{K}^{\alpha}_{fea}(\mathbf{x}_t)$ becomes a proper subset of $\mathcal{K}_\infty$ and the function α has a lower bound at state $\mathbf{x}_t$ to guarantee the point-wise feasibility.

From above, we could pick an appropriate α function to guarantee point-wise feasibility at a given state $\mathbf{x}_t$. Beside feasibility, we are also interested in whether the CBF constraint is activated during the optimization.

Remark 3.1. *We have* $\forall i, v_i(\mathbf{x}_t) \in \mathcal{U}_{cbf}(\mathbf{x}_t)$ *when*

$$\alpha(h(\mathbf{x}_t)) \geq \max_i(-L_f h(\mathbf{x}_t) - L_g h(\mathbf{x}_t) v_i(\mathbf{x}_t)), \tag{3.16}$$

which means $\mathcal{U}_{adm}(\mathbf{x}_t) \subset \mathcal{U}_{cbf}(\mathbf{x}_t)$*, therefore in this case, the CBF constraint does not confine the input constraint during the optimization. This case is illustrated by the green* $\mathcal{U}_{cbf}(\mathbf{x}_t)$ *in Figure 3.1. When* (3.16) *is not satisfied, we have the CBF constraint confine the input constraint, i.e., the intersection between* $\mathcal{U}_{adm}(\mathbf{x}_t)$ *and* $\mathcal{U}_{cbf}(\mathbf{x}_t)$ *becomes a proper set of* $\mathcal{U}_{adm}(\mathbf{x}_t)$*, indicated by the blue* $\mathcal{U}_{cbf}(\mathbf{x}_t)$ *in Figure 3.1. When the intersection is empty, illustrated by the red* $\mathcal{U}_{cbf}(\mathbf{x}_t)$ *in Figure 3.1, the optimization problem is infeasible.*

Remark 3.2. *We say a constraint is active in the optimization when the optimal solution lies on the constraint line [27]. When the CBF constraint does not confine the input constraint, the CBF constraint becomes inactive during the optimization. However, when the CBF constraint confines the input constraint, it does not necessarily guarantee the CBF constraint activation, as the value of optimal solution also depends on the design of cost function in the optimization. Therefore, the choice of* α *function together with the design of cost function determine the activation of CBF constraint in the safety-critical optimal control, shown in Figure 3.1.*

Remark 3.3. *In this section, the admissible set is assumed as a bounded convex polytope. In fact, these discussions could be easily generalized for an unbounded convex set, as any unbounded set could be regarded as a limit of a sequence of bounded sets [19, Chap. 11]. Moreover, our proposed optimal-decay approach in this chapter that guarantees point-wise feasibility will only rely on the admissible set being convex while not strictly assuming it as a bounded convex polytope, see Theorem 3.1.*

We have seen point-wise feasibility in input-space in this section. Next, we will look at point-wise feasibility in the state-space in Sec. 3.4. After that, we provide a formulation which allows us to guarantee the point-wise feasibility without parameter tuning in Sec. 3.5.

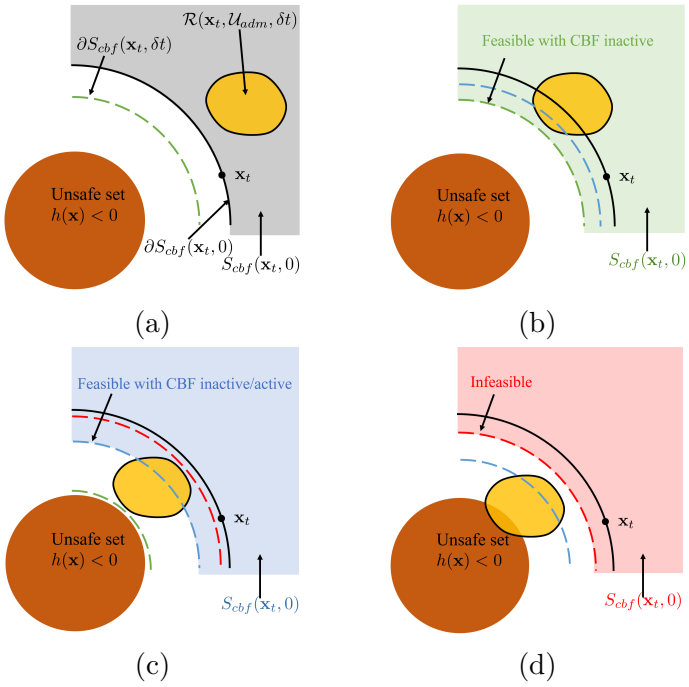


Figure 3.2: Point-wise feasibility problem in different scenarios.

3.4 Point-wise Feasibility in State-Space

The point-wise feasibility problem could also be understood through qualitative illustration in the state-space. Given a state $\mathbf{x}_t$ at time t, we will define $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ to represent the set of reachable states after infinitesimal time δt while satisfying system dynamics (3.1) and input constraint $\mathcal{U}_{adm}(\mathbf{x}_t)$ starting from state $\mathbf{x}(t) = \mathbf{x}_t$, *i.e.*,

$$\begin{aligned}\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) = \{\mathbf{x}(t+\delta t) \in \mathbb{R}^n : \forall \bar{t} \in [t, t+\delta], \dot{\mathbf{x}}(\bar{t}) = f(\mathbf{x}(\bar{t})) + g(\mathbf{x}(\bar{t}))\mathbf{u}(\bar{t}), \\ \mathbf{u}(\bar{t}) \in \mathcal{U}_{adm}(\mathbf{x}(\bar{t})), \mathbf{x}(t) = \mathbf{x}_t\}.\end{aligned} \tag{3.17}$$

The evolution of the system dynamics also needs to be safe and thus satisfy the definition of control barrier function (3.4) during time segment $[t, t+\delta t]$, thus we have

$$h(\mathbf{x}(t+\delta t)) \geq h(\mathbf{x}(t)) - \int_t^{t+\delta t} \alpha(h(\mathbf{x}(\bar{t})))d\bar{t}. \tag{3.18}$$

This allows us to define the superlevel set in state-space for $\mathbf{x}(t+\delta t)$ satisfying the CBF constraints,

$$\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t) = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq h(\mathbf{x}_t) - \int_t^{t+\delta t} \alpha(h(\mathbf{x}(\bar{t})))d\bar{t}, \mathbf{x}(t) = \mathbf{x}_t\}. \tag{3.19}$$

We also define

$$\mathcal{S}_{cbf}(\mathbf{x}_t, 0) = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq h(\mathbf{x}_t)\}, \tag{3.20}$$

motivated by $\int_t^{t+\delta t} \alpha(h(\mathbf{x}(\bar{t})))d\bar{t} = 0$ when $\delta t = 0$. The set $\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ corresponds to the set of all possible $\mathbf{x}$ for which $h(\mathbf{x}) \geq h(\mathbf{x}_t)$.

Since the state $\mathbf{x}(t + \delta t)$ should satisfy the system dynamics and control barrier function constraint, the optimization problem is then point-wise feasible at state $\mathbf{x}_t$ when the intersection between $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ and $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ is not empty. We are interested in whether the intersection between $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ and $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ is empty or not under different circumstances. However, it is numerically complicated or even impossible to calculate $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ for a general nonlinear affine system. Thus, we provide an intuition of understanding point-wise feasibility problem through geometry in state-space.

Remark 3.4. *Notice that we always have $\mathcal{S}_{cbf}(\mathbf{x}_t, 0) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ as α is class $\mathcal{K}_\infty$ function and $h(.)$ is positive.*

In practice, we usually define the safety set $\mathcal{C}$ in (3.3) corresponding to the free space outside the obstacle, illustrated in Figure 3.2. $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ and $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ are illustrated with black solid and colorful dashed curves respectively, and $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$, $\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ are illustrated as the regions on the top-right side of them. Since we always have $\mathcal{S}_{cbf}(\mathbf{x}_t, 0) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$, $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ is always closer to the obstacle, lying on the bottom-left side of $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ for any choice of α function. We classify the point-wise feasibility problem into three scenarios as follows.

Moving away from obstacles: when $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, 0)$, *i.e.*, the system is moving away from obstacles. The scenario is illustrated in Figure 3.2a. In this scenario, we have $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, 0) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$. This means that for any class $\mathcal{K}_\infty$ function α, the optimization will always be point-wise feasible at state $\mathbf{x}_t$.

Moving around obstacles: when $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ intersects with $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$, we have the reachable state-space lies partly on the top-right side of $\partial\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$, shown in Figure 3.2b. In this scenario, the optimization is always feasible for any class $\mathcal{K}_\infty$ function α, as $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \cap \mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ is not empty and is always a subset of $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$.

Moving close to obstacles: when $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \cap \mathcal{S}_{cbf}(\mathbf{x}_t, 0) = \emptyset$, this usually happens when the system is moving close to obstacles, shown in Figure 3.2c and 3.2d. In this scenario, when α becomes too small (CBF level set in red), the intersection between $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ and $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ is empty and the optimization problem becomes infeasible at state $\mathbf{x}_t$. This indicates that $\alpha(h(\mathbf{x}_t))$ needs to be greater than a lower bound to make the optimization point-wise feasible at state $\mathbf{x}_t$.

Remark 3.5. *When $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \subset \mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$, the CBF constraint does not confine the reachable set which means the CBF constraint is inactive in the optimization, shown with green level sets $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ in Figure 3.2. When the intersection between $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ and $\mathcal{S}_{cbf}(\mathbf{x}_t, \delta t)$ is non-empty and becomes as a proper subset of $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t)$ shown with blue level sets $\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ in Figure 3.2, the CBF constraint does confine the reachable set. This*

does not necessarily guarantee CBF constraint activation, as the constraint activation also depends on the design of cost function, which is similar to what we discussed in Remark 3.2. For the red level sets $\mathcal{S}_{cbf}(\mathbf{x}_t, 0)$ in Figure 3.2c and 3.2d, the optimization is infeasible.

3.5 Point-wise Feasible Formulation

3.5.1 Formulation and Point-wise Feasibility

In Sec. 3.3 and 3.4, we have seen that the optimization in safety-critical control might become point-wise infeasible at a given state $\mathbf{x}_t$, if either $\mathcal{U}_{cbf}(\mathbf{x}_t) \cap \mathcal{U}_{adm}(\mathbf{x}_t) = \emptyset$ or $\mathcal{R}(\mathbf{x}_t, \mathcal{U}_{adm}, \delta t) \cap \mathcal{S}_{cbf}(\mathbf{x}_t, \delta t) = \emptyset$ which means the convergence rate of control barrier function is less than a lower bound. In other words, when the decay rate of the lower bound of $h(\mathbf{x})$ is not large enough, we might encounter infeasibility in the optimization problem. To solve this problem, we could manually tune the form of α function to make the optimization feasible, however, this tuning process becomes relatively difficult when the system dynamics becomes complicated. This motivates us to introduce an optimal-decay form of CBF constraint to guarantee the point-wise feasibility for any $\mathbf{x}$ with $h(\mathbf{x}) > 0$.

With the same notation as CBF-QP in (3.8) and CLF-CBF-QP in (3.9), we introduce an optimal-decay form of CBF-QP and CLF-CBF-QP in this section. The optimal decay CBF-QP is formulated as follows

---

Optimal-decay CBF-QP:

$$\mathbf{u}(\mathbf{x}) = \underset{(\mathbf{u},\omega)\in\mathbf{R}^{m+1}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{u} - k(\mathbf{x})||^2 + p_\omega(\omega - \omega_0)^2 \tag{3.21a}$$

$$\text{s.t. } L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\omega\alpha(h(\mathbf{x})), \tag{3.21b}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}). \tag{3.21c}$$

Compared with CBF-QP, we optimize the decay rate of the CBF constraint with a new variable ω in (3.21b) and add a quadratic cost in (3.21a). p_ω is a positive scalar and a scalar ω_0 could usually be chosen to tune the performance of the controller. Similarly, we develop the optimal form of CLF-CBF-QP as follows

---

Optimal-decay CLF-CBF-QP:

$$\mathbf{u}(\mathbf{x}) = \underset{(\mathbf{u},\delta,\omega)\in\mathbf{R}^{m+2}}{\operatorname{argmin}} \frac{1}{2}\mathbf{u}^T H(\mathbf{x})\mathbf{u} + p\delta^2 + p_\omega(\omega - \omega_0)^2 \tag{3.22a}$$

$$\text{s.t. } L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} \leq -\gamma(V(\mathbf{x})) + \delta, \tag{3.22b}$$

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\omega\alpha(h(\mathbf{x})), \tag{3.22c}$$

$$\mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}). \tag{3.22d}$$

We call (3.21) and (3.22) as optimal-decay form of CBF-QP and CLF-CBF-QP, since it actually optimizes the decay rate of lower bound of the control barrier function with variable ω in the optimization. The point-wise feasibility of optimization for any $\mathbf{x} \in \text{Int}(\mathcal{C})$ in (3.21), (3.22) is illustrated with the following theorem.

Theorem 3.1. *When $\mathcal{U}_{adm}(\mathbf{x})$ is convex, the optimizations in optimal-decay CBF-QP and CLF-CBF-QP are point-wise feasible for any $\mathbf{x}$ lying inside $\mathcal{C}$, i.e., the optimizations are solvable with a unique solution for any $\mathbf{x}$ when $h(\mathbf{x}) > 0$.*

Proof. For optimal-decay CBF-QP, since $\omega \in \mathbb{R}$ is a variable to optimize and $h(\mathbf{x}) > 0$, $-\omega\alpha(h(\mathbf{x}))$ could vary between $-\infty$ and $+\infty$ for any $\mathbf{x}$. Then, for any $\mathbf{x}$, there always exists a value ω such that the feasible region between (3.21b) and (3.21c) is non-empty.

Moreover, we notice that the CBF constraints and input constraints are convex with respect to optimization variables $\mathbf{u}$ and ω. Hence, the feasible region in the optimal-decay CBF-QP is convex and not empty. Additionally, our cost function (3.21a) is a quadratic form and positive-definite, which is strictly convex. Therefore, the minimization in (3.21) is convex (convex cost and constraints) with non-empty feasible region, which is solvable and holds a unique solution [27, Chap. 5].

Compared to optimal-decay CBF-QP, the optimal-decay CLF-CBF-QP is nothing different except there is a CLF constraint in (3.22b), which is also convex with respect to optimization variables and is optimized with the relaxation variable δ. Therefore, the optimization in optimal CLF-CBF-QP is also convex with a non-empty feasible region, which is feasible with an unique solution. □

In fact, we could generalize the point-wise feasibility in Theorem 3.1 for any $\mathbf{x}$ when $h(\mathbf{x}) \neq 0$, as $\omega\alpha(h(\mathbf{x}))$ could vary between $-\infty$ and ∞ when $h(\mathbf{x}) \neq 0$ with $\omega \in \mathbb{R}$. However, this is not interesting since the system is already unsafe when $h(\mathbf{x}) < 0$.

Remark 3.6. *The optimization variable ω is not required to be positive. In fact, when the optimized value of ω is negative, this implies that the control barrier function is increasing with the optimized control input, correpsonding to the safety with set invariance. In the scenario of ω as negative, the system is keeping further away from the obstacles.*

3.5.2 Convergence of CBF constraint

The variable ω optimizes the convergence rate of the CBF constraint, with $\omega\alpha(.)$ still being a class $\mathcal{K}_\infty$ function. Therefore, with optimal-decay CBF-QP and CLF-CBF-QP, we equivalently use a state-dependent rate for the convergence of the CBF constraint, since the optimized value of ω will be calculated differently at different state $\mathbf{x}$.

Moreover, smaller ω_0 and larger p_ω would make the control barrier function decay slower. This makes sense from a mathematical perspective, since the larger ω_0 optimizes the nominal CBF constraint with larger decay rate. Similarly, for smaller p_ω, we will have optimized value

of ω deviating more from the value ω_0, which also brings a larger decay rate on the control barrier function.

3.5.3 Persistent Feasibility and Safety

Note that point-wise feasibility for all $\mathbf{x}$ with $h(\mathbf{x}) > 0$ does not guarantee that our system is persistently feasible, *i.e.*, the system might become unsafe even if the system is initialized at $\mathbf{x}(0) = \mathbf{x}_0$ with $h(\mathbf{x}_0) > 0$. This is due to the fact that when we reach $h(\mathbf{x}) = 0$, we can't change ω to make the optimization problem feasible while simultaneously enforcing the optimal-decay CBF constraint (3.22c) and the input constraint. This makes the control barrier function invalid with respect to the safe set defined in (3.3) due to the input constraint. In this case, the CBF-based control policy isn't safe, i.e., it is not forward complete, see [12, Definition 1]. However, given any invariant set that is a subset of the safe set $\mathcal{C}$, our optimal-decay CBF-QP/CLF-CBF-QP will be persistently feasible if the initial state lies inside this safe invariant set.

The maximum invariant set for an optimal-decay controller could be different with respect to hyperparameters, such as p_ω and ω_0, which will be illustrated in Sec. 3.6.4. Notice that solving an explicit representation of this control invariant set for a nonlinear system is usually intractable [151, 15, 164] and this exceeds the scope of this chapter. We will discuss the invariant safety and related performance in future work.

Remark 3.7. *When p_ω tends to $+\infty$ in* (3.21) *and* (3.22)*, then given a state $\mathbf{x}$, if the original CBF constraint is satisfied, i.e.,*

$$\exists \mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}),\ L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})),$$

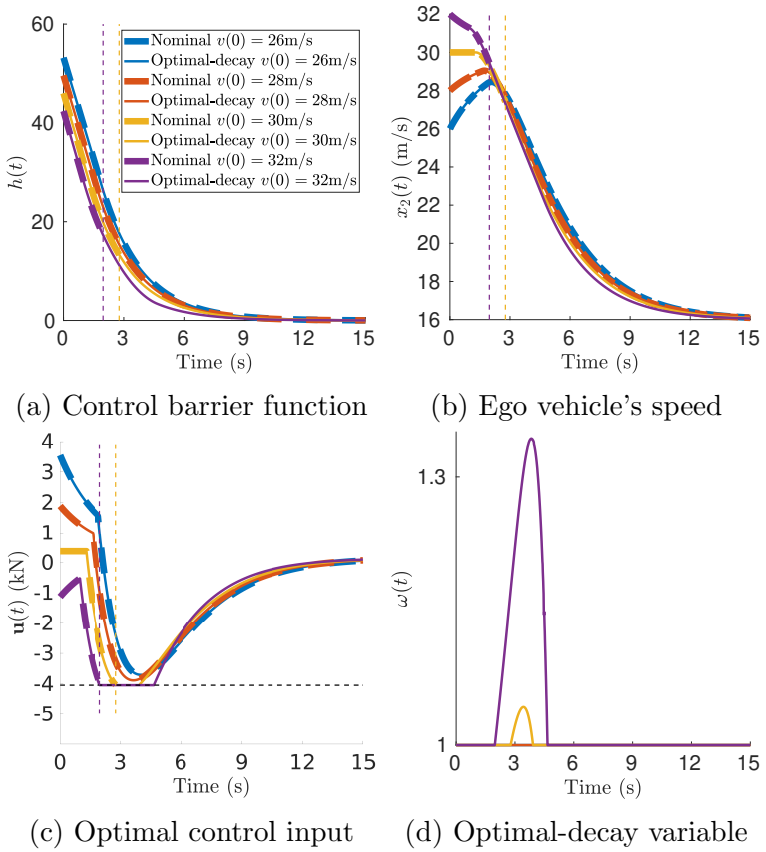
we will have the optimized value of ω as $\omega^*(\mathbf{x}) = \omega_0$. Specifically, when $\omega_0 = 1$, it will make the optimal-decay CBF-QP and CLF-CBF-QP as the original CBF-QP/CLF-CBF-QP. However, when the original CBF-QP and CLF-CBF-QP become infeasible, i.e.,

$$\forall \mathbf{u} \in \mathcal{U}_{adm}(\mathbf{x}),\ L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} < -\alpha(h(\mathbf{x})),$$

our optimal-decay form is still feasible and the optimal value of ω is as follows

$$\omega^*(\mathbf{x}) = \sup_{\mathbf{u} \in \mathcal{U}_{adm}} \frac{L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}}{-\alpha(h(\mathbf{x}))}.$$

To sum up, when $p_\omega = +\infty$ and $\omega_0 = 1$, the optimal-decay CBF-QP and optimal-decay CLF-CBF-QP are equivalent to original CBF-QP and CLF-CBF-QP, if the original ones are already feasible. When the original constraint is infeasible, the optimization variable ω makes the proposed one still feasible.



(a) Control barrier function (b) Ego vehicle's speed

(c) Optimal control input (d) Optimal-decay variable

Figure 3.3: Point-wise feasibility of adaptive cruise control under input constraints.

3.6 Case Study: Adaptive Cruise Control

Having presented our optimal-decay CBF-QP/CLF-CBF-QP formulation, we proceed to validate the proposed strategy using an adaptive cruise control (ACC) example, which has been commonly used to validate safety-critical control strategies [9, 214].

3.6.1 Simulation Setup

Consider a point-mass model of an ego vehicle moving along a straight line to follow a lead vehicle. The dynamics are given as follows

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\frac{1}{m} F_r(\mathbf{x}) \\ v_l - x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \\ 0 \end{bmatrix} \mathbf{u}, \tag{3.23}$$

where (x_1, x_2) are the position and velocity ($x_2 = \dot{x}_1$) of the ego vehicle, m is the mass of the vehicle, and x_3 is the distance between the vehicle and the lead vehicle traveling at a

velocity of v_l. F_r represents the aerodynamic drag and we use the empirical form given as below

$$F_r = f_0 + f_1 x_2 + f_2 x_2^2, \tag{3.24}$$

where f_0, f_1, and f_2 are empirical constants.

A CLF-CBF-QP controller is designed to solve this ACC problem. In order to regulate speed, we pick a control Lyapunov function as follows

$$V = (x_2 - v_d)^2. \tag{3.25}$$

A control barrier function is used to guarantee that the ego vehicle will not collide with the leading vehicle. We consider a control barrier function as

$$h = x_3 - 1.8x_2, \tag{3.26}$$

where the factor of 1.8 is a result of converting to SI units. Besides speed regulation and safe distance maintenance, we also consider input constraints as follows

$$c_d mg \leq \mathbf{u} \leq c_a mg, \tag{3.27}$$

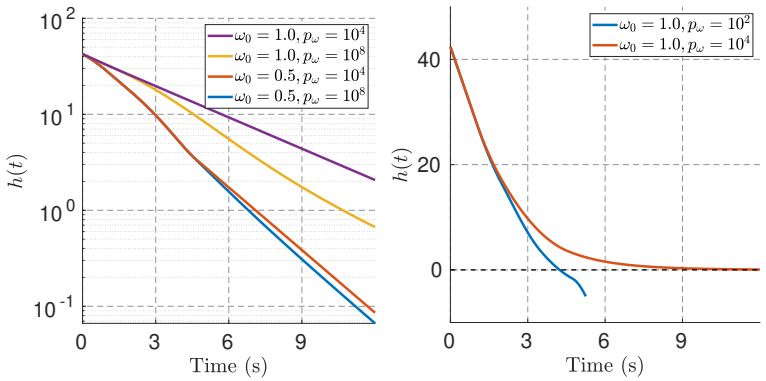
where c_a and c_d are coefficients for maximum and minimum wheel forces.

Using the control Lyapunov function (3.25) and the control barrier function (3.26), we design an original CLF-CBF-QP controller and an optimal-decay CLF-CBF-QP controller with formulation in (3.9) and (3.22), respectively. The system is simulated with the two controllers with a frequency at 100Hz, and the values of parameters in the simulations are shown in Table 3.1. Notice that in the simulations, we consider $\alpha(.)$ and $\gamma(.)$ in (3.9) and (3.22) as linear functions with constant coefficients α and γ.

3.6.2 Point-wise Feasibility

To compare the performance between our optimal-decay CLF-CBF-QP and the original form, we simulate the system with different initial conditions and illustrate them together in Figure 3.3, where solid and dashed lines represents the results from our optimal-decay CLF-CBF-QP and the original form, respectively. For our optimal-decay form, we firstly pick the hyperparameters ω_0 and p_ω as 1 and 10^8, which corresponds to the choice we discussed in Sec. 3.5.2.

Precisely, the ego vehicle is initialized at origin $x_1(0) = 0$ m, the initial distance between the ego car and the leading one is set as $x_3(0) = 100$ m and initial speed $x_2(0)$ ranges from 26 m/s to 32 m/s, shown in Figure 3.3b. We can see that CLF-CBF-QP with initial speed as 30 m/s and 32 m/s will become infeasible during the simulations. This infeasibility comes from the conflict between input constraint and original CBF constraint and we can see clearly that the CLF-CBF-QP fails with input saturation and our optimal-decay form handles the problem properly, shown in Figure 3.3c. The simulation with initial speed as 26 m/s and 28 m/s does not encounter any infeasible issues where the CLF-CBF-QP and our optimal-decay



(a) Different decay rates (b) Different safety performances

Figure 3.4: Influence of hyperparameters and CBF decay rates on system safety performance.

form shared the same control performance. Moreover, we notice ω becomes not equal to one only when there is a conflict between the input constraint and the original CBF constraint, shown in Figure 3.3d. The evolution of control barrier function for two controllers are shown in Figure 3.3a. To sum up, the optimal-decay CLF-CBF-QP is equivalent to the original one when $\omega_0 = 1$ if the original one is already feasible, while still being as a feasible problem even when the original CLF-CBF-QP is infeasible.

3.6.3 Convergence rate with hyperparameters

As we have stated in Sec. 3.5.3, the system might become unsafe if it is initialized outside its invariant set. To illustrate this property, we simulate the controller from an initial speed $x_2(0) = 32$ m/s and leading distance $x_3(0) = 100$ m with different values of hyperparameters ω_0 and p_ω. We keep the same Lyapunov function and control barrier function with the same parameters in Table 3.1 and the simulation results are shown in Figure 3.4. In Figure 3.4a, we can observe that smaller ω_0 and larger p_ω would make the control barrier function decay slower, which verified exactly what was stated in Sec. 3.5.2.

3.6.4 Safety with hyperparameters

The simulations in Sec. 3.6.1 and 3.6.3 that we have seen are all safe with set invariance on $\mathcal{C}$ defined in (3.3). However, for the same initial condition $x_2(0) = 32$ m/s, if we decrease p_ω, the system could be unsafe after a while. Specifically, we show that when $\omega_0 = 1, p_\omega = 10^2$, our system actually becomes unsafe after around 4s. This happens since the initial condition no longer lies inside the invariant set for the optimal-decay CLF-CBF-QP with this configuration of hyperparameters, which was previously discussed in Sec. 3.5.3. It must be noted that while a formulation guarantees point-wise feasibility for $h(\mathbf{x}) > 0$, this does not guarantee

System Parameters					
m	1650 kg	v_l	16 m/s	v_d	30 m/s
f_0	0.1 N	f_1	5 Ns/m	f_2	0.25 Ns^2/m^2
c_d	-0.25	c_a	0.25	g	9.81 m/s^2
Controller Parameters					
γ	1.0	p	1.0	α	0.5

Table 3.1: Simulation parameter values for adaptive cruise control.

persistent feasibility for $h(\mathbf{x}) > 0$ in a sufficiently long-time trajectory. This is seen in Figure 3.4b, where point-wise feasibility for all states with $h(\mathbf{x}) > 0$ still leads to the system unsafe after a while.

3.7 Chapter Summary

In this chapter, we have analyzed the point-wise feasibility problem of safety critical control from the perspective of input-space and state-space. Then, in order to deal with the conflict between CBF constraint and input constraint, we proposed an optimal-decay form for CBF-QP and CLF-CBF-QP to guarantee point-wise feasibility inside the safe set. We numerically verified this control design in an ACC example.

In next chapter, we will see that the feasibility problem not only exists for control barrier functions in the continuous-time domain but also in the discrete-time domain, especially in the formulation about model predictive control with control barrier functions.

Chapter 4

Model Predictive Control with Control Barrier Functions [2]

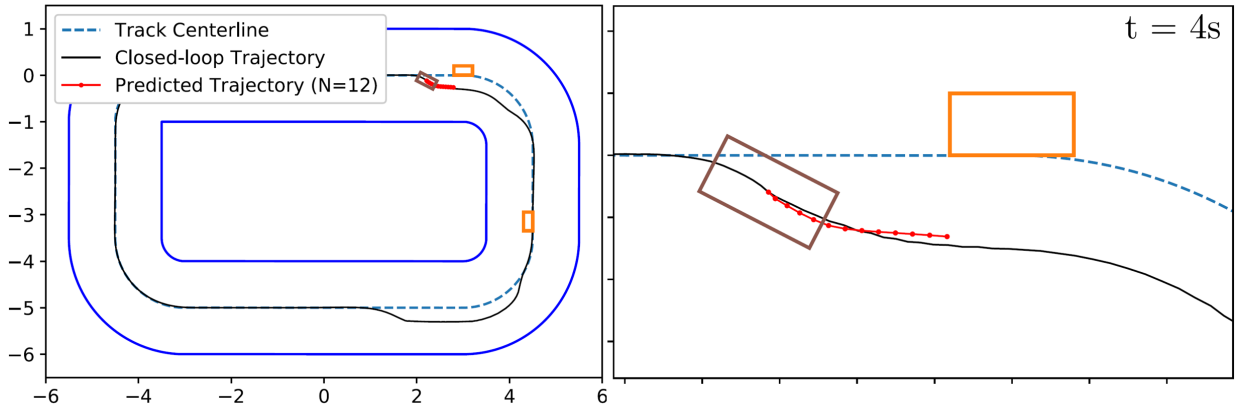
4.1 Introduction

4.1.1 Motivation

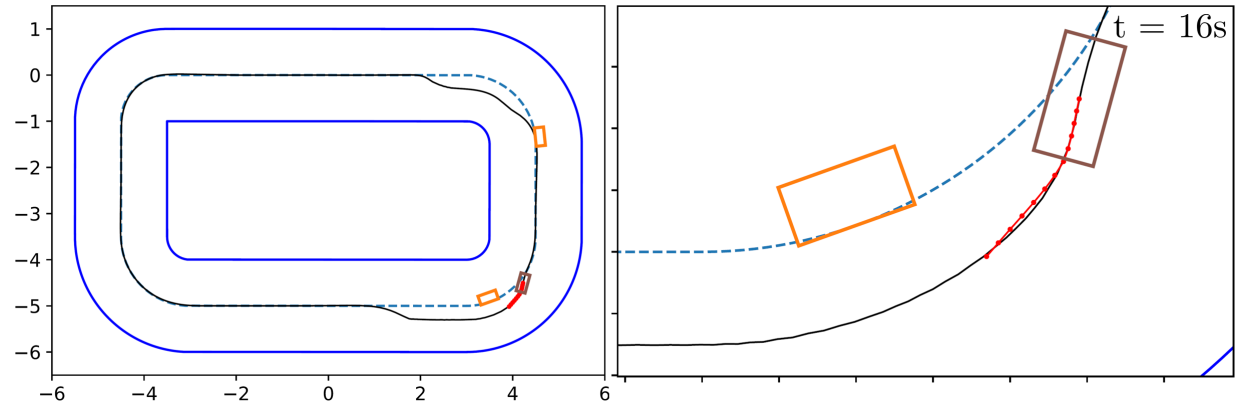
Safety-critical optimal control and planning is one of the fundamental problems in robotic applications. In order to ensure the safety of robotic systems while achieving optimal performance, the tight coupling between potentially conflicting control objectives and safety criteria is considered in an optimization problem. Some recent work formulates this problem using control barrier functions, but only using current state information without prediction, see [9, 2], which yields a greedy control policy. Model predictive control can give a less greedy policy, as it takes future state information into account. However, the safety criteria in a predictive control framework is usually enforced as distance constraints defined under Euclidean norms, such as the distance between the robot and obstacles being larger than a safety margin. This distance constraint will not confine the optimization until the reachable set along the horizon intersects with the obstacles. In other words, the robot will not take actions to avoid the obstacles until it is close to them. One way to solve this problem is to use a larger horizon, but that will increase the computational complexity in the optimization.

We address this challenge above by directly unifying model predictive control with discrete-time control barrier functions together into one optimization problem. This results in a safety-critical model predictive control formulation, called MPC-CBF in this chapter. In this formulation, the CBF constraints could enforce the system to avoid obstacles even when the reachable set along the horizon is far away from the obstacles. We validate this control design using a 2D double integrator for obstacle avoidance, and also demonstrate that this method enables a racing car to safely compete with other cars in a racing competition, shown

[2] The material of this chapter is from "Safety-critical model predictive control with discrete-time control barrier function" published in 2021 American Control Conference (ACC) [224].



(a) Snapshot of first overtake [whole track (left) vs. zoom-in (right)]



(b) Snapshot of second overtake [whole track (left) vs. zoom-in (right)]

Figure 4.1: Safety-critical model predictive control applied to competitive car racing.

in Figure 4.1.

4.1.2 Related Work

4.1.2.1 Model Predictive Control

MPC is widely used for robotic systems, such as robotic manipulation and locomotion [79, 169], to achieve optimal performance while satisfying different constraints. One of the most important criteria to deploy robots for real-world tasks is safety. There is some existing work about model predictive control considering system safety [175, 159]. The safety criteria in the context of MPC is usually formulated as constraints in an optimization problem [221, 57], such as obstacle constraints and actuation limits. One concrete scenario regarding safety criteria for robots is obstacle avoidance. The majority of literature focuses on collision avoidance using simplified models, and considers distance constraints with various Euclidean

norms [190, 162, 227], which we call MPC-DC in this chapter.

However, these obstacle avoidance constraints under Euclidean norms will not confine the robot's movement unless the robot is relatively close to the obstacles. To make the robot take actions to avoid obstacles even far away from it, we usually need a larger horizon which increases the computational time in the optimization. This encourages us to formulate a new type of model predictive control, which can guarantee safety in the context of set invariance with CBF constraints confining the robot's movement during the optimization. Recently in [70], model predictive control is introduced with control Lyapunov functions to ensure stability.

4.1.2.2 Control Barrier Functions

CBFs have recently been introduced as a promising way to ensure set invariance by considering the system dynamics and implying forward invariance of the safe set. Furthermore, a safety-critical control design for continuous-time systems was proposed by unifying a control Lyapunov function (CLF) and a control barrier function through a quadratic program (CLF-CBF-QP) [12]. This method could be deployed as a real-time optimization-based controller with safety-critical constraints, shown in [136, 9, 209]. Besides the continuous-time domain, the formulation of CBFs was generalized into discrete-time systems (DCLF-DCBF) in [2].

4.1.2.3 Model Predictive Control with Control Barrier Functions

There is some existing work that tries to combine the advantages of MPC and CBFs. Barrier functions have been used in MPC in [206], which converts constraints to cost but not related to safety-critical control. In [175], they use continuous-time CBFs as constraints inside a discrete-time MPC. MPC and CBFs are organized as a high-level planner and a low-level tracker in [159]. This method treats MPC and CBFs separately at different levels.

Inspired by the previous work of model predictive control and control barrier functions, DCLF-DCBF can be improved by taking future state prediction into account, yielding a better control policy. This motivates us to investigate the control design of predictive control under the constraints imposed by CBFs. In this chapter, we focus on the discrete-time formulation of control barrier functions applied to model predictive control, which encodes the safety obtained from discrete-time CBFs in MPC.

4.1.3 Contribution

The contributions of this chapter are as follows.

- We present a MPC-CBF control design for safety-critical tasks, where the safety-critical constraints are enforced by discrete-time control barrier functions.

- We analyze the stability of our control design, and qualitatively discuss the feasibility in terms of set intersections between reachable sets of MPC and safe sets enforced by CBF constraints along the horizon.
- Our proposed method is shown to outperform both MPC-DC and DCLF-DCBF. It enables prediction capability to DCLF-DCBF for performance improvement, and it also guarantees safety via discrete-time CBF constraints in the context of set invariance.
- We verify the properties of our control design using a 2D double integrator for obstacle avoidance. Our algorithm is generally applicable and also validated in a more complex scenario, where MPC-CBF enables a car racing on a track while safely overtaking other cars.

4.2 Background

Our proposed safety-critical model predictive control design builds on model predictive control and control barrier functions. We now present necessary preliminaries.

4.2.1 Model Predictive Control

Consider the problem of regulating to the origin of a discrete-time control system described by,

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

where $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ represents the state of the system at time step $t \in \mathbb{Z}^+$, $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, and f is locally Lipschitz.

Assume that a full measurement or estimate of the state $\mathbf{x}_t$ is available at the current time step t. Then a finite-time optimal control problem is solved at time step t. When there are safety criteria, such as obstacle avoidance, the obstacles are usually formulated using distance constraints. The finite-time optimal control formulation is shown in (4.2).

MPC-DC:

$$J_t^*(\mathbf{x}_t) = \min_{\mathbf{u}_{t:t+N-1|t}} p(\mathbf{x}_{t+N|t}) + \sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) \tag{4.2a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}),\ k = 0, ..., N-1 \tag{4.2b}$$

$$\mathbf{x}_{t+k|t} \in \mathcal{X},\ \mathbf{u}_{t+k|t} \in \mathcal{U},\ k = 0, ..., N-1 \tag{4.2c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t, \tag{4.2d}$$

$$\mathbf{x}_{t+N|t} \in \mathcal{X}_f, \tag{4.2e}$$

$$g(\mathbf{x}_{t+k|t}) \geq 0,\ k = 0, ..., N-1. \tag{4.2f}$$

Here $\mathbf{x}_{t+k|t}$ denotes the state vector at time step $t+k$ predicted at time step t obtained by starting from the current state $\mathbf{x}_t$ (4.2d), and applying the input sequence $\mathbf{u}_{t:t+N-1|t}$ to the system dynamics (4.2b). In (4.2a), the terms $q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t})$ and $p(\mathbf{x}_{t+N|t})$ are referred to as stage cost and terminal cost respectively, and N is the time horizon. The state and input constraints are given by (4.2c), and distance constraints for safety criteria are represented by function g, in (4.2f), which could be defined under various Euclidean norms. The terminal constraint is enforced in (4.2e).

Let $\mathbf{u}^*_{t:t+N-1|t} = \{\mathbf{u}^*_{t|t}, ..., \mathbf{u}^*_{t+N-1|t}\}$ be the optimal solution of (4.2) at time step t. The resulting optimized trajectory using $\mathbf{u}^*_{t:t+N-1|t}$ is referred as an open-loop trajectory. Then, the first element of $\mathbf{u}^*_{t:t+N-1|t}$ is applied to system (4.1). This feedback control law is given below,

$$\mathbf{u}(t) = \mathbf{u}^*_{t|t}(\mathbf{x}_t). \tag{4.3}$$

The finite-time optimal control problem (4.2) is repeated at next time step $t+1$, based on the new estimated state $\mathbf{x}_{t+1|t+1} = \mathbf{x}_{t+1}$. It yields the model predictive control strategy. The resulting trajectory using (4.3) is referred as a closed-loop trajectory. More details can be referred to in [26].

4.2.2 Control Barrier Functions

We now present discrete-time control barrier functions that will be used together with model predictive control for our control design, which will be introduced in Sec. 4.3.

For safety-critical control, we consider a set $\mathcal{C}$ defined as the superlevel set of a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$:

$$\mathcal{C} = \{\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n : h(\mathbf{x}) \geq 0\}. \tag{4.4}$$

Throughout this chapter, we refer to $\mathcal{C}$ as a safe set. The function h is a control barrier function (CBF) [9] if $\dfrac{\partial h}{\partial \mathbf{x}} \neq 0$ for all $\mathbf{x} \in \partial\mathcal{C}$ and there exists an extended class $\mathcal{K}_\infty$ function γ such that for the control system (4.1), h satisfies

$$\exists\ \mathbf{u} \text{ s.t. } \dot{h}(\mathbf{x}, \mathbf{u}) \geq -\gamma(h(\mathbf{x})),\ \gamma \in \mathcal{K}_\infty. \tag{4.5}$$

This condition can be extended to the discrete-time domain which is shown as follows

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) \geq -\gamma h(\mathbf{x}_k),\ 0 < \gamma \leq 1, \tag{4.6}$$

where $\Delta h(\mathbf{x}_k, \mathbf{u}_k) := h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k)$. Satisfying constraint (4.6), we have $h(\mathbf{x}_{k+1}) \geq (1-\gamma)h(\mathbf{x}_k)$, i.e., the lower bound of control barrier function $h(x)$ decreases exponentially with the rate $1-\gamma$.

Remark 4.1. *Note that in* (4.6)*, we defined γ as a scalar instead of a $\mathcal{K}_\infty$ function as in* (4.5)*. Generally, for the discrete-time domain, γ could also be considered as a class $\mathcal{K}$ function that also satisfies $0 < \gamma(h(\mathbf{x})) \leq h(\mathbf{x})$ for any $h(\mathbf{x})$. However, we will continue to use the scalar form γ in this chapter to simplify the notations for further discussions.*

Besides the system safety, we are also interested in stabilizing the system with a feedback control law $\mathbf{u}$ under a control Lyapunov function V, i.e.,

$$\exists\ \mathbf{u} \text{ s.t. } \dot{V}(\mathbf{x}, \mathbf{u}) \leq -\alpha(V(\mathbf{x})),\ \alpha \in \mathcal{K}. \tag{4.7}$$

We can also generalize it to the discrete-time domain,

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) \leq -\alpha V(\mathbf{x}_k),\ 0 < \alpha \leq 1, \tag{4.8}$$

where $\Delta V(\mathbf{x}_k, \mathbf{u}_k) := V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k)$. Similarly as above, the upper bound of control Lyapunov function decreases exponentially with the rate $1 - \alpha$.

The discrete-time control Lyapunov function and control barrier function can be unified into one optimization program (DCLF-DCBF), which achieves the control objective and guarantees system safety. This formulation was first introduced in [2] and is presented as follows

---

DCLF-DCBF:

$$\mathbf{u}_k^* = \underset{(\mathbf{u}_k, \delta) \in \mathbb{R}^{m+1}}{\operatorname{argmin}} \quad \mathbf{u}_k^T H(\mathbf{x}) \mathbf{u}_k + l \cdot \delta^2 \tag{4.9a}$$

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) + \alpha V(\mathbf{x}_k) \leq \delta, \tag{4.9b}$$

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) + \gamma h(\mathbf{x}_k) \geq 0, \tag{4.9c}$$

$$\mathbf{u}_k \in \mathcal{U}, \tag{4.9d}$$

---

where $H(\mathbf{x})$ is a positive definite matrix, that is pointwise differentiable in $\mathbf{x}$. l is positive, and $\delta \geq 0$ is a slack variable that allows the control Lyapunov function to grow when the CLF and CBF constraints are conflicting. The safe set $\mathcal{C}$ in (4.4) is invariant along the trajectories of the discrete-time system with controller (4.9) if $h(\mathbf{x}_0) \geq 0$ and $0 < \gamma \leq 1$.

4.3 Control Design

After presenting a background of model predictive control and control barrier functions, we formulate the safety-critical model predictive control logic in this section.

4.3.1 Formulation

Consider the problem of regulating to a target state for the discrete-time system (4.1) while ensuring safety in the context of set invariance. The proposed control logic MPC-CBF solves the following constrained finite-time optimal control problem with horizon N at each time step t,

MPC-CBF:

$$J_t^*(\mathbf{x}_t) = \min_{\mathbf{u}_{t:t+N-1|t}} p(\mathbf{x}_{t+N|t}) + \sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) \tag{4.10a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}),\ k = 0, ..., N-1 \tag{4.10b}$$

$$\mathbf{x}_{t+k|t} \in \mathcal{X}, \mathbf{u}_{t+k|t} \in \mathcal{U},\ k = 0, ..., N-1 \tag{4.10c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t, \tag{4.10d}$$

$$\mathbf{x}_{t+N|t} \in \mathcal{X}_f, \tag{4.10e}$$

$$\Delta h(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) \geq -\gamma h(\mathbf{x}_{t+k|t}),\ k = 0, ..., N-1 \tag{4.10f}$$

where (4.10d) represents the initial condition constraint, (4.10b) describes the system dynamics, and (4.10c) shows the state/input constraints along the horizon. The terminal set constraint is imposed in (4.2e). The CBF constraints in (4.10f) are designed to guarantee the forward invariance of the safe set $\mathcal{C}$ associated with the discrete-time control barrier function, where Δh is as introduced in (4.6). Here we have

$$\Delta h(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) = h(\mathbf{x}_{t+k+1|t}) - h(\mathbf{x}_{t+k|t}).$$

The optimal solution to (4.10) at time t is a sequence of inputs as

$$\mathbf{u}^*_{t:t+N-1|t} = [\mathbf{u}^*_{t|t}, ..., \mathbf{u}^*_{t+N-1|t}].$$

Then, the first element of the optimizer vector is applied

$$\mathbf{u}(t) = \mathbf{u}^*_{t|t}(\mathbf{x}_t). \tag{4.11}$$

This constrained finite-time optimal control problem (4.10) is repeated at time step $t+1$, based on the new state $\mathbf{x}_{t+1|t+1}$, yielding a receding horizon control strategy: safety-critical model predictive control (MPC-CBF).

The dynamics constraint in (4.10b) could be linear if we have a linear system, and (4.10c) could also be linear if $\mathcal{X}$ and $\mathcal{U}$ are defined as polytopes in the state and input space, respectively. The discrete-time control barrier functions constraints in (4.10f) are generally non-convex unless the CBFs are linear. This makes the whole optimization in (4.10) generally become a nonlinear programming problem (NLP).

4.3.2 Stability

In DCLF-DCBF, control Lyapunov functions are introduced as optimization constraints (4.9b) with corresponding slack variable as additional term in the cost function (4.9a). This allows for the achievement of control objectives represented by CLFs and unifies the formulation under one optimization with CBFs. In our MPC-CBF control design, we have

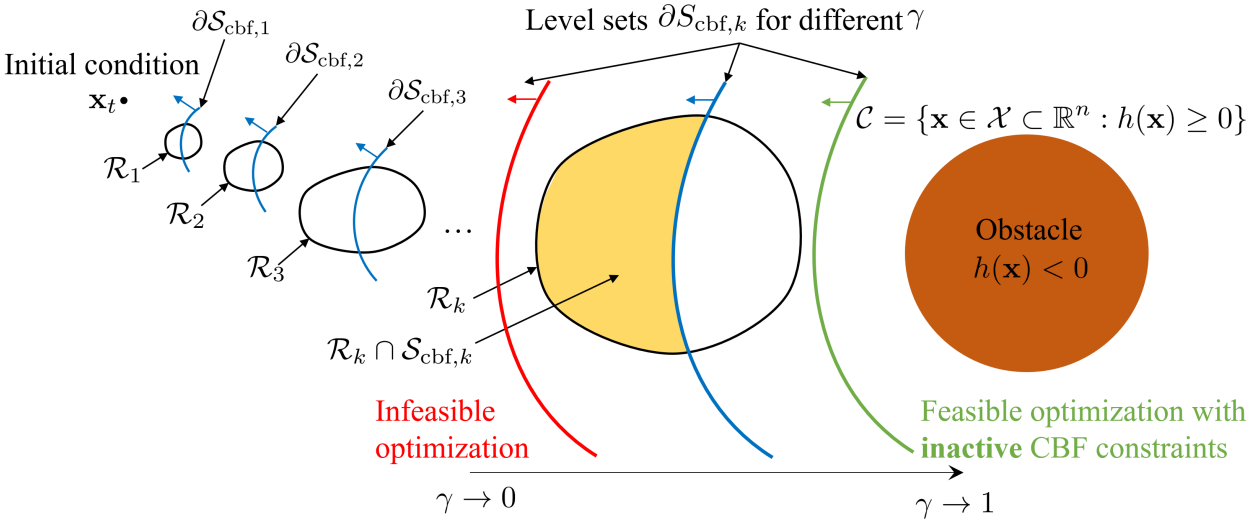


Figure 4.2: Feasibility and reachability analysis of MPC-CBF.

the terminal cost $p(\mathbf{x}_{t+N|t})$ in (4.10a) as a control Lyapunov function, which can be used to guarantee the stability of the closed-loop system by satisfying mild assumptions in the context of linear systems [26, Theorem 12.2]. In general, a rigorous proof of nonlinear MPC stability as well as MPC-CBF still remains an open challenge. More detailed analysis could be found in [6].

4.3.3 Feasibility

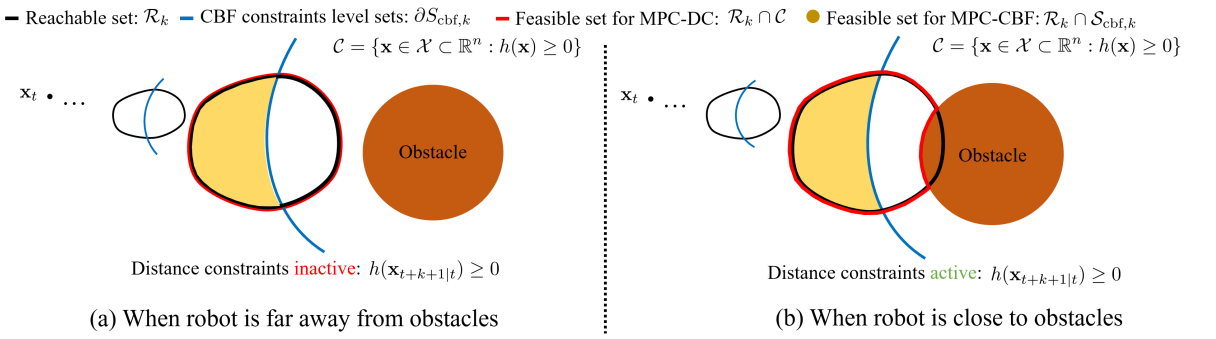
Recursive feasibility is generally not guaranteed for MPC-DC defined in (4.2) [26, 108] and other general NLP [27, Chap. 11]. In this chapter, we qualitatively analyze the feasibility problem of MPC-CBF based on set analysis. Since the optimization (4.10) could be a NLP, we are interested in finding under which circumstances this optimization becomes feasible, i.e., the feasible set under the constraints (4.10b)-(4.10f) is not empty.

Given the current state $\mathbf{x}_t$ in (4.10d), the reachable set at horizon step k is defined as a reachable region in the state space, satisfying system dynamics in (4.10b), input/state constraints in (4.10c) and initial condition in (4.10d). This reachable set $\mathcal{R}_k$ is defined as follows for the horizon step k:

$$\begin{aligned}\mathcal{R}_k = \{\mathbf{x}_{t+k|t} \in \mathcal{X} : \forall i = 0, ..., k-1, \mathbf{x}_{t+i+1|t} = f(\mathbf{x}_{t+i|t}, \mathbf{u}_{t+i|t}),\\ \mathbf{x}_{t+i|t} \in \mathcal{X}, \mathbf{u}_{t+i|t} \in \mathcal{U}, \mathbf{x}_{t|t} = \mathbf{x}_t\}.\end{aligned} \tag{4.12}$$

We also define the set of state space satisfying the CBF constraints in (4.10f) and initial condition in (4.10d) as

$$\mathcal{S}_{\text{cbf},k} = \left\{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) - h(\mathbf{x}_{t+k-1|t}) \geq -\gamma h(\mathbf{x}_{t+k-1|t})\right\}, \tag{4.13}$$



(a) When robot is far away from obstacles

(b) When robot is close to obstacles

Figure 4.3: Constraint activation for MPC-CBF and MPC-DC

where $\mathcal{S}_{\text{cbf},k}$ describes superlevel sets of h satisfying the control barrier function constraints (4.10f) at each time step along the open-loop trajectory. $\mathcal{S}_{\text{cbf},k}$ also depends on the value of optimal value $\mathbf{x}_{t+k-1|t}$, which depends on the states and the inputs of previous nodes before the index $k-1$.

We illustrate $\mathcal{R}_k$ and $\mathcal{S}_{\text{cbf},k}$ in the state space shown in Figure 4.2, given the initial condition $\mathbf{x}_t$. Then, the feasibility of the optimization in (4.10) turns out to be whether the intersection between the feasible set at each horizon step, $\mathcal{R}_k$, and the superlevel set of $h(\mathbf{x})$ satisfying the CBF constraints, $\mathcal{S}_{\text{cbf},k}$, is nonempty for all $k = 1, ..., N$.

Remark 4.2. *Note that $\mathcal{S}_{cbf,k}$ is not empty when h is a valid control barrier function. Furthermore, $\mathcal{R}_k$ is also guaranteed to be nonempty, if we choose $\mathcal{X}$ as a control invariant set as discussed in [26, Theorem 11.2].*

In order to better understand this problem, we illustrate the level sets of control barrier function constraints as

$$\partial\mathcal{S}_{\text{cbf},k} = \{x \in \mathcal{X} : h(\mathbf{x}) = (1-\gamma)h(\mathbf{x}_{t+k-1|t})\},$$

with several choices of γ. The superlevel set $\mathcal{S}_{\text{cbf},k}$ are the regions illustrated on the left hand side of these level sets, with an example where the robot approaches the obstacle from the left to the right, shown in Figure 4.2. We can see that, if γ becomes relatively small, $\mathcal{S}_{\text{cbf},k}$ will be smaller. In this case, the system tends to be safer as the decar-rate of CBF becomes smaller, but the intersection between $\mathcal{R}_k$ and $\mathcal{S}_{\text{cbf},k}$ might be infeasible if γ becomes too small. When the γ becomes larger, the region of $\mathcal{S}_{\text{cbf},k}$ in the state space will be increased. This will make the optimization more likely to be feasible, however, the CBF constraints might not be active during the optimization, if γ is too large. In this case, $\mathcal{R}_k$ will become a proper subset of $\mathcal{S}_{\text{cbf},k}$.

Remark 4.3. *When γ becomes relatively small, the MPC-CBF controller makes a smaller subset of the safe set $\mathcal{C}$ in (4.4) invariant, and thus is more safer, but this might also make the optimization infeasible. A larger γ will make the optimization more likely to be feasible,*

but the CBF constraints might not be active during the optimization. We expect that γ is chosen appropriately, such that the intersection between these two sets will not be empty and becomes a proper subset of $\mathcal{R}_k$. This leads to a tradeoff between safety and feasibility in terms of the choice of γ. However, it still remains an open challenge for how to automatically choose the γ.

Remark 4.4. *Given $\mathbf{x}_t$, $\mathcal{X}$, $\mathcal{U}$, we could pick a value of γ to find a tradeoff between safety and feasibility. However, when the system evolves, this γ might no longer satisfy our safety demand or guarantee the optimization feasibility. Therefore, for a given fixed γ, we generally only have pointwise feasibility and a persistently feasible formulation is still an open problem and is part of future work.*

4.3.4 Relation of MPC-CBF with DCLF-DCBF

When $N = 1$, the formulation in (4.10) could be simplified as an optimization over one step system input $\mathbf{u}_k^*$

$$\begin{aligned} \mathbf{u}_k^* = \underset{\mathbf{u}_k \in \mathbb{R}^m}{\operatorname{argmin}} \quad & p(f(\mathbf{x}_k, \mathbf{u}_k)) + q(\mathbf{x}_k, \mathbf{u}_k) && \text{(4.14a)} \\ & \Delta h(\mathbf{x}_k, \mathbf{u}_k) + \gamma h(\mathbf{x}_k) \geq 0, && \text{(4.14b)} \\ & \mathbf{u}_k \in \mathcal{U}, && \text{(4.14c)} \end{aligned}$$

where $p(f(\mathbf{x}_k, \mathbf{u}_k))$ and $q(\mathbf{x}_k, \mathbf{u}_k)$ are the terminal cost and stage cost which we have seen previously in (4.10a). The optimization (4.14) is similar to the DCLF-DCBF formulation (4.9). The stage cost $q(\mathbf{x}_k, \mathbf{u}_k)$ minimizes the system input, similar as $\mathbf{u}_k^T H(\mathbf{x})\mathbf{u}_k$ in (4.9a). The terminal cost minimizes the control Lyapunov function $p(f(\mathbf{x}_k, \mathbf{u}_k))$ in (4.14a), instead of using CLF constraints as (4.9b). As we no longer use the CLF constraint, we do not need a slack variable, such as δ in (4.9a), to guarantee the feasibility.

Remark 4.5. *As the CLF constraints in* (4.9b) *are transferred into the cost function in* (4.14) *with $N = 1$, the formulation in* (4.14) *becomes similar to DCLF-DCBF. To sum up, our MPC-CBF formulation operates in a similar manner of DCLF-DCBF with prediction $N = 1$.*

4.3.5 Relation of MPC-CBF with MPC-DC

When γ approaches its upper bound of 1, the CBF constraints in (4.10f) becomes,

$$h(\mathbf{x}_{t+k+1|t}) \geq 0.$$

If $g(\mathbf{x})$ in (4.2f) and $h(\mathbf{x})$ are the same, these CBF constraints are almost the same as distance constraints defined in (4.2f) except for one horizon step difference. In other words, the CBF constraints are at the next predicted horizon step instead of the current horizon

step. Moreover, the feasible set at each horizon step k in MPC-DC formulation becomes $\mathcal{R}_k \cap \mathcal{C}$. While, as we have seen previously, the feasible set in MPC-CBF formulation at each horizon step k is $\mathcal{R}_k \cap \mathcal{S}_{\text{cbf},k}$. Note that $\mathcal{S}_{\text{cbf},k}$ is a subset of $\mathcal{C}$. Therefore, the MPC-CBF formulation in (4.10) has a smaller safe set than MPC-DC.

In the case the reachable set $\mathcal{R}_k$ is a proper subset of the safe set $\mathcal{C}$, then the distance constraints in (4.2f) are not active in the optimization (4.2), as shown in Figure 4.3a. In other words, the distance constraints will not be active in the optimization (4.2) until the reachable set along the horizon intersects with the unsafe regions, i.e., the reachable set intersects the obstacles as shown in Figure 4.3b. Using our MPC-CBF formulation, the CBF constraints in (4.10f) could always confine the reachable set $\mathcal{R}_k$ with an appropriate choice of γ whenever the robot tends to approach the obstacles. In this case, even when the reachable set $\mathcal{R}_k$ is a proper subset of the safe set $\mathcal{C}$, the reachable set could still be constrained by its intersection with $\mathcal{S}_{\text{cbf},k}$, as shown in Figure 4.3a.

Remark 4.6. *We discuss the constraint activation only in the cases when the robot is moving towards the obstacles, i.e., there exists a control input* $\mathbf{u}$ *such that* $h(f(\mathbf{x}, \mathbf{u})) < h(\mathbf{x})$. *When the robot is moving away from the obstacles, both distance constraints and CBF constraints are inactive, which is intuitive since the robot is always safe in this case.*

Remark 4.7. *Our MPC-CBF formulation is safer than MPC-DC in the context of smaller set invariance, where we can see that* $\mathcal{R}_k \cap \mathcal{S}_{cbf,k}$ *is a proper subset of* $\mathcal{R}_k \cap \mathcal{C}$, *as shown in Figure 4.3.*

Remark 4.8. *In practice, we may need to use a smaller horizon for speeding up the optimization. To achieve a similar performance compared to the MPC-DC formulation in* (4.2), *we could apply smaller* γ *in our MPC-CBF control design in* (4.10) *and use a smaller horizon. This could help us to reduce the complexity of the optimization for obstacle avoidance, as will be shown in Figure 4.4e.*

4.4 Examples

Having presented the MPC-CBF control design, we now numerically validate it using a 2D double integrator for obstacle avoidance and analyze its properties. We also apply this controller to a competitive car racing problem. We use the solver IPOPT [196] in MPC-CBF/MPC-DC/DCLF-DCBF problems [22].

4.4.1 2D Double Integrator for Obstacle Avoidance

Consider a linear discrete-time 2D double integrator

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \tag{4.15}$$

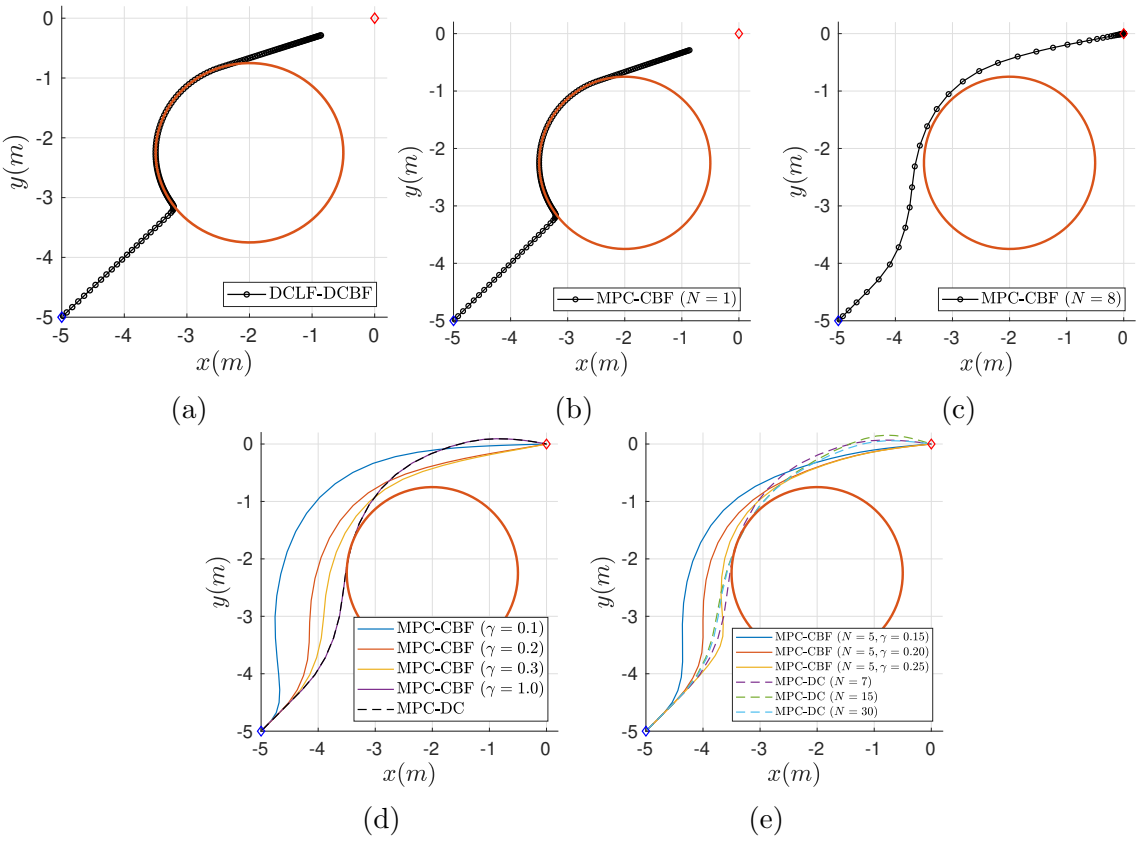


Figure 4.4: A 2D double integrator avoids an obstacle using different control designs.

where the sampling time Δt is set as $0.2s$.

A MPC-CBF is designed as in (4.10) and (4.11) for 2D double integrator to avoid an obstacle, where the stage cost and and terminal cost are

$$q(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k' Q \mathbf{x}_k + \mathbf{u}_k' R \mathbf{u}_k, \; p(\mathbf{x}_N) = \mathbf{x}_N' P \mathbf{x}_N, \tag{4.16}$$

where $Q = 10 \cdot \mathcal{I}_4$, $R = \mathcal{I}_2$ and $P = 100 \cdot \mathcal{I}_4$. The system is subject to state constraint $\mathcal{X}$ and input constraints $\mathcal{U}$,

$$\mathcal{X} = \{\mathbf{x}_k \in \mathbb{R}^n : \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}\},$$
$$\mathcal{U} = \{\mathbf{u}_k \in \mathbb{R}^m : \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}\}.$$

The lower and upper bounds are

$$\mathbf{x}_{\max}, \mathbf{x}_{\min} = \pm 5 \cdot \mathcal{I}_{4\times 1}, \; \mathbf{u}_{\max}, \mathbf{u}_{\min} = \pm \mathcal{I}_{2\times 1}.$$

For discrete-time control barrier function constraint (4.10f), we choose a quadratic barrier function for obstacle avoidance

$$h_k = (\mathbf{x}_k(1) - x_{obs})^2 + (\mathbf{x}_k(2) - y_{obs})^2 - r_{obs}^2, \tag{4.17}$$

where x_{obs}, y_{obs}, and r_{obs} describe x/y-coordinate and radius of the obstacle with $x_{obs} = -2m$, $y_{obs} = -2.25m$ and $r_{obs} = 1.5m$, shown as a red circle in Figure 4.4. The start and target positions are $(-5, -5)$ and $(0, 0)$, which are labelled as blue and red diamonds in Figure 4.4, respectively.

4.4.1.1 Comparison of MPC-CBF with DCLF-DCBF

In order to compare the performance between our proposed MPC-CBF and DCLF-DCBF, we develop a DCLF-DCBF controller for the same obstacle avoidance task, which is based on (4.9).

For the design of DCLF-DCBF, we use R in (4.16) as H in (4.9) to ensure that we have the same penalty on inputs. The discrete-time CBF constraints of DCLF-DCBF are the same as the ones used in MPC-CBF. Since the terminal cost p in the model predictive control serves as a control Lyapunov function, we choose p in MPC-CBF example as the control Lyapunov function that is used to construct the discrete-time CLF constraints in (4.9). Based on these choices, it is fair to compare a DCLF-DCBF controller with a MPC-CBF controller.

The simulation result of MPC-CBF and DCLF-DCBF comparison is shown in Figure 4.4a, 4.4b and 4.4c. The trajectory is denoted in black line with small circles representing each step. The trajectory of DCLF-DCBF controller with $\gamma = 0.4$ is presented in Figure 4.4a, where the system does not start to avoid the obstacle until it is close to it. Figure 4.4b shows the trajectory for MPC-CBF controller with horizon $N = 1$ and $\gamma = 0.4$. This trajectory is similar to that of DCLF-DCBF controller. Based on our analysis in Sec. 4.3.4, the performances of DCLF-DCBF and MPC-CBF with $N = 1$ are almost the same, which is validated in this simulation. Figure 4.4c shows the trajectory of MPC-CBF controller with horizon $N = 8$ and $\gamma = 0.4$. We can see that this controller can drive the system to avoid the obstacle earlier than the DCLF-DCBF controller. Also, among these three controllers, it is the only one that can reach the goal position in the limited simulation time.

4.4.1.2 Comparison of MPC-CBF with MPC-DC

A MPC-DC controller is developed based on (4.2) using the same parameters as MPC-CBF presented before except for the discrete-time CBF constraint, which is replaced by a Euclidean norm distance constraint g shown in (4.2f). The function g has the same expression as h, defined in (4.17).

Figure 4.4d and 4.4e show the simulation result of the comparison between MPC-CBF and MPC-DC. In Figure 4.4d, MPC-CBF controllers with $\gamma = 0.1, 0.2, 0.3, 1.0$ are described in blue, orange, yellow and purple lines, respectively. The trajectory of MPC-DC controller is shown in black dashed line. As γ decreases, the system starts to avoid the obstacle earlier, which means a smaller safe set as analyzed in Sec. 4.3.3, while on the other hand the trajectory of MPC-DC is the closest to the obstacle. We also notice that the trajectories of MPC-DC and MPC-CBF with $\gamma = 1$ are almost the same, which validates our analysis in Sec. 4.3.5.

controller	status	N	γ	mean/std (s)	min dist	cost
MPC-CBF	solved	5	0.1	0.028±0.012	1.483	7.620
MPC-CBF	solved	5	0.2	0.028±0.011	0.791	7.464
MPC-CBF	solved	5	0.3	0.028±0.011	0.441	8.314
MPC-CBF	solved	5	0.4	0.028±0.011	0.288	8.292
MPC-CBF	solved	5	0.5	0.028±0.010	0.110	8.813
controller	status	N		mean/std (s)	min dist	cost
MPC-DC	infeas.	5		NaN	NaN	NaN
MPC-DC	solved	7		0.033±0.013	0.000	9.102
MPC-DC	solved	15		0.048±0.016	0.000	8.537
MPC-DC	solved	30		0.062±0.031	0.000	8.528

Table 4.1: MPC-DC and MPC-CBF benchmark in terms of prediction horizon, computational time, minimal distance with respect to obstacle and cost integral.

The trajectories of MPC-CBF controllers with $N = 5$ and different choices of γ and MPC-DC controllers with different values of horizon N are shown in Figure 4.4e. We notice that MPC-CBF controller with smaller γ and MPC-DC with larger horizon N can make the system avoid obstacles earlier. This verifies our analysis in Sec. 4.3.3, since smaller γ in MPC-CBF and larger horizon N in MPC-DC make the trajectory deviate from obstacles earlier. We also observe that even with an extremely large horizon N, e.g. $N = 30$, the system only has noticeable obstacle avoidance behavior when it is close to obstacles. In contrary, a relatively small γ is able to make the system avoid obstacles even when far away from obstacles.

In Figure 4.4e, MPC-CBF with $N = 5$ and $\gamma = 0.25$ starts to turn to avoid the obstacle with a similar behavior as MPC-DC with $N = 7$. This property is discussed in Remark 4.8. Since discrete-time CBF enforces the invariance of safe set, it allows a smaller N for MPC-CBF with a smaller γ to achieve a comparable performance as MPC-DC with a larger N.

In Table 4.1, we benchmark the MPC-CBF and MPC-DC in terms of prediction horizon, computation time, minimal distance to the obstacle and cost integral $\sum_k \mathbf{u}_k^T \mathbf{u}_k \Delta t$ over the trajectory. We can observe that less prediction horizon of MPC-CBF leads to less computational time. MPC-DC always reaches to the boundary of the obstacle, however, MPC-CBF could automatically set a safety margin depending on different choices of γ. For this specific scenario, the cost integral over the trajectory of MPC-CBF is generally less than the one of MPC-DC.

4.4.2 Competitive Car Racing

We have evaluated the MPC-CBF design using a 2D double integrator and compared its performance with DCLF-DCBF and MPC-DC. We proceed to implement MPC-CBF in a

more complex scenario: competitive car racing. In some previous car racing control work [117, 193], they only consider static obstacles on the track while we deal with dynamic obstacles such as other cars using MPC-CBF.

4.4.2.1 Vehicle Model

We use curvilinear coordinates to describe vehicle states of the ego and other cars in a racing competition. In this chapter, we use the nonlinear lateral vehicle dynamics model in [150, Ch. 2] for system dynamics

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{4.18}$$

where x_t and u_t represent the state and input of the vehicle at time step t and their definitions are as follow

$$\mathbf{x}_t = [v_{x_t},\ v_{y_t},\ \phi_t,\ e_{\phi_t},\ s_t,\ e_{y_t}]^T,\ \mathbf{u}_t = [a_t,\ \delta_t]^T, \tag{4.19}$$

where s_t represents the curvilinear distance travelled along the centerline of the track, e_{y_t} and e_{ϕ_t} are the deviation distance and heading angle error between vehicle and path. v_{x_t}, v_{y_t}, ϕ_t are the vehicle longitudinal velocity, lateral velocity and yaw rate in the curvilinear coordinates, respectively. A representation of the state in the curvilinear coordinate is shown in Figure 4.5. The inputs are longitudinal acceleration a_t and steering angle δ_t. In the car racing, we assume to have K racing cars competing with the ego car, and we use the superscript i to distinguish the i-th ($i = 1, 2, ..., K$) competiting vehicle from the ego one, shown in Figure 4.5.

We note that this kinematic model is highly nonlinear and this model cannot be applied to formulate system dynamics constraints in (4.10b). Instead, we follow the data-driven approach proposed in [160]. We firstly simulate the vehicle system to track the centerline by using a PID controller

$$\begin{aligned}\delta_t &= -k_1 e_{y_t} - k_2 e_{\psi_t},\\ a_t &= k_3(v_d - v_{x_t}),\end{aligned}$$

where v_d could be any positive user-defined target speed. This PID controller allows the vehicle to track the centerline of the track with a target speed.

Then, the linearized time-invariant dynamics is shown as below

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t, \tag{4.20}$$

where A and B are calculated with a regression-based approach [160] over the trajectory we simulated using a PID controller. This process allows us to approximate the highly nonlinear dynamics with the time-invariant linearized one in (4.20). The whole process is purely data-driven without a prior knowledge of the system parameters. This linearized dynamics (4.20) is used in CBF constraints (4.10b), which reduces the burden of the computational complexity for the numerical simulation.

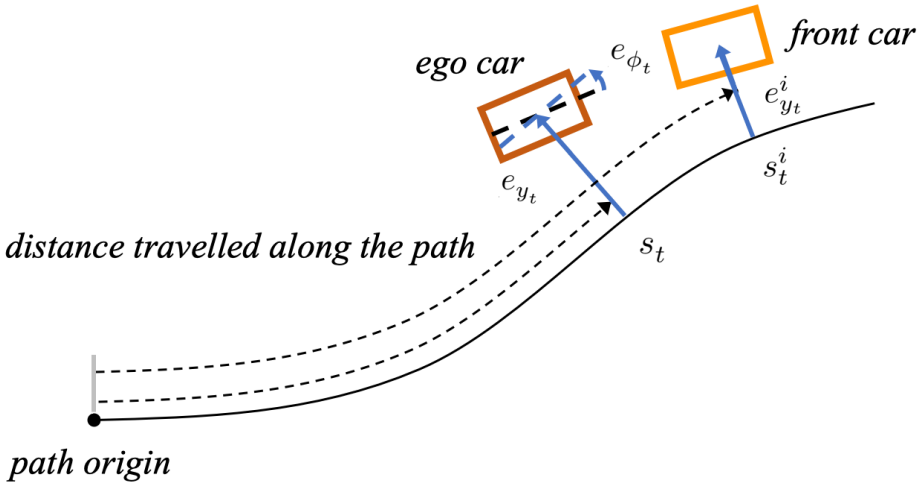


Figure 4.5: Representation of the ego and the leading car in the curvilinear coordinate frame.

4.4.2.2 Control Design

A MPC-CBF is developed for this competitive car racing example using (4.10). The stage cost function is designed as follows

$$q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) = (\mathbf{x}_{t+k|t} - \mathbf{x}_r)^T Q (\mathbf{x}_{t+k|t} - \mathbf{x}_r) + \mathbf{u}_{t+k|t}^T R \mathbf{u}_{t+k|t}, \tag{4.21}$$

where $\mathbf{x}_r = (v_t, 0, 0, 0, 0, 0)$, $Q = diag(10, 0, 0, 0, 0, 10)$ and $R = diag(1, 1)$. This cost function allows the ego car to track the centerline with a target speed v_t while minimizing the tracking error from the centerline.

The motion of overtaking other racing cars is considered as CBF constraints in (4.10f). At time step t, each CBF h_t^i represents the safety criterion between ego car at (s_t, e_{y_t}) and i-th other racing car at $(s_t^i, e_{y_t}^i)$, described in the curvilinear coordinates, shown in Figure 4.5. We choose CBF in a quartic form as follows

$$h_t^i = \frac{(s_t - s_t^i)^4}{(2l_1)^4} + \frac{(e_{y_t} - e_{y_t}^i)^4}{(2l_2)^4} - 1, \tag{4.22}$$

where we assume all racing cars including ego car hold the shape of rectangle with a length as $2l_1$ and width as $2l_2$. Notice that we assume that we have perfect estimation about (s_t, e_{y_t}) and $(s_t^i, e_{y_t}^i)$.

4.4.2.3 Simulation & Results

During the competition, we expect ego car to track the centerline with a target speed $v_t = 0.6$m/s. MPC-CBF with a horizon $N = 12$ updates at 10 Hz. The system dynamics is simulated at 1000 Hz and the controller sampling time is $0.1s$. Our ego vehicle is simulated

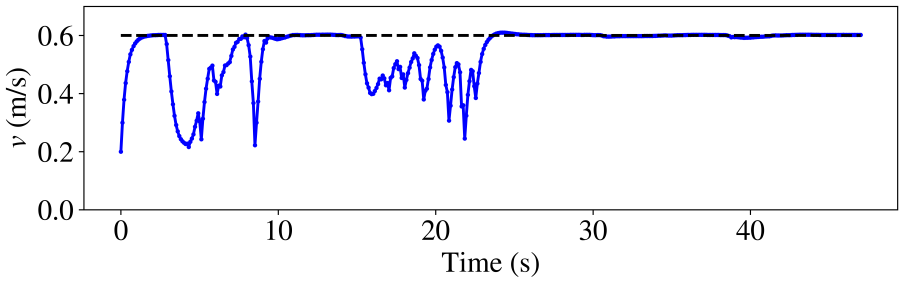


Figure 4.6: Speed profile during the car racing in one lap of the simulation.

with the nonlinear lateral vehicle dynamics model and we use the linearized dynamics along the centerline to formulate our control design. In the simulation, we deploy several racing cars to compete with the ego car. In order to better illustrate results, a snapshot of overtaking motion with a zoom-in view is shown in Figure 4.1. The ego car begins with an initial speed $v_0 = 0.2$ m/s at the origin of the centerline and two other racing cars start in front of the ego car. Two cars are simulated to move at 0.2 m/s while keeping a constant distance deviation e_y^i from the centerline, where $e_y^1 = 0.1$ m and $e_y^2 = -0.1$ m. Figure 4.1 demonstrates that the MPC-CBF allows ego car to safely race and overtake other cars in both left and right directions. Figure 4.6 shows the speed profile, where the dashed black line shows the desired speed. We can see that the ego car always tries to catch up to the target speed during the competitive car racing. In Figure 4.1, we observe two motions of overtaking front racing cars. Since two racing vehicles hold opposite distance deviations from the centerline, ego car overtakes them with right and left turns respectively.

4.5 Chapter Summary

In this chapter, a safety-critical model predictive control design is proposed in this chapter, where discrete-time control barrier function constraints are used in a receding horizon fashion to ensure safety. We present an analysis of its stability and feasibility, and describe its relation with MPC-DC and DCLF-DCBF. To verify our analysis, we use a 2D double integrator for obstacle avoidance, where we can see that MPC-CBF outperforms both MPC-DC and DCLF-DCBF. The proposed control logic is also applied to a more complex scenario: competitive car racing, where our ego car can race and safely overtake other racing cars.

In the next chapter, we will focus specifically on the feasibility problem in the formulation of model predictive control with control barrier functions.

Chapter 5

Feasibility Enhancement in Discrete-time Control Barrier Function [3]

5.1 Introduction

5.1.1 Motivation

Safety-critical optimal control and planning is one of the fundamental problems in robotic applications. In order to ensure the safety of robotic systems while achieving optimal performance, tight coupling between potentially conflicting control objectives and safety criteria is considered in an optimization problem. Some researchers formulate this problem using control barrier functions under the continuous dynamics of the system [9, 11], where the optimal performance is achieved by the control Lyapunov functions and safety criteria is guaranteed through control barrier functions. Recently, this methodology is also introduced in the discrete-time domain and the optimal control problem can be formulated to calculate the current optimal input [2], or a sequence of ones in the fashion of model predictive control [224]. However, feasibility and safety are still the main challenges for these formulations using discrete-time control barrier functions. In this chapter, our proposed formulations focus on how to handle these problems and provides a detailed analysis comparing to existing methods using discrete-time control barrier functions.

5.1.2 Related Work

Designing controllers to ensure provable safety guarantees for autonomous systems is vital. One approach to provide safety guarantees in control is to draw inspirations from control

[3]The material of this chapter is from "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions" published in 2021 60th IEEE Conference on Decision and Control (CDC) [223].

barrier functions [12]. The CBF-QP formulation [13] permits us to find the minimum perturbation for a given feedback controller to guarantee safety. Control Lyapunonv functions (CLFs) [158] can be applied to stabilize the closed-loop dynamics of both linear and nonlinear dynamical systems [61]. Together with CBFs, the CLF-CBF-QP formulation [11] enables handling safety-critical constraints effectively in real-time. This approach is also generalized for high-order systems [133, 214]. Robust or adaptive optimal control are also applied with this technique [134, 90, 185, 122].

5.1.2.1 Optimal Control with Discrete-time CBFs

Besides the continuous-time domain, the formulations of CBFs are generalized into discrete-time systems. An optimization problem can be formulated to calculate the current optimal control input, proposed in the DCLF-DCBF formulation in [2]. A type of model predictive control is also recently introduced to enhance performance. The model predictive control with control Lyapunov functions (NMPC-DCLF) is proposed to ensure stability in [70], where discrete-time CLF constraints are considered under nonlinear model predictive control (NMPC). A control design (MPC-DCBF) for safety-critical tasks is firstly presented in [224], where the safety-critical constraints are enforced by discrete-time control barrier functions. This approach could also be applied to a multi-layer control framework [159, 69], where the safety-critical control with discrete-time CBF serves as a mid-level controller or planner.

5.1.2.2 Feasibility & Safety

Among the formulations in the discrete-time domain, optimal control with discrete-time CBFs also encounter feasibility issues. The feasibility issues arise due to the potential empty intersection between the reachable set and the safe region confined by CBF constraints at each time step. Moreover, with current approaches, there exists a tradeoff between safety performance and feasibility and they cannot be enhanced at the same time, as discussed in [224]. In other words, reducing the decay rate of CBF constraints increases the system safety, but comes at the possibility of infeasibility. A potential way to partly handle the feasibility issue is to adopt the CBF constraint only on the first time step as a one-step constraint, as presented in the formulation MPC-GCBF in [124]. This approach is shown to enhance the feasibility, however, we are still under the restriction between choosing feasibility and safety, as the intersection between the reachable set and the safe region at the first step could still potentially be empty. A soft constrained predictive safety filter with a terminal control barrier function is proposed in [195], which enhances the safety with respect to the model uncertainty. However, the tradeoff between feasibility and safety is unsolved.

Similarly, feasibility is challenging to be guaranteed in the continuous domain. Many existing approaches relax the CLF constraint to resolve the conflict between CLF and CBF constraints, as summarized in [12]. However, the QP-based problem could become infeasible as the CBF constraint might violate the input constraint even when we relax the CLF constraint. In [11], a valid CBF is specifically designed for the adaptive cruise control

scenario based on the system dynamics under input constraints, which could ensure the feasibility in the optimization. This approach solves the problem specifically for this system but not for general nonlinear systems. Recently in [226], the decay rates of CBF constraints are relaxed with optimization variables, which generally resolves the conflict between the CBF constraint and the input constraint and guarantees point-wise feasibility.

In this chapter, the proposed formulations draw inspiration from the decay-rate relaxing technique for the CBF constraint in the continuous-time domain, and will be shown to be able to enhance the feasibility and the safety at the same time. The proposed formulations are generalized for both one-step or multi-step constraints, and don't specifically require only a one-step constraint to increase the feasibility as done with MPC-GCBF [124].

5.1.3 Contributions

The contributions of this chapter are as follows:

- We propose two control frameworks for guaranteeing stability and safety using nonlinear MPC (NMPC), where the safety criteria are considered as discrete-time CBF constraints, and stability criteria appear as CLFs formulated either as a terminal cost or discrete-time constraints.
- The decay rates of the discrete-time control barrier function constraints are relaxed in the optimization problem, which allow the proposed formulations to enhance the feasibility and the safety at the same time for both one-step and multi-step input optimization.
- The proposed formulations are shown theoretically to enhance the feasibility and safety performance compared with existing approaches, and also validated with numerical examples.

5.2 Background

Having introduced the problem, we next present background on CLFs and CBFs and revisit some existing optimal control formulations using discrete-time CLFs and CBFs.

5.2.1 CLFs and CBFs

In this chapter, we consider a discrete-time control system as follows,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{5.1}$$

with $\mathbf{x} \in \mathcal{X}$ representing the system state with the control input $\mathbf{u}$ confined by admissible input set $\mathcal{U}$. For safety-critical control, we consider a set $\mathcal{C}$ defined as the superlevel set of a

continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} \mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}, \\ \partial\mathcal{C} &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) = 0\}, \\ \text{Int}(\mathcal{C}) &= \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) > 0\}. \end{aligned} \tag{5.2}$$

Throughout this chapter, we refer to $\mathcal{C}$ as a safe set. The safe set can be regarded as the ensemble of states satisfying distance constraints

$$h(\mathbf{x}) \geq 0. \tag{5.3}$$

In a stricter manner, the function h becomes a control barrier function in the discrete-time domain if it satisfies the following relation,

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) \geq -\gamma_k h(\mathbf{x}_k),\ 0 < \gamma_k \leq 1, \tag{5.4}$$

where $\Delta h(\mathbf{x}_k, \mathbf{u}_k) := h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k)$. Satisfying constraint (5.4), we have $h(\mathbf{x}_{k+1}) \geq (1 - \gamma_k)h(\mathbf{x}_k)$, i.e, the lower bound of control barrier function $h(\mathbf{x})$ decreases exponentially at time k with the rate $1 - \gamma_k$.

Besides the system safety, we are also interested in stabilizing the system with a feedback control law $\mathbf{u}$ under a control Lyapunov function V in the discrete-time domain,

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) \leq -\alpha_k V(\mathbf{x}_k),\ 0 < \alpha_k \leq 1, \tag{5.5}$$

where $\Delta V(\mathbf{x}_k, \mathbf{u}_k) := V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k)$. Similarly as above, the upper bound of control Lyapunov function decreases exponentially at time k with the rate $1 - \alpha_k$.

5.2.2 Existing Approaches Revisited

In this section, we will revisit some existing optimal control formulations using discrete-time CLFs or CBFs.

5.2.2.1 DCLF-DCBF [2]

The discrete-time control Lyapunov function and control barrier function can be unified into one optimization program, which achieves the control objective and guarantees system safety. This formulation was introduced in [2] and is presented as follows,

---

DCLF-DCBF:

$$\mathbf{u}_k^* = \underset{(\mathbf{u}_k, s) \in \mathbb{R}^{m+1}}{\text{argmin}} \quad \mathbf{u}_k^T H(\mathbf{x}) \mathbf{u}_k + \phi(s) \tag{5.6a}$$

$$\Delta V(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k V(\mathbf{x}_k) \leq s \tag{5.6b}$$

$$\Delta h(\mathbf{x}_k, \mathbf{u}_k) + \gamma_k h(\mathbf{x}_k) \geq 0 \tag{5.6c}$$

$$\mathbf{u}_k \in \mathcal{U}, \tag{5.6d}$$

where $H(\mathbf{x})$ is any positive definite matrix and $s \geq 0$ is a slack variable together with an additional cost term $\phi(s) \geq 0$ that allows the Lyapunov function to grow when the CLF and CBF constraints are conflicting. The safe set $\mathcal{C}$ in (5.2) is invariant along the trajectories of the discrete-time system with controller (5.6) if $h(\mathbf{x}_0) \geq 0$ and $0 < \gamma_k \leq 1$.

5.2.2.2 MPC-DCBF [224]

Inspired by the previous work of model predictive control and control barrier functions, DCLF-DCBF can be improved by taking future state prediction into account, yielding the form of MPC-DCBF firstly introduced in [224]:

MPC-DCBF:

$$J_t^*(\mathbf{x}_t) = \min_U p(\mathbf{x}_{t+N|t}) + \sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) \tag{5.7a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}),\ k = 0, ..., N-1 \tag{5.7b}$$

$$\mathbf{u}_{t+k|t} \in \mathcal{U},\ \mathbf{x}_{t+k|t} \in \mathcal{X},\ k = 0, ..., N-1 \tag{5.7c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t, \tag{5.7d}$$

$$\Delta h(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) \geq -\gamma_k h(\mathbf{x}_{t+k|t}).\ k = 0, ..., N-1 \tag{5.7e}$$

Here, at each time t, the optimized cost-to-go function is denoted as $J_t^*(\mathbf{x}_t)$ and $U = [\mathbf{u}_{t|t}^T, ..., \mathbf{u}_{t+N-1|t}^T]^T$. In the cost function, $p(\mathbf{x}_{t+N|t})$ and $q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t})$ represent terminal cost and stage cost at each time step. The constraint (5.7b) describes the system dynamics, (5.7c) shows the input constraints along the horizon and (5.7d) represents the initial condition constraint. The CBF constraints imposed in (5.7e) are designed to guarantee the forward invariance of the safe set $\mathcal{C}$ associated with the discrete-time control barrier function. Here we have

$$\Delta h(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) = h(\mathbf{x}_{t+k+1|t}) - h(\mathbf{x}_{t+k|t}).$$

The optimal solution to (5.7) at time t is a sequence of inputs as $U^* = [\mathbf{u}_{t|t}^{*T}, ..., \mathbf{u}_{t+N-1|t}^{*T}]^T$. Then, the first element of the optimizer vector is applied, i.e.,

$$\mathbf{u}_t = \mathbf{u}_{t|t}^*(\mathbf{x}_t). \tag{5.8}$$

This constrained finite-time optimal control problem (5.7) is repeated at time step $t+1$, based on the new state $\mathbf{x}_{t+1|t+1}$, yielding a receding horizon control strategy. In [224], MPC-DCBF has been shown to have better exploration performance with predictive horizon, which can be beneficial in maneuvering through deadlock conditions. However, MPC-DCBF has a smaller region of feasibility than the DCLF-DCBF controller as more constraints are used to confine the optimal control input.

5.2.2.3 MPC-GCBF [124]

In the formulation (5.7), the CBF constraints are imposed on multi-steps along the horizon. This increases the computational complexity with a large horizon and the possibility of infeasibility also increases with more constraints. One way to suppress the computational complexity and enhance the feasibility is to apply a one-step CBF constraint, which is done by modifying the CBF constraint in (5.7e) as follows,

$$\Delta h(\mathbf{x}_{t+1|t}, \mathbf{u}_{t|t}) \geq -\gamma h(\mathbf{x}_{t|t}). \tag{5.9}$$

This approach is firstly shown in the MPC-GCBF formulation in [124], and enhances the feasibility and reduces the computational time at the same time due to fewer constraints. This approach is also generalized with consideration over the high relative-degree constraint, where constraints in (5.7e) are modified as constraints posed on two nonadjacent steps,

$$h(\mathbf{x}_{t+m|t}) \geq (1-\gamma)^m h(\mathbf{x}_{t|t}), \tag{5.10}$$

where m represents the relative degree of the high-order safety constraint.

5.2.2.4 CLF-NMPC [70]

Beside the control barrier functions, the model predictive control is also unified with control Lyapunov functions, where stability constraints with CLFs are considered in the proposed CLF-NMPC formulation [70]. The CLF constraint can be imposed during one-step or multi-step,

$$\Delta V(\mathbf{x}_{t+k|t}) \leq -\alpha_k V(\mathbf{x}_{t+k|t}) + s_k, \tag{5.11}$$

where s_k is the slack variable. The safety criteria is not considered in CLF-NMPC. For more details see [70].

5.3 CLFs and CBFs unified with NMPC

In this section, we are going to present two proposed formulations: NMPC-DCBF and NMPC-DCLF-DCBF unifying discrete-time CLFs and CBFs with NMPC.

5.3.1 Formulations

Firstly, we incorporate discrete-time CBF constraints of decay rates under a relaxing technique into a nonlinear model predictive control framework, denoted as NMPC-DCBF with the decision variables $U = [\mathbf{u}_{t|t}^T, ..., \mathbf{u}_{t+N-1|t}^T]^T$ and $\Omega = [\omega_1, ..., \omega_{M_{\text{CBF}}-1}]^T$.

NMPC-DCBF:

$$J_t^*(\mathbf{x}_t) = \min_{U,\Omega} \beta V(\mathbf{x}_{t+N|t}) + \sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) + \psi(\omega_k) \tag{5.12a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}),\ k{=}0, ..., N{-}1 \tag{5.12b}$$

$$\mathbf{u}_{t+k|t} \in \mathcal{U},\ \mathbf{x}_{t+k|t} \in \mathcal{X},\ k = 0, ..., N-1 \tag{5.12c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t, \tag{5.12d}$$

$$h(\mathbf{x}_{t+k+1|t}) \geq \omega_k(1-\gamma_k)h(\mathbf{x}_{t+k|t}),\ \omega_k \geq 0\ k = 0, ..., M_{\text{CBF}} - 1 \tag{5.12e}$$

The system dynamics (5.12b), the input constraint (5.12c) and the initial condition (5.12d) are imposed on the optimization. The control Lyapunov function $V(\mathbf{x}_{t+N|t})$ is used as a terminal cost scaled up with the parameter β, together with the cumulative stage cost along the horizon $\sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t})$. The terminal cost as CLF adopts the fashion of work from the field of MPC, as noted in [152], where stability usually can be achieved without the need to specify a terminal state constraint if β is selected large enough.

Different from the formulation in (5.7), the decay rates of control barrier functions are relaxed from the fixed value $1-\gamma_k$ into optimization variables $\omega_k(1-\gamma_k)$ and an additional cost about the relaxing rate variables $\psi(\omega_k) \geq 0$ is included in the optimization. This function ψ can be tuned for different performance. The slack variable ω_k for relaxing is constrained by (5.12e) such that the following relation is guaranteed,

$$h(\mathbf{x}_{t+k+1|t}) \geq \omega_k(1-\gamma_k)h(\mathbf{x}_{t+k|t}) \geq 0. \tag{5.13}$$

This results in the safety guarantee for the first M_{CBF} steps in the open-loop trajectory but not for the entire horizon N. Here, one horizon length N is designed for dynamics constraint, input constraint and stage cost, and another horizon length $M_{\text{CBF}} \leq N$ is applied for CBF constraints, which allows us to choose the appropriate value of M_{CBF} to reduce the computational complexity. We denote this formulation as NMPC-DCBF as the optimization is always nonlinear since the constraints (5.12e) are nonlinear even if system dynamics and the $h(.)$ function are linear. Note that the constraints inside a MPC-DCBF could become linear if the system dynamics and the $h(.)$ function are linear.

Remark 5.1. *Here, we hypothesize that the existence of the closed-loop trajectory can still be guaranteed by iterations. Formal guarantee of this property requires analysis of recursive feasibility and reachability, which will be proved in the subsequent work.*

Remark 5.2. *NMPC-DCBF represents a generalized form of MPC-DCBF and MPC-GCBF with the relaxing technique of decay rates of safety constraints. When the slack variable ω_k is fixed as 1, NMPC-DCBF becomes the same as MPC-DCBF when $M_{CBF} = N$ and the same as MPC-GCBF when $M_{CBF} = 1$.*

Remark 5.3. *The fixed decay rates for safety constraints existing in MPC-DCBF and MPC-GCBF are relaxed and become optimization variables in NMPC-DCBF, which increase the optimization feasibility.*

Remark 5.4. *MPC-GCBF reduces the computational complexity and increases feasibility by reducing multi-step constraints into one-step. However, one-step constraint might not confine the system sufficiently and the optimization problem may become infeasible after a while in the closed-trajectory, as shown in Figure 5.4b. Moreover, its set invariance for high-relative degree constraint relies on additional assumptions that* $h(\mathbf{x}_{t+i|t}) > 0$ *for* $i = 0, 1, ...m-1$*, shown in [124, Theorem 2]. This could be invalid as it depends on the initial condition of the system. Additionally, identifying the high-relative degree of general complex dynamical systems could be difficult. In our proposed approach, we continue to use the multi-step constraints, which guarantee stronger set invariance with less assumptions. To apply NMPC-DCBF on a high-relative degree system, we just simply need* $M_{CBF} \geq m$*, where* m *represents the relative-degree defined in* (5.10)*.*

Alternatively, the stability criteria with CLF could be posed as constraints instead of as a terminal cost, which leads to the formulation as follows,

---

NMPC-DCLF-DCBF:

$$J_t^*(\mathbf{x}_t) = \min_{U,\Omega,S} \sum_{k=0}^{N-1} q(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}) + \psi(\omega_k) + \phi(s_k) \tag{5.14a}$$

$$\text{s.t.} \quad \mathbf{x}_{t+k+1|t} = f(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}),\ k = 0, ..., N-1 \tag{5.14b}$$

$$\mathbf{u}_{t+k|t} \in \mathcal{U},\ \mathbf{x}_{t+k|t} \in \mathcal{X},\ k = 0, ..., N-1 \tag{5.14c}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_t, \tag{5.14d}$$

$$h(\mathbf{x}_{t+k+1|t}) \geq \omega_k(1-\gamma_k)h(\mathbf{x}_{t+k|t}),\ \omega_k \geq 0\ k = 0, ..., M_{\text{CBF}} - 1 \tag{5.14e}$$

$$V(\mathbf{x}_{t+k+1|t}) \leq (1-\alpha_k)V(\mathbf{x}_{t+k|t}) + s_k,\ k = 0, ..., M_{\text{CLF}} - 1 \tag{5.14f}$$

---

where M_{CLF} and M_{CBF} are the horizon length for CLF and CBF constraints respectively, which can be chosen to be less than the prediction horizon N. The slack variables $S = [s_1, ..., s_{M_{\text{CLF}}}]^T$ are introduced to avoid infeasibility between CLF and CBF constraints and additional term $\phi(s_k) \geq 0$ is added into the cost function to minimize those slack variables. This formulation is denoted as NMPC-DCLF-DCBF for later discussions.

Remark 5.5. *NMPC-DCLF-DCBF represents a generalized form for DCLF-DCBF with horizon lengths for cost and constraints. When the horizon lengths for CLF and CBF constraints equal to one, i.e.,* $M_{CLF} = M_{CBF} = 1$*, the NMPC-DCLF-DCBF becomes exactly as DCLF-DCBF except the decay rate of CBF constraint is relaxed. This relaxation is necessary as it enhance feasibility. Compared with NMPC-CLF in [70], NMPC-DCLF-DCBF represents an extended form by adding safety constraints.*

Remark 5.6. *On one hand, the proposed formulations* (5.12), (5.14) *have additional computational complexity generated due to the introduction of the optimization variables* ω_k*. On the other hand, reducing the horizon length for constraints (*M_{CBF} *and* M_{CLF}*) could reduce the computational complexity. The joint influence on computational complexity arising from the additional optimization variables and reduced constraint horizons for the optimization problem depends on the system dynamics and non-linearity of the control barrier functions. When the control barrier functions are nonlinear, the majority of non-linearity in the optimization comes from the CBF constraints, and therefore the reduction in complexity arising from the reduced constraint horizons would dominate the increase in complexity that arises from the introduction of additional optimization variables* ω_k*.*

5.4 Theoretical Analysis

In this section, we are going to illustrate theoretically the advantages about feasibility and safety of the proposed approaches. These advantages compared with the state-of-the-art are summarized in Table 5.1.

5.4.1 Theoretical Analysis of Feasibility

In this section, we are going to illustrate the enhancement of feasibility with reachability analysis by comparing MPC-DCBF and NMPC-DCBF.

For the MPC-DCBF formulation, the reachable set and safe region confined by the CBF constraint at each time are defined respectively as follows:

$$\begin{aligned}\mathcal{R}_k^{\text{MPC-DCBF}} = \{\mathbf{x}_{t+k|t} \in \mathbb{R}^n : \forall i = 0, ..., k-1, \mathbf{x}_{t+i+1|t} = f(\mathbf{x}_{t+i|t}, \mathbf{u}_{t+i|t}),\\ \mathbf{u}_{t+i|t} \in \mathcal{U}, \mathbf{x}_{t+k|t} \in \mathcal{X}, \mathbf{x}_{t|t} = \mathbf{x}_t\},\end{aligned} \tag{5.15}$$

$$\mathcal{S}_k^{\text{MPC-DCBF}} = \{\mathbf{x}_{t+k|t} \in \mathbb{R}^n : h(\mathbf{x}_{t+k|t}) - h(\mathbf{x}_{t+k-1|t}) \geq -\gamma_k h(\mathbf{x}_{t+k-1|t})\}. \tag{5.16}$$

The optimization of MPC-DCBF is feasible when the intersections between the reachable set $\mathcal{R}_k^{\text{MPC-DCBF}}$ and safe set $\mathcal{S}_k^{\text{MPC-DCBF}}$ at time $t+k$ are non-empty for all k. We denote the safe region at each time step as $\mathcal{S}_k^{\text{MPC-DCBF}}$, but notice that it also depends on the value of optimal value $\mathbf{x}_{t+k-1|t}$, which depends on the states and the inputs of previous nodes before the index $k-1$. $\mathcal{S}_k^{\text{MPC-DCBF}}$ could be rewritten as a function of reachable set at the one-step before,

$$\mathcal{S}_k^{\text{MPC-DCBF}} = \{\mathbf{x}_{t+k|t} \in \mathbb{R}^n : h(\mathbf{x}_{t+k|t}) \geq (1-\gamma_k) \inf_{\mathbf{x} \in \mathcal{R}_{k-1}^{\text{MPC-DCBF}}} h(\mathbf{x})\}, \tag{5.17}$$

as (5.16) leads to the following equation being valid

$$h(\mathbf{x}_{t+k|t}) \geq (1-\gamma_k) h(\mathbf{x}_{t+k-1|t}),$$

Table 5.1: Comparison among existing and proposed optimal control methods with respect to a variety of attributes.

Existing & Proposed Approaches	Optimization Structure				Performance	
	stability criteria	safety criteria	cost function	constraint(s)	feasibility	safety
DCLF-DCBF [2]	constraint	constraint	one-step	one-step	medium	**strong**
MPC-DCLF [70]	constraints	none	multi-step	one/multi-step	**high**	none
MPC-DCBF [224]	cost	constraints	multi-step	multi-step	low	**strong**
MPC-GCBF [124]	cost	constraint	multi-step	one-step	medium	weak
NMPC-DCBF (5.12)	cost	constraint(s)	multi-step	one/multi-step	**high**	**strong**
NMPC-DCLF-DCBF (5.14)	constraint(s)	constraint(s)	multi-step	one/multi-step	**high**	**strong**

with at least one value of $\mathbf{x}_{t+k-1|t}$.

For the NMPC-DCBF formulation, the reachable set is the same as MPC-DCBF as they share the same initial condition, system dynamics and input constraints, i.e.,

$$\mathcal{R}_k^{\text{NMPC-DCBF}} = \mathcal{R}_k^{\text{MPC-DCBF}}.$$

The corresponding safe region at each step is as follows,

$$\mathcal{S}_k^{\text{NMPC-DCBF}} = \{\mathbf{x}_{t+k|t} \in \mathbb{R}^n : \omega_k \geq 0, h(\mathbf{x}_{t+k|t}) \geq \omega_k(1-\gamma_k) \inf_{\mathbf{x}\in\mathcal{R}_{k-1}^{\text{NMPC-DCBF}}} h(\mathbf{x})\}. \tag{5.18}$$

As ω_k is an optimization variable with constraint $\omega_k \geq 0$, this leads us to rewrite $\mathcal{S}_k^{\text{NMPC-DCBF}}$ as follows,

$$\mathcal{S}_k^{\text{NMPC-DCBF}} = \{\mathbf{x}_{t+k|t} \in \mathbb{R}^n : h(\mathbf{x}_{t+k|t}) \geq 0\} = \mathcal{C}. \tag{5.19}$$

We can see that any value of γ_k for NMPC-DCBF won't affect the feasibility, as $\mathcal{S}_k^{\text{NMPC-DCBF}}$ equals the safe set $\mathcal{C}$ defined in (5.2), which is independent of γ_k. Hence, we have $\mathcal{S}_k^{\text{MPC-DCBF}}$ in (5.17) always as a subset of $\mathcal{S}_k^{\text{NMPC-DCBF}}$ in (5.19), as

$$\mathcal{S}_k^{\text{MPC-DCBF}} \subset \mathcal{S}_k^{\text{NMPC-DCBF}} = \mathcal{C} \tag{5.20}$$

which results in showing that the feasible regions of decision variable $\mathbf{x}_{t+k|t}$ is always larger when applying NMPC-DCBF approach compared to MPC-DCBF

$$\mathcal{R}_k^{\text{MPC-DCBF}} \cap \mathcal{S}_k^{\text{MPC-DCBF}} \subset \mathcal{R}_k^{\text{NMPC-DCBF}} \cap \mathcal{S}_k^{\text{NMPC-DCBF}}, \tag{5.21}$$

where feasibility region at step k is the intersection between reachable set $\mathcal{R}_k$ and safe set $\mathcal{S}_k$ for both approaches. To sum up, as the relaxing technique for decay rates is applied in NMPC-DCBF and NMPC-DCLF-DCBF, we can state they outperform MPC-DCBF / MPC-GCBF / DCLF-DCBF from the perspective of feasibility.

Remark 5.7. *In* (5.19)*, we see that* $\mathcal{S}_k^{NMPC\text{-}DCBF} = \mathcal{C}$*, which is the same as the region enforced by the distance constraint* (5.3)*. This reveals the NMPC-DCBF holds the same feasible region as MPC-DC [224]. Therefore, the feasible regions for NMPC-DCBF and NMPC-DCLF-DCBF are constant with respect to different hyperparameters* γ_k*, which will be shown in Figure 5.1, 5.2 and 5.3.*

5.4.2 Theoretical Analysis of Safety

The system safety can be influenced by many factors, including the safety function, the cost function, and other hyperparameters, etc. In this section, we focus on the influence from the hyperparameter γ_k and the additional cost function $\psi(\omega_k)$ for the slack variable ω_k of decay rate.

By reducing γ_k for MPC-DCBF / MPC-GCBF / DCLF-DCBF, the system safety will increase as the smaller γ_k represents a slower decay rate of lower bound of control barrier function, see (5.4). However, from (5.17), we can see that reducing γ_k for MPC-DCBF makes $\mathcal{S}_k^{\text{MPC-DCBF}}$ smaller, which leads the optimization more likely to be infeasible along the trajectory as the intersection between the reachable set and the region constrained by safety constraint decreases. This leads to a tradeoff between feasibility and safety. This tradeoff also happens among DCLF-DCBF and MPC-GCBF with similar reasons and forces us to choose either feasibility or safety for performance. However, in the case of our proposed NMPC-DCBF, the region confined by safety constraint won't be affected by changing the value of γ_k, as shown in (5.19). Hence, the intersection between the reachable set and the region constrained by safety constraint is independent of γ. This allows us to enhance the safety by reducing γ_k while not harming feasibility, which resolves the tradeoff between feasibility and safety.

The design of the additional cost function $\psi(\omega_k)$ for the decay-rate slack variable ω_k could also affect the safety performance. For example, the function $\psi(\omega_k)$ can be in the form as follows,

$$\psi(\omega_k) = P_\omega(\omega_k - 1)^2 \tag{5.22}$$

which keeps ω_k close to 1 and thus minimizes the deviation of the CBF constraint from the nominal decay rate of $1 - \gamma_k$. When the hyperparameter P_ω becomes larger, the optimized value of ω tends to be closer to 1, which implies the deviation from the nominal decay rate $1 - \gamma_k$ is smaller. Numerically, γ_k tends to be optimized as value smaller than 1 to increase the safe region (5.19) confined by CBF constraint at each time step. When $\omega_k = 0$, the relaxed CBF constraint becomes equivalent to a simple distance constraint and MPC with distance constraints (MPC-DC [224]) needs longer horizon to generate an expected obstacle avoidance performance in a closed-loop trajectory. Therefore, it's not recommended to set a relatively too small value for P_ω, which would over-relax the CBF constraint, i.e., the optimized value of ω_k could be too small.

To sum up, by reducing γ_k and utilizing an appropriate form of the additional cost function for decay-rate slack variable ω_k, the proposed approach would outperform the existing formulations in term of safety while not harming feasibility performance.

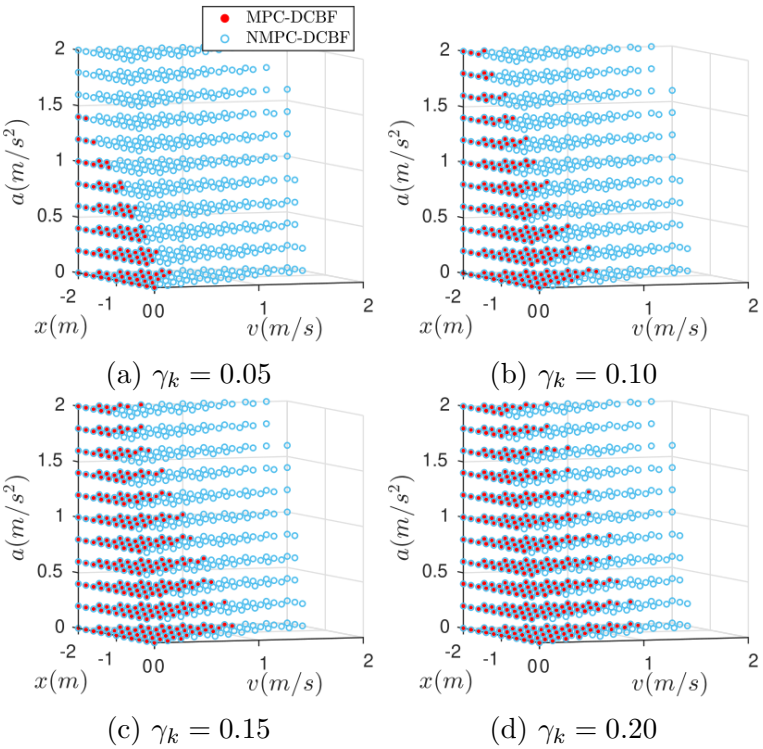
5.5 Numerical Examples & Results

In this section, we are going to show numerical results to illustrate the advantages of our proposed formulations with respect to the existing approaches. Consider the discrete-time linear triple-integrator system,

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \tag{5.23}$$

where $\mathbf{x} = [x, v, a]^T$ and $\mathbf{u} = [j]^T$ represent position (x), velocity (v), acceleration (a) and jerk (j), respectively. The admissible input set is $\mathcal{U} = \{j \in \mathbb{R} : j_{\min} \leq j \leq j_{\max}\}$.

For numerical simulations in this section, we set the sampling time as $\Delta t = 0.1s$ together with input lower and upper bounds as $j_{\min,\max} = -1m/s^3, 1m/s^3$. All simulations run in



(a) $\gamma_k = 0.05$ (b) $\gamma_k = 0.10$

(c) $\gamma_k = 0.15$ (d) $\gamma_k = 0.20$

Figure 5.1: Feasibility comparison with high-order CBF between MPC-DCBF and NMPC-DCBF with different values of decay rates.

MATLAB and the optimal control is formulated with Yalmip [120] as modelling language and solved with IPOPT [196].

5.5.1 Numerical Results for Feasibility

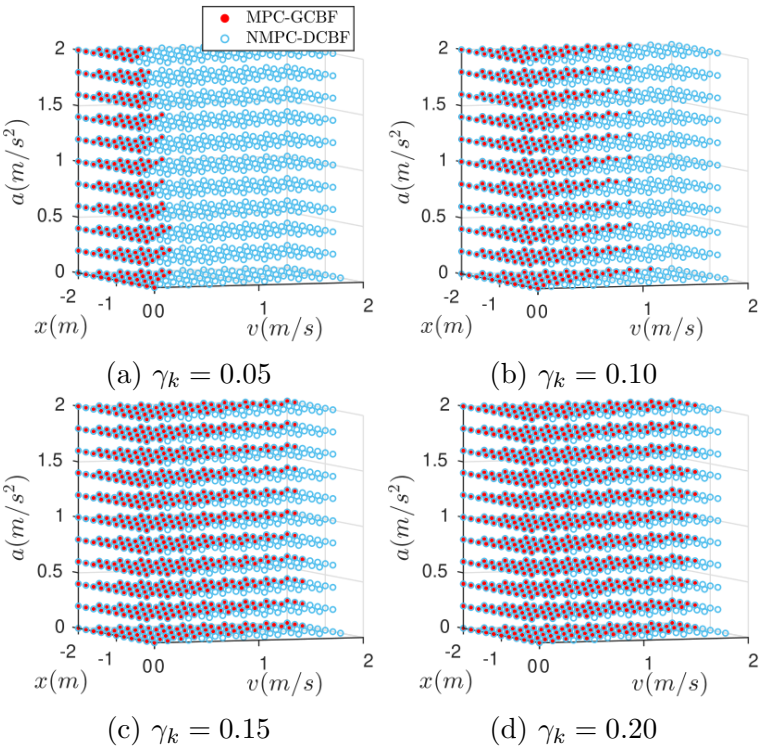
Our proposed formulations along with existing ones is compared by solving the optimization problems at all sampling states in a closed space. Precisely, we iterate over sampling states in the closed space $\mathcal{X}$ as

$$\mathcal{X} = \{(x, v, a) \in \mathbb{R}^3 \ : \ x_{\min} \leq x \leq x_{\max}, v_{\min} \leq v \leq v_{\max}, a_{\min} \leq a \leq a_{\max}\}$$

and run these optimal controllers to see whether the optimization problems are feasible at a given state $\mathbf{x}_t$. For simulation, we set $x_{\min,\max} = -2m, 0m$, $v_{\min,\max} = 0m/s, 2m/s$ and $a_{\min,\max} = 0m/s^2, 2m/s^2$. All the feasibility performance comparison is evaluated between approaches with the same horizon N and same form of stage cost and terminal cost.

To compare the feasibility performance among MPC-DCBF, MPC-GCBF and NMPC-DCBF, we choose a high-order control barrier function

$$h(\mathbf{x}) = -x, \tag{5.24}$$



(a) $\gamma_k = 0.05$ (b) $\gamma_k = 0.10$

(c) $\gamma_k = 0.15$ (d) $\gamma_k = 0.20$

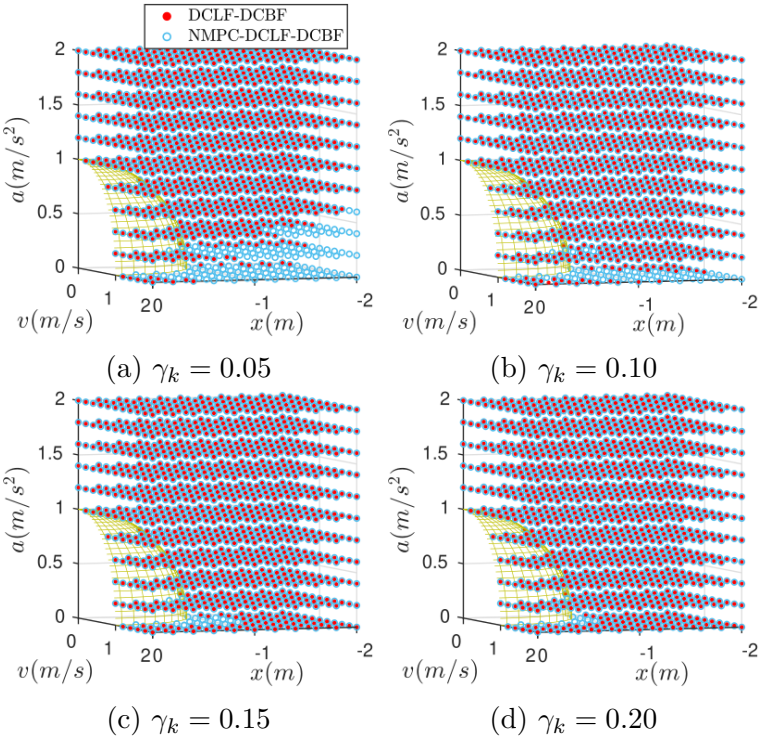
Figure 5.2: Feasibility comparison with high-order CBF between MPC-GCBF and NMPC-DCBF with different values of decay rates.

which enforces the system to stay on one side of the yz plane ($x \leq 0$). In Figs. 5.1, 5.2, results from NMPC-DCBF are compared with MPC-DCBF and MPC-GCBF for feasibility analysis. The comparisons are validated with different values of $\gamma_k = 0.05, 0.10, 0.15, 0.20$. For a reasonable comparison, the horizon length of CBF constraints is assumed as $M_{\text{CBF}} = 3$ for NMPC-DCBF to compare with MPC-GCBF, as the relative-degree of the CBF in (5.24) is 3 for a triple integrator system. The formulations of MPC-DCBF and MPC-GCBF are shown to enhance the feasibility with larger value of γ. MPC-GCBF does enhance the feasibility compared with MPC-DCBF as more states are feasible for MPC-GCBF with any value of γ_k. The feasibility of the proposed NMPC-DCBF is shown to consistently outperform MPC-DCBF and MPC-GCBF for any value of γ_k, where the feasible state region for MPC-DCBF or MPC-GCBF lies always inside the one for NMPC-DCBF. Additionally, the feasible state region of NMPC-DCBF is independent of the value of γ_k, shown in Figs. 5.1, 5.2, which verifies $\mathcal{S}_k^{\text{NMPC-DCBF}}$ is independent of γ_k as shown in (5.19).

To compare the feasibility performance between DCLF-DCBF and NMPC-DCLF-DCBF, we choose a relative-degree one control barrier function

$$h(\mathbf{x}) = x^2 + v^2 + a^2 - 1, \tag{5.25}$$

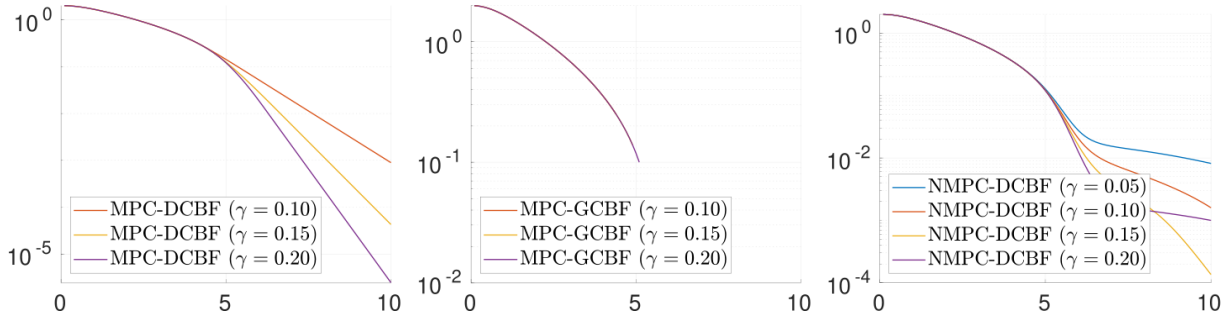
as the DCLF-DCBF method can only optimize one-step control input. The comparison result



(a) $\gamma_k = 0.05$ (b) $\gamma_k = 0.10$

(c) $\gamma_k = 0.15$ (d) $\gamma_k = 0.20$

Figure 5.3: Feasibility comparison with CBF between DCLF-DCBF and NMPC-DCLF-DCBF with different values of decay rates.



(a) MPC-DCBF ($N = 8$) (b) MPC-GCBF ($N = 8$) (c) NMPC-DCBF ($N = 8, M_{\text{CBF}} = 8$)

Figure 5.4: Evolution of control barrier function in the closed-loop trajectory by using controllers MPC-DCBF, MPC-GCBF and NMPC-DCBF with various decay rates.

is shown in Figure 5.3, where NMPC-DCLF-DCBF outperforms DCLF-DCBF in terms of feasibility for any values of γ_k. Similar to what we have seen previously, DCLF-DCBF enhances the feasibility with larger γ_k, while the feasible state region for NMPC-DCLF-DCBF is independent of γ_k. Notice that the unsafe states, which are inside the sphere region defined by (5.25), are excluded from the state sampling test for feasibility. The zero-level surface of the control barrier function is colored in yellow in Figure 5.3.

We also remark that the number of safety constraints for NMPC-DCBF and NMPC-DCLF-DCBF are larger than the ones for MPC-GCBF and DCLF-DCBF, but the feasibility performance are enhanced in our proposed approach, which demonstrates the importance of decay-rate relaxing technique that are introduced in the two proposed formulations. To sum up, the proposed formulations outperform the state-of-the-art in terms of feasibility.

5.5.2 Safety

The safety performance between controllers are compared numerically in this section. Given the same initial condition $\mathbf{x}_0 = [-2.0, 0.0, 1.0]^T$, we test each controller performance by using hyperparameters $\gamma_k = 0.05, 0.10, 0.15, 0.20$. The results for comparison among these approaches are shown in Figure 5.4. Among MPC-DCBF and NMPC-DCBF, it can seen that by reducing the value of γ_k, the value of CBF decreases slower, which implies a safer closed-loop trajectory. We notice that NMPC-DCBF is the only approach that maintains feasibility along the trajectory with $\gamma = 0.05$, while the other two approaches are infeasible right at the initial condition. Additionally, as illustrated in Figure 5.4b, MPC-GCBF becomes infeasible after around 5 seconds in a closed-loop trajectory starting from the initial condition. This arises from the fact that the one-step constraint doesn't sufficiently confine the system for safety and leads the system into an infeasible state after a while. We also notice that the control barrier function for NMPC-DCBF with $\gamma_k = 0.20$ is larger than $\gamma_k = 0.15$ after $t = 8s$. This happens due to numerical errors as after $t = 8s$, the CBF $h(\mathbf{x})$ is very close to zero and its derivative becomes almost zero and the solver tends to optimize the additional cost for relaxing decay-rate variable instead of the stage and terminal cost. Together with feasibility analysis in Sec. 5.5.1, we have shown that by reducing γ_k, NMPC-DCBF could enhance the safety of the closed-loop trajectory while not adversely affecting feasibility, which resolves the tradeoff between feasibility and safety.

5.6 Chapter Summary

In this chapter, we have proposed formulations to unify control Lyapunov function and control barrier functions under the framework of nonlinear model predictive control. Compared with previous work, the decay-rate of the CBF constraints are relaxed and different horizon lengths are considered for the cost function and the constraints. The proposed formulations are shown both theoretically and numerically to outperform the state-of-the-art from the perspective of feasibility and safety.

The proposed formulations in this chapter largely reduces the possibility of infeasibility and will be used for control and trajectory generations in later parts, especially for Chapters 7, 9 and 10.

Part II

Obstacle Avoidance for Control and Trajectory Generation

Chapter 6

Polytopic Obstacle Avoidance for Continuous-time Control Systems [4]

6.1 Introduction

Achieving safety-critical navigation for autonomous robots in an environment with obstacles is a vital problem in robotics research. Recently, control barrier functions (CBFs) together with quadratic program (QP) based optimizations have become a popular method to design safety-critical controllers. In this chapter, we propose a novel duality-based approach to formulate the obstacle avoidance problem between polytopes into QPs in the continuous domain using CBFs, which could then be deployed in real-time.

6.1.1 Related Work

6.1.1.1 Control Barrier Functions

One approach to provide safety guarantees for obstacle avoidance in control problems is to draw inspiration from control barrier functions. CBF-QPs [12] permit us to find the minimum deviation from a given feedback control input to guarantee safety. The method of CBFs can also be generalized for high-order systems [133, 214], discrete-time systems [2, 223, 7] and input-bounded systems [224, 3, 99, 28]. It must be noted that early work on nonovershooting control in [107] could also have been used to obtain results similar to control barrier functions. Specifically, CBFs are widely used for obstacle avoidance [222, 179, 83, 126, 74, 127] with a variety of applications for autonomous robots, including autonomous cars [38], aerial vehicles [210] and legged robots [81]. The shapes of robots and obstacles are usually approximated as points [38], paraboloids [52] or hyper-spheres [224], where the distance function can be calculated explicitly as an analytic expression from their geometric

[4]The material of this chapter is from "Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions" published in 2022 American Control Conference (ACC) [187]. This paper is was led by Akshay Thirugnanam.

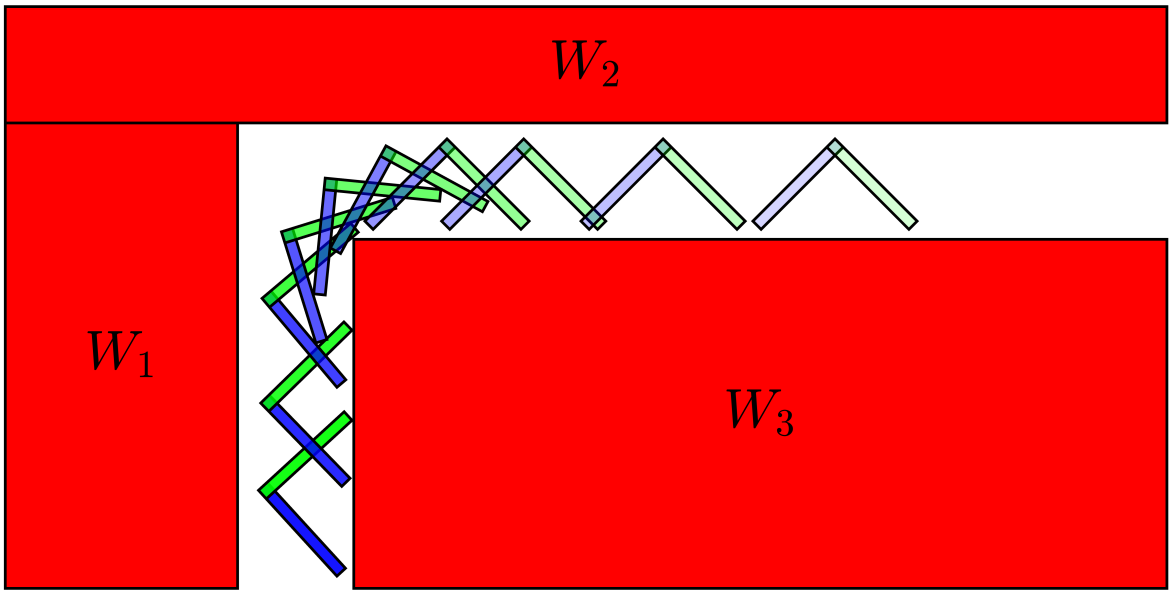


Figure 6.1: Snapshots of solving the moving sofa (piano) problem where the controlled object can maneuver through a tight corridor whose width is smaller than the diagonal length of the controlled object.

configuration. The distance functions for these shapes are differentiable and can be used as control barrier functions to construct a safety-critical optimal control problem.

However, these approximations usually over-estimate the dimensions of the robot and obstacles, e.g., a rectangle is approximated as the smallest circle that contains it. When a tight-fitting obstacle avoidance motion is expected, as shown in Figure 6.1, robots and obstacles are usually approximated as polytopes. While this makes maneuvers less conservative for obstacle avoidance, computing the distance between two polytopes requires additional effort [63]. Moreover, since this distance is not in an explicit form, it cannot be used directly as a CBF. Furthermore, the distance between polytopes is non-differentiable [1], which necessitates the use of nonsmooth control barrier functions (NCBFs) to guarantee safety [66], [65].

6.1.1.2 Obstacle Avoidance between Polytopes

We narrow our discussions about obstacle avoidance between polytopes into optimization-based approaches. In [110], obstacle avoidance between rectangle-shaped objects in an offline planning problem is studied, where collision avoidance is ensured by keeping all vertices of the controlled object outside the obstacle. Generally, when controlled objects are polyhedral, the collision avoidance constraints can be reformulated with integer variables [71]. This method applies well for linear systems using mixed-integer programming but cannot be deployed

as real-time controllers for general nonlinear systems due to the complexity arising from integer variables. The obstacle avoidance problem between convex regions could also be solved by using sequential programming [167], where penalizing collisions with a hinge loss is considered through an offline optimization problem.

Recently, a duality-based approach [227] was introduced to non-conservatively reformulate obstacle avoidance constraints as a set of smooth non-convex ones, which is validated on navigation problems in tight environments [225, 171, 64, 53]. This idea does optimize the computational time compared with other ideas, but nonlinear non-convex programming is still involved. Moreover, this approach can only be used for offline planning for nonlinear systems. This philosophy has been extended into discrete-time control barrier functions (DCBFs) to enforce obstacle avoidance constraints between polytopes in real-time [188]. However, the resulting DCBF formulation is still non-convex with non-convex DCBF or nonlinear system dynamics, and the computation time of the DCBF formulation might not be sufficiently low enough to be real-time. On the other hand, a continuous-time formulation can result in a convex optimization formulation even for nonlinear systems, which leads to faster computation times. Thus, continuous-time obstacle avoidance between polytopes requires a computationally efficient implementation, such as CBF-QPs and proper analysis on the nonsmooth nature of distance between polytopes to guarantee safety. To summarize, real-time obstacle avoidance between polytopes with convex programming for general nonlinear systems is still a challenging problem.

6.1.2 Contributions

The contributions of this chapter are as follows:

- We propose a novel approach to reformulate a minimization problem for obstacle avoidance between polytopes for nonlinear affine systems into a duality-based quadratic program with CBFs.
- We establish the obstacle avoidance algorithm for the minimum distance between polytopes under a QP-based control law that guarantees safety, where the nonsmooth nature of the minimum distance between polytopes is resolved in the dual space. This formulation is used for real-time safety-critical obstacle avoidance.
- Our proposed algorithm demonstrates real-time obstacle avoidance at 50 Hz in the *moving sofa (piano) problem* [80] with nonlinear dynamics, where an L-shaped controlled object can maneuver safely in a tight L-shaped corridor, whose width is less than the diagonal length of the controlled object.

6.2 Background

We consider N robots, with the i-th robot having states $x^i \in \mathcal{X} \subset \mathbb{R}^n$ and nonlinear, control affine dynamics:

$$\dot{x}^i(t) = f^i(x^i(t)) + g^i(x^i(t))u^i(t), \;\; i \in [N], \tag{6.1}$$

where, $u^i(t) \in \mathcal{U} \subset \mathbb{R}^m$, $f^i : \mathcal{X} \to \mathbb{R}^n$ and $g^i : \mathcal{X} \to \mathbb{R}^{n\times m}$ are continuous, and $[N] = \{1, ..., N\}$. We assume $\mathcal{X}$ to be a connected set and $\mathcal{U}$ a convex, compact set. While the dimensions of the states and inputs for each system can be different, we assume them to be the same across the systems for simplicity. Throughout the chapter, superscripts of variables denote the robot index and subscripts denote the row index of vectors or matrices.

6.2.1 Closed-loop Trajectory for Discontinuous Inputs

Polytopes have non-differentiable surfaces, and the minimum distance between polytopes could be non-differentiable at these points of non-differentiability. Hence, enforcing safety constraints with the nonsmooth distance could result in the loss of continuity property of the feedback control. Since f^i and g^i might not be Lipschitz continuous and $u^i(t)$ may be discontinuous, the solution of (6.1) need not be unique or even well-defined. In this case, to have a well-defined notion of a solution to (6.1), the dynamical system (6.1) is turned into a differential inclusion. Let $u^i : \mathcal{X} \to \mathcal{U}$ be some measurable feedback control law. A valid solution for the closed loop trajectory is defined via the Filippov map [65] as

$$\dot{x}^i(t) \in F[f^i + g^i u^i](x^i(t)) := \text{co}\{\lim_{k\to\infty} (f^i + g^i u^i)(x_{(k)}) : x_{(k)} \to x^i(t), x_{(k)} \notin \mathcal{Q}_f, \mathcal{Q}\} \tag{6.2}$$

where 'co' stands for convex hull, $x_{(k)}$ denotes the k-th element of the sequence $\{x_{(k)}\}$, $\mathcal{Q}_f$ is a system-dependent zero-measure set, and $\mathcal{Q}$ is any zero-measure set. The resulting map $F[f^i + g^i u^i] : \mathcal{X} \to 2^{\mathbb{R}^n}$ is a non-empty convex, compact, and upper semi-continuous set-valued map. Here $2^{\mathbb{R}^n}$ denotes the power set of $\mathbb{R}^n$. For a set-valued map $\Gamma : \mathcal{X} \to 2^{\mathbb{R}^n}$ to be upper semi-continuous at $a \in \mathcal{X}$, we require, $\forall\ \{a_{(m)}\} \in \mathcal{X}$ and $\{b_{(m)}\}$ such that $b_{(m)} \in \Gamma(a_{(m)})$, $\lim_{m\to\infty} a_{(m)} = a$ and $\lim_{m\to\infty} b_{(m)} = b \Rightarrow b \in \Gamma(a)$. Then for all $x_0^i \in \mathcal{X}$, a Filippov solution exists for the differential inclusion (6.2) with $x^i(0) = x_0^i$ [45, Prop. 3]. A Filippov solution on $[0, t]$ is an absolutely continuous map $x^i : [0, T] \to \mathcal{X}$ which satisfies (6.2) for almost all $t \in [0, T]$. Throughout the chapter, the term "almost all" means for all but on a set of measure zero.

6.2.2 Minimum Distance between Polytopes

For robot i, we define the polytope $\mathcal{P}^i(x^i)$ as the l-dimensional physical domain associated to the robot at state $x^i \in \mathcal{X}$ with

$$\mathcal{P}^i(x^i) := \{z \in \mathbb{R}^l : A^i(x^i)z \leq b^i(x^i)\}, \tag{6.3}$$

where $A^i : \mathcal{X} \to \mathbb{R}^{r^i \times l}$ and $b^i : \mathcal{X} \to \mathbb{R}^{r^i \times 1}$ represent the half spaces that define the geometry of robot i at some state, shown in Figure 6.2. We assume that the following properties hold for A^i and b^i:

- $\exists\ R, r > 0$ such that $\forall\ x^i \in \mathcal{X},\ \exists\ c \in \mathbb{R}^l$ such that $B_r(c) \subset \mathcal{P}^i(x^i) \subset B_R(c)$, where $B_r(c)$ represents the open ball with radius r centered at c.
- A^i, b^i are continuously differentiable, and $\forall\ x \in \mathcal{X}$, the set of inequalities $A^i(x)z \leq b^i(x)$ does not contain any redundant inequality.
- $\forall\ x \in \mathcal{X}$, the set of active constraints at any vertex of $\mathcal{P}^i(x)$ are linearly independent.

The first assumption requires that the geometry of the robot $\mathcal{P}^i(x)$ be uniformly bounded with a non-empty interior for all $x \in \mathcal{X}$. This assumption guarantees regularity conditions [20, Definition 2.3] are met for minimum distance computations, which in turn guarantee the essential conditions of continuity [20, Theorem 2.3] and differentiability [20, Theorem 2.4] of minimum distance. The last assumption requires that no more than l half spaces intersect at any vertex. As an example, a square pyramid in 3D does not satisfy this criterion. Any polytope that does not satisfy this assumption can be tessellated into smaller polytopes, such as tetrahedra.

The square of the minimum distance between $\mathcal{P}^i(x^i)$ and $\mathcal{P}^j(x^j)$ is defined as $h^{ij}(x^i, x^j)$, where $h^{ij} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be computed using the following QP:

$$\begin{aligned} h^{ij}(x^i, x^j) := \min_{\{z^i, z^j\}} \ & \|z^i - z^j\|_2^2 \\ \text{s.t.} \quad & A^i(x^i)z^i \leq b^i(x^i),\ A^j(x^j)z^j \leq b^j(x^j), z^i, z^j \in \mathbb{R}^l. \end{aligned} \tag{6.4}$$

Note that compared to the prior work on CBFs, the distance h^{ij} is implicit and is a solution of a minimization problem. By the regularity and smoothness assumptions on the polytopes, h^{ij} is locally Lipschitz continuous [103, Lemma 1]. Variables $z^i, z^j \in \mathbb{R}^l$ denote points inside the polytopes $\mathcal{P}^i(x^i)$ and $\mathcal{P}^j(x^j)$ respectively. Since the feasible sets of (6.4) are non-empty (by assumption), convex, and compact, the solution to the QP (6.4) always exists and is non-negative. When the robots intersect with each other the minimum distance is uniformly zero, and there is no measure of penetration between the polytopes.

6.2.3 Nonsmooth Control Barrier Functions

For obstacle avoidance, we want to design a controller such that the minimum distance between any pair of robots i and j should be strictly greater than 0. We define a safe set of states $\mathcal{S}^{ij}$ as the zero-superlevel set of the minimum distance between robots i and j, h^{ij} [12],

$$\mathcal{S}^{ij} := \{(x^i, x^j) : h^{ij}(x^i, x^j) > 0\}^c, \tag{6.5}$$

where $(\cdot)^c$ denotes closure of a set. Note that since $h^{ij}(x^i, x^j) = 0$ for intersecting robot geometries, we use closure to obtain the correct safe set. The closed-loop system is considered safe if $(x^i(t), x^j(t)) \in \mathcal{S}^{ij}\ \forall\ t \in [0, T]$, where T is time till which the solution is defined.

Let $u^i(x^i), u^j(x^j) \in \mathcal{U}$ be feedback control laws that are measurable, with corresponding Filippov solutions $x^i(t), x^j(t)$ for $t \in [0, T]$. Since h^{ij} is locally Lipschitz and the state trajectories are absolutely continuous, $h^{ij}(t) := h^{ij}(x^i(t), x^j(t))$ is an absolutely continuous function, and is thus differentiable at almost all $t \in [0, T]$.

Lemma 6.1. *[66, Lemma 2] Let $\alpha : \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz class-$\mathcal{K}$ function. If*

$$\dot{h}^{ij}(t) \geq -\alpha(h(t)) \tag{6.6}$$

for almost all $t \in [0, T]$ and $h(0) > 0$, then $h(t) > 0\ \forall\ t \in [0, T]$, making the system safe.

In this case the absolutely continuous function $h^{ij}(t)$ is called a Nonsmooth Control Barrier Function (NCBF), which is a generalization of Control Barrier Functions (CBFs) to nonsmooth functions. The constraint (6.6) is called the NCBF constraint. In the following section we derive a safety-critical feedback control law that satisfies this property.

Remark 6.1. *For simplicity of discussion, the later analysis will be illustrated for a pair of robots i and j, and results will be generalized where necessary. Further, for simplicity of notation, we denote $x := (x^i, x^j)$, $h(x) := h^{ij}(x^i, x^j)$, $\mathcal{S} := \mathcal{S}^{ij}$, and $u := (u^i, u^j)$ for the pair of systems i and j.*

6.3 NCBFs for Polytopes

In this section, we will illustrate a general approach for imposing NCBF constraints for polytopes. In order to enforce the NCBF constraints, we need to be able to write $\dot{h}(t)$ explicitly in terms of $\dot{A}^i(t), \dot{A}^j(t), \dot{b}^i(t), \dot{b}^j(t)$, which in turn depend on u. In the following sub-section, we attempt to construct an explicit formula for $\dot{h}(t)$.

6.3.1 Primal Approach

To explicitly compute $\dot{h}(t)$, we first need to show that it exists. Let $x(t), t{\in}[0, T]$ be the Filippov solution corresponding to a feedback control law $u(x)$. Let $\mathcal{O}(t)$ be the set of all optimal solutions to (6.4) at time t. For any pair of optimal solutions $(z^{*i}(t), z^{*j}(t)) \in \mathcal{O}(t)$, let $s^*(t) := z^{*i}(t) - z^{*j}(t)$ be the separating vector. $s^*(t)$ is fundamental for the distance formulation between polytopes since it is related to the minimum distance $h^{ij}(t) = \|s^*(t)\|_2^2$. We will next prove that $s^*(t)$ is unique, continuous, and right-differentiable.

Lemma 6.2. *For all $t{\in}[0, T]$, the separating vector $s^*(t)$ is unique for all pairs of primal optimal solutions $(z^{*i}(t), z^{*j}(t))$.*

Proof. The feasible set of (6.4) is convex and compact, and the cost function $\|z^i - z^j\|_2^2$ is convex. Projecting the feasible set onto the $z^i + z^j = 0$ surface, the resulting set is also convex and compact. Note that $s = z^i - z^j \in \mathbb{R}^l$ represents the projected co-ordinates on the $z^i + z^j = 0$ surface. The resulting cost function is $\|s\|_2^2$, is strictly convex, and therefore, the optimal solution $s^*(t)$ of the projected problem exists and is unique. Since for any pair of primal optimal solutions $(z^{*i}(t), z^{*j}(t))$, $z^{*i}(t) - z^{*j}(t)$ is an optimal solution to the projected problem, $z^{*i}(t) - z^{*j}(t)$ equals $s^*(t)$ for all pairs of optimal solutions of (6.4). □

For all $(z^{*i}(t), z^{*j}(t)) \in \mathcal{O}(t)$, we define $\text{Act}^i(t) \subset [r^i]$ ($\text{Act}^j(t) \subset [r^j]$) as the set of indices of constraints that are active for all $z^{*i}(t)$ ($z^{*j}(t)$) for $\mathcal{P}^i(t)$ ($\mathcal{P}^j(t)$). Then,

$$\begin{aligned} \text{Aff}^i(t) &:= \{z^i : A^i_{\text{Act}^i(t)}(t) z^i = b^i_{\text{Act}^i(t)}(t)\} \\ \text{Aff}^j(t) &:= \{z^j : A^j_{\text{Act}^j(t)}(t) z^j = b^j_{\text{Act}^j(t)}(t)\} \end{aligned} \tag{6.7}$$

represent two parallel affine spaces such that the minimum distance between robots i and j at t is the distance between $\text{Aff}^i(t)$ and $\text{Aff}^j(t)$. These affine spaces can be points, hyperplanes, or even the entire space if the two polytopes intersect. An example is pictorially depicted in Figure 6.2.

We now assume that for almost all $t \in [0, T]$, $\exists\ \epsilon > 0$ such that the $\dim(\text{Aff}^i(t))$, $\dim(\text{Aff}^j(t))$, and dimension of the othogonal subspace common between $\text{Aff}^i(t)$ and $\text{Aff}^j(t)$ are constant for $\tau \in [t, t + \epsilon)$. This is true when the set of times when states of the system oscillate infinitely fast has zero-measure. In practice the states of the system do not oscillate infinitely fast due to limited control frequency and inertia of the system. Then, under this assumption, we have the following result:

Lemma 6.3. *The separating vector $s^*(t)$ is continuous and right-differentiable for almost all $t \in [0, T]$.*

Proof. Let t be a time when the dimensions of the affine spaces and null space is constant for $\tau \in [t, t + \epsilon)$. The separating vector $s^*(\tau)$ is the unique vector from $\text{Aff}^i(\tau)$ to $\text{Aff}^j(\tau)$ that is perpendicular to both of them. These three constraints (that define $s^*(\tau)$ and establish orthogonality of $s^*(\tau)$ to $\text{Aff}^i(\tau)$ and $\text{Aff}^j(\tau)$) can be written in the form of a system of linear equalities with the matrix right-differentiable at $\tau = t$. By the assumption on constant dimensions of the above spaces ($\text{Aff}^i(t), \text{Aff}^j(t)$, and their common orthogonal subspace), this matrix has constant rank for $\tau \in [t, t + \epsilon)$. By Lemma 6.2, there is at least one solution to this system $\forall\ \tau \in [t, t + \epsilon)$. A right-differentiable solution to $s^*(\tau)$ can be found from this linear system using Gauss elimination. Since $s^*(t)$ is unique by Lemma 6.2, $s^*(t)$ must be right-differentiable, and thus continuous, at t. □

To impose the NCBF constraint we then compute $\dot{h}(t) = \lim_{\delta \to 0^+} 1/\delta(h(t + \delta) - h(t))$ as:

$$\dot{h}(t^+) = \frac{d}{d\tau} \|z^{*i}(\tau) - z^{*j}(\tau)\|_2^2 \Big|_{\tau = t^+}. \tag{6.8}$$

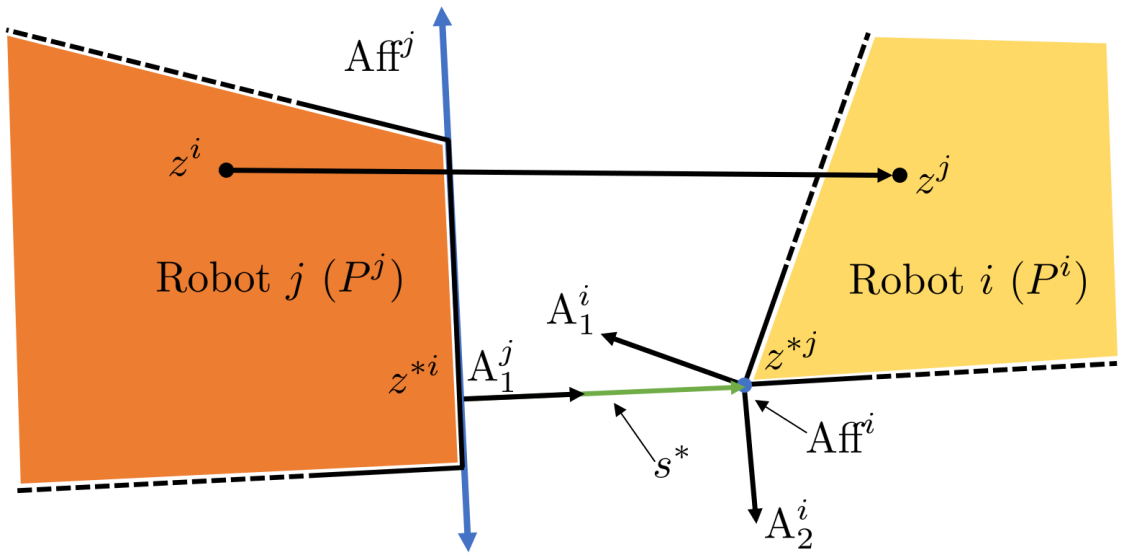


Figure 6.2: At any two configurations, the minimum distance between the robots i and j is the same as the minimum distance between the affine spaces Aff^i and Aff^j.

Although, $h(t) = \|s^*(t)\|_2^2$ is right-differentiable for almost all times, the primal optimal solutions $z^{*i}(t)$, $z^{*j}(t)$ could be non-differentiable or even discontinuous. So, $\dot{h}(t)$ cannot be written explicitly as a minimization problem from (6.8), since $\dot{z}^{*i}(t)$ and $\dot{z}^{*j}(t)$ need not be well-defined. This leads us to consider the dual formulation instead.

As a motivation, consider enforcing the collision avoidance constraint $\mathcal{P}^i \cap \mathcal{P}^j = \emptyset$ explicitly. Constraining the distance between any two points in $\mathcal{P}^i$ and $\mathcal{P}^j$ to be greater than 0 is not sufficient, since they may not be the closest points. This is due to the fact that (6.4) is a minimization problem. However, constraining a plane to separate $\mathcal{P}^i$ and $\mathcal{P}^j$ is sufficient to guarantee $\mathcal{P}^i \cap \mathcal{P}^j = \emptyset$ even if it is not the maximal separating plane. The dual formulation of (6.4) allows us to explicitly compute this separating plane constraint, which can be used in the NCBF constraint (6.4). The reason for using the dual formulation instead of the primal is further elaborated upon later in Remark 6.2.

6.3.2 Dual Formulation

The dual program of a minimization problem is a maximization problem in terms of the corresponding dual variables. For a quadratic optimization problem, as in (6.4), the dual program has the same optimal solution as that of (6.4). So, differentiating the dual program will give $\dot{h}(t)$ as a maximization problem. Since any feasible solution to a maximization problem gives us a lower bound of the optimum, we use the dual problem of (6.4) to get a lower bound of $\dot{h}(t)$. This enables us to express the NCBF constraint as a feasibility problem rather than an optimization problem.

To obtain the dual program, we first transform the constrained optimization problem (6.4) to an unconstrained one by adding the constraints to the cost with weights λ^i and λ^j, which are called the dual variables. The unconstrained problem is optimized in terms of z^i and z^j to obtain the Lagrangian function $L(\lambda^i, \lambda^j)$ as

$$L(\lambda^i, \lambda^j) = -\frac{1}{4}\lambda^i A^i(x^i) A^i(x^i)^T \lambda^{i\,T} - \lambda^i b^i(x^i) - \lambda^j b^j(x^j). \tag{6.9}$$

The dual program is then defined as the maximization of the Lagrangian function.

Lemma 6.4. *The dual program corresponding to* (6.4) *is:*

$$\begin{aligned} h(x) &= \max_{\{\lambda^i, \lambda^j\}} L(\lambda^i, \lambda^j) \\ s.t. \quad & \lambda^i A^i(x^i) + \lambda^j A^j(x^j) = 0,\ \lambda^i, \lambda^j \geq 0. \end{aligned} \tag{6.10}$$

Proof. The Lagrangian function for (6.4) is [27, Chap. 5]:

$$\begin{aligned} \Lambda(z^i, z^j, \lambda^i, \lambda^j) = & \lambda^j (A^j(x^j) z^j - b^j(x^j)) + \\ & \lambda^i (A^i(x^i) z^i - b^i(x^i)) + \|z^i - z^j\|^2, \end{aligned} \tag{6.11}$$

where $\lambda^i \in \mathbb{R}^{1 \times r^i}, \lambda^j \in \mathbb{R}^{1 \times r^j}$ are the dual variables. The Lagrangian dual function can be written as:

$$L(\lambda^i, \lambda^j) = \inf_{z^i, z^j \in \mathbb{R}^l} \left(\Lambda(z^i, z^j, \lambda^i, \lambda^j) \right), \tag{6.12}$$

and together with the Weak Duality Theorem [27, Chap. 5], we have,

$$L(\lambda^i, \lambda^j) \leq h(x), \ \ \forall\ \lambda^i, \lambda^j \geq 0. \tag{6.13}$$

As the constraints in (6.4) are affine in z^i and z^j at any given time, constraint qualification holds and the Strong Duality Theorem can be applied, resulting in

$$\max_{\lambda^i, \lambda^j \geq 0} L(\lambda^i, \lambda^j) = h(x). \tag{6.14}$$

$L(\lambda^i, \lambda^j)$ can be explicitly computed. Let the minimizer for (6.12) be z^i, z^j. Then,

$$\begin{aligned} \frac{\partial \Lambda}{\partial z^i} = 0 \Rightarrow 2(z^i - z^j)^T + \lambda^i A^i(x^i) = 0, \\ \frac{\partial \Lambda}{\partial z^j} = 0 \Rightarrow 2(z^j - z^i)^T + \lambda^j A^j(x^j) = 0. \end{aligned} \tag{6.15}$$

Hence, for the minimum of (6.12) to exist, from (6.15) the following must hold:

$$\begin{aligned} \lambda^i A^i(x^i) + \lambda^j A^j(x^j) = 0, \\ \lambda^i A^i(x^i) = -2(z^i - z^j)^T. \end{aligned} \tag{6.16}$$

Substituting the value of $(z^i - z^j)$ from (6.16) in (6.11) and using (6.12), we have

$$\begin{aligned} L(\lambda^i, \lambda^j) = &-\frac{1}{4}\lambda^i A^i(x^i) A^i(x^i)^T \lambda^{i\,T} - \lambda^i b^i(x^i) \\ &- \lambda^j b^j(x^j). \end{aligned} \tag{6.17}$$

The dual formulation (6.25) then follows from (6.16) and (6.17). □

Since an optimal solution to (6.4) always exists, an optimal solution to (6.10) also always exists. Let $(z^{*i}(t), z^{*j}(t)) \in \mathcal{O}(t)$ and the active set of constraints at $z^{*i}(t)$ be $\mathrm{Act}^i(z^{*i}(t), t)$. Linear independence constraint qualification (LICQ) is said to be held at $z^{*i}(t)$ if $A^i_{\mathrm{Act}^i(z^{*i}(t),t)}$ is full rank [26, Definition 2.1]. By the linear independence assumption on $\mathcal{P}^i(t)$, since the set of the active constraints at any vertex of $\mathcal{P}^i(t)$ are linearly independent, $A^i_{\mathrm{Act}^i(z^{*i}(t),t)}$ is also full-rank for all $z^{*i}(t)$ [26, Lemma 2.1]. Then, the primal problem (6.4) is considered non-degenerate and there exists a unique dual optimal solution for (6.10) [26, Lemma 2.2]. The dual optimal solution $(\lambda^{*i}(t), \lambda^{*j}(t))$ along with any primal optimal solution $(z^{*i}(t), z^{*j}(t))$ must then satisfy the KKT optimality conditions at t:

$$\begin{aligned} 2\lambda^{*i}(t) A^i(t) &= z^{*i}(t)^T - z^{*j}(t)^T = s^*(t)^T, \\ 2\lambda^{*j}(t) A^j(t) &= z^{*j}(t)^T - z^{*i}(t)^T = -s^*(t)^T, \\ \lambda^{*i}_k(t) &= 0 \quad \text{for } k \notin \mathrm{Act}^i(z^{*i}(t), t), \\ \lambda^{*j}_k(t) &= 0 \quad \text{for } k \notin \mathrm{Act}^j(z^{*j}(t), t). \end{aligned} \tag{6.18}$$

Then, by LICQ, $A^i_{\mathrm{Act}^i(t)}$ and $A^j_{\mathrm{Act}^j(t)}$ have full rank and the non-zero components of the dual optimal solutions at t can be written explicitly as:

$$\lambda^{*i}(t) = s^*(t)^T A^{i\,\dagger}_{\mathrm{Act}^i(t)}, \;\; \lambda^{*j}(t) = -\, s^*(t)^T A^{j\,\dagger}_{\mathrm{Act}^j(t)}, \tag{6.19}$$

where $(\cdot)^\dagger$ is the generalized inverse. By assumption $\mathrm{Aff}^i(\tau)$ and $\mathrm{Aff}^j(\tau)$ have full rank for $\tau \in [t, t+\epsilon)$ for almost all $t \in [0, T]$, and thus $\lambda^{*i}(t)$ and $\lambda^{*j}(t)$ are right-differentiable at almost all $t \in [0, T]$. Since $\lambda^{*i}(t), \lambda^{*j}(t), A^i(t)$ and $A^j(t)$ are right-differentiable, we can differentiate the constraints of (6.10). Finally, we can explicitly write a linear program, which is obtained by differentiating the cost and constraints of (6.10), to calculate $\dot{h}(t)$ as:

Lemma 6.5. *Let,*

$$\begin{aligned} g(t) = \max_{\{\dot\lambda^i, \dot\lambda^j\}} \; & \dot{L}(t, \lambda^{*i}(t), \lambda^{*j}(t), \dot\lambda^i, \dot\lambda^j) \\ s.t. \quad & \dot\lambda^i A^i(t) + \lambda^{*i}(t)\dot{A}^i(t) + \dot\lambda^j A^j(t) + \lambda^{*j}(t)\dot{A}^j(t) = 0, \\ & \dot\lambda^i_k \geq 0 \quad \textit{if } \lambda^{*i}(t)_k = 0, \\ & \dot\lambda^j_k \geq 0 \quad \textit{if } \lambda^{*j}(t)_k = 0. \end{aligned} \tag{6.20}$$

where $\dot{L}(t, \lambda^i, \lambda^j, \dot{\lambda}^i, \dot{\lambda}^j)$ *represents the time-derivative of Lagrangian function* $L(\lambda^i, \lambda^j)$ *and is as follows,*

$$\begin{aligned}\dot{L} = &-\frac{1}{2}\lambda^i A^i(t) A^i(t)^T \dot{\lambda}^{i\,T} - \frac{1}{2}\lambda^i A^i(t) \dot{A}^i(t)^T \lambda^{i\,T} \\ &- \dot{\lambda}^i b^i(t) - \lambda^i \dot{b}^i(t) - \dot{\lambda}^j b^j(t) - \lambda^j \dot{b}^j(t).\end{aligned} \tag{6.21}$$

Then, for almost all $t \in [0, T]$, $\dot{h}^{ij}(t) = g(t)$.

Proof. Since $(\lambda^{*i}(t), \lambda^{*j}(t))$ is the optimal solution to (6.10), its derivative $(\dot{\lambda}^{*i}(t), \dot{\lambda}^{*j}(t))$ is a feasible solution to (6.20) for almost all $t \in [0, T]$. So, $\dot{h}^{ij}(t) = \frac{d}{dt} L(\lambda^{*i}(t), \lambda^{*j}(t)) \leq g(t)$ for almost all $t \in [0, T]$. Let $(\dot{\lambda}^i, \dot{\lambda}^j)$ be a feasible solution to (6.20). We can integrate the constraints of (6.20) to find $(\bar{\lambda}^i(\tau), \bar{\lambda}^j(\tau)), \tau \in [t, t+\epsilon)$ which satisfy:

$$\begin{aligned}(\bar{\lambda}^i(t), \bar{\lambda}^j(t)) &= (\lambda^{*i}(t), \lambda^{*j}(t)) \\ (\bar{\lambda}^i(t), \bar{\lambda}^j(t)) &\text{ is dual feasible for (6.10)}\end{aligned} \tag{6.22}$$

So, $(\bar{\lambda}^i(\tau), \bar{\lambda}^j(\tau))$ are dual feasible and have cost less than $h^{ij}(\tau)$, i.e. $L(\bar{\lambda}(\tau), \bar{\lambda}(\tau)) \leq h^{ij}(\tau)$ $\forall \tau \in [t, t+\epsilon)$ and $L(\bar{\lambda}(t), \bar{\lambda}(t)) = h^{ij}(t)$. Differentiating the cost yields

$$\dot{h}^{ij}(t) \geq \dot{L}(t, \lambda^{*i}(t), \lambda^{*j}(t), \dot{\lambda}^i, \dot{\lambda}^j)$$

for all feasible $(\dot{\lambda}^i, \dot{\lambda}^j)$, and thus $\dot{h}^{ij}(t) \geq g(t)$. So, $g(t) = \dot{h}^{ij}(t)$ for almost all $t \in [0, T]$. □

Based on the linear program in (6.20), we can conservatively implement the NCBF constraint by enforcing, for some $(\dot{\lambda}^i, \dot{\lambda}^j)$ feasible for (6.20),

$$\dot{L}(t, \lambda^{*i}(t), \lambda^{*j}(t), \dot{\lambda}^i, \dot{\lambda}^j) \geq -\alpha(h(t)). \tag{6.23}$$

Lemma 6.5 then guarantees that

$$\dot{h}(t) \geq \dot{L}(t, \lambda^{*i}(t), \lambda^{*j}(t), \dot{\lambda}^i, \dot{\lambda}^j) \geq -\alpha(h(t)) \tag{6.24}$$

which is the required NCBF constraint.

Remark 6.2. *Due to the direction of inequality required for the NCBF constraint,* $\dot{h}(t)$ *needs to be expressed as a maximization problem. This is the primary motivation for considering the dual problem, since writing* $\dot{h}(t)$ *using the primal problem results in a minimization problem* (6.8).

We use (6.23) to motivate a feedback control law to guarantee safety of the system. The input u implicitly affects $\dot{L}$ via the derivatives of the boundary matrices A^i, A^j, b^i, b^j. Note that $\dot{L}$ is affine in $\dot{\lambda}^i$, $\dot{\lambda}^j$, and u. So, (6.23) is a linear constraint in $\dot{\lambda}^i$, $\dot{\lambda}^j$, and u. So, $\forall\, x \in \mathcal{S}$, (6.10) is used to compute $h(x)$, $\lambda^{*i}(x)$, and $\lambda^{*j}(x)$ and the optimal solution of the following quadratic program is used as the feedback control:

$$
\begin{aligned}
u^*(x) = &\underset{\{u,\dot\lambda^i,\dot\lambda^j\}}{\operatorname{argmin}} \|u-u^{nom}(x)\|_Q^2 && \text{(6.25a)}\\
\text{s.t.}\quad &\dot L(t,\lambda^{*i}(x),\lambda^{*j}(x),\dot\lambda^i,\dot\lambda^j,u)\geq -\alpha(h(x)-\epsilon_1^2) && \text{(6.25b)}\\
&\dot\lambda^i A^i(x)+\lambda^{*i}(x)(\mathcal{L}_{f^i}A^i(x)+\mathcal{L}_{g^i}A^i(x)u) && \text{(6.25c)}\\
&\quad +\lambda^{*j}(x)(\mathcal{L}_{f^j}A^j(x)+\mathcal{L}_{g^j}A^j(x)u)=-\dot\lambda^j A^j(x)\\
&\dot\lambda^i_k\geq 0 \text{ if } \lambda^{*i}(x)_k<\epsilon_2,\ \dot\lambda^j_k\geq 0 \text{ if } \lambda^{*j}(x)_k<\epsilon_2, && \text{(6.25d)}\\
&|\dot\lambda^i|\leq M, |\dot\lambda^j|\leq M, && \text{(6.25e)}\\
&u\in\mathcal{U}, && \text{(6.25f)}
\end{aligned}
$$

where $u^{nom}(x)$ is a non-safe nominal feedback control law, $\mathcal{L}_{(\star)}(\cdot)$ represents the Lie derivative of $(\cdot)$ along $(\star)$, $Q\succ 0$ is the cost matrix, M is a large number, and ϵ_1 and $\epsilon_2>0$ are small constants. $u^{nom}(x)$ can be obtained by a control Lyapunov function or by any tracking controller. Note that (6.25) is a feedback law which does not assume existence of Filippov solutions, since it is only a function of x and not t.

Remark 6.3. *Since $h(x)$ is quadratic in nature, its gradient (if it exists) can be zero at $\partial\mathcal{S}$, which can affect forward invariance of $\mathcal{S}$ [12, Remark 5]. For example, in 1D, let $\dot x = u$ and $h(x)=x^2$. Then at $x=0$, the NCBF constraint reduces to $0\cdot u\geq 0$, which is true $\forall\, u\in\mathcal{U}$. This would imply that the system remains safe irrespective of the input, which is not true. This problem can be solved by setting $h(x)=x^2-\epsilon_1^2$, which would result in non-zero gradient at $x=\epsilon_1$. So, we consider an $\epsilon_1>0$ and the ϵ_1^2-level set of h as*

$$\Omega_{\epsilon_1^2}=\{x : h(x)=\epsilon_1^2\}. \tag{6.26}$$

We can then redefine $h(x)$ as $(h(x)-\epsilon_1^2)$. If the new h satisfies (6.6)*, then the ϵ_1^2-superlevel set is safe.*

Remark 6.4. *The constant ϵ_2 is used to ensure upper semi-continuity of the feasible set of control inputs for* (6.25)*. This is similar to the almost-active gradient method used to prove safety in [65, Theorem 3].*

We now prove that (6.25) results in system safety.

Theorem 6.1. *Let $x(0)\in\mathcal{S}$ and* (6.25) *be feasible $\forall\, x\in\mathcal{S}$ for some locally Lipschitz class-$\mathcal{K}$ function α. Then, using the feedback control law* (6.25)*, the system remains safe irrespective of the cost function.*

Proof. Let the feasible set of control inputs for (6.25) be $\mathcal{F}_u(x)$. The control input $u^*(x)$ is feasible for (6.25), and hence satisfies (6.25b), which implies safety by (6.24). However, it shall be noted that since u^* need not be continuous, the Filippov operator (6.2) could

be required to obtain a valid closed loop trajectory. The control inputs obtained from the Filippov operation might not be feasible for (6.25), and we might lose the safety property.

The overview of the proof is as follows: We show that: (1) for any control law chosen from the feasible set of inputs $\mathcal{F}_u(x)$, the control input obtained after applying the Filippov operator is still feasible for (6.25), and (2) any control input feasible for (6.25) results in a safe trajectory.

Therefore, we first need to show that inputs obtained from the Filippov operator F is feasible for (6.25). We can prove this by showing $F[\mathcal{F}_u](x) = \mathcal{F}_u(x)$.

6.3.2.1 $F[\mathcal{F}_u](x) = \mathcal{F}_u(x)$

Recall, from the definition of the Filippov operator (6.2), that the Filippov operator applied on a set-valued function makes it closed and convex point-wise and upper semi-continuous. To show that $F[\mathcal{F}_u](x) = \mathcal{F}_u(x)$, we can equivalently show that $\mathcal{F}_u$ is closed and convex point-wise, and upper semi-continuous, thus making it invariant to the Filippov operator.

By assumption, (6.25) is always feasible. So, $\mathcal{F}_u(x) \neq \emptyset \ \forall \ x \in \mathcal{S}$. Further, since $\mathcal{U}$ is convex and compact, $\mathcal{F}_u(x)$ is convex and compact pointwise $\forall \ x \in \mathcal{S}$. Finally, we need to show that $\mathcal{F}_u(x)$ is upper semi-continuous. Consider any sequences $\{x_{(p)}\} \to x$ and $\{u_{(p)}\} \to u$, where $u_{(p)}$ are feasible for (6.25) at $x_{(p)}$, i.e. $u_{(p)} \in \mathcal{F}_u(x_{(p)})$. To show that $\mathcal{F}_u$ is upper semi-continuous, we have to show that u is feasible for (6.25) at x, i.e. $u \in \mathcal{F}_u(x)$. This is not trivial to show because the number of constraints in (6.25d) changes depending on x.

We now prove some general properties for optimization problems, using which we can conclude upper semi-continuity of $\mathcal{F}_u$. Consider an optimization problem O of the form $\rho^*(\xi) = \max_v\{\rho(\xi, v) : v \in \Pi(\xi, v)\}$, where $\Pi(\xi, v) = \{v : \pi_k(\xi, v) \leq 0, k \in K(\xi)\}$, where ρ is the continuous cost function, π is a continuous vector function, K is an index set, and the inequality is applied element-wise. Let $\Pi(\xi, v)$ be non-empty and uniformly bounded in ξ, and let $v^*(\xi)$ be an optimal solution at ξ. Note that the constraints enforced in the optimization can vary with ξ since the index set K depends on ξ. Here are the three properties as follows,

(a) If the index set is reduced, the optimal cost increases:

Define $\bar{\Pi}(\xi, v) = \{v : \pi_k(\xi, v) \leq 0, k \in \bar{K}\}$, where $\bar{K} \subseteq K(x) \ \forall \xi$. Define the corresponding optimization problem $\bar{O}$ as $\bar{\rho}^*(\xi) = \max_v\{\rho(\xi, v) : v \in \bar{\Pi}(\xi, v)\}$, with optimal solution $\bar{v}^*(\xi)$. Since $\Pi(\xi, v) \subseteq \bar{\Pi}(\xi, v)$, $\bar{\rho}^*(\xi) \geq \rho^*(\xi) \ \forall \xi$.

(b) Continuity property of optimal cost $\bar{\rho}^*$:

Consider a sequence $\{\xi_{(p)}\} \to \xi$ and optimal solutions $\{\bar{v}^*(\xi_{(p)})\} \to \bar{v}$. By continuity of π and ρ, $\pi_{\bar{K}}(\xi, \bar{v}) = \lim_{p\to\infty} \pi_{\bar{K}}(\xi_{(p)}, \bar{v}^*(\xi_{(p)})) \leq 0$, i.e. $(\xi, \bar{v})$ is a feasible solution for $\bar{O}$, and $\bar{\rho}^*(\xi) \geq \rho(\xi, \bar{v}) = \lim_{p\to\infty} \rho(\xi_{(p)}, \bar{v}^*(\xi_{(p)})) = \lim_{p\to\infty} \bar{\rho}^*(\xi_{(p)})$.

(c) Limit property of the index set, $K(\xi)$:

Let $K(\xi)$ be of the form $K(\xi) = \{k : \eta_k(\xi) < \epsilon\}$, where η is a continuous, non-negative vector function and $\epsilon > 0$. Consider a sequence $\{\xi_{(p)}\} \to \xi$. By continuity of η, $\exists P \in \mathbb{N}$ such that $\forall p \geq P$, and $\forall k \in K(\xi)$, $\eta_k(\xi_{(p)}) < \epsilon$. Thus, $K(\xi) \subseteq K(\xi_{(p)})\ \forall p \geq P$.

We now use the above properties to show upper semi-continuity of $\mathcal{F}_u$, which is the feasible set of inputs for (6.25). The variables ξ and v in the optimization problem O correspond to the pair (x, u) and $\dot{\lambda}$ in (6.25) respectively. Define the index set as $K(y) = \{(k^i, k^j) : \lambda^{*i}_k(y) < \epsilon_2, \lambda^{*j}_k(y) < \epsilon_2\} \subseteq [r^i] \times [r^j]$ corresponding to (6.25d). Note that λ^{*i} and λ^{*j}, corresponding to η, are continuous functions (Sec. 6.3.2).

Now consider the following optimization problem, denoted as $\mathrm{LP}(x, y, u)$ relating O to (6.25):

$$g(x,y,u) = \max_{\{\dot{\lambda}^i, \dot{\lambda}^j\}} \dot{L}(t, \lambda^{*i}(x), \lambda^{*j}(x), \dot{\lambda}^i, \dot{\lambda}^j, u) \tag{6.27a}$$

$$\text{s.t.} \quad \dot{\lambda}^i A^i(x) + \lambda^{*i}(x)(\mathcal{L}_{f^i} A^i(x) + \mathcal{L}_{g^i} A^i(x) u) \tag{6.27b}$$

$$+ \lambda^{*j}(x)(\mathcal{L}_{f^j} A^j(x) + \mathcal{L}_{g^j} A^j(x) u) = -\dot{\lambda}^j A^j(x)$$

$$\dot{\lambda}^i_k \geq 0 \quad \text{if } k \in K^i(y), \tag{6.27c}$$

$$\dot{\lambda}^j_k \geq 0 \quad \text{if } k \in K^j(y),$$

$$|\dot{\lambda}^i| \leq M, |\dot{\lambda}^j| \leq M. \tag{6.27d}$$

Note that the cost (6.27a) of $\mathrm{LP}(x, y, u)$ corresponds to the LHS of (6.25b), and the constraints (6.27b)-(6.27d) correspond to (6.25c)-(6.25e). The parameter y in $\mathrm{LP}(x, y, u)$ only affects the index set in (6.27c). Thus, u if feasible at x only if $g(x, x, u) \geq -\alpha(h(x) - \epsilon_1^2)$. Also, since $u_{(p)}$ is feasible at $x_{(p)}$, $g(x_{(p)}, x_{(p)}, u_{(p)}) \geq -\alpha(h(x_{(p)}) - \epsilon_1^2)$.

The optimization $\mathrm{LP}(x, x, u)$ corresponds to the optimization problem O as defined above. Note that by continuity of A^i, A^j, b^i, b^j, λ^{*i}, and λ^{*j}, the cost (6.27a) and constraint matrices (6.27b)-(6.27d) are continuous, similar to O. Moreover, the feasible set of (6.27) is non-empty (by the assumption in Theorem 6.1) and is uniformly bounded due to (6.27d). Thus, the optimization problem $\mathrm{LP}(x, x, u)$ fits the framework of O.

By Property (c), $\exists P \in \mathbb{N}$ such that $K(x) \subseteq K(x_{(p)})\ \forall p \geq P$. Truncate the sequences $\{x_{(p)}\}$ and $\{u_{(p)}\}$ before P. Define the optimization problem $\mathrm{LP}(x_{(p)}, x, u_{(p)})$ corresponding to $\bar{O}$. By Property (a), $g(x_{(p)}, x, u_{(p)}) \geq g(x_{(p)}, x_{(p)}, u_{(p)})\ \forall p$. Let the optimal solution to $\mathrm{LP}(x_{(p)}, x_{(p)}, u_{(p)})$ be $\dot{\lambda}_{(p)}$. Since $\dot{\lambda}_{(p)}$ is uniformly bounded (by (6.27d)), by Bolzano-Weierstrass Theorem, there exists a sub-sequence $\dot{\lambda}_{(q)}$ such that it converges to $\dot{\lambda}$. We now restrict the analysis to the q sequence. By Property (b) for $\bar{O}$, we have $g(x, x, u) \geq \lim_{q \to \infty} g(x_{(q)}, x, u_{(q)})$.

Combining all the inequalities from above, and by continuity of $h(x)$ (Lemma 6.3),

$$\begin{aligned} g(x,x,u) &\geq \lim_{q\to\infty} g(x_{(q)},x,u_{(q)}) \\ &\geq \limsup_{q\to\infty} g(x_{(q)},x_{(q)},u_{(q)}) \\ &\geq \limsup_{q\to\infty} -\alpha(h(x_{(q)})-\epsilon_1^2) = -\alpha(h(x)-\epsilon_1^2). \end{aligned}$$

Thus, u is feasible at x and $\mathcal{F}_u(x)$ is upper-semicontinuous.

6.3.2.2 Safety

Finally, we show that the Filippov operator does not affect the safety of the system. Let $u^*(x) = (u^{*i}(x), u^{*j}(x)) \in \mathcal{F}_u(x)$ be any measurable feedback control law, obtained as the solution of (6.25). Then, taking the Filippov operator,

$$F[u^*](x) \subset F[\mathcal{F}_u](x) = \mathcal{F}_u(x).$$

Given $x(0)$ such that $h(0) > \epsilon_1^2$, a closed loop trajectory $(x^i(t), x^j(t)), t \in [0,T]$ can be obtained using the Filippov solution, where $(x^i(t), x^j(t))$ are absolutely continuous in $[0,T]$ and for almost all $t \in [0,T]$, satisfy,

$$\begin{aligned} \dot{x}^i(t) &= f^i(x(t)) + g^i(x(t))u^i(t) \\ \dot{x}^j(t) &= f^j(x(t)) + g^j(x(t))u^j(t), \end{aligned}$$

where $u(t) = (u^i(t), u^j(t)) \in F[u^*](x(t)) \subset \mathcal{F}_u(x(t))$. Note that the constraints of (6.25) are stronger than those of (6.20), meaning that for any $(u, \dot{\lambda}^i, \dot{\lambda}^j)$ feasible for (6.25), $(\dot{\lambda}^i, \dot{\lambda}^j)$ is feasible for (6.20), and by (6.25b) and Lemma 6.5, $(\dot{\lambda}^i, \dot{\lambda}^j)$ satisfies (6.24). Since $u(t)$ is feasible for (6.25), $\exists\ (\dot{\lambda}^i(t), \dot{\lambda}^j(t))$ such that the closed loop trajectory satisfies (6.25b):

$$\dot{L}(t, \lambda^{*i}(t), \lambda^{*j}(t), \dot{\lambda}^i(t), \dot{\lambda}^j(t)) \geq -\bar{\alpha}(h(t)-\epsilon_1^2),$$

for almost all $t \in [0,T]$. By (6.24), $\dot{h}(t) \geq -\alpha(h(t)-\epsilon_1^2)$ for almost all $t \in [0,T]$. By Lemma 6.1, $h(t) > \epsilon_1^2\ \forall\ t \in [0,T]$, and thus the system remains safe. □

Remark 6.5. *This formulation can be extended to more than 2 robots by introducing a new pair of dual variables for each pair of robots, and the corresponding constraints in* (6.25)*. Note that the dual variable for robot i corresponding to robot j is different from that for robot k. Thus, the analysis in Sec. 6.3 remains valid. Additionally, static or dynamic obstacles can be represented as uncontrolled robots and non-convex shaped robots can be represented through unions of different polytopes with each using the same states and inputs.*

6.4 Numerical Examples

In this section, we consider the problem of moving an L-shaped sofa through an L-shaped corridor. The approach presented in Sec. 6.3 is applied to solve the problem in real-time.

6.4.1 Simulation Setup

The sofa is the controlled object, with the side length of the L-shape as 1 m and the width as 0.1 m, as illustrated in Figure 6.1. To implement our controller, we consider the sofa as the union of two 1 m $\times$ 0.1 m arms perpendicular to each other. The width of the corridor is 1 m.

The states of the sofa are $x = (z_1, z_2, \theta) \in \mathbb{R}^2 \times \mathbb{R}$, where (z_1, z_2) is the position of the vertex at the intersection of the two arms, and θ is the angle of rotation. The inputs to the sofa are (v, ω), where v is the speed of the sofa and ω the angular velocity. The velocity of the sofa is assumed to be along a $\frac{\pi}{4}$ angle to the arm. The nonlinear control affine dynamics of the sofa is then,

$$\dot{z}_1 = v\cos(\theta + \frac{\pi}{4}),\ \dot{z}_2 = v\sin(\theta + \frac{\pi}{4}),\ \dot{\theta} = \omega. \tag{6.28}$$

Further, we impose input bounds as $|v| \leq 0.3\,\text{m/s}$ and $|\omega| \leq 0.2\,\text{rad/s}$. The corridor is represented as the union of three rectangular walls, shown in Figure 6.1. The polytopes corresponding to these walls are defined as follows:

$$\mathcal{W}^i := \{z \in \mathbb{R}^2 : A^{W^i} z \leq b^{W^i}\},\ i \in [3] \tag{6.29}$$

We note the general problem description in Sec. 7.2 allows for such static obstacles by eliminating the dependence of A^{W^i} and b^{W^i} on the state, as mentioned in Remark 6.5.

Similarly, the sofa is represented as:

$$\mathcal{P}^j(x) := \{z \in \mathbb{R}^2 : A^{S^j}(x) z \leq b^{S^j}(x)\},\ j \in [2], \tag{6.30}$$

where $\mathcal{P}^j(x)$ is denoted as Arm j. Arm 1 is the blue-colored polytope in Figure 6.1, whereas Arm 2 is the green-colored one. Again, our description allows for this by choosing the same states and inputs for the two polytopes.

The NCBF is chosen as the square of the minimum distance as in (6.4). Notice that the QP-based program (6.25) will always have a solution, as $v = 0, \omega = 0$ is always a feasible solution. Since we have static obstacles and the controlled object consists of two polytopes, we only enforce NCBF constraints between $\mathcal{W}^i$ and $\mathcal{P}^j$ for $i \in [3], j \in [2]$.

A control Lyapunov function V_{clf} [12] is introduced in the final QP formulation in place of the nominal controller with the form as follows:

$$V_{clf}(x) = (z_1 - z_1^d)^2 + (z_2 - z_2^d)^2 + k(\theta - \theta^d)^2, \tag{6.31}$$

where $\theta_d = -\frac{\pi}{4}$. The desired position (z_1^d, z_2^d) is chosen to be at the end of the corridor, and the initial orientation is $\frac{\pi}{4}$. The CLF constraint $\dot{V}_{clf} \geq -\alpha_1 V_{clf} - s$ is then enforced, where $s \geq 0$ is a slack variable. The slack variable ensures that the feasibility of the QP-based program is not affected by the CLF constraint. The margin ϵ_1 as defined in Remark 6.3 is chosen as 1.5 cm and ϵ_2 is chosen as 10^{-5}.

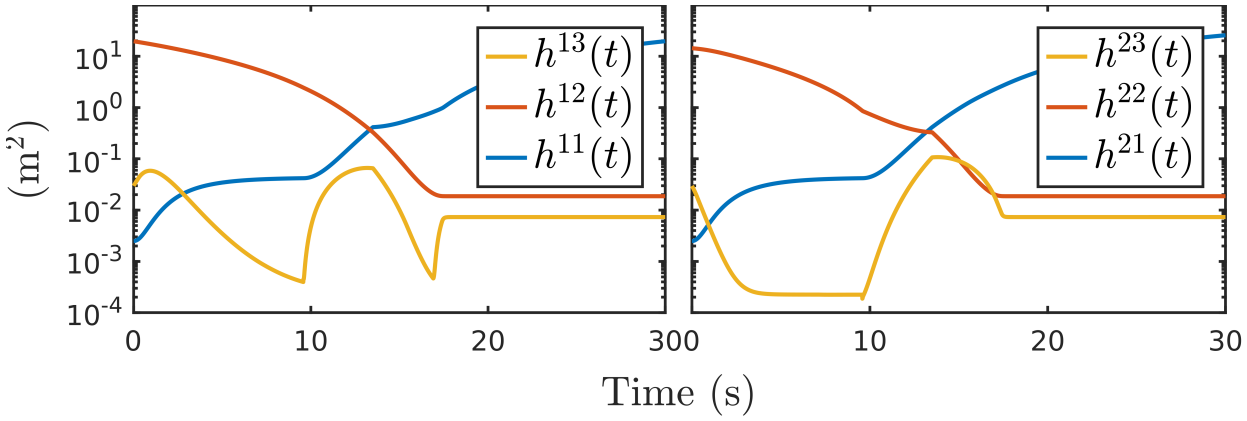


Figure 6.3: Square of minimum distance (NCBF) between the arms of the sofa and the walls.

6.4.2 Results

The simulations are performed on a Virtual Machine with 4 cores of a 2.20 Ghz Intel Core i7 processor, running IPOPT [196] on MATLAB, and the visualization is generated by MPT3 [77]. The snapshots of the sofa trajectory is shown in Figure 6.1 and also illustrated in the multimedia attachment.

6.4.2.1 Enforcement of NCBF constraint

The system remains safe throughout the simulation since the NCBF h^{ij} is greater than ϵ_1^2 for all possible robot-obstacle interactions as depicted in Figure 6.3. In Figure 6.4, the values of $\dot{h}(t)$, and LHS and RHS terms in the NCBF constraints from (6.25) are shown. We can see that by Lemma 6.5, the LHS term in the NCBF constraint is always a lower bound to $\dot{h}(t)$. This verifies the safety property of our duality-based approach.

6.4.2.2 Computation time

Table 6.1: Statistical analysis of computation time (ms) per iteration

Timing (ms)	mean ± std	p50	p99	max
Distance QPs (6.10)	1.07 ± 0.29	0.99	1.94	20.1
Polytope-NCBF-QP (6.25)	14.5 ± 1.55	14.2	19.3	37.6
Total ($2 \times 3 + 1 = 7$ QPs)	21.1 ± 1.69	20.8	26.9	41.6

From Table 6.1, we can see that there are large outliers in computation time per iteration, but they occur in less than 0.1% iterations. Nevertheless, from Table 6.1, we can apply our

controller at 50Hz. The fast computation time allows us to directly implement the control inputs from (6.25) on the robot in real-time.

The optimization problem for the sofa problem has 51 variables and at most 72 constraints. In general consider N controlled robots in a d-dimensional space with f facets and m control inputs. Then, the polytope-NCBF-QP (6.25) has $(1+d)\frac{N(N-1)}{2}$ constraints along with at most $2f\frac{N(N-1)}{2}$ non-negativity constraints and $N(m+2f)$ variables, whereas a CBF-QP formulation using spherical over-approximation would have $\frac{N(N-1)}{2}$ constraints and Nm variables.

6.4.2.3 Continuity of derivative of control barrier function and dual variables

Figure 6.4 also shows that both $\dot{h}(t)$ and the lower bound of $\dot{h}(t)$ can have discontinuities. For the case of the moving sofa problem, a discontinuity in $\dot{h}(t)$ can arise when the sofa is rotating and the point of minimum distance on the sofa with any wall jumps from one vertex to another, since the end points of the sofa arm need not have the same velocity when rotating. The dual optimal variables $\lambda^{*i}(t)$ and $\lambda^{*j}(t)$ are always continuous and right-differentiable as shown in Sec. 6.3. The dual variables are plotted in Figure 6.5, which demonstrate this property.

6.4.2.4 Deadlocks

For various initial conditions, the sofa can get stuck in a deadlock in the corridor. This can happen when the arms of the sofa are so large that it cannot turn at the corner. It can also happen when the two arms of the sofa get too close to the wall and the sofa cannot turn because it would cause one of its arms to penetrate the wall. Our controller still ensures safety in this case. A high-level planner could help by generating a deadlock free trajectory at low frequency that then serves as input to our control law.

6.4.3 Discussions

6.4.3.1 Nonlinearity of the system dynamics

The duality-based formulation (6.25) is a convex quadratic program even when the system dynamics is nonlinear, as long as it is control affine. This allows us to achieve dynamically-feasible obstacle avoidance with QPs for polytopes.

6.4.3.2 Optimality of solution vs computation time

As noted in Sec. 6.3, the cost function of the QP-based program (6.25) does not affect the safety of the system. If the optimization solver does not converge to the optimal solution, the current solution can be used if it is feasible. This can be useful in real-time implementations where both control frequency and safety matter. A feasible solution to (6.25) can be directly

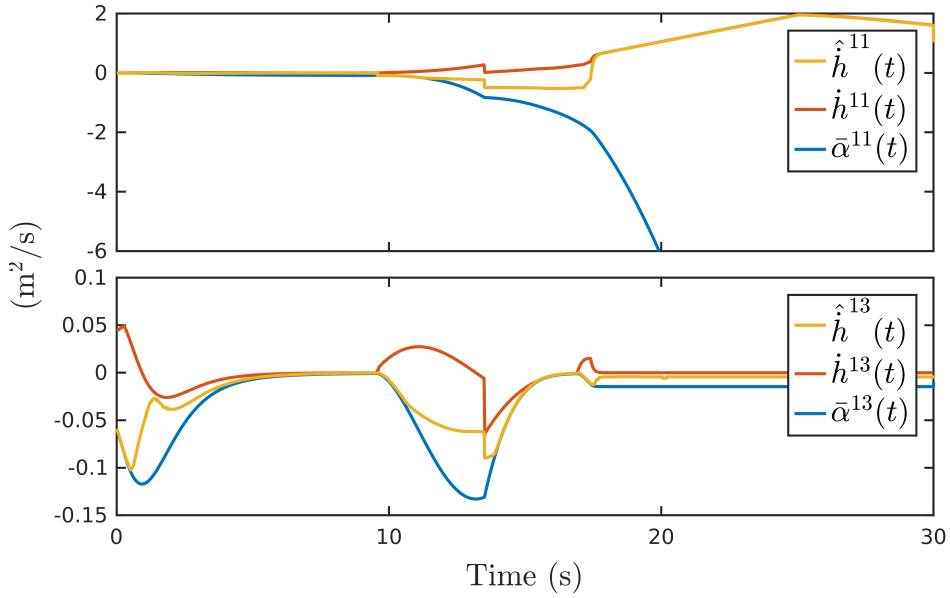


Figure 6.4: NCBF constraint enforcement between Arm 1 and Wall 1 (top figure) and Arm 1 and Wall 3 (bottom figure).

found in some cases, such as when the NCBF is constructed using a safe backup controller [178].

6.4.3.3 Trade-off between computation speed and tight maneuvering

A polytope with f^i faces would require f^i dual variables in the dual formulation (6.25) and additional constraints. Using a hyper-sphere as an over-approximation to the polytope would require fewer dual variables, as in [214]. However, such an approximation can be too conservative and completely ignore the rotational geometry of the polytope. Using the full polytope structure can prove beneficial in dense environments, such as the sofa problem, where a spherical approximation cannot work. So, there is a trade-off between computation speed and maneuverability. In practice, a hybrid approach should be used: a hyper-sphere approximation when two obstacles are far away, and the polytope structure when closer, which could find a good trade-off between computation speed and maneuverability for tight obstacle avoidance in dense environments.

6.4.3.4 Robustness with respect to the safe set

Since we use the minimum distance between two polytopes as the NCBF and not the signed distance, the minimum distance is uniformly zero when two polytopes intersect. So, the proposed controller is not robust in the sense that if the state leaves the safe set, it will not converge back to it.

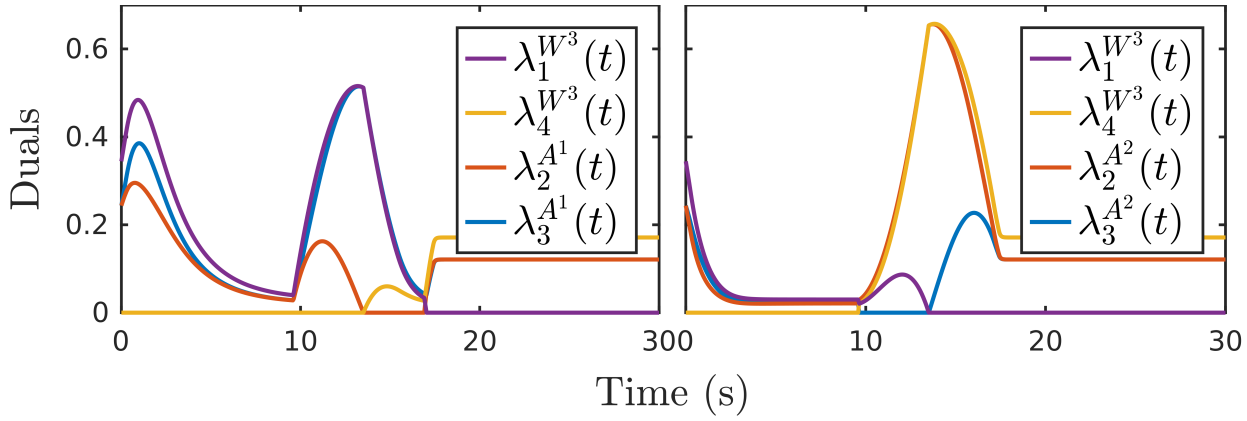


Figure 6.5: Dual optimal solutions for Arm 1 (left figure) and Arm 2 (right figure) of the robot corresponding to the obstacle wall 3.

6.5 Chapter Summary

In this chapter, we have presented a general framework for obstacle avoidance between polytopes using the dual program of a minimum distance QP problem. We have shown that the control input using our method can be computed using a QP for systems with control affine dynamics, enabling real-time implementation. We have numerically verified the safety performance of our controller for the problem of moving an L-shaped sofa through a tight corridor. We also have explored properties such as robustness and deadlock avoidance. Further work in this topic would involve extending our method to other convex shapes, and deriving safety guarantees for more general classes of systems.

In this chapter, we focused on the dual optimization for polytopic obstacle avoidance with control barrier functions in the continuous-time domain. The discrete-time domain formulations will be discussed in the next chapter.

Chapter 7

Polytopic Avoidance for Discrete-time Control Systems [5]

7.1 Introduction

Obstacle avoidance in optimization-based control and trajectory planning has received significant attention in the robotics community. When a tight-fitting obstacle avoidance motion is expected, the robot and the obstacles need to be considered as polyhedral. In this chapter, we propose an optimization formulation to consider obstacle avoidance between polytopes using discrete-time control barrier function (DCBF) constraints with dual variables. The proposed formulation is shown to be a computationally fast algorithm that can serve as a local planner to generate dynamically-feasible and collision-free trajectories, or even directly as a safety-critical controller for general dynamical systems.

7.1.1 Related Work

7.1.1.1 Graph Search-based and Sampling-based Approaches

Motion planning techniques in real-world applications often consider high-level path planning and low-level control synthesis, given safety requirements and dynamical constraints. Graph search-based and sampling-based approaches such as PRM [24], A* [49], RRT* [87] have been explored, and many variant approaches have also been proposed based on them, which could be applied as efficient strategies for high-dimensional kinematic planning. However, generally, these algorithms assume that a low-level controller exists, and is able to track kinematically feasible trajectories in real time. This leads to trajectories that are dynamically infeasible and results in large tracking errors on dynamical systems. Other approaches such as kinodynamic RRT* [200], LQR-RRT* [145] try to bridge the gap between path planning

[5]The material of this chapter is from "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions" published in 2022 IEEE International Conference on Robotics and Automation (ICRA) [188].

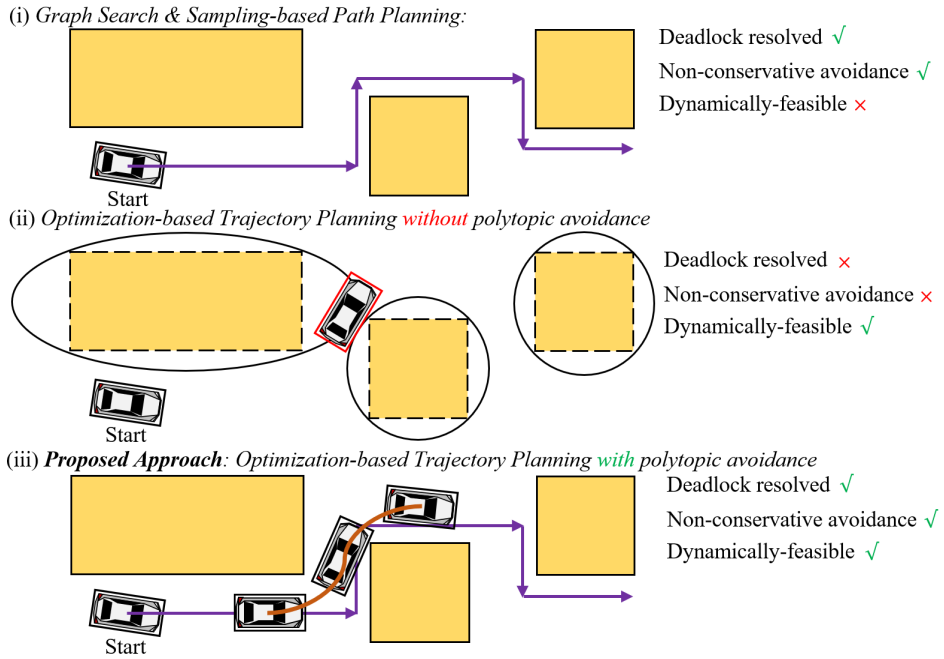


Figure 7.1: Comparison of various approaches for trajectory planning with polytopic obstacle avoidance.

and control synthesis by finding appropriate steering inputs to go between two vertices in the sampling graph. However, these approaches cannot do dynamic collision checking with respect to the exact nonlinear dynamics of the robot. For general dynamical systems, we still need to locally generate a dynamically feasible and collision-free trajectory.

7.1.1.2 Optimization-based Control and Trajectory Planning

We now narrow down our discussion to optimization-based approaches for generating collision-free trajectories. The existing methods in this sub-area can be classified under two categories: those that generate obstacle avoidance behaviors with additional cost terms, and those that apply constraints to achieve a similar behavior. Additional cost terms were first introduced under the philosophy of potential fields [85], and were later generalized to be named as "barrier function" [206]. This approach has been applied to solve optimal control and trajectory generation problems with broad applications [167, 117, 158, 139, 212]. Other methods consider the obstacle avoidance criteria as constraints in the optimization problem. An example of such a constraint is the distance constraint, enforced using inequality constraints on the distance function between the robot and obstacles, where the robots and the obstacles are usually approximated as points [142], lines [23], paraboloids [52], ellipsoids [162], or hyperspheres [224]. The distance functions for these shapes have analytical expressions and are differentiable so that nonlinear optimization (NLP) solvers can easily compute the gradients.

7.1.1.3 Obstacle Avoidance between Polytopes

When a tight-fitting obstacle avoidance motion is expected, the above over-approximations of the shape of the robot can lead to deadlock maneuvers, shown in Figure 4.1. A tight polytopic approximation of the shape of the robot enables obstacle avoidance maneuvers that are less conservative, see [101]. However, the distance function between two polytopes is implicit and not analytic [71], and requires a large amount of numerical computation [62, 47]. Moreover, this distance function between polytopes is also non-differentiable with respect to the robot's configuration, which makes it hard to be treated as a constraint directly for a nonlinear programming problem. For collision avoidance between two polytopes, mixed-integer programming [71, 154, 48, 213] applies well for linear systems but cannot be deployed as a real-time controller or trajectory planner for general nonlinear systems due to the added complexity from integer variables [119]. To handle the non-differentiability of the distance function between two polytopes, a duality-based approach [228] is introduced to reformulate constraints as a set of smooth non-convex ones. However, obstacle avoidance behavior with this method can only be achieved with a relatively long horizon and needs to be solved offline for nonlinear systems [227, 225, 171, 54, 64]. Recently, a dual optimization formulation [187] was introduced to construct a differentiable control barrier function (CBF) [12] for polytopes, but it only optimizes one-step control input and is only applicable for continuous-time affine systems with a relative-degree of one. This formulation could also run into a deadlock for general high relative-degree systems.

7.1.1.4 Obstacle Avoidance with DCBFs

To resolve the problems mentioned above, it's required to propose a computationally fast multi-step optimization formulation for systems with nonlinear discrete-time dynamics. Recently, it has been shown that considering discrete-time control barrier function (DCBF) constraints instead of distance constraints can handle this challenge, where the DCBF constraints can regulate the obstacle avoidance behavior with a smaller horizon and prevent local deadlock in trajectory generation, see [223]. The control and planning problems with one-step [2] or multi-step [223] optimization using DCBF constraints have been studied, and various applications on different platforms, including car racing [224], autonomous vehicles [125], and bipedal robots [186] have validated this approach. In the work mentioned above, the robots and the obstacles are only considered as points or hyper-spheres, while obstacle avoidance constraint between polytopes is still an unsolved problem in all previous work by using discrete-time control barrier functions.

7.1.2 Contributions

The contributions of this chapter are as follows:

- We formulate the dual form of the obstacle avoidance constraint between polytopes as DCBF constraints for safety. These proposed DCBF constraints are incorporated into

an NMPC formulation which enables fast online computation for control and planning for general nonlinear dynamical systems.

- The proposed NMPC-DCBF formulation for polytopes is validated numerically. Different convex and non-convex shaped robots are shown to be able to navigate with tight maneuvers through maze environments with polytopic obstacles using fast real-time control and trajectory generation.

7.2 Background

In this section, we present a brief background on optimization formulations using discrete-time control barrier functions and obstacle avoidance between polytopic sets.

7.2.1 Optimization Formulation using DCBFs

Consider a discrete-time dynamical system with states $x \in \mathcal{X} \subset \mathbb{R}^n$ and inputs $u \in \mathcal{U} \subset \mathbb{R}^m$, as

$$x_{k+1} = f(x_k, u_k), \tag{7.1}$$

where $x_k := x(k)$, $u_k := u(k)$, $k \in \mathbb{Z}^+$, $\mathcal{U}$ is a compact set and f is continuous.

7.2.1.1 Discrete-time CBFs

Obstacle avoidance for safety for this dynamical system is defined in terms of invariance of its trajectories with respect to a connected set. In other words, if the dynamical system (7.1) is safe with respect to a set $\mathcal{S} \subset \mathcal{X}$, then any trajectory starting inside $\mathcal{S}$ remains inside $\mathcal{S}$. The set $\mathcal{S}$ is defined as the 0-superlevel set of a continuous function $h : \mathcal{X} \to \mathbb{R}$ as:

$$\mathcal{S} := \{x \in \mathcal{X} \subset \mathbb{R}^n : h(x) \geq 0\}. \tag{7.2}$$

We refer to $\mathcal{S}$ as the safe set and it represents the region outside the obstacle. h is defined as a discrete-time control barrier function (DCBF) if $\forall\, x \in \mathcal{S}, \exists\, u \in \mathcal{U}$ such that

$$h(f(x,u)) \geq \gamma(x)h(x), \quad 0 \leq \gamma(x) < 1, \tag{7.3}$$

Let $\gamma_k := \gamma(x_k)$. Satisfying (7.3) implies $h(x_{k+1}) \geq \gamma_k h(x_k)$, i.e., the lower bound of the DCBF decreases exponentially with the decay rate γ_k [2]. Given a choice of $\gamma(x)$, we denote $\mathcal{K}(x)$ as

$$\mathcal{K}(x) := \{u \in \mathcal{U} : h(f(x,u)) - \gamma(x)h(x) \geq 0\}. \tag{7.4}$$

Then, if $x_0 \in \mathcal{S}$ and $u_k \in \mathcal{K}(x_k)$, then $x_k \in \mathcal{S}$ for $\forall\, k \in \mathbb{Z}^+$, i.e., the resulting trajectory is safe [2].

Given a valid DCBF h [12], imposing DCBF constraint (7.3) in an optimization problem could guarantee system safety, i.e., collision-free trajectories. If $\gamma(x)$ is close to 1, the system

converges to $\partial\mathcal{S}$ slowly but can easily become infeasible. On the other hand, if $\gamma(x)$ is close to 0, the constraint (7.3) is feasible in a larger domain but can approach $\partial\mathcal{S}$ quickly and become unsafe. The later proposed formulation in [223] (discussed in Chapter 6) introduces a relaxing form of DCBF constraint as follows,

$$h(f(x,u)) \geq \omega(x)\gamma(x)h(x), \quad 0 \leq \gamma(x) < 1. \tag{7.5}$$

where the relaxing variable ω resolves the tradeoff between feasibility and safety and is optimized with other variables inside an optimization formulation.

When one-step control input is optimized [2], it could lead to a deadlock situation such that the robot is safe but unable to track the reference command. A nonlinear model predictive control formulation [224] can overcome these problems, shown as follows,

---

NMPC-DCBF [224] (discussed in Chapter 6):

$$\min_{U,\Omega} p(x_{t+N|t}) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) + \psi(\omega_k) \tag{7.6a}$$

$$\text{s.t. } x_{t|t} = x_t, \tag{7.6b}$$

$$x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t}), \quad k = 0, ..., N-1 \tag{7.6c}$$

$$u_{t+k|t} \in \mathcal{U},\ x_{t+k|t} \in \mathcal{X}, \qquad k = 0, ..., N-1 \tag{7.6d}$$

$$h(x_{t+k+1|t}) \geq \omega_k \gamma_k h(x_{t+k|t}),\ \omega_k \geq 0 \quad k = 0, ..., N_{\text{CBF}} - 1, \tag{7.6e}$$

---

where $x_{t+i|t}$ and $u_{t+i|t}$ denote the predicted state and input at time $t+i$ evaluated at the current time t. N and $N_{\text{CBF}} \leq N$ denote the prediction and safety horizons respectively, which allows us to control the optimization computation complexity, and $U = [u_{t|t}^T, ..., u_{t+N-1|t}^T]^T$ and $\Omega = [\omega_0, ..., \omega_{N_{\text{CBF}}-1}]$ are the joint input and relaxation variables respectively. $p(\cdot)$ and $q(\cdot,\cdot)$ are the terminal and stage costs respectively, and ψ is the penalty function for the relaxation variable.

The optimization formulation (7.6) can be regarded as a control problem with $U^* = [u^{*T}_{t|t}, ..., u^{*T}_{t+N-1|t}]^T$ as the optimized control inputs, as well as a trajectory planning problem with $X^* = [x^{*T}_{t|t}, ..., x^{*T}_{t+N|t}]^T$ as the optimized trajectory. From the joint optimal input vector U^* the first input $u^*_{t|t}$ is applied at time $t \in \mathbb{Z}^+$, and the optimization (7.6) is solved again at time $t+1$ with x_{t+1}.

7.2.2 Obstacle Avoidance between Polytopic Sets

In this work, we assume that there are $N_{\mathcal{O}}$ static obstacles together with a single controlled robot. We further assume that the geometry of all the obstacles and the robot can be over-approximated with a union of convex polytopes, which is defined as a bounded polyhedron.

Let the state of the robot be $x \in \mathbb{R}^n$ with its discrete-time dynamics as defined in (7.1) and the geometry of the robot and obstacles be in a l-dimensional space. We denote the

geometry of the i-th static obstacle and the dynamic robot at some state $x \in \mathcal{X}$ by the polytopes:

$$\begin{aligned} \mathcal{O}_i &:= \{y \in \mathbb{R}^l : A^{\mathcal{O}_i} y \leq b^{\mathcal{O}_i}\} \\ \mathcal{R}(x) &:= \{y \in \mathbb{R}^l : A^{\mathcal{R}}(x) y \leq b^{\mathcal{R}}(x)\}, \end{aligned} \tag{7.7}$$

respectively, where $b^{\mathcal{O}_i} \in \mathbb{R}^{s^{\mathcal{O}_i}}, i \in \{1, ...N_{\mathcal{O}}\}$ and $b^{\mathcal{R}}(x) \in \mathbb{R}^{s^{\mathcal{R}}}$, and $A^{\mathcal{R}}, b^{\mathcal{R}}$ are continuous. Inequalities on vectors are enforced element-wise. We assume that $\mathcal{O}_i$, $i \in \{1, ...N_{\mathcal{O}}\}$ and $\mathcal{R}(x)$ $\forall$ $x \in \mathcal{X}$ are bounded and non-empty. $s^{\mathcal{O}_i}$, $s^{\mathcal{R}}$ represent the number of facets of polytopic sets for the i-th obstacle and the robot, respectively.

Then $\mathcal{O}_i$, $\forall$ $i \in \{1, ..., N_{\mathcal{O}}\}$, and $\mathcal{R}(x)$, $\forall$ $x \in \mathcal{X}$, are non-empty, convex, and compact sets, and the minimum distance between any pair $(\mathcal{O}_i, \mathcal{R}(x))$ is well-defined. The minimum distance is 0 if and only if $\mathcal{O}_i$ and $\mathcal{R}(x)$ intersect. The square of the minimum distance between $\mathcal{O}_i$ and $\mathcal{R}(x)$, denoted by $h_i(x)$, can be computed using a QP as follows:

$$h_i(x) = \min_{(y^{\mathcal{O}_i}, y^{\mathcal{R}}) \in \mathbb{R}^{2l}} \quad \|y^{\mathcal{O}_i} - y^{\mathcal{R}}\|_2^2 \tag{7.8a}$$

$$\text{s.t.} \quad A^{\mathcal{O}_i} y^{\mathcal{O}_i} \leq b^{\mathcal{O}_i}, \; A^{\mathcal{R}}(x) y^{\mathcal{R}} \leq b^{\mathcal{R}}(x). \tag{7.8b}$$

where (7.8) is a convex optimization problem. To ensure safe motion of the robot, we enforce DCBF constraints (7.3) pairwise between each robot-obstacle pair. Then, the safe set corresponding to the pair $(\mathcal{O}_i, \mathcal{R})$ is defined as:

$$\mathcal{S}_i := \{x \in \mathbb{R}^n : h_i(x) > 0\}^c, \tag{7.9}$$

where $(\cdot)^c$ denotes the closure of a set. Enforcing the DCBF constraint for each h_i ensures that the state remains in $\mathcal{S}_i$ for all i, and thus the state remains in $\mathcal{S} := \cap_{i=1}^{N_{\mathcal{O}}} S_i$. Note that, due to the relaxation variable ω, enforcing multiple DCBF constraints for each h_i is equivalent to enforcing a single DCBF constraint on $h(x) := \min_i\{h_i(x)\}$. Thus, we focus on how to enforce the DCBF constraint for a given pair $(\mathcal{O}_i, \mathcal{R})$.

However, (7.3) requires computation of $h_i(f(x, u))$ via (7.8), which can only be solved numerically. This results in an optimization formulation with non-differentiable implicit constraints, which results in a significant increase in computation time. In the following section we derive explicit differentiable constraints which guarantee that the DCBF constraint is satisfied without affecting the feasible set of safe inputs $\mathcal{K}(x)$ $\forall$ $x \in \mathcal{X}$.

7.3 Optimization with Dual DCBF Constraints

In this section, we derive the duality-based optimization which allows generating obstacle avoidance maneuvers for the controlled robot in tight environments with obstacles.

7.3.1 Dual Optimization Problem

Corresponding to the minimization problem (7.8), we can define a dual problem. The dual problem is always a convex optimization problem, and a maximization problem if the original primal problem is a minimization one.

The dual formulation of (7.8) can be explicitly computed as [27, Chap. 8]:

$$g_i(x) = \max_{(\lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}})} \quad -\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(x) \tag{7.10a}$$

$$\text{s.t.} \quad \lambda^{\mathcal{O}_i} A^{\mathcal{O}_i} + \lambda^{\mathcal{R}} A^{\mathcal{R}}(x) = 0, \tag{7.10b}$$

$$\|\lambda^{\mathcal{O}_i} A^{\mathcal{O}_i}\|_2 \leq 1,\ \lambda^{\mathcal{O}_i} \geq 0,\ \lambda^{\mathcal{R}} \geq 0. \tag{7.10c}$$

Here $\lambda^{\mathcal{O}_i} A^{\mathcal{O}_i}$ represents the normal vector of the plane of maximum separation between the polytopes.

The Weak Duality Theorem [27, Chap. 5] states that $g_i(x) \leq h_i(x)$ holds for all optimization problems. Since (7.8) is a convex optimization with linear constraints and has a well-defined optimum solution in $\mathbb{R}^+$, the Strong Duality Theorem [27, Chap. 5] also holds, which states that

$$g_i(x) = h_i(x). \tag{7.11}$$

7.3.2 Obstacle Avoidance with Dual Variables

In order to remove the implicit dependence of $h_i(x)$ on x via (7.8), we enforce a constraint stronger than (7.3) which does not require explicit computation of $h_i(x)$ via (7.8). The dual formulation of (7.8), (7.10), can be used to achieve this.

Let $\bar{g}_i(x, \lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}})$ be the cost corresponding to any feasible solution $(\lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}})$ of (7.10). Since (7.10) is a maximization problem and the Strong Duality Theorem (7.11) holds for the primal problem (7.8),

$$\bar{g}_i(x, \lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}}) := -\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(x) \leq h_i(x). \tag{7.12}$$

Then, at time k, we can enforce the stronger constraint

$$-\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(f(x_k, u)) \geq \gamma_k h_i(x_k) \tag{7.13}$$

which, using (7.12), implies

$$h_i(f(x_k, u)) \geq \gamma_k h_i(x_k), \tag{7.14}$$

as required. Hence, the DCBF constraint can be enforced by (7.13) subject to $(f(x_k, u), \lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}})$ being feasible, i.e., the stronger DCBF constraint (7.13) along with the feasibility constraints (7.10b), (7.10c) should be satisfied

$$-\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(f(x_k, u)) \geq \gamma_k h_i(x_k), \tag{7.15a}$$

$$\lambda^{\mathcal{O}_i} A^{\mathcal{O}_i} + \lambda^{\mathcal{R}} A^{\mathcal{R}}(f(x_k, u)) = 0, \tag{7.15b}$$

$$\|\lambda^{\mathcal{O}_i} A^{\mathcal{O}_i}\|_2 \leq 1,\ \lambda^{\mathcal{O}_i} \geq 0,\ \lambda^{\mathcal{R}} \geq 0. \tag{7.15c}$$

By the Strong Duality Theorem (7.11), $\exists\ \lambda^{\mathcal{O}_i *}, \lambda^{\mathcal{R}*}$ satisfying (7.10b)-(7.10c) such that for all $x \in \mathcal{X}$,

$$\bar{g}_i(x, \lambda^{\mathcal{O}_i *}, \lambda^{\mathcal{R}*}) = -\lambda^{\mathcal{O}_i *} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}*} b^{\mathcal{R}}(x) = h_i(x). \tag{7.16}$$

This means that for any fixed $x \in \mathcal{X}$, the input u satisfies the DCBF constraint (7.3) with the implicit definition of h_i if and only if the tuple $(u, \lambda^{\mathcal{O}_i *}, \lambda^{\mathcal{R}*})$ satisfies (7.13). So, the feasible set of inputs is not reduced at any $x \in \mathcal{X}$.

7.3.3 Optimization Formulation

We adopt the philosophy of the NMPC-DCBF method to enforce safety constraints between polytopes, as shown in Sec. 7.3.2, and construct a multi-step optimization formulation. For simplicity of notation, we drop the explicit dependence of $h_i(x_{t+k|t})$ on $x_{t|t}$ and $[u_{t|t}^T, ..., u_{t+k-1|t}^T]^T$. Since $h_i(x_{t+k|t})$ is also not explicitly known at time t, to impose DCBF constraints along the horizon, we extend the idea in Sec. 7.3.2 by enforcing a constraint stronger than $h_i(x_{t+k+1|t}) \geq \gamma_k h_i(x_{t+k|t})$ which does not rely on computations of $h_i(x_{t+k|t})$ and $h_i(x_{t+k+1|t})$ via (7.8).

The primal optimization problem (7.8) provides an upper bound to $h_i(x)$, which can be used in the stronger DCBF constraints. Let $(y^{\mathcal{O}_i}, y^{\mathcal{R}})$ be any feasible solution of (7.8). Since (7.8) is a minimization problem and its solution is well-defined for all $x \in \mathcal{X}$,

$$\bar{h}_i(x, y^{\mathcal{O}_i}, y^{\mathcal{R}}) := \|y^{\mathcal{O}_i} - y^{\mathcal{R}}\|_2^2 \geq h_i(x). \tag{7.17}$$

Then at time t, we can enforce

$$-\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(x_{t+k+1|t}) \geq \gamma_k \|y^{\mathcal{O}_i} - y^{\mathcal{R}}\|_2^2 \tag{7.18}$$

which, using (7.12) and (7.17), implies

$$\begin{aligned} h_i(x_{t+k+1|t}) &\geq -\lambda^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda^{\mathcal{R}} b^{\mathcal{R}}(x_{t+k+1|t}) \\ &\geq \gamma_k \|y^{\mathcal{O}_i} - y^{\mathcal{R}}\|_2^2 \geq \gamma_k h_i(x_{t+k|t}), \end{aligned} \tag{7.19}$$

as required. Additionally, we can introduce the relaxation variables without affecting the analysis in this section.

Remark 7.1. *The primal problem* (7.8) *and the dual problem* (7.10) *provide upper and lower bounds respectively for the distance function h, which is implicit in nature. These bounds can be calculated implicitly since they are equal to the cost functions subject to the corresponding constraints of each optimization problem. Thus, any implicit constraint on h can be converted into a weaker explicit set of constraints using these bounds.*

Then at time t, we first calculate the optimal solutions $(y_t^{\mathcal{O}_i *}, y_t^{\mathcal{R}*})$ to the minimum distance QP (7.8) using $x = x_t$, hence the NMPC-DCBF formulation for polytopes is shown as follows:

NMPC-DCBF for Polytopes:

$$\min_{U,\Omega} p(x_{t+N|t}) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) + \psi(\omega_k) \tag{7.20a}$$

$$\text{s.t. } x_{t|t} = x_t, \quad y_0^{\mathcal{O}_i} = y_t^{\mathcal{O}_i *}, \quad y_0^{\mathcal{R}} = y_t^{\mathcal{R}*} \tag{7.20b}$$

$$x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t}),\ k{=}0, ..., N{-}1 \tag{7.20c}$$

$$u_{t+k|t} \in \mathcal{U},\ x_{t+k|t} \in \mathcal{X}, \qquad k{=}0, ..., N{-}1 \tag{7.20d}$$

$$-\lambda_{k+1}^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda_{k+1}^{\mathcal{R}} b^{\mathcal{R}}(x_{t+k+1|t}) \geq \omega_k \gamma_k \| y_k^{\mathcal{O}_i} - y_k^{\mathcal{R}} \|_2^2, \quad \text{for } k{=}0, ..., N_{\text{CBF}}{-}1 \tag{7.20e}$$

$$A^{\mathcal{R}}(x_{t+k+1|t}) y_k^{\mathcal{R}} \leq b^{\mathcal{R}}(x_{t+k+1|t}),\ A^{\mathcal{O}_i} y_k^{\mathcal{O}_i} \leq b^{\mathcal{O}_i}, \quad \text{for } k{=}1, ..., N_{\text{CBF}}{-}1 \tag{7.20f}$$

$$\lambda_k^{\mathcal{O}_i} A^{\mathcal{O}_i} + \lambda_k^{\mathcal{R}} A^{\mathcal{R}}(x_{t+k|t}) = 0,\ \| \lambda_k^{\mathcal{O}_i} A^{\mathcal{O}_i} \|_2 \leq 1, \quad \text{for } k{=}1, ..., N_{\text{CBF}} \tag{7.20g}$$

$$\lambda_{k+1}^{\mathcal{O}_i} \geq 0, \quad \lambda_{k+1}^{\mathcal{R}} \geq 0, \quad \omega_k \geq 0, \quad \text{for } k{=}0, ..., N_{\text{CBF}}{-}1 \tag{7.20h}$$

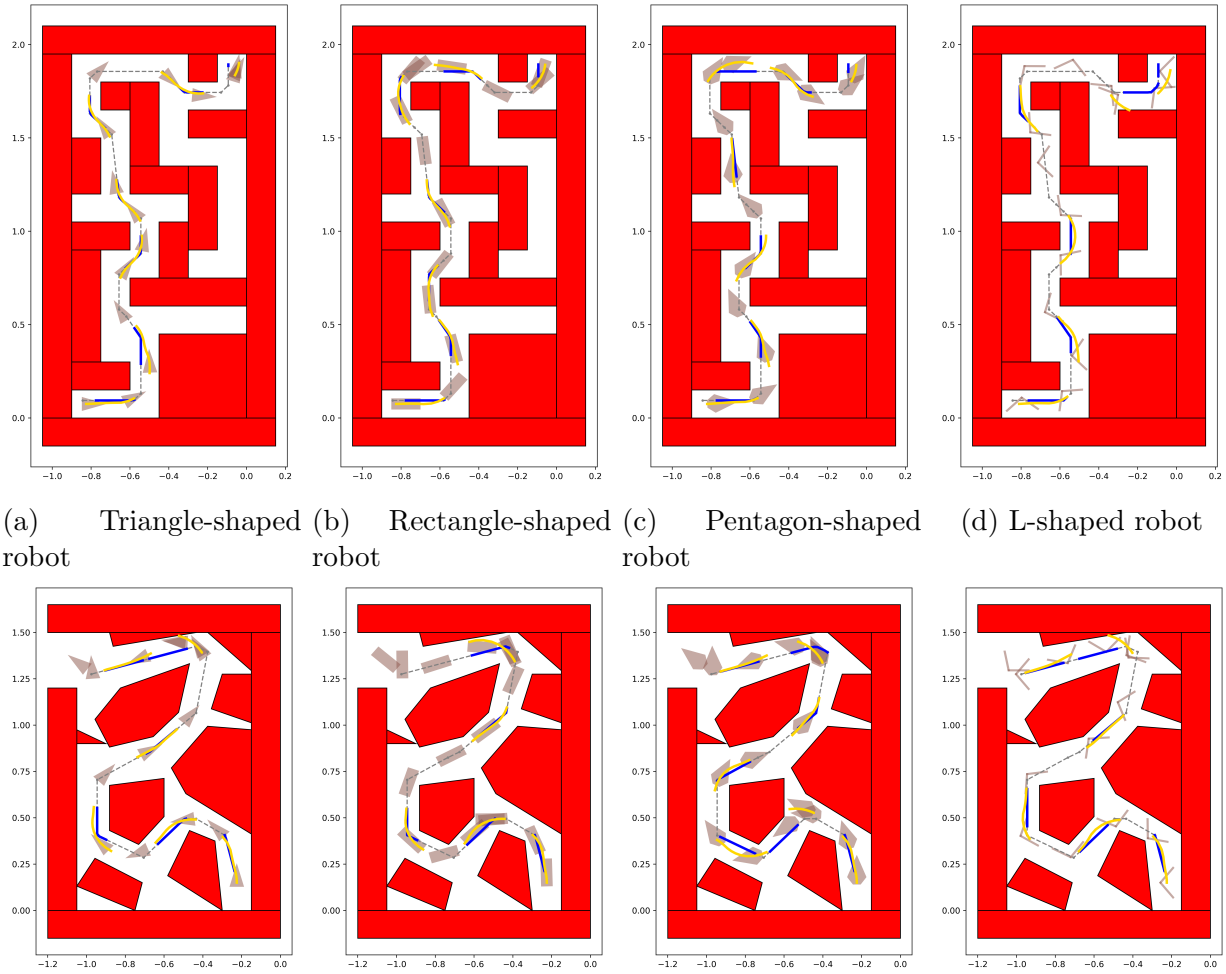
The subscripts of $\lambda^{\mathcal{O}_i}, \lambda^{\mathcal{R}}, y^{\mathcal{O}_i}, y^{\mathcal{R}}$ denote the time, (7.20e) is the DCBF constraint between polytopes, (7.20f) is the primal feasibility condition, and (7.20g)-(7.20h) are the dual feasibility conditions. The initial conditions, system dynamics constraints, and input and state constraints are represented by (7.20b), (7.20c), and (7.20d) respectively. This NMPC-DCBF formulation (7.20) corresponds to enforcing safety constraints only between the pair $(\mathcal{O}_i, \mathcal{R})$ of polytopes. To enforce DCBF constraints between every pair of robot and obstacle, we introduce corresponding dual and primal variables and enforce the constraints (7.20e)-(7.20h) for each pair.

Remark 7.2. *The mathematical intuition behind* (7.20) *is different from the previous work in [187] (discussed in Chapter 6) that focuses on duality in the continuous domain. Moreover, the system dynamics is not required to be control affine in our proposed* (7.20)*. Furthermore, differentiability of dual variables is required in [187] and not needed in this work.*

7.3.4 Complexity and Performance

7.3.4.1 Exponential DCBF Constraint

To reduce complexity of the NMPC-DCBF shown in (7.20), we can modify the DCBF constraint $h_i(x_{t+k|t}) \geq \omega_k \gamma_k h_i(x_{t+k-1|t})$ by rolling out time k and removing the dependence



(a) Triangle-shaped robot (b) Rectangle-shaped robot (c) Pentagon-shaped robot (d) L-shaped robot

(e) Triangle-shaped robot (f) Rectangle-shaped robot (g) Pentagon-shaped robot (h) L-shaped robot

Figure 7.2: Snapshots from simulation of tight maneuvers of obstacle avoidance with a controlled robot with different shapes in two maze environments.

on $x_{t+k-1|t}$ from the LHS of the DCBF constraint, thus enforcing the following exponential DCBF constraints:

$$h_i(x_{t+k|t}) \geq \omega_k(\Pi_{j=0}^{k}\gamma_j)h_i(x_{t|t}). \tag{7.21}$$

The exponential decay rate $\Pi_{j=0}^{k}\gamma_j$ arises due to rolling out the decay rate at time j, γ_j. The DCBF constraint (7.21) only contains $h_i(x_{t|t})$ on the RHS for all k, which can be explicitly computed at each time step using $x_t = x_{t|t}$. Note that the RHS of the DCBF constraint (7.19) at time $k \geq 1$ cannot be computed explicitly, since $x_{t+k|t}$ implicitly depends on the control inputs. This leads to modifying (7.20e) with,

$$-\lambda_k^{\mathcal{O}_i} b^{\mathcal{O}_i} - \lambda_k^{\mathcal{R}} b^{\mathcal{R}}(x_{t+k|t}) \geq \omega_k(\Pi_{j=0}^{k}\gamma_j)h_i(x_{t|t}) \tag{7.22}$$

Such a formulation speeds up the computational time, as only $h(x_{t|t})$ needs to be calculated at each time t. This change affects neither the feasibility nor the safety of the system.

7.3.4.2 Horizon Length Selection

Compared with distance constraints, DCBF constraints allow effective obstacle avoidance behavior with smaller horizon length. Our NMPC-DCBF formulation does not require the obstacle avoidance horizon length N_{CBF} be equal to N, which additionally reduces the complexity [223]. Additionally, maneuvers such as deceleration for obstacle avoidance or reversing motion for deadlock avoidance are motion primitives that only require a small horizon in the MPC formulation. To sum up, DCBF constraints with dual variables enable fast optimization for obstacle avoidance between polytopes.

7.4 Numerical Results

In this section, we consider an autonomous navigation problem. We model the controlled robot with different shapes, including triangle, rectangle, pentagon and L-shape. The proposed optimization-based planning algorithm allows us to successfully generate dynamically-feasible collision-free trajectories even in tight maze environments, shown in Figure 7.2. Animation of the navigation problems can be found in the video attachment.

7.4.0.1 Environment

The tight maze environment is described by a combination of multiple convex obstacles, including shapes like triangle, rectangle pentagon, etc. The controlled robot is also modelled with different shapes, including triangle, rectangle, pentagon and L-shape, whose orientation is determined by the yaw angle. The L-shape is non-convex, but can be represented by two convex polytopes. The dimensions for each robot shape are mentioned in Figure 7.2.

7.4.0.2 System Dynamics

The controlled robot is described by the kinematic bicycle model, which is typical for testing trajectory planning algorithms in tight environments. The continuous-time dynamics of the robot is given as follows,

$$\dot{c}_x = v\cos(\phi),\ \dot{c}_x = v\sin(\phi),\ \dot{v} = a,\ \dot{\phi} = \frac{v\tan\delta}{l}, \tag{7.23}$$

where system states are $x = (c_x, c_y, v, \phi)$ with (c_x, c_y) as the center of rear axes, ϕ as yaw angle and v as the velocity, and $u = (a, \delta)$ are inputs with steering angle δ and acceleration a. The wheel base of the robot is $l = 0.1m$. The steering angle and acceleration are limited between $\pm 0.5rad$ and $\pm 1m/s^2$.

7.4.0.3 Global Planning

The global path from the starting position to the goal is generated using the A* algorithm. The 2-D space is sub-divided into grids and obstacle collision checks are performed at each grid point during the algorithm. A safety margin, which is smaller than at least one dimension of the robot, is used for the collision checks. Finally, the generated optimal path is reduced to fewer waypoints using line-of-sight reductions, similar to the θ^* algorithm [132]. The generated global path is not dynamically feasible, and is only safe at the node points.

7.4.0.4 Local Trajectory Generation

The local trajectory planning is formulated using the NMPC-DCBF formulation (7.20) to track the local reference trajectory while avoiding obstacles. The local reference trajectory $\bar{X} = [\bar{x}_{t|t}^T, \bar{x}_{t+1|t}^T, ..., \bar{x}_{t+N|t}^T]^T$ is generated from a start point with a constant speed v_0 and the same orientation as the global planner. The start point is found by local projection from the current robot's position to the global path. For tracking the local reference trajectory, the cost function (7.20) in the optimizer is constructed with terminal cost, stage cost, and relaxing cost function, respectively as

$$p(x_{t+N|t}) = ||x_{t+N|t} - \bar{x}_{t+N|t}||_{Q_T}^2, \tag{7.24a}$$

$$q(x_{t+k|t}, u_{t+k|t}) = ||x_{t+k|t} - \bar{x}_{t+k|t}||_Q^2 + ||u_{t+k|t}||_R^2 + ||u_{t+k|t} - u_{t+k-1|t}||_{dR}^2, \tag{7.24b}$$

$$\psi(\omega_k) = p_\omega(\omega_k - 1)^2, \tag{7.24c}$$

where $u_{t-1|t} = u_{t-1|t-1}^*$ represents the last optimized control input. The dynamics constraints (7.20c) is applied with the discrete-time forward Euler formulation from the continuous-time dynamics (7.23). The input constraints (7.20d) is imposed by the steering angle and acceleration limits mentioned above. A prediction horizon of $N = 11$ is used, with the DCBF horizon as $N_{\text{CBF}} = 6$, and the decay rate as $\gamma_k = 0.8$. The obstacle avoidance constraints (7.20e)-(7.20h) can be applied directly to each pair of the convex-shaped (triangle, rectangle,

pentagon) robot and each convex obstacle. When the robot is non-convex shaped (L-shape), these constraints are applied to the convex parts of the robot and each convex obstacle.

To reduce the complexity of the optimization formulation, we only consider the obstacles which are within a specified radius from the robot at any given time. The radius is calculated using the reference tracking velocity, prediction horizon and the maximum deceleration of the robots.

7.4.0.5 Warm Start

The NMPC-DCBF formulation is a non-convex optimization, and hence computationally challenging to solve in general. Although the DCBF constraints help to reduce the complexity, as discussed in Sec. 7.3.4.2, it still requires a good initial guess to lead to faster numerical convergence. The initial guess trajectory and control inputs are generated using a braking controller. An acceleration input equal to the maximum deceleration is provided and the steering angle is set to zero. Once the robot comes to a halt, the acceleration input is also set to zero. The braking control inputs, along with the trajectory generated from it are provided as an initial guess to the optimization at each time step. Since at each time step $h_i(x_{t|t})$ is solved using (7.10), the dual optimal solution from this computation is provided as an initial guess for the dual variables for the entire horizon.

7.4.0.6 Simulation Results

To evaluate the performance of (7.20), we study the navigation problem with two different maze environments with four choices of robot shapes. The optimization problems are implemented in Python with CasADi [14] as modelling language, solved with IPOPT [22] on Ubuntu 18.04 with Intel Xeon E-2176M CPU with a 2.7GHz clock. From the snapshots we can observe tight-fitting obstacle avoidance motion of the robot, and also reversing motion to avoid deadlock. These examples highlight the safety and planning features of our implementation. We also analyze the computational time of trajectory generation using (7.20), which is shown in Table 7.1. This illustrates that optimization (7.20) can be solved sufficiently fast to be deployed on different-shaped robots for trajectory generation in different maze environments. The details of hyperparameter selections can be found in the open-source repository.

The specific choice of the parameters in the optimization formulation (7.20) can influence safety and deadlock behaviors in the robot. Qualitatively, for safety, the reference tracking velocity should be such that the robot can come to a halt within the prediction horizon with the input as the maximum deceleration. There is also a trade-off between deadlock avoidance and safety: Higher value of the terminal cost weight Q_T improves deadlock avoidance, but can also increase velocity of the robot, leading to unsafe motion.

To tune the hyperparameters for a given system, consider the set of feasible trajectories of (7.20) for some reference trajectory $\bar{X}$. First, to capture the affect of all control inputs on the DCBF h, the safety horizon N_{CBF} must be larger than the relative degree of the system

with respect to h. If there exists any safe trajectory at the current state that brings the robot to rest, the resulting control law guarantees that the closed-loop trajectory is also safe. Thus, it is desirable to choose the safety horizon N_{CBF} high enough so that the robot can come to rest within N_{CBF} steps. However, choosing a large safety horizon may result in an increased computation time and safety could be achieved in practice with proper tuning of the decay rate γ. Choosing γ closer to 1 prioritizes safety over reference tracking while choosing γ closer to 0 ensures faithful tracking at the cost of safety. Note that the relaxation variable ω_k is essential for feasibility of (7.20) especially for γ closer to 1, see [223]. Selecting a large terminal cost parameter p_ω in (7.24c) is also key to ensure that the effective decay rate does not deviate, thus prioritizing safety of the system. Finally, the stage cost parameter Q in (7.24b) and the terminal cost parameter Q_T in (7.24a) present a trade-off between reference tracking and exploration. Larger values of Q result in the closed-loop trajectory closely following the reference trajectory. Notice that since the reference trajectory may not be dynamically feasible, this may still result in deadlocks. Selecting larger values of Q_T mostly penalizes deviation from the terminal state and thus promotes exploration during timesteps $k = 1, ..., N_{\text{CBF}-1}$. This can result in better deadlock avoidance maneuvers as depicted in Figure 7.2.

Table 7.1: Solver time statistics of NMPC-DCBF with polytopic obstacles.

Env	Robot Shape	median	std	min	max
Maze 1	triangle (Figure 7.2a)	51ms	19ms	14ms	149ms
	rectangle (Figure 7.2b)	49ms	25ms	15ms	185ms
	pentagon (Figure 7.2c)	71ms	15ms	15ms	272ms
	L-shape (Figure 7.2d)	85ms	13ms	13ms	215ms
Maze 2	triangle (Figure 7.2e)	33ms	24ms	13ms	121ms
	rectangle (Figure 7.2f)	29ms	25ms	13ms	122ms
	pentagon (Figure 7.2g)	30ms	29ms	13ms	150ms
	L-shape (Figure 7.2h)	29ms	49ms	13ms	233ms

7.5 Chapter Summary

In this chapter, we proposed a nonlinear optimization formulation using discrete-time control barrier function based constraints for polytopes. The proposed formulation has been shown to be applied as a fast optimization for control and planning for general nonlinear dynamical systems. We validated our approach on navigation problems with various robot shapes in maze environments with polytopic obstacles.

We have seen discussions involving optimization, control and trajectory generation about control barrier functions in different chapters until now. After this chapter, We will present different robotics applications with control barrier functions.

Part III

Applications about Motion planning and Navigation

Chapter 8

Safety-Critical Autonomous Driving with Finite State Machine [6]

8.1 Introduction

Safety is critical in the automobile industry, and significant progress in this area has been made via (semi-)autonomous driving in the past few decades. While systems like adaptive cruise control or lane departure warning systems have proven to improve vehicle's safety in simple scenarios, it is desirable in the future that autonomous driving could handle safety-critical tasks in more complex driving scenarios, for example, autonomous lane change maneuvers. Lane change is challenging for both drivers and autonomous driving controllers since multiple surrounding vehicles should be considered and their future movements should be predicted. The more challenging fact is that drivers or controllers have little time to respond to avoid a crash if a threat occurs. According to statistics provided by US National Highway Traffic Safety Administration (NHTSA) in [170], about 9% of all vehicle crashes involved lane change maneuvers.

8.1.1 Related Work

Several recent results focus on this topic. In [207, 82], new methods are proposed to predict the intent of other surrounding vehicles. Research in [202, 166] focuses on the generation of optimal trajectories for lane change maneuvers. Various low-level controllers have been developed to track the optimal path generated by planners. Model predictive control (MPC) is a commonly used method. In [229, 36, 16], MPC based controllers are implemented. Additionally, simple linear [92] or nonlinear [37] feedback control method has also been proven to be suitable for this task.

[6] The material of this chapter is from "Rule-based safety-critical control design using control barrier functions with application to autonomous lane change" published in 2021 American Control Conference (ACC) [74].

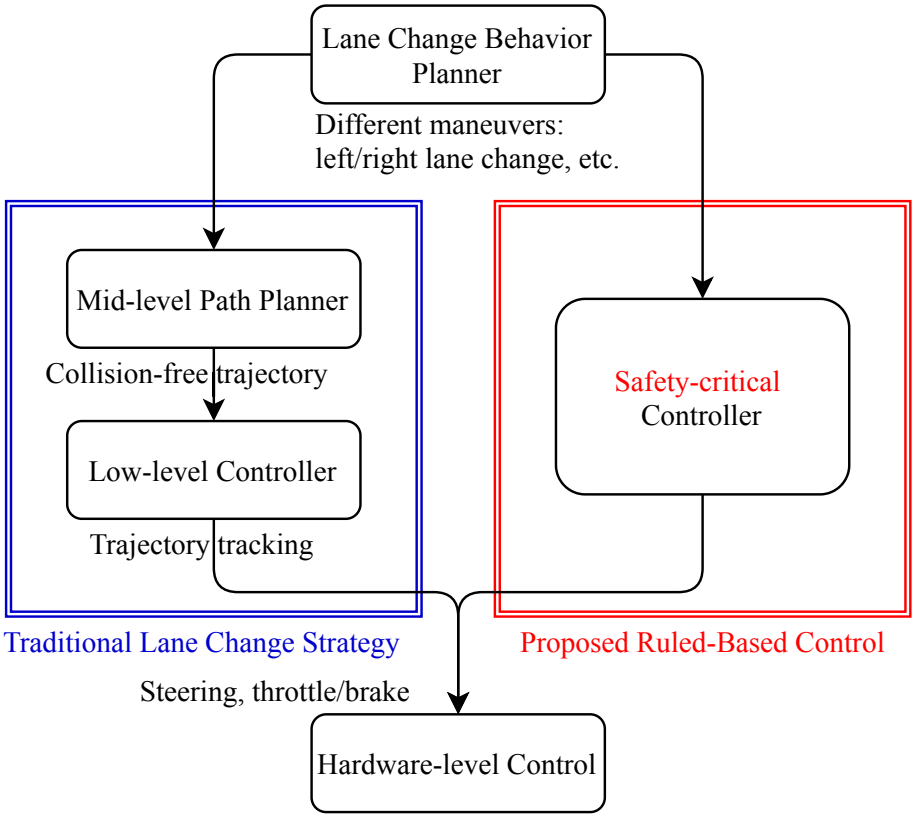


Figure 8.1: Control and planning strategy for lane change.

Based on previous work like [229, 36], we can summarize the traditional autonomous lane change strategy as shown in the left side of Figure 8.1. A high-level behavior planner makes the decision on the ego vehicle should stay in the current lane or change to the left (right) adjacent lane. This can be done by human drivers (e.g. activating the turn signal) or by algorithms [191]. Once a specific lane change maneuver is triggered, a mid-level path planner generates a collision-free optimal trajectory for the ego vehicle, which will be tracked by a low-level controller. However, the mid-level path planner usually can not be updated at a high frequency and may not respond to a sudden threat immediately, which means its planned trajectory may not always meet safety-critical requirements in a fast changing environment. Furthermore, there are always notable tracking errors in the low-level controller, which means tracking a collision-free trajectory could still be potentially unsafe. Therefore, in a safety-critical autonomous lane change strategy, safety relevant constraints should be considered in the low-level controller directly and the controller must work at a high update frequency.

In order to guarantee the system's safety in the low-level controller, control barrier functions (CBFs) have recently been introduced to ensure set invariance by considering the

system dynamics and several researchers have used CBFs to ensure the system's safety for the vehicle control design problems. In [38, 175, 224, 159], CBFs are used to implement obstacle avoidance. A CBF-based controller to supervise the safety of a learning based lane keeping controller is proposed in [39]. In [9], CBFs are unified with control Lyapunov functions (CLFs) via a quadratic program, and it illustrates this CLF-CBF-QP formulation in the context of adaptive cruise control. Due to its low computational complexity, this quadratic program allows the low-level controller to work at a high update frequency, which motivates us to investigate the safety-critical lane change control design through the CLF-CBF-QP formulation.

Rule-based strategies are also widely used in control design. One example is the finite state machine (FSM), which can make decisions according to input signals and transition conditions [131]. Since its computational complexity is lower than that of a traditional path planner, the FSM can be used in a low-level controller and work at a high update frequency, which inspires us to use a well-designed FSM to replace the mid-level path planner in traditional lane change strategies.

Based on the above analysis, we propose a rule-based safety-critical lane change control design. A FSM works as the basic structure, where a quadratic program based optimization problem using CLF and CBF constraints (CLF-CBF-QP) is formulated to achieve control objectives and guarantee the system's safety. The FSM can unify the mid-level path planner and low-level controller in traditional lane change strategy as one: a low-level safety-critical controller, see Figure 8.1. The FSM is used to make the decision of whether the ego vehicle can do a collision-free lane change maneuver or not. If the current traffic environment isn't suitable for a lane change maneuver, the ego vehicle will keep the current lane until a safe situation occurs. Additionally, if a threat arises during a lane change maneuver, the FSM will enter another state to drive the ego vehicle back to its current lane. Since the safety relevant constraints are considered in the low-level controller directly and a quadratic program allows the controller to run with high update frequency, our proposed strategy can guarantee the ego vehicle's safety in complex environments.

8.1.2 Contribution

The contribution of this chapter is as follows:

- We present a rule-based safety-critical design by using CLF-CBF-QP formulation, which can achieve the control objective and guarantee the vehicle's safety via low-level control in lane change maneuvers. The quadratic program allows the controller to work at a high update frequency.
- A finite state machine is used to unify mid-level planner and low-level controller into one optimization problem, a low-level safety-critical controller. The FSM helps the proposed strategy do switches between multiple CBF constraints in a complicated environment.

- We verify the performance of our controller using both typical and randomly generated driving scenarios. The result shows that our approach is generally applicable in both highway and city driving scenarios.

8.2 Background

8.2.1 Vehicle Model

In this chapter, we use a kinematic bicycle model (8.1) in [105] for our numerical validation and its dynamics is described as follows,

$$\dot{x} = v\cos(\psi+\beta) \tag{8.1a}$$
$$\dot{y} = v\sin(\psi+\beta) \tag{8.1b}$$
$$\dot{\psi} = \frac{v}{l_r}\sin\beta \tag{8.1c}$$
$$\dot{v} = a \tag{8.1d}$$
$$\beta = \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta_f)\right) \tag{8.1e}$$

where acceleration at vehicle's center of gravity (c.g.) a and front steering angle δ_f are the inputs of the system. x and y denote the coordinates of the vehicle's c.g. in an inertial frame (X, Y). ψ represents the orientation of the vehicle. l_f and l_r describe the distance from vehicle's c.g. to the front and real axles, respectively. β represents the slip angle of the vehicle.

Remark 8.1. *Both kinematic and dynamic bicycle models are commonly used vehicle models in the field of autonomous driving research. From [147], authors show that the kinematic bicycle model works under small lateral acceleration and also recommend $0.5\mu g$ as a limitation for the validity of the kinematic bicycle model (μ is friction coefficient and g is gravitational acceleration). In this work, we assume our controller will result in small longitudinal acceleration and this criteria will be considered as a constraint in our optimal control problem in Sec. 8.3*

Notice that the kinematic model in (8.1) represents a nonlinear nonaffine system. For our later usage, we assume that the slip angle β is constrained with a small angle assumption in our control design together, where we approximate $\cos\beta = 1$ and $\sin\beta = \beta$. Hence, (8.1) could be simplified as a nonlinear affine form as follows,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\\ \dot{v}\end{bmatrix} = \begin{bmatrix}v\cos\psi\\ v\sin\psi\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0 & -v\sin\psi\\ 0 & v\cos\psi\\ 0 & v/l_r\\ 1 & 0\end{bmatrix}\begin{bmatrix}a\\ \beta\end{bmatrix} \tag{8.2}$$

This nonlinear affine model in (8.2) with inputs a and β will be used for our safety-critical control design. Input δ_f in (8.1) can be calculated through (8.1e)

8.2.2 Safety-Critical Control

Consider a nonlinear affine control system

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{8.3}$$

where $\mathbf{x} \in D \subset \mathbb{R}^n$, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$ represent the system's state and input, and $\mathcal{U}$ is the admissible input set of the system, f and g are locally Lipschitz.

For this nonlinear affine system, we are interested in the system's safety. For the same control system (8.3), we define a superlevel set $\mathcal{C} \subset D \subset \mathbb{R}^n$ of a differentiable function $h : D \times \mathbb{R} \to \mathbb{R}$,

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}, t) \geq 0\}, \tag{8.4}$$

and we refer to $\mathcal{C}$ as a safe set. The function h becomes a control barrier function (CBF) if it can satisfy the following condition [210]:

$$\sup_{\mathbf{u}\in\mathcal{U}}[\frac{\partial h}{\partial t} + L_f h(\mathbf{x}, t) + L_g h(\mathbf{x}, t)\mathbf{u}] \geq -\gamma h(\mathbf{x}, t). \tag{8.5}$$

When $h(\mathbf{x}, t)$ is time-invariant, denoted as $h(\mathbf{x})$, then the condition above could be simplified [12] as follows,

$$\sup_{\mathbf{u}\in\mathcal{U}}[L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}] \geq -\gamma h(\mathbf{x}). \tag{8.6}$$

The control barrier function guarantees the set invariance of $\mathcal{C}$ for the system's safety. Besides the system's safety, we have a control Lyapunov function for the system's stability, where a function $V : D \in \mathbb{R}^n \to \mathbb{R}_+$ is a control Lyapunov function (CLF) if (i) V is positive definite and (ii) it can satisfy the following condition:

$$\inf_{\mathbf{u}\in\mathcal{U}}[L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} \leq -\alpha V(\mathbf{x})]. \tag{8.7}$$

Notice that γ and α in (8.5) and (8.7) could be generalized into extended class $\mathcal{K}_\infty$ and $\mathcal{K}$ functions, respectively; and we only treat them as linear functions with constant coefficients in this chapter.

To guarantee the system's safety and achieve its control objective simultaneously, the control Lyapunov function and control barrier function can be unified as a quadratic program (CLF-CBF-QP) [9] and written as follows:

$$\mathbf{u}(\mathbf{x}) = \underset{(\mathbf{u},\delta)\in\mathbb{R}^{m+1}}{\operatorname{argmin}} \frac{1}{2}\mathbf{u}^T H(\mathbf{x})\mathbf{u} + p\delta^2 \tag{8.8a}$$

$$\text{s.t.} \quad L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u} \leq -\alpha V(\mathbf{x}) + \delta \tag{8.8b}$$

$$\frac{\partial h}{\partial t} + L_f h(\mathbf{x}, t) + L_g h(\mathbf{x}, t)\mathbf{u} \geq -\gamma h(\mathbf{x}, t) \tag{8.8c}$$

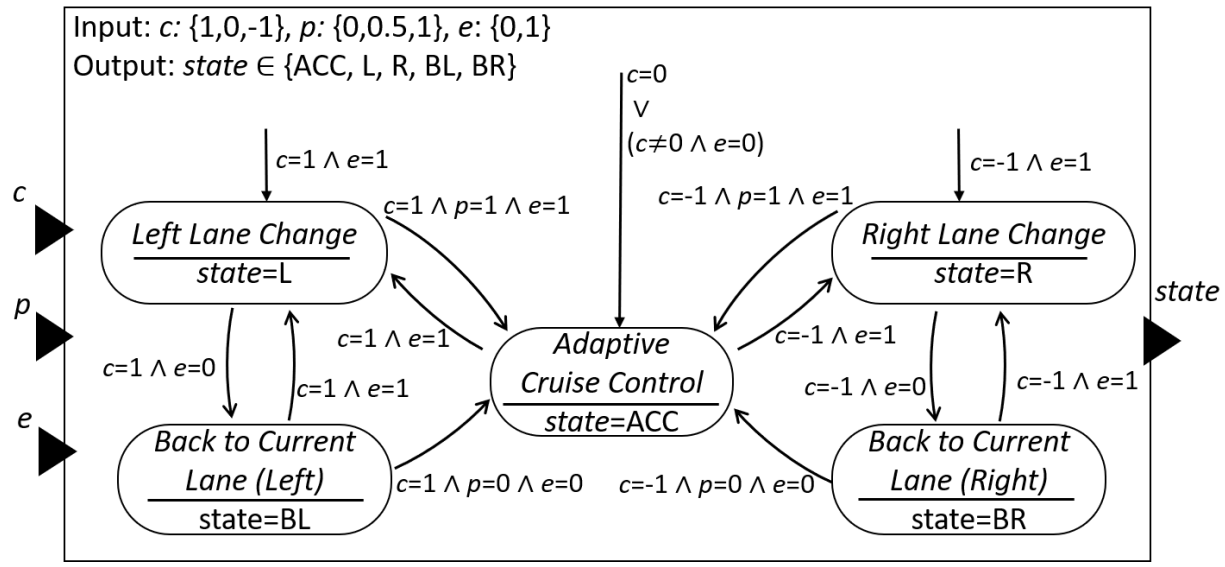


Figure 8.2: Finite state machine of lane change controller.

where $H(\mathbf{x})$ is a positive definite matrix, and δ is a relaxation variable to make the CLF constraint become a soft constraint to mediate stability for safety, and p is the penalty for this relaxation variable. This formulation will be used for our control design, which will be introduced in Sec. 8.3.

8.3 Control Design

After introducing the vehicle model and CLF-CBF-QP formulation, a safety-critical lane change controller will be presented in this section. The proposed controller is a finite state machine (FSM), where rule-based CLF-CBF-QP formulations are used to calculate the system's optimal input. This FSM will be introduced in Sec. 8.3.1. Then we will present the safety-based conditions for switches of constraints in Sec. 8.3.2. Finally, details about the CLF-CBF-QP formulations will be shown in Sec. 8.3.3.

8.3.1 Finite State Machine

Figure 8.2 shows the proposed FSM. The FSM's output is a state representing one of the following:
Adaptive Cruise Control State - **ACC**: The ego vehicle maintains a desired speed and follows a leading vehicle in its current lane at a safe distance.
Left or Right Lane Change State - **L** or **R**: The ego vehicle is expected to do a collision-free lane change maneuver to the left or right adjacent lane, respectively.
Back to Current Lane From Left or Right State - **BL** or **BR**: The ego vehicle drives back to its current lane to avoid a potential crash if a threat arises during a lane change maneuver.

Having presented the FSM's state definitions, we next introduce the FSM's input signals (c, p, e) and reveal how they make the FSM switch between the above states.
Command from High-Level Behaviour Planner (c) : This indicates the high-level planner's expected maneuver for the ego vehicle. Value 0 will set the controller in **ACC** state; value 1 or -1 will make the controller work in **L** or **R** state, respectively.
Positional Information - (p) : This represents the ego vehicle's relative lateral position. Value 0 means the ego vehicle is in its current lane; if it moves across the edge between current and target lanes, p will change to 0.5; finally, when the ego vehicle is totally in its target lane for more than some duration of time, e.g. 1.5s, p will become 1, which represents the success of a lane change maneuver and will bring the controller back to **ACC** state.
Traffic Environment Information - (e) : This shows whether the ego vehicle can do a lane change maneuver under safety-critical constraints. When the CLF-CBF-QP formulation is in the **L** or **R** state and is numerically unsolvable due to a potential future collision, e will change from 1 to 0. When c is not 0 but e is 0, then if the FSM is in **ACC** state, it will continue working in this state; otherwise, as shown in Figure 8.2, the FSM will go back to **ACC** state via **BL** or **BR** state.

When c's value is -1 or 1 but the ego vehicle is in **ACC** state, a predictive calculation as in (8.10) will be made to determine if the ego vehicle can get enough space for a lane change maneuver after accelerating to the speed limit. When the result shows this is possible, the speed limit will be the desired speed for **ACC** state and the ego vehicle will re-enter **L** or **R** state once the lane change CLF-CBF-QP is solvable.

These switches between different states implement the function of a planner in our proposed strategy. According to the input signals, the FSM will decide when is the best opportunity to do the lane change maneuver. Additionally, if this maneuver is interrupted, the FSM will drive the ego vehicle to change to its target lane again once it is safe.

8.3.2 Safety-Based Conditions for Switches of CBFs

In this section, we will show the continuity of the system's safety under different CBF constraints. In our controller, the switches between different CBF constraints could happen when the FSM changes state or a CBF constraint is removed from the CLF-CBF-QP formulation (see Sec. 8.3.3). This asks the controller to guarantee the continuity of system's safety. As shown in Figure 8.3, system's CBF constraints before and after a switch can be described as safe sets $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. When $h_2(\mathbf{x}, t)$ replaces $h_1(\mathbf{x}, t)$ as the new CBF constraint, if the system's state $\mathbf{x}$ is in the intersection of $\mathcal{C}_1$ and $\mathcal{C}_2$, the new barrier function $h_2(\mathbf{x}, t)$ will make the new safe set $\mathcal{C}_2$ invariant after the switch, which will guarantee the system's safety. This condition is used to design switches between different CBF constraints in this chapter.

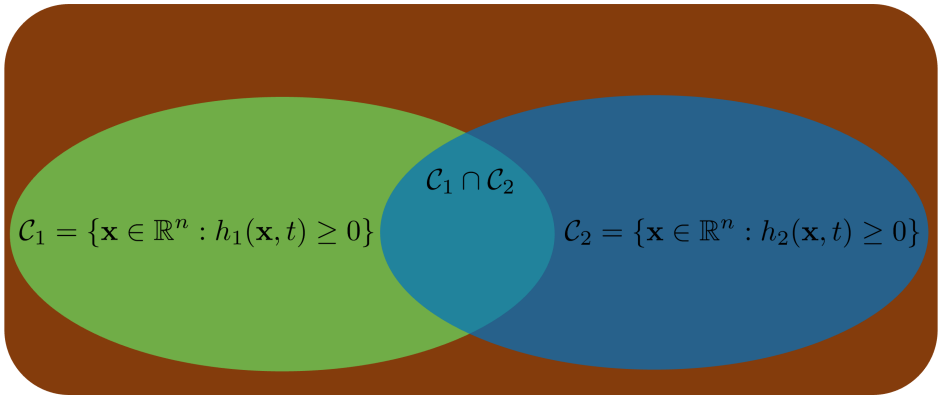


Figure 8.3: Safe sets' invariance for switches of CBF constraints.

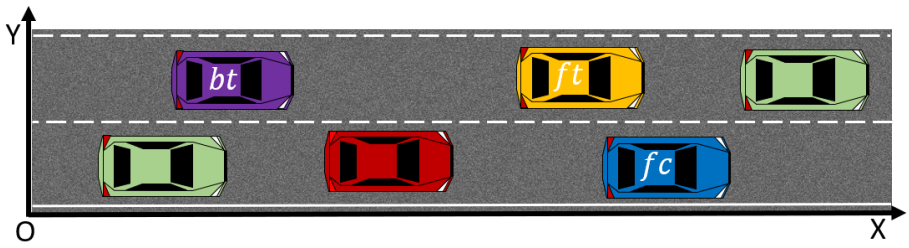


Figure 8.4: Typical lane change scenario.

8.3.3 Rule-Based CLF-CBF-QP Formulation

We now present the CLF-CBF-QP formulations in different FSM states. Firstly, we consider a typical lane change scenario as shown in Figure 8.4, where the red vehicle is our ego vehicle. In the controller, up to three vehicles will be selected as vehicles of interest (vehicles with letters in Figure 8.4): vehicle fc represents the vehicle immediately in *front* of the ego vehicle in its *current lane* (blue vehicle in Figure 8.4); vehicle bt represents the vehicle immediately *behind* the ego vehicle in the *target lane* (purple vehicle in Figure 8.4); vehicle ft denotes the vehicle immediately in *front* of the ego vehicle in the *target lane* (yellow vehicle in Figure 8.4). Other surrounding vehicles like green ones in the Figure 8.4 will not be considered in the controller. The frame in Figure 8.4 is the inertial frame used in further discussion, which is called E.

Remark 8.2. *In this work, we are interested in the safety-critical control of the autonomous lane change maneuver. To explore this problem, we assume the controller has access to accurate data of all surrounding vehicles. This could be done through Lidar, vision, radar, and ultrasonic sensors.*

Table 8.1: Notations and Symbols for control design.

Notation	Description
	Vehicle's Dimension Data
l_{fc}	length of vehicle's body that is in front of the c.g.
l_{rc}	length of vehicle's body that is behind the c.g.
w_{lc}	width of vehicle's body that is on the left of c.g.
w_{rc}	width of vehicle's body that is on the right of c.g.
	Ego Vehicle's Data
(x, y)	coordinates of c.g. in frame E
v	ego vehicle's speed
ψ	ego vehicle's yaw angle
v_d	ego vehicle's desired speed
v_l	ego vehicle's speed limit of current scenario
a_l	ego vehicle's acceleration limit
ϵ	a safety factor between 0.1-1 (1 is safest)
	Other Vehicles' Data
(x_k, y_k)	coordinates of vehicle k's c.g. in frame E
v_k	vehicle k's speed
a_k	vehicle k's acceleration
Δx_k	time varying longitudinal distance between vehicle k and the ego vehicle, equal to $\lvert x - x_k \rvert - l_{fc} - l_{rc}$.
Δy_k	time varying lateral distance between vehicle k and the ego vehicle, equal to $\lvert y - y_k \rvert - w_{rc} - w_{lc}$.

Table 8.1 shows notations that will be used for further discussion and the subscript k can represent fc, bt or ft, which indicates the corresponding vehicle with respect to the ego vehicle.

Remark 8.3. *The vehicle's position, velocity or yaw angle information is a function of time. In order to simplify the notations for discussion, in this chapter, we omit time information in the notations. For example, v represents the speed of the ego vehicle at the current time.*

We next present constraints used in the optimization problem, divided into two groups:
Hard Constraints: These represent the safety-critical relevant constraints. In our controller, these constraints are used to keep the ego vehicle at a safe distance from the surrounding vehicles, which is defined by us as $1+\epsilon$ times following vehicle's speed. For example, if the ego vehicle changes its lane, the distance between it and the vehicle bt should be greater than $(1+\epsilon)v_{bt}$. Hard constraints should never be violated under any conditions and will be guaranteed through CBF constraints.
Soft Constraints: These introduce the control objectives related constraints. The control goals will only be achieved through CLFs when the hard constraints are satisfied, for example

the speed or position of the ego vehicle. Through CLF constraints, it is possible to track a desired position without a reference trajectory.

Next we present details of the CLF-CBF-QP formulations in each FSM state.

For all FSM states, the following CLFs will be used to regulate the ego vehicle to track the desired speed v_d and reach its lateral dynamics' control objective:

$$V_v(\mathbf{x}) = (v - v_d)^2, \tag{8.9a}$$
$$V_y(\mathbf{x}) = (y - y_l)^2, \tag{8.9b}$$
$$V_\psi(\mathbf{x}) = \psi^2, \tag{8.9c}$$

where y_l is the y coordinate of current lane's center line for states **ACC**, **BL** and **BR** or y coordinate of target lane's center line for states **L** and **R** in frame E.

As mentioned in Sec. 8.3.1, if input c is 1 or -1 but the FSM is in **ACC** state, a simple predictive calculation will be done to determine the ego vehicle's desired speed by first computing the distances between the ego vehicle and the three vehicles of interest:

$$\Delta x'_{fc} = \Delta x_{fc} + v_{fc}\frac{v_l - v}{a_l} - \frac{v_l^2 - v^2}{2a_l} - (1+\epsilon)v, \tag{8.10a}$$
$$\Delta x'_{ft} = \Delta x_{ft} + v_{ft}\frac{v_l - v}{a_l} - \frac{v_l^2 - v^2}{2a_l} - (1+\epsilon)v, \tag{8.10b}$$
$$\Delta x'_{bt} = \Delta x_{bt} - v_{bt}\frac{v_l - v}{a_l} + \frac{v_l^2 - v^2}{2a_l} - (1+\epsilon)v_{bt}. \tag{8.10c}$$

If all equations above are greater than 0, which means the ego vehicle will have enough space to change the lane through accelerating to the current scenario's speed limit v_l, v_l will become the new desired speed v_d in **ACC** state; otherwise, the original desired speed v_d will be used in **ACC** state.

CBF in ACC state: In this state, safety-critical control should keep the distance between the ego vehicle and vehicle fc greater than a pre-defined value. We refer to the distance constraints and force based constraints in [9] and construct the following CBF:

$$h_{fc}(\mathbf{x},t) = \begin{cases} \Delta x_{fc} - (1+\epsilon)v - \dfrac{(v_{fc}-v)^2}{2a_l} & \text{if } v \geq v_{fc} \\ \Delta x_{fc} - (1+\epsilon)v & \text{else} \end{cases} \tag{8.11}$$

If the ego vehicle is faster than its leading vehicle fc, the traveling distance during deceleration process will be considered in the $h_{fc}(\mathbf{x},t)$. Otherwise, $h_{fc}(\mathbf{x},t) \geq 0$ indicates the ego vehicle meets the safety-critical requirement directly. Converting this CBF into its corresponding constraint in the CLF-CBF-QP formulation will guarantee $h_{fc}(\mathbf{x},t)$ always greater than 0. This condition will also be used for similar expressions later.

CBFs in L or R state: In this state, all three vehicles of interest should be considered in the safety-critical control design. Following CBFs will be constructed:

$$h_{fc}(\mathbf{x},t)=\begin{cases}\Delta x_{fc}-(1+\epsilon)v-\dfrac{(v_{fc}-v)^2}{2a_l} & \text{if } v\geq v_{fc}\\ \Delta x_{fc}-(1+\epsilon)v & \text{else}\end{cases} \tag{8.12a}$$

$$h_{ft}(\mathbf{x},t)=\begin{cases}\Delta x_{ft}-(1+\epsilon)v-\dfrac{(v_{ft}-v)^2}{2a_l} & \text{if } v\geq v_{ft}\\ \Delta x_{ft}-(1+\epsilon)v & \text{else}\end{cases} \tag{8.12b}$$

$$h_{bt}(\mathbf{x},t)=\begin{cases}\Delta x_{bt}-(1+\epsilon)v_{bt}-\dfrac{(v_{bt}-v)^2}{2a_l} & \text{if } v_{bt}\geq v\\ \Delta x_{bt}-(1+\epsilon)v_{bt} & \text{else}\end{cases} \tag{8.12c}$$

Similarly to the (8.11), (8.12c) can be used to keep a safe distance between the ego vehicle and vehicle bt (or after the ego vehicle accelerates to the same speed as vehicle bt). Additionally, during a lane change maneuver, after the ego vehicle changes to its target lane, vehicle fc and bt will no longer be the vehicles of interest. Therefore, $h_{fc}(\mathbf{x},t)$ and $h_{bt}(\mathbf{x},t)$ will not be used if the ego vehicle is totally in its target lane.

CBFs in BL or BR state: In this state, since the ego vehicle will go back to its current lane, it should keep a safe distance from vehicle fc by using the CBF as in (8.13a). More importantly, hard constraints should be used to prevent a potential crash with interrupted vehicles, which can be either vehicle ft or bt. Equations (8.13b) and (8.13c) are used to prevent a crash in both longitudinal and lateral directions (if the vehicle ft or bt is overlapping with the ego vehicle longitudinally). Following CBFs are used in this case:

$$h_{fc}(\mathbf{x},t)=\begin{cases}\Delta x_{fc}-(1+\epsilon)v-\dfrac{(v_{fc}-v)^2}{2a_l} & \text{if } v\geq v_{fc}\\ \Delta x_{fc}-(1+\epsilon)v & \text{else}\end{cases} \tag{8.13a}$$

$$h_{ft}(\mathbf{x},t)=\begin{cases}\Delta x_{ft}-\dfrac{(v_{ft}-v)^2}{2a_l} & \text{if } \Delta x_{ft}\geq 0\ ,\ v\geq v_{ft}\\ \Delta x_{ft} & \text{if } \Delta x_{ft}\geq 0\ ,\ v<v_{ft}\\ \Delta y_{ft}-0.1\epsilon & \text{else}\end{cases} \tag{8.13b}$$

$$h_{bt}(\mathbf{x},t)=\begin{cases}\Delta x_{bt}-\dfrac{(v_{bt}-v)^2}{2a_l} & \text{if } \Delta x_{bt}\geq 0\ ,\ v_{bt}\geq v\\ \Delta x_{bt} & \text{if } \Delta x_{bt}\geq 0\ ,\ v_{bt}<v\\ \Delta y_{bt}-\epsilon & \text{else}\end{cases} \tag{8.13c}$$

Remark 8.4. *For the switches between different FSM states, we take the change from* $\boldsymbol{L}$ *to* $\boldsymbol{ACC}$ *state as an example to show the continuity of the system's safety under different CBF constraints. We assume that the ego vehicle does a left lane change maneuver and only vehicle* fc, ft *exist. The corresponding safe sets of CBF* (8.12a) *and* (8.12b) *are called* $\mathcal{C}_{fc}$ *and* $\mathcal{C}_{ft}$, *respectively. In* $\boldsymbol{L}$ *state, the ego vehicle's state* $\mathbf{x}$ *is in the intersections of these two sets. When the FSM enters* $\boldsymbol{ACC}$ *state, CBF* (8.11) *will be the only hard constraint in the*

controller, which will build the same safe set as $\mathcal{C}_{ft}$ since vehicle ft becomes the new leading vehicle. In this case, the intersection of sets $\mathcal{C}_{ft}$ and $\mathcal{C}_{fc}$ is the subset of $\mathcal{C}_{ft}$, which meets the proposed safety-based conditions for switches of CBF constraints in Sec. 8.3.2.

Finally, the system's optimal input will be calculated through a quadratic program, where the CLFs and CBFs in different states will be used to construct the constraints,

$$u = \underset{\left[u\ \delta_v\ \delta_y\ \delta_\psi\right]^T \in \mathbb{R}^5}{\operatorname{argmin}} \frac{1}{2} u^T H u + p_v \delta_v^2 + p_y \delta_y^2 + p_\psi \delta_\psi^2 \tag{8.14a}$$

$$\text{s.t.} \quad L_f V_j(\mathbf{x}) + L_g V_j(\mathbf{x}) u \leq -\alpha_j V_j(\mathbf{x}) + \delta_j \tag{8.14b}$$

$$\frac{\partial h_k(\mathbf{x}, t)}{\partial t} + L_f h_k(\mathbf{x}, t) + L_g h_k(\mathbf{x}, t) u \geq -\gamma_k h_k(\mathbf{x}, t) \tag{8.14c}$$

where j can represent subscript v, y or ψ and k can denote subscript fc, ft or bt. f and g are corresponding items in nonlinear affine kinematic bicycle model (8.2). According to the FSM's current state, different combinations of CBFs and CLFs will be used for the optimization problem (8.14). This quadratic program carries a low computational cost, which makes the controller work at a high update frequency.

Remark 8.5. *Notice that the barrier functions used in this chapter are time-varying functions. When we construct the CBF constraints in* (8.14)*, we must use surrounding vehicle's time-varying speed and acceleration information to calculate corresponding partial derivatives.*

8.4 Results

Having introduced the ruled-based safety-critical lane change controller, we will validate our algorithm through numerical simulations in this section. Simulations are also illustrated in a supplement video[1]. The controller is simulated to work at an update rate of 100Hz and parameters in Table 8.2 will be used.

As discussed in Remark 8.1, we apply constraints on the system's input to satisfy the requirement of the kinematic bicycle model and physical limitations of the ego vehicle, which are shown in Table 8.3.

8.4.1 Simulation with Pre-designed Typical Scenarios

To evaluate our controller's performance, we test the algorithm on the ego vehicle in the following pre-designed typical scenarios, which can be used to mimic most driving scenes. For these scenarios, our ego vehicle is simulated to start from initial position $(x(0), y(0)) =$ (0m, 1.75m) with an initial speed $v(0) = 27.5$m/s and the target lane is the adjacent one

[1] `https://youtu.be/icmy9u2a4z4`

Table 8.2: Vehicle parameters and control hyperparameters.

Vehicle Parameter		**Hyperparameter**			
Parameter	Value	Parameter	Value	Parameter	Value
l_f	1.11m		$\begin{bmatrix} 0.01 & 0 \\ 0 & 0 \end{bmatrix}$	α_v	1.7
l_r	1.74m	H		α_p	0.8
l_{fc}	2.15m	ϵ	0.5	α_s	12
l_{rc}	2.77m	p_v	0.1	γ_{fc}	1
w_{lc}	0.93m	p_p	15	γ_{ft}	1
w_{rc}	0.93m	p_s	400	γ_{rt}	1

Table 8.3: Vehicle input bounds.

	β	$\dot{\beta}$	a	a_y
minimum	15	15s	$0.3g$	$0.3g$
maximum	-15	-15s	$-0.3g$	$-0.3g$

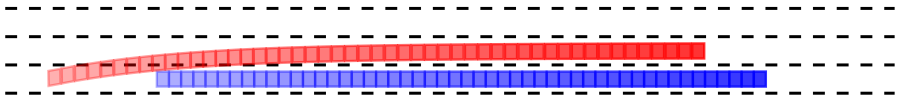
Table 8.4: Initial setups for lane change numerical simulations.

Simulation	Initial Position $(x_1(0), y_1(0))$	Constant Speed $v_1(t)$
1	(55 m, 1.75 m)	22 m/s
2	(-15 m, 5.25 m)	19 m/s
3	(3 m, 8.75 m)	33 m/s

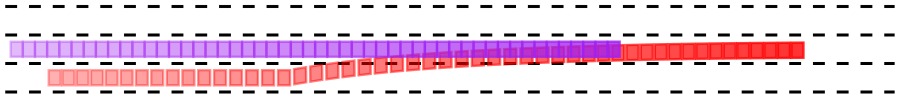
on the left. The width of each lane is 3.5m, the ego vehicle's speed limit is $v_l = 33.33$ m/s and its desired speed is set as $v_d = 27.5$ m/s. Another surrounding vehicle is also simulated and it is initialized with a different position and constant speed in each simulation, shown in Table 8.4. This surrounding vehicle is set to stay on its original lane for first two simulations and it is required to change to the same target lane as the ego vehicle in the third one.

The numerical simulations of these scenarios are shown in Figure 8.5 by snapshots. The ego vehicle's speed, front steering angle and FSM's state are plotted in Figure 8.6.

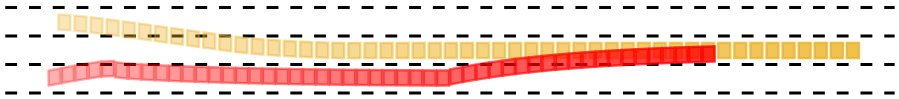
Overtaking a slow leading vehicle occurs frequently and the first simulation is a simple example of this overtaking maneuver. The ego vehicle decelerates to keep away from the slow leading vehicle and changes the lane. Additionally, during the lane change maneuver, the ego vehicle should pay attention to the approaching vehicle in its target lane. Our controller is shown to guarantee the safety of both ego vehicle and this coming vehicle in our second simulation. The ego vehicle accelerates to gain enough space to change the lane. Finally, a dangerous scenario that may happen during a lane change maneuver is that two vehicles try to change to the same lane at the same time due to lack of observation before changing the lane. In the third simulation, the ego vehicle avoids the other vehicle by driving back to its current lane and then changes to its target lane once it is safe. Our ego vehicle is able to



(a) Simulation 1: ego vehicle overtakes a slow leading one.



(b) Simulation 2: a target lane's surrounding vehicle approaches ego one.



(c) Simulation 3: ego vehicle avoids another surrounding vehicle.

Figure 8.5: Snapshots of three lane change simulations.

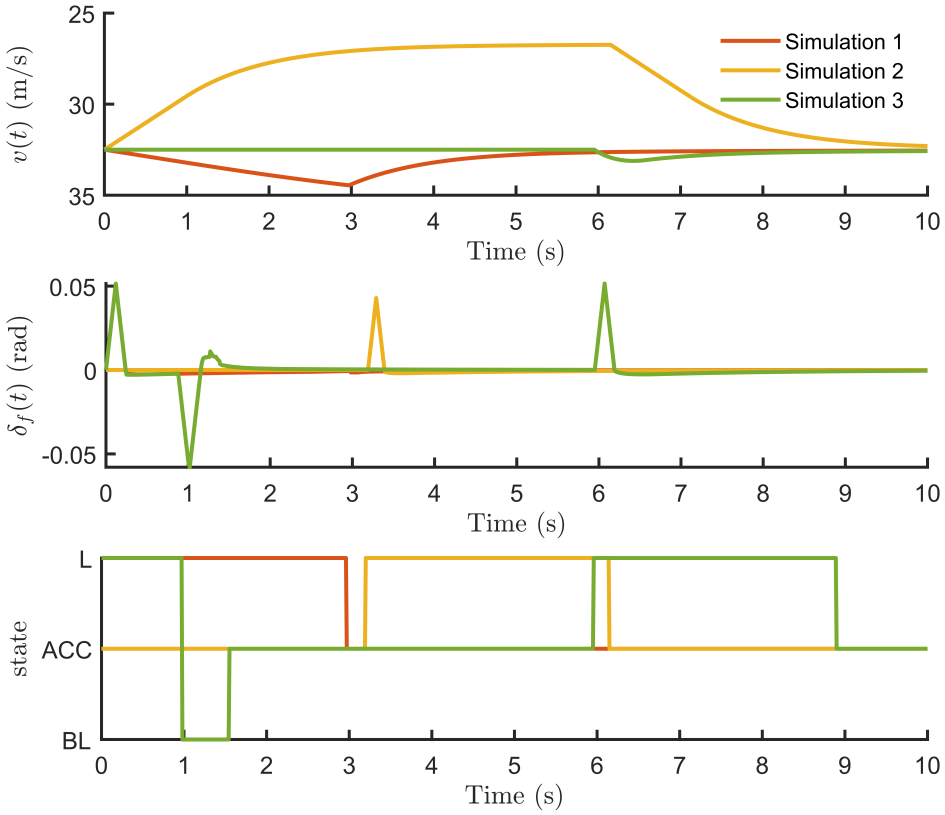


Figure 8.6: Ego vehicle's speed, front steering angle and FSM's state during simulations.

achieve the lane change maneuver and ensure its safety along the trajectory.

8.4.2 Simulations with Randomly Generated Tests

To show statistically the robustness of our control design in different environments, we also deploy tests with numerous driving scenarios with randomly generated surrounding vehicles. We will mimic highway and urban road environments where two groups of random scenarios will be generated. Notice that we adopt the suggested values of speed ranges and lane widths provided by US Federal Highway Administration (FHWA) [182] to generate two groups of random tests.

We can summarize the simulations' basic settings of two groups as follows: only ego vehicle is equipped with our controller and will start from initial position $x(0) = 0$m and $y(0)$ is equal to the y coordinate of the initial lane's center line in frame E with initial speed $v(0)$, whose value is determined by the driving scenario. The target lane of the ego vehicle is the adjacent one on the left. There will be six vehicles generated randomly around the ego vehicle. One vehicle (denoted as 1st vehicle) is in the initial lane and another four vehicles (denoted as 2nd to 5th vehicle, respectively) are in the ego vehicle's target lane. These five vehicles will move with random initial speed and random acceleration. Finally, one vehicle (denoted as 6th vehicle) is simulated to be in the left adjacent lane of target lane. It will change to the same target lane with a constant speed and begin at a random position. All of above six surrounding vehicles have a vehicle speed lower bound and upper bound, which are determined by the corresponding scenario. Examples of randomly generated scenarios are shown in Figure 8.7, where red is the ego vehicle. Random tests' relevant data are summarized in Table 8.5. We do 5000 simulations with a length of 60s for every scenario and a statistical analysis of our controller's performance in urban roads and highway are shown in Table 8.6.

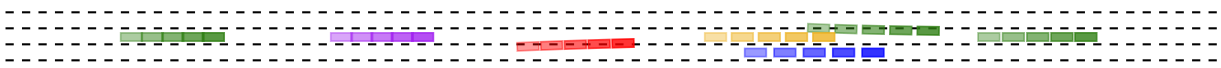
From the results, we find that more than 55% of the simulations finish the lane change maneuver under 60 seconds. Less than 0.5% of simulations fail to find a solution for the CLF-CBF-QP formulation. In other cases, the ego vehicle does not change to the target lane but meets all safety-critical requirements during the simulations. In order to explore the reasons of later two kinds of results, we look back on details of simulations. The randomly generated vehicles may move at a similar speed with the ego vehicle. Some of them in ego vehicle's target lane have small longitudinal distance with the ego vehicle, which makes it impossible to change the lane (see Figure 8.7b). Our controller fails when a travelled vehicle, like vehicle ft, passes "through" a slow moving vehicle since we don't constraint the distance between normal vehicles in our simulator (see Figure 8.7c). When this happens, the slow vehicle becomes vehicle ft and the CBF (8.13b) consequently becomes negative, which will make the quadratic program infeasible. Note that vehicles passing "through" other vehicles will not occur in real life. Additionally, in the real world, drivers will respond to surrounding vehicle's lane change maneuver to ensure the safety and avoid potential threats, for example, decelerate to keep away from cut-in vehicles. This kind of response is not considered in our random test, which makes it more challenging for the controller.

Table 8.5: Simulation setup for random tests for urban road and highway.

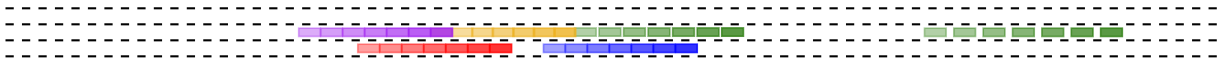
Notations	Values	Values
	Urban Road	**Highway**
Driving Lane's Width	3m	3.6m
Ego Vehicle's $v(0)$	13m/s	29m/s
Ego Vehicle's v_d	13m/s	29m/s
Ego Vehicle's v_l	16.67m/s	33.33m/s
1st vehicle's $x_1(0)$ in E	[25m,40m]	[50m,65m]
2nd-5th vehicle's $x_k(0)$ in E	[-50m,50m]	[-85m,85m]
6th vehicle's $x_6(0)$ in E	[-50m,50m]	[-85m,85m]
1st-6th vehicle's $v_k(0)$	[11m/s,15m/s]	[26m/s,32m/s]
1st-5th vehicle's a_k	[-2m/s^2,2m/s^2]	[-3m/s^2,3m/s^2]
Speed lower and upper bound	[10m/s,16.67m/s]	[23m/s,33.33m/s]

Table 8.6: Simulation results of 5000 groups of lane-change simulations.

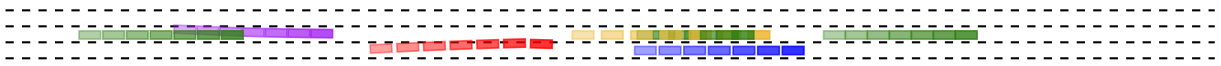
Result	In City	On Highway
Change the lane successfully	62.46%	55.58%
Still in the current lane after 60s	37.06%	44.22%
CLF-CBF-QP formulation is not solvable	0.48%	0.20%



(a) The ego vehicle changes to its target lane in a randomly generated driving scenario successfully.



(b) The ego vehicle is still in its current lane after 60s since surrounding vehicles move with similar speed and the distance between them is small.



(c) The ego vehicle fails to do a lane change maneuver. A fast vehicle (yellow one) drives "through" a slow vehicle (one of the green ones). The change of vehicle ft in the controller makes the barrier function $h_{ft}(\mathbf{x}, t)$'s value change suddenly. This scenario will not happen in real world.

Figure 8.7: Selected examples of randomly generated scenarios.

8.5 Discussion

In our proposed algorithm, we use a kinematic bicycle model with small angle assumption on the slip angle for control design. However, since the kinematic bicycle model is based on the zero slip angle assumption, it asks us to limit the ego vehicle's lateral acceleration, which limits the controller's performance. The small angle assumption also causes a mismatch between the real dynamics model and our approximated nonlinear affine dynamics model. Therefore, it is desirable to explore the use of a nonlinear non-affine dynamic bicycle model in the future work. Besides, in this work, we assume the perception system is able to sense surrounding environment accurately and the controller can receive the sensors' measurements without time delay. To enhance the performance of our safety-critical controller, we need to incorporate a state estimator and introduce a safety margin for the noisy data from distance estimation between the ego vehicle and surrounding ones. The communication time cost between sensors and controller should also be considered in the future work.

8.6 Chapter Summary

In this chapter, we presented a safety-critical lane change control design using rule-based CLF-CBF-QP formulation. Utilizing a FSM, we divide a lane change maneuver into different states, which allows the proposed strategy to function as both a planner and a controller. A quadratic program based safety-critical control is applied to achieve optimal lane change motion while guaranteeing safety. We tested the proposed control design using three pre-designed typical driving scenarios and 5000 randomly generated tests in different scenarios to show the effectiveness and robustness.

In this chapter, we focused on the finite state machine approach for motion planning in autonomous driving problems. In the next chapter, we will discuss how parallelism together with control barrier functions can enhance the performance about autonomous racing problems.

Chapter 9

Autonomous Racing using Control Barrier Functions with Parallel Computation for Trajectory Generation [7]

9.1 Introduction

9.1.1 Motivation

Recently, autonomous racing is an active subtopic in the field of autonomous driving research. In autonomous racing, the ego car is required to drive along a specific track with an aggressive behavior, such that it is capable of competing with other agents on the same track. By overtaking other leading vehicles and moving ahead, the ego vehicle can finish the racing competition with a smaller lap time. While the behavior of overtaking other vehicles has been studied in autonomous driving on public roads, however, these techniques are not effective on a race track. This is because autonomous vehicles are guided by dedicated lanes on public roads to succeed in lane follow and lane change behaviors, while the racing vehicles compete in the limited-width tracks without guidance from well-defined lanes. Existing work focuses on a variety of algorithms for autonomous racing, but most of them could not provide a time-optimal behavior with high update frequency in the presence of other moving agents on the race track. In order to generate racing behaviors for the ego racing car, we propose a racing algorithm for planning and control that enables the ego vehicle to maintain time-optimal maneuvers in the absence of local vehicles, and fast overtake maneuvers when local vehicles exist, as shown in Figure 9.1.

[7] The material of this chapter is from "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions" published in 2022 IEEE International Conference on Robotics and Automation (ICRA) [73].

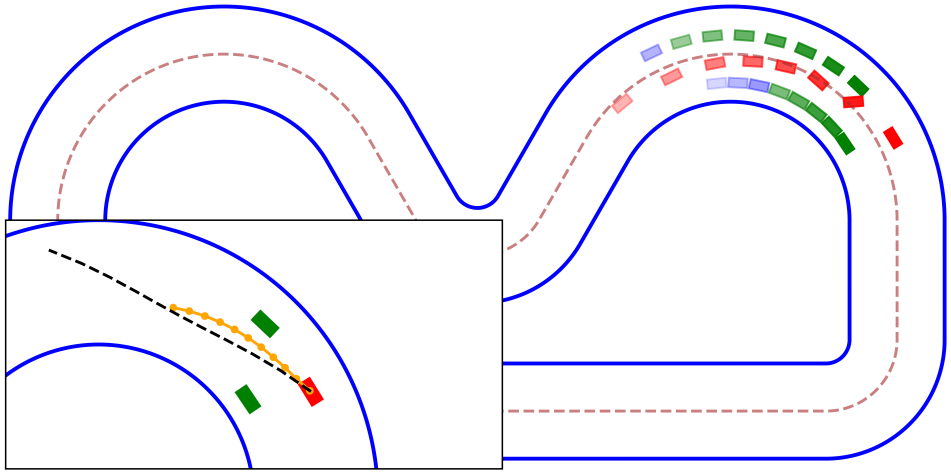


Figure 9.1: Snapshots from simulation of the overtaking behavior.

9.1.2 Related Work

In recent years, researchers have been focusing on planning and control for autonomous driving on public roads. For competitive scenarios like autonomous lane change or lane merge, both model-based methods [166] and learning-based methods [220] have been demonstrated to generate the ego vehicle's desired trajectory. Similarly, control using model-based methods [192, 74, 123] and learning-based methods [106] has also been developed. However, the criteria to evaluate planning and control performance are different for car racing compared to autonomous driving on public roads. For autonomous racing [121], when the ego racing car competes with other surrounding vehicles, most on-road traffic rules are not effective. Instead of maneuvers that offer a smooth and safe ride, aggressive maneuvers that push the vehicle to its dynamics limit [148] or even beyond its dynamics limit [67] are sought to win the race. In order to quickly overtake surrounding vehicles, overtake maneuvers with tiny distances between the cars and large orientation changes are needed. Moreover, due to the bigger slip angle caused by changing the steering orientation more quickly during racing, more accurate dynamical models should be used for autonomous racing planning and control design. We next enumerate the related work in several specific areas.

9.1.2.1 Planning Algorithms

For autonomous racing, the planner is desired to generate a time-optimal trajectory. Although some work using convex optimization problems [117, 97, 25, 31, 75, 180, 4] or Bayesian optimization (BO) [89] reduces the ego vehicle's lap time impressively, either no obstacles [97, 25, 31, 75, 180, 4, 89] or only static obstacles [117] are assumed to be on the track. When moving vehicles exist on the track, nonlinear dynamic programming (NLP)

Approach	*GP*	*DRL*	*Graph-Search*	*Game Theory*		*Model-Based*						
Publication	[78]	[59]	[181]	[118]	[198]	[75]	[161]	[95]	[98]	[51]	[29]	[224]	**Ours**
Lap Timing	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No	**Yes**
Static Agent	No	No	**Yes**	**Yes**	**Yes**	No	No	No	No	**Yes**	**Yes**	**Yes**	**Yes**
Moving Agent	No	No	One	One	**Multiple**	No	No	No	No	**Multiple**	One	One	**Multiple**
Update Frequency (Hz)	N/A	N/A	15	**30**	2	Offline	**20**	**20**	Offline	<1	10	<10	**>25**
Planner	No	No	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No	**Yes**	**Yes**	No	**Yes**
Dynamics Accuracy	N/A	N/A	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Table 9.1: A comparison of recent work on autonomous racing and their attributes.

[51], graph-search [181] and game theory [118, 198, 94] based approaches have demonstrated their capabilities to generate collision-free trajectories. Additionally, in order to improve the chance of overtaking, offline policies are learnt for the overtake maneuvers at different portions of a specific track [21]. However, these approaches don't solve all challenges. For instance, work in [51, 198, 94, 21] does not take lap timing enhancement into account. In [181], the ego vehicle is assumed to compete on a straight track with one constant-speed surrounding vehicle. These assumptions are relatively simple for a real car racing competition. In [118], it is assumed that the planner knows the other vehicle's strategy and the complexity of the planner increases excessively when multiple vehicles compete with each other on the track.

9.1.2.2 Control Algorithms

Researchers focus on enhancing performance of the ego vehicle by achieving its speed and steering limits through better control design, e.g., obtaining the optimal lap time by driving fast. The majority of existing work focuses on developing controllers with no other vehicles on the track. The learning-based controllers [161, 95, 98, 199] leverage the control input bounds to achieve optimal performance in iterative tasks. Model-free methods like Bayesian optimization (BO) [140], Gaussian processes (GPs) [78], deep neural networks (DNN) [146, 197] and deep reinforcement learning (DRL) [153, 59] have also been exploited to develop controllers that result in agile maneuvers for the ego car. To deal with other surrounding vehicles, DRL has also been used in [176] to control the ego vehicle during overtake maneuvers. Recently, model predictive based controllers (MPC) with nonlinear obstacle avoidance constraints have become popular to help the ego vehicle avoid other vehicles in the free space. A nonconvex nonlinear optimization based controller is implemented in [162] to help the ego vehicle avoid static obstacles. Researchers in [111] use mixed-integer quadratic programs (MIQP) to help the ego vehicle compete with one moving vehicle. In [29], GPs was applied to formulate the distance constraints of a stochastic MPC controller with a kinematic bicycle model. However, large slip angles under aggressive maneuvers will cause a mismatch between real dynamics model and the kinematic model used in the controller, resulting in the controller being unable to guarantee the system's safety in some cases. In [224], a safety-critical control design by using control barrier functions is proposed to generate a collision-free trajectory without a high-level planner, where infeasibility could arise due to the high nonlinearity of the optimization problem. Moreover, due to the lack of a trajectory planner, deadlock could happen very often during overtake maneuvers, such as in [224, 29]. A comparison of various approaches and their features are enumerated in Table 9.1.

As mentioned above, all the previous work on planning and control design for autonomous racing could not enhance the lap timing performance and simultaneously compete with multiple vehicles. Inspired by the work on iterative learning-based control and optimization-based planning, we propose a novel racing strategy to resolve the challenges mentioned above with a steady low computational complexity.

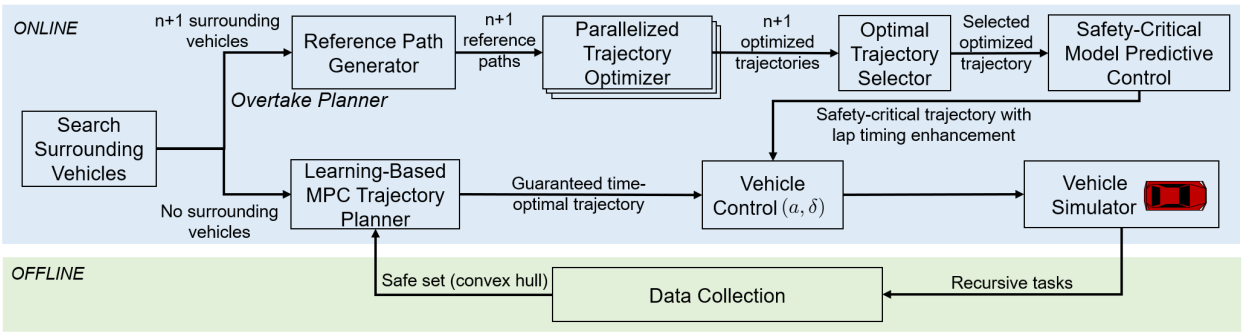


Figure 9.2: Autonomous Racing Strategy.

9.1.3 Contribution

The contributions of this chapter are as follows:

- We present an autonomous racing strategy that switches between a learning-based MPC trajectory planner (in the absence of surrounding vehicles) and optimization-based homotopic trajectory planner with a low-level safety-critical controller (when the ego vehicle competes with surrounding vehicles).
- The learning-based MPC approach guarantees time-optimal performance in the absence of surrounding vehicles. When the ego vehicle competes with surrounding vehicles, multiple homotopic trajectories are optimized in parallel with different geometric reference paths and the best time-optimal trajectory is selected to be tracked with an optimization-based controller with obstacle avoidance constraints.
- We validate the robust performance together with steady low computational complexity of our racing strategy in numerical simulations where randomly moving vehicles are generated on a simulated race track. It is shown that our proposed strategy allows the ego vehicle to succeed in overtaking tasks without deadlock when there are multiple vehicles moving around the ego vehicle. We also demonstrate that our strategy would work for various racing environments.

9.2 Background

In this section, we revisit the vehicle model and learning-based MPC for iterative tasks. The learning-based MPC will be used as the trajectory planner when no surrounding vehicles exist.

9.2.1 Vehicle Model

In this work, we use a dynamic bicycle model with decoupled Pacejka tire model under Frenet coordinates. The system dynamics is described as follows,

$$\dot{x} = f(x, u), \tag{9.1}$$

where x and u show the state and input of the vehicle, f is a nonlinear dynamic bicycle model in [150]. The definition of state and input is as follows,

$$x = [v_x, v_y, \omega_z, e_\psi, s_c, e_y]^T, \ u = [a, \delta]^T, \tag{9.2}$$

where acceleration at vehicle's center of gravity a and steering angle δ are the system's inputs. s_c denotes the curvilinear distance travelled along the track's center line, e_y and e_ψ show the deviation distance and heading angle error between vehicle and center line. v_x, v_y and ω_z are the longitudinal velocity, lateral velocity and yaw rate, respectively.

In this chapter, this model (9.1) is applied for precise numerical simulation using Euler discretization with sampling time 0.001s (1000Hz). Through linear regression from the simulated reference path, an affine time-invariant model as below,

$$x_{t+1} = A(\bar{x})x_t + B(\bar{x})u_t \tag{9.3}$$

will be used in the trajectory planner to avoid excessive complexity from nonlinear optimization, where $\bar{x}$ represents the equilibrium point for linearized dynamics. On the other hand, an affine time-varying model as below,

$$x_{t+1} = A_t(\bar{x}_k)x_t + B_t(\bar{x}_k)u_t + C_t(\bar{x}_k) \tag{9.4}$$

where matrices $A_t(\bar{x}_k)$, $B_t(\bar{x}_k)$, and $C_t(\bar{x}_k)$ are obtained at local equilibrium point $\bar{x}_k$ on reference trajectory with iterative data which is close to x_t. The dynamics (9.4) will be used on racing controller design for better tracking performance.

9.2.2 Iterative Learning Control

A learning-based MPC [161], which improves the ego vehicle's lap timing performance through iterative tasks, will be used in this chapter. This has the following components:

9.2.2.1 Data Collection

The learning-based MPC optimizes the lap timing through historical states and inputs from iterative tasks. To collect initial data, a simple tracking controller like PID or MPC can be used for the first several laps. During the data collection process, after the j-th iteration (lap), the controller will store the ego vehicle's closed-loop states and inputs as vectors. Meanwhile, through offline calculation, every point of this iteration will be associated with a cost, which describes the time to finish the lap from this point.

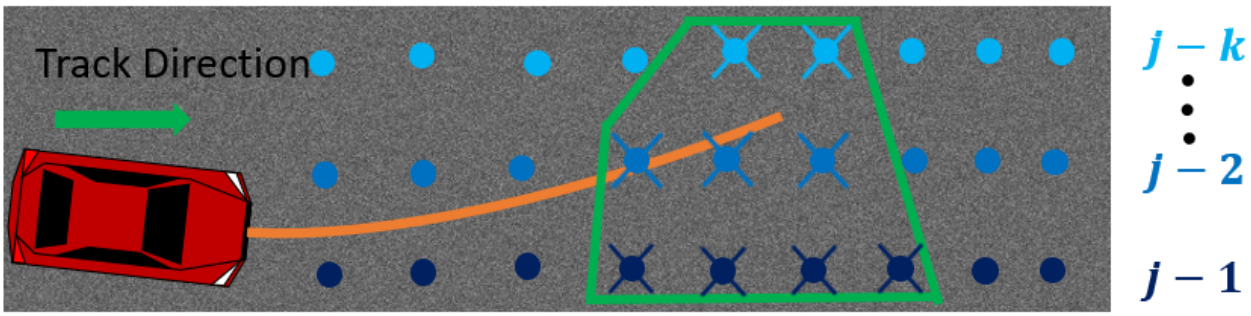


Figure 9.3: Illustration of learning-based MPC.

9.2.2.2 Online Optimization

After the initial laps, the learning-based MPC optimizes the vehicle's behavior based on collected data. At each time step, the terminal constraint is formulated as a convex set (green convex hull in Figure 9.3). This convex set includes the states that can drive the ego vehicle to the finish line in the previous laps. By constructing the cost function to create a minimum-time problem, an open-loop optimized trajectory can be generated. Since the cost function is based on the previous states' timing data, the vehicle is able to drive to the finish line with time that is no greater than the time from the same position during previous laps. As a result, the ego vehicle will reach the time-optimal performance after several laps.

More details of this method can be found in [161]. In our work, this approach will be used for trajectory planning when the ego vehicle has no surrounding vehicles. This helps with better lap timing without surrounding vehicles. Notice that the data for iterative learning control will be collected through offline simulation with no obstacles on the track, as shown in Figure 9.2.

9.3 Racing Algorithm

After introducing the background of vehicle modeling and learning-based MPC, we will present an autonomous racing strategy that can help the ego vehicle enhance lap timing performance while overtaking other moving vehicles.

9.3.1 Autonomous Racing Strategy

There are two tasks in autonomous racing: enhancing the lap timing performance and competing with other vehicles. To deal with these two problems, our proposed strategy will switch between two different planning strategies. When there are no surrounding vehicles, trajectory planning with learning-based MPC is used to enhance the timing performance through historical data. Once the leading vehicles are close enough, an optimization-based trajectory planner optimizes several homotopic trajectories in parallel and the collision-free

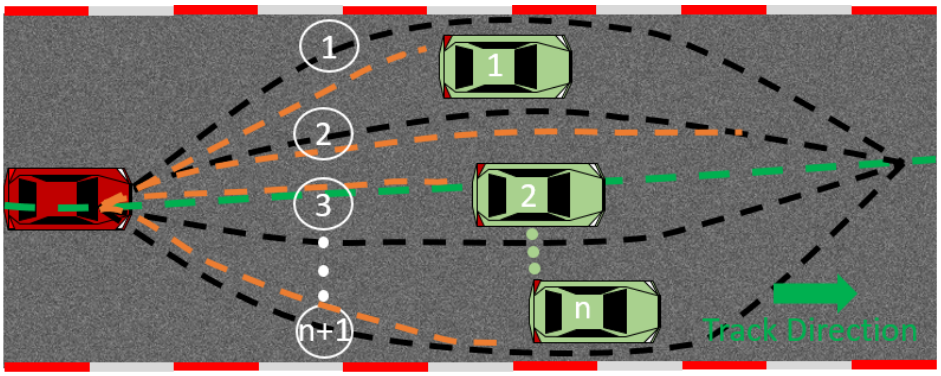


Figure 9.4: A typical overtaking scenario when there are n vehicles in the range of overtaking.

optimal trajectory is selected with an optimal-time criteria, which will be tracked by a low-level MPC controller. By adding obstacle avoidance constraints to the low-level controller, it has the ability to guarantee the system's safety. The racing strategy is summarized in Figure 9.2.

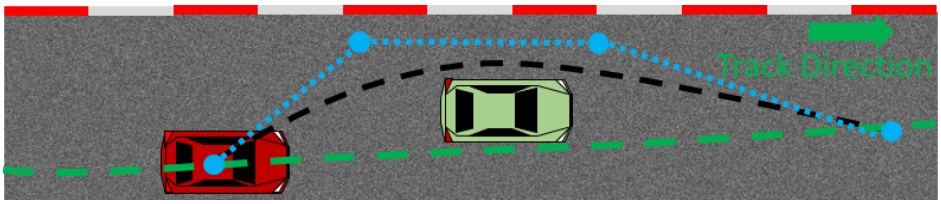
9.3.2 Overtaking Planner

To determine if a surrounding vehicle is in the ego vehicle's range of overtaking, following condition must be satisfied:

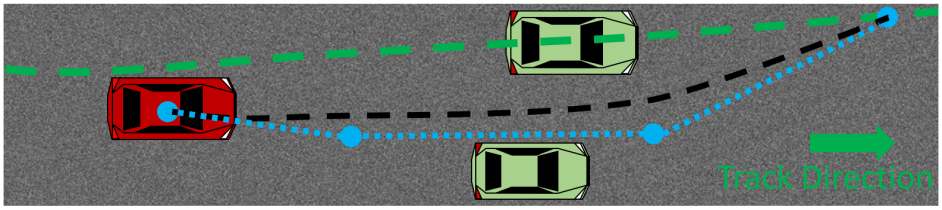
$$-\epsilon l \leq s_{c,i} - s_c \leq \epsilon l + \gamma |v_x - v_{x,i}| \tag{9.5}$$

where s_c and $s_{c,i}$ are ego vehicle's and i-th surrounding vehicle's traveling distance, v_x and $v_{x,i}$ are ego vehicle's and i-th surrounding vehicle's longitudinal speed. l indicates the vehicle's length. ϵ and γ are safety-margin factor and prediction factor which we can tune for different performance.

As shown in Figure 9.4, when there are n vehicles in the ego vehicle's range of overtaking, there exists $(n+1)$ potential areas, each leading to paths with a different homotopy, that the ego vehicle can use to overtake these surrounding vehicles. These areas are the one below the n-th vehicle, the one above the 1st vehicle, and the ones between each group of adjacent vehicles. n+1 groups of optimization-based trajectory planning problems are solved in parallel, enabling steady low computational complexity even when competing with different numbers of surrounding vehicles. To reduce each optimization problem's computational complexity through fast convergence, geometric paths with a distinct homotopy class that laterally lay between vehicles or vehicle and track boundary (black dashed curves in Figure 9.4) are used as reference paths in the optimization problems. By comparing the optimization problems' costs, the optimal trajectory is selected from $n+1$ optimized solutions. For example, as the case shown in Figure 9.4, the dashed orange line in Area 2 will be selected since it avoids surrounding vehicles and finishes overtake maneuver with smaller time. The function to



(a) Control points for the Bezier-curve next to the track boundary.



(b) Control points for the Bezier-curve between two adjacent vehicles.

Figure 9.5: Third order Bezier-curves (dashed black lines) and their control points (blue points) for two different cases.

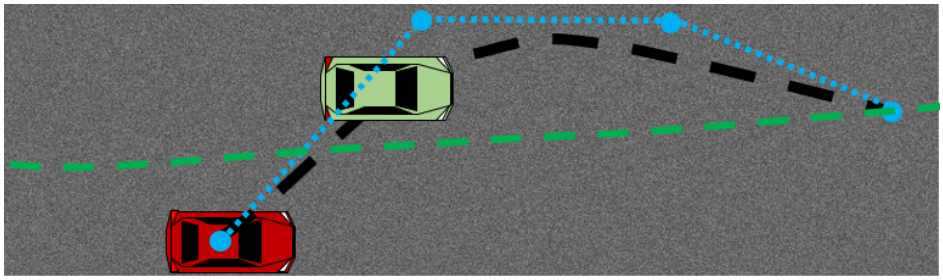


Figure 9.6: One typical scenario where the reference Bezier-curve has a conflict with the surrounding vehicle when approaching with a large lateral difference.

minimize during the selection is shown as follows,

$$J_s(x_t) = \min_{x_t} -K_s(s_{c_{t+N}} - s_{c_t}) - \sum_{k=1}^{N_p}((s_{c_{t+k}} - s_{c,i_{t+k}})^2 + (e_{y_{t+k}} - e_{y,i_{t+k}})^2 - l^2 - d^2) + b \tag{9.6}$$

where K_s is a scalar used in metric for timing and b is a non-zero penalty cost if the new potential area of overtaking is different from the area of overtaking in the last time step. A bigger value of K_s is applied such that the ego vehicle is optimized to reach a farther point during the overtake maneuver, which results in a shorter overtaking time since the planner's prediction horizon and sampling time are fixed. Additionally, the other terms in (9.6) prevents the ego vehicle from changing direction abruptly during an overtake maneuver and guarantees the ego vehicle's safety.

Bezier-curves are widely used in path planning algorithms in autonomous driving research [72, 34, 149] because it is easy to tune and formulate. Third-order Bezier-curves are used in

this work. Each Bezier-curve is interpolated from four control points, including shared start and end points with two additional intermediate points, shown in Figure 9.5. Specifically, the start point for the Bezier curve is the ego vehicle's current position and end point is on the time-optimal trajectory generated from learning-based MPC planner. The selection of end point makes vehicle's state as close as possible to the time-optimal trajectory after overtake behavior. To make all curves smoother and have no or fewer conflicts with surrounding vehicles, the other two control points will be between the track's boundary and vehicle for Areas 1 and n+1, shown Figure 9.5a, or between two adjacent vehicles, shown in Figure 9.5b. These two intermediate control points will have the same lateral deviation from the center line. The key advantage of our selection of control points is that the interpolated geometric curve won't cross the connected lines between control points with its convexity, shown in Figure 9.5. This property makes our reference paths collision-free with respect to surrounding vehicles in most cases, which speeds up the computational time of the trajectory generation at each area.

Remark 9.1. *In order to avoid mixed-integer and nonlinear optimizations to speed up computation, the planners are separated into (a) the overtaking planner that generates different homotopic reference paths; and (b) trajectory generator that generates dynamically-feasible and collision-free trajectories (in parallel) based on the generated homotopic reference paths. The overtaking planner generates potential reference geometric paths for obstacle avoidance by choosing control points of the Bezier curves in different homotopic regions. However, this does not guarantee the reference path is collision free. For instance, when the ego vehicle is approaching other surrounding vehicles with a big lateral difference (see Figure 9.6), the Bezier-curve path used in the planner might not be collision-free with other surrounding vehicles (while the Bezier control points are.) The parallelized trajectory optimizer then runs multiple trajectory optimizations in parallel for each homotopy class, warm starting from the corresponding Bezier curve. One of the generated dynamically-feasible and collision-free trajectories is then chosen to be sent to the overtaking controller.*

The details of the optimization formulation for trajectory generation will be illustrated in the next section.

9.3.3 Trajectory Generation

After illustrating the planing strategy, this subsection will show the details about the optimization problem used for trajectory generation for each potential area with different homotopic paths that the ego vehicle can use to overtake the surrounding vehicles.

The optimization problem is formulated as follows,

$$\begin{aligned}
&\underset{x_{t:t+N_p|t}, u_{t:t+N_p-1|t}}{\operatorname{argmin}} \quad p(x_{t+N|t}) + \sum_{k=0}^{N_p-1} q(x_{t+k|t}) \\
&\quad + \sum_{k=1}^{N_p-1} r(x_{t+k|t}, u_{t+k|t}, x_{t+k-1|t}, u_{t+k-1|t}) && \text{(9.7a)} \\
&\text{s.t. } x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t}, k = 0, ..., N_p{-}1 && \text{(9.7b)} \\
&\quad x_{t+k+1|t} \in \mathcal{X}, u_{t+k|t} \in \mathcal{U}, k = 0, ..., N_p{-}1 && \text{(9.7c)} \\
&\quad x_{t|t} = x_t, && \text{(9.7d)} \\
&\quad g(x_{t+k+1|t}) \geq d + \epsilon, k = 0, ..., N_p{-}1 && \text{(9.7e)}
\end{aligned}$$

where (9.7b), (9.7c), (9.7d) are constraints for system dynamics, state/input bounds and initial condition. The system dynamics constraint describes the affine linearized model described in (9.3). The cost function (9.7a) is composed with three parts along the horizon length N_p, the terminal cost $p(x_{t+N|t})$, the stage cost $q(x_{t+k|t})$ and the state/input changing rate cost $r(x_{t+k|t}, u_{t+k|t})$. The construction of cost function and constraints in the optimization will be presented in detail in the following subsections.

9.3.3.1 Terminal Cost

Terminal cost is related to the ego vehicle's traveling distance along the track during overtaking process:

$$p(x_{t+N|t}) = K_d(s_{c_{t+N|t}} - s_{c_t}) \tag{9.8}$$

This compares the open-loop predicted traveling distance at the N-th step $s_{c_{t+N|t}}$ with the ego vehicle's current traveling distance s_{c_t}. This works as the cost metric for timing during the overtaking process.

9.3.3.2 Stage Cost

The stage cost introduces the lateral position differences between the open-loop predicted trajectory and other two paths along the horizon:

$$q(x_{t+k|t}) = ||x_{t+k|t} - x_R(s_{c_k})||^2_{Q_1} + ||x_{t+k|t} - x_T(s_{c_k})||^2_{Q_2} \tag{9.9}$$

Here x_R and x_T are the reference path and time-optimal trajectory in Frenet coordinates. The time-optimal trajectory is generated by the learning-based MPC trajectory planner used on a track without other agents, as discussed in Sec. 9.2.2. s_{c_k} is an initial guess for the traveling distance at the k-th step, which is equal to $s_{c_k} = s_{c_t} + v_{x_t} k\Delta_t$, where a constant longitudinal speed is assumed along the prediction horizon.

9.3.3.3 State/Input Changing Rate Cost

To make the predicted trajectory smoother, the state/input changing rate cost $r(x_{t+k|t}, u_{t+k|t})$ is formulated as follows:

$$r(x_{t+k|t}, u_{t+k|t}, x_{t+k-1|t}, u_{t+k-1|t}) = ||x_{t+k|t} - x_{t+k-1|t}||^2_{R_1} + ||u_{t+k|t} - u_{t+k-1|t}||^2_{R_2} \tag{9.10}$$

9.3.3.4 Obstacle Avoidance Constraint

In order to generate a collision-free trajectory, collision avoidance constraint (9.7e) is added in the optimization problem. To reduce computational complexity, only linear lateral position constraint will be added when the ego vehicle overlapes with other vehicles longitudinally. $|s_c(t) + v_x(t)k\Delta t - s_{c,i}(t+k)| < l + \epsilon$ will be used to check if the ego vehicle is overlapping with other vehicle longitudinally along the horizon. In (9.7e), $g(x) = |e_{y,i} - e_y|$ shows the lateral position difference, l and d are the vehicle's length and width, ϵ is a safe margin.

After parallel computation, the optimized trajectory $x^*_{t:t+N|t}$ with the minimum cost $J_s(x_t)$ discussed in (9.6) will be selected from $n+1$ groups of optimization problems. It will be tracked by the MPC controller introduced in 9.3.4.

9.3.4 Overtaking Controller

After introducing the algorithm for trajectory generation, a low-level tracking controller with model predictive control used for overtaking will be discussed in this part. The constrained optimization problem is described as follows:

$$\underset{\tilde{u}_{t:t+N_c-1|t}, \omega_{1:N_c-1}}{\operatorname{argmin}} \sum_{k=0}^{N_c-1} \tilde{q}(\tilde{x}_{t+k|t}, \tilde{u}_{t+k|t}) + p_\omega(1-\omega_k)^2 \tag{9.11a}$$

$$\text{s.t. } \tilde{x}_{t+k+1|t} = A_{t+k|t}\tilde{x}_{t+k|t} + B_{t+k|t}\tilde{u}_{t+k|t} + C_{t+k|t},\ k = 0, ..., N_c-1 \tag{9.11b}$$

$$\tilde{x}_{t+k+1|t} \in \mathcal{X}, \tilde{u}_{t+k|t} \in \mathcal{U},\ k = 0, ..., N_c-1 \tag{9.11c}$$

$$\tilde{x}_{t|t} = \tilde{x}_t, \tag{9.11d}$$

$$h(\tilde{x}_{t+k+1|t}) \geq \gamma\omega_k h(\tilde{x}_{t+k|t}),\ k = 0, ..., N_c-1 \tag{9.11e}$$

where (9.11b), (9.11c), (9.11d) describe the constraints for system dynamics (9.4), input/state bounds and initial conditions, respectively. The $\tilde{q}(\tilde{x}_{t+k|t}, \tilde{u}_{t+k|t}) = ||\tilde{x}_{t+k|t} - x^*_{t+k|t}||^2_{\tilde{Q}_1} + ||\tilde{u}_{t+k|t}||^2_{\tilde{Q}_2}$ represents the stage cost, which tracks the desired trajectory $x^*_{t:t+N|t}$ optimized by the trajectory planner. Equation (9.11e) with $0 \leq \gamma < 1$ represents discrete-time control barrier function constraints [223] with relaxation ratio ω_k for feasibility [226], which could guarantee the system's safety by guaranteeing $h(\tilde{x}_{t+k|t}) > 0$ along the horizon with forward invariance. In this project, $h(\tilde{x}_{t+k|t}) = (\tilde{s}_{c,i} - \tilde{s}_c)^2 + (\tilde{e}_{y,i} - \tilde{e}_y)^2 - l^2 - d^2$ is used to represent the distance between the ego vehicle and other vehicles. The optimization (4.10) allows us to find the optimal control $u^*_t = \tilde{u}_{t|t}$ in a manner similar to MPC.

Remark 9.2. *Notice that* (9.7) *uses a distance constraint for obstacle avoidance while* (4.10) *uses control barrier functions. The reason why we don't combine these optimizations into one under dynamics* (9.11b) *is that this cascaded approach for trajectory generation and control is shown to be more computationally efficient and less likely to generate deadlock behavior.*

9.4 Results

Having illustrated our autonomous racing strategy in the previous section, we now show the performance of proposed algorithm. The setup and results of numerical simulations will be presented in the following part.

9.4.1 Simulation Setup

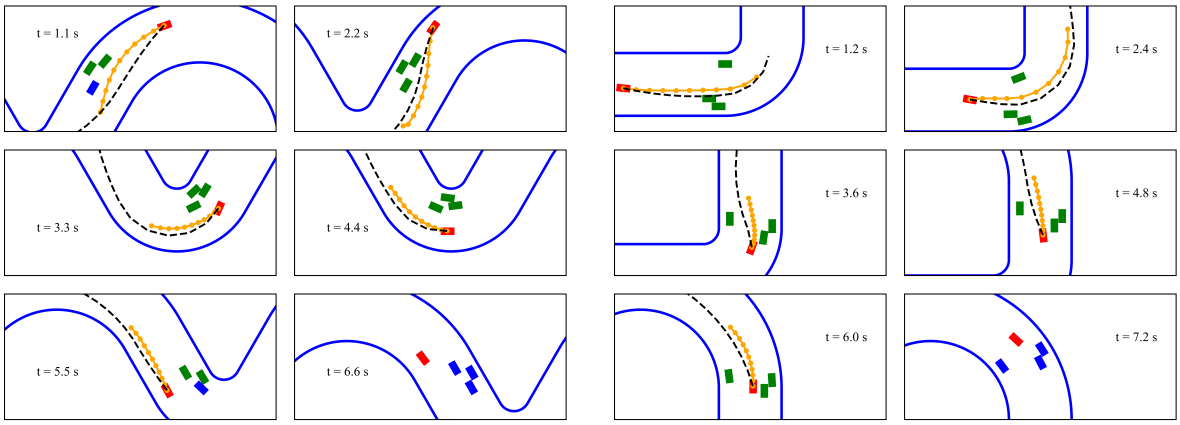
In all simulations, all vehicles are 1:10 scale RC cars with a length of 0.4m and a width of 0.2m. Other vehicles drive along trajectories with fixed lateral deviation from track's center line. The track's width is set to 2 m. The horizon lengths for trajectory planner and controller are $N_p = 12$, $N_c = 10$ and shared discretization time $\Delta t = 0.1s$. In our custom-designed simulator, both state and input noises are considered. Surrounding vehicles' speed and position information is used for trajectory generation. However, no interaction between the ego vehicle and other surrounding vehicles is included.

The optimization problems (9.7) and (4.10) are implemented in Python with CasADi [14] used as modeling language, are solved with IPOPT [22] on Ubuntu 18.04 on a laptop with a CPU i7-9850 processor at a 2.6Ghz clock rate.

9.4.2 Racing With Other Vehicles

Snapshots shown in Figure 9.7 illustrate examples of the overtaking behavior in both straight and curvy track segments when competing with other vehicles. When the ego vehicle competes with three surrounding vehicles, it could overtake them on one side of all vehicles (Figure 9.7a) or between them (Figure 9.7b). The animations of more challenging overtaking behavior can be found at `https://youtu.be/1zTXfzdQ8w4`. As shown in Table 9.1, the proposed racing planner could update at 25 Hz and could help the ego vehicle overtake multiple moving vehicles. By switching to a trajectory planning based on learning-based MPC, the ego vehicle is able to reach its speed and steering limit when there are no surrounding vehicles.

To better analyze the performance and limitations of our autonomous racing strategy in different scenarios, random tests are introduced under two groups. The first group of simulation aims to show the overtaking time for passing one leading vehicle with different speeds, and statistical results are summarized in Table 9.2. We can observe that when the surrounding vehicles' speed reaches between 1.2m/s and 1.6m/s, much more time is needed for the ego vehicle to overtake the leading vehicle. This is because as the leading vehicle's



(a) Overtaking snapshots on curvy track segment. Other surrounding vehicles move at 1.3 m/s, 1.3 m/s and 1.35 m/s.

(b) Overtaking snapshots on straight track segment. Other surrounding vehicles all move at 1.2 m/s.

Figure 9.7: Snapshots from simulation of the overtaking behavior.

Speed Range [m/s]	0 - 0.4	0.4 - 0.8	0.8 - 1.2	1.2 - 1.6
mean [s]	1.613	2.312	3.857	13.095
min [s]	0.8	1.2	1.8	3.5
max [s]	3.6	5.2	21.6	36.1

Table 9.2: Time taken to overtake the leading vehicle traveling at different speeds.

speed increases, less space becomes available for the ego vehicle to drive safely. Especially in a curve, the ego vehicle's speed limit decreases when less space can be used to make a turn. Since more than half of our track is with curves, the ego vehicle needs to wait for a straight segment to accelerate to pass the leading vehicle.

Speed Range [m/s]	0 - 0.4	0.4 - 0.8	0.8 - 1.2	1.2 - 1.6
Single	100 %	100 %	96 %	84 %
Two	100 %	100 %	98 %	66 %
Three	100 %	98 %	84 %	36 %

Table 9.3: Overtaking success rate for the ego vehicle after one lap.

The second group of simulation shows the proposed racing strategy's success rate to overtake multiple leading vehicles in one lap, and statistical results are summarized in Table 9.3. We can find that when more than one surrounding vehicle exists, much more space

would be occupied by other vehicles. As a result, the ego vehicle might not have enough space to accelerate to high speed to pass surrounding vehicles. Although in these cases, the ego vehicle can not overtake all surrounding vehicles after one lap, our proposed racing strategy can still guarantee the ego vehicle's safety along the track.

During our simulation, the mean solver time for our planner for single, two or three surrounding vehicles is 39.21ms, 39.41ms and 40.23ms. We also notice that when the number of surrounding vehicles is larger than three, the steady complexity still holds but the track becomes too crowded for the ego vehicle to achieve high success rate of overtake maneuver. This validates the steady low computational complexity of proposed planning strategy thanks to the parallel computation for multiple trajectory optimizations.

Moreover, the optimized multiple trajectory candidates offer more choices for overtaking, which helps the ego vehicle avoid deadlock.

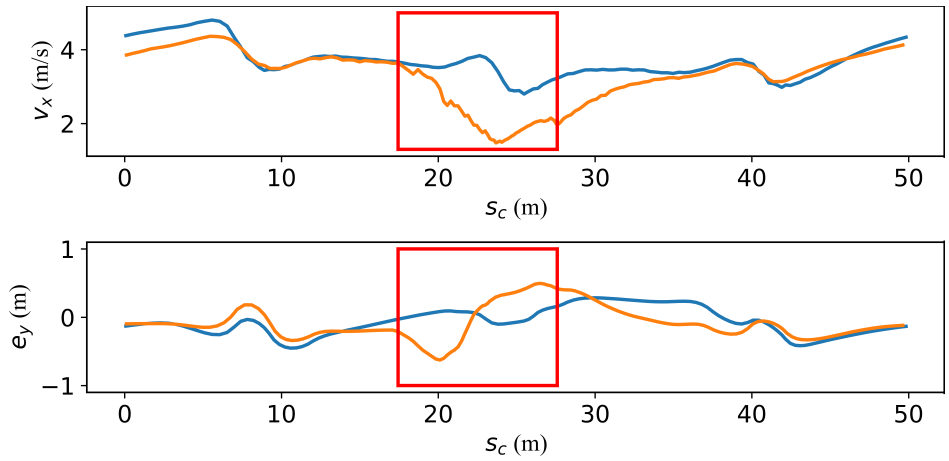


Figure 9.8: Speed and lateral deviation from the track's center line.

9.4.3 Racing Without Other Vehicles

As discussed in Sec. 9.3.1, when there are no other surrounding vehicles, the ego vehicle adopts the learning-based MPC formulation for trajectory generation and control. In this chapter, the learning-based MPC uses historical data from two previous laps (implying $k = 2$ in Figure 9.3) and the initial data are calculated offline before the racing tasks. For the same setup as shown in Figure 9.7a, the ego vehicle's speed and deviation from track's center line along the track is shown with learning-based MPC's profile in Figure 9.8. The overtake maneuver happens in a hairpin curve and the curve's apex is occupied by other moving vehicles, resulting in less space being available for the ego vehicle and thus causing it to slow down to avoid a potential collision. After it passes all surrounding vehicles, the ego vehicle goes back to drive at its speed and steering limit to achieve time-optimal behavior.

9.5 Chapter Summary

In this chapter, we have presented an autonomous racing strategy that enables an ego vehicle to enhance its lap timing performance while overtaking other moving vehicles. The proposed racing strategy switches between two modes. When there are no surrounding vehicles, a learning-based model predictive control (MPC) trajectory planner is used to guarantee that the ego vehicle achieves better lap timing performance. When the ego vehicle is competing with other surrounding vehicles to overtake, an optimization-based planner generates multiple dynamically-feasible trajectories through parallel computation. Each trajectory is optimized under a MPC formulation with different homotopic Bezier-curve reference paths lying laterally between surrounding vehicles. The time-optimal trajectory among these different homotopic trajectories is selected and a low-level MPC controller with control barrier function constraints for obstacle avoidance is used to guarantee system's safety-critical performance. The proposed algorithm has the capability to generate collision-free trajectories and track them while enhancing the lap timing performance with steady low computational complexity, outperforming existing approaches in both timing and performance for a autonomous racing environment. We have verified the performance of our proposed algorithm through numerical simulation, where several surrounding vehicles are simulated to start from random positions with random speeds on a track.

In the next chapter, we focus on bridging model-based safety and model-free RL, and this leads to a navigation task with control barrier functions, which can represented by a low-dimensional model with RL policy.

Chapter 10

Safety Navigation for Legged Robots with Reinforcement Learning [8]

10.1 Introduction

It is challenging for robotic systems to achieve control objectives with coupled dynamical models while considering input, state and safety constraints. ode-based safety-critical optimal control methods that combine control barrier functions (CBFs) [12] or Hamilton-Jacobi (HJ) reachability analysis [18] are able to provide formal safety guarantees for control, planning and navigation problems on different platforms such as autonomous vehicles [9, 175], aerial systems [35, 174] and legged robots [41, 69]. However, previous work on applying explicit formulation for such safety-critical methods are only able to be validated on relatively simple or low-dimensional systems such as a 5 Degree-of-Freedom (DoF) $2D$ bipedal robot [137] or a quadrotor with decoupled dynamics [76]. When encountering the safety-critical control problem on complex and high-dimensional systems, such as a $3D$ bipedal robot Cassie [112] which has 20 DoF and hybrid walking dynamics, model-based methods will face challenges since the full-order dynamics model of the robot is computationally not tractable for online implementation with current computing tools.

Model-free reinforcement learning (RL) methods, on the other hand, are able to leverage full-order dynamics model of the robot during offline training in simulation to provide a policy for the control purposes online. With the recent progress on solving the sim-to-real problem, RL demonstrates the capacity to control a large range of dynamic robots in the real world [86, 144, 216, 109, 143, 173, 114, 32, 219]. For example, with the help of RL, a robust and versatile walking controller for a bipedal robot Cassie is obtained in [114] to reliably track given commands of walking velocities, walking height, and turning yaw rate in real life. This RL-based controller shows significant improvements over traditional model-based

[8] The material of this chapter is from "Bridging Model-based Safety and Model-free Reinforcement Learning through System Identification of Low Dimensional Linear Models" which will appear in 2022 Robotics: Science and Systems (RSS) [113]. This paper was led by Zhongyu Li.

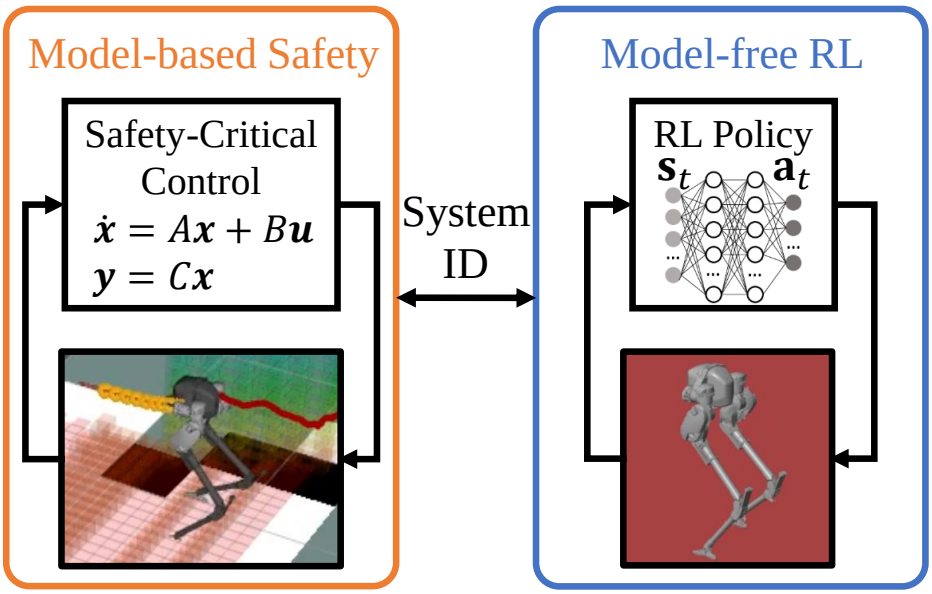


Figure 10.1: Proposed method to bridge model-based safety and model-free reinforcement learing (RL).

walking controllers [68, 112] by showing boarder feasible commands and the ability to stay robust to random perturbations.

Such robust controllers create interesting questions in the community: where does the robustness come from and whether we can formally assess the stability of such a RL-based controller/policy? Addressing such questions is very interesting as studying properties of a RL policy can further our understanding of the reason that RL demonstrates advantages on controlling highly dynamic systems. Moreover, if we can find an explicit dynamic model of a system controlled by RL, we may be able to utilize such a model to formally guarantee safety for such autonomous systems. However, this is very challenging as the policy obtained by RL is usually represented by a high dimensional nonlinear neural network and explicit analysis on nonlinear systems with RL with rigorous proofs is still an unsolved problem. In this chapter, we seek to ascertain the feasibility to find and study a low dimensional representation of a complex dynamic robot driven by a RL policy and to utilize such a simple model to provide formal guarantees on stability and safety during the task of autonomous navigation, as abstractly illustrated in Figure 10.1.

10.1.1 Related Work

10.1.1.1 Safety & Learning

There has been some exciting progress to bridge safety and learning. Previous approaches can be summarized into three classes: (a) learning dynamics to improve model-based per-

formance, (b) increasing robustness in RL by model-based safety, and (c) providing learned stability and safety with existing model-based controllers. We provide more details on each of these approaches next.

Learning Dynamics in Model-based Control Safety can be guaranteed based on modern control frameworks. One approach is to integrate adaptive control with standard machine learning or data-driven methods, such as frequency filter [211], NN [194], GP [60] and DNN [93]. The safety properties are usually considered on the whole system with some parts being learned in MPC problems [30, 104], shielding [8], Control Barrier Functions (CBFs) [40], Hamiltonian analysis [17]. These approaches can guarantee safety for tasks such as stabilization [96, 204] and tracking [141, 50]. However, these approaches usually make assumptions to apply on a known model structure, such as control affine or linear system with bounded uncertainty, which becomes challenging to apply on high-dimensional nonlinear systems.

Model-based Safety in Reinforcement Learning Another approach to bridge safety and learning is to increase robustness and safety in RL tasks by model-based methods. When using RL to solve a control problem through trials and errors, safety is considered by imposing input constraints or safety rewards. Safe exploration [130] and safe optimization [183] of MDPs under unknown or selected cost function can be formulated. Constrained MDPs are also common in RL tasks with Lagrangian methods [43] and generalized Lyapunov/barrier functions [42, 40] and shielding [8]. Nonetheless much of the work remains confined to rather naive simulated tasks, such as moving a $2D$ agent on a grid map.

Learned Certifications for Model-based Controllers Remaining methods for connecting safety and learning are to provide certification on learned stability and constraint set using existing model-based controllers. Stability criterion can be achieved by Lyapunov analysis on region of attraction (RoA) [155, 46] or of Lipschitz-based analysis for safety [91] during reinforcement learning. Safety with constraint set certifications can be learned using control synthesis such as feedback linearization controllers [203], control barrier functions (CBFs) [218, 184, 156, 128], and Hamilton-Jacobi (HJ) reachability analysis [55, 88, 17]. However, all these approaches are only validated on simple dynamic systems such as 5 DoF bipedal robot in simulation or 7 DoF static robot arm in the real world. This is because these approaches usually suffer from a curse of dimensionality, in other words, finding a valid control barrier function [9] or a backward reachable set [18] for high dimensional systems remain unsolved problems. As we will see, the proposed method in this chapter is an "inverse" of such a methodology: we use model-based methods to find certifications on closed-loop systems with RL-based controllers, which is more practical to apply on high order and nonlinear systems.

10.1.1.2 Low-dimensional Structure of Deep Learning

Finding low-dimensional structures of high-dimensional data is widely used in the statistical learning field, such as PCA, kernels, etc [208]. More recently, researchers in the deep learning field realize that learning on nonlinear functions in high-dimensional space may tend to linearize and compress the system and obtain a low-dimensional linear representation of it [5, 201, 33] within the learning components. But this is still an open question and under debate. However, there is little effort exerted on finding the low-dimensional structure of a system driven by RL as the optimization of RL is usually realized through trial and error. In this work, we show some evidence that policy optimization RL may learn through linearizing the nonlinear system.

10.1.2 Contributions

In this chapter, we propose a new direction to bridge model-based safety and model-free reinforcement learning by finding a low dimensional model of a closed-loop system controlled by a learned policy, as shown in Figure 10.1. We use the walking controller of a person-sized bipedal robot Cassie, which is represented by a deep neural network optimized by reinforcement learning in [114], as an example. According to the best of our knowledge, this is the first attempt to apply system identification using a low-dimensional model on the complex system controlled by a RL policy. We find out that a linear model is sufficient to describe this low-dimensional representation. This linear model shows desirable properties such as all dimensions are decoupled, minimum phase, and asymptotic stability. We further show that linearity exists across multiple RL policies under a provided criterion. Such a method shows the possibility to answer the questions from control community to learning community: to analyze the stability of the policies obtained by RL. Additionally, we demonstrate that the linearity analysis can reflect the convergence of RL. Furthermore, based on the found linear model, we propose one of the first safe navigation framework using a high-dimensional nonlinear robot, a bipedal robot Cassie, that combines low-level RL policy for locomotion controller and high-level safety-critical optimal control.

10.2 Methodology

In this section, the experimental platform, the bipedal robot Cassie, and its RL policy for locomotion controller are briefly introduced. Moreover, the method utilized to find the low-dimensional model by system identification through input-output pairs is also presented.

10.2.1 Cassie and Its RL policy for Walking Controller

Cassie is a person-size bipedal robot. It has 10 motors and 4 underactuated joints connected by springs. A detailed introduction of Cassie can be found in [68, 112]. There has been some exciting progress on applying model-free reinforcement learning to obtain locomotion

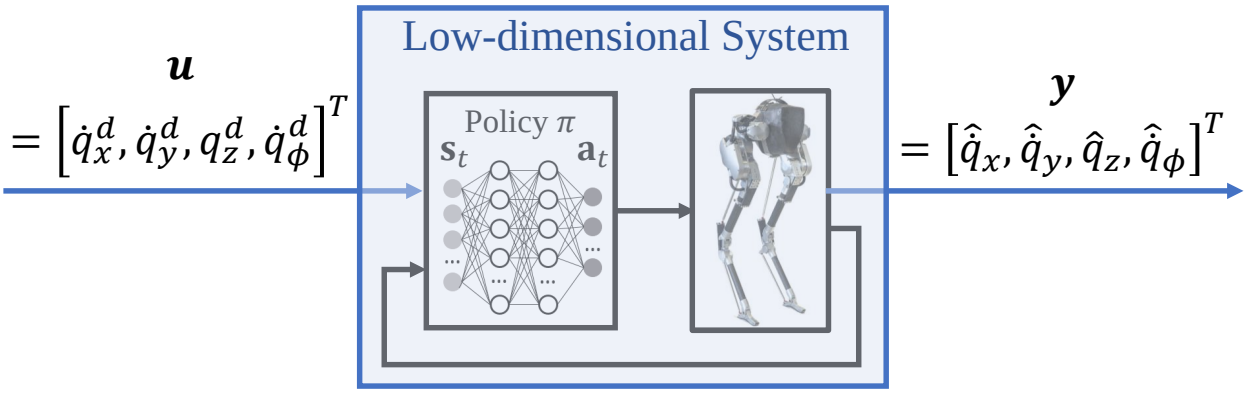


Figure 10.2: System identification on the closed-loop system.

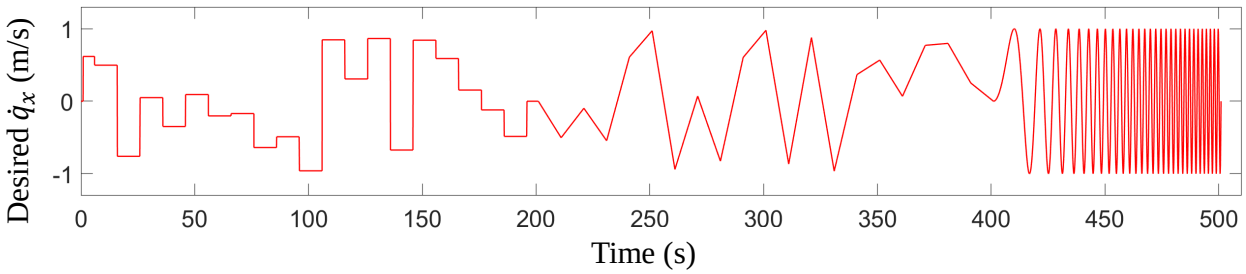
controllers on Cassie in the real world [216, 114, 173, 172]. Among these work, [114] develops a robust and versatile walking controller on Cassie that can track variable commands via RL. This policy is represented by a 2-layer fully-connected neural network with 512 *tanh* nonlinear activations in each layer, and directly outputs 10 dimensional desired motor positions which are then used in a joint-level PD controller to generate motor torques in real time. The policy observation includes reference motion to imitate, robot current states, and 4 timestep past robot states and actions. This single policy is able to track a 4 dimensional command, which is comprised of the desired sagittal walking speed $\dot{q}_x^d$, lateral walking speed $\dot{q}_y^d$, walking height q_z^d, and turning yaw rate $\dot{q}_\phi^d$. The agent is trained in Mujoco [189] which is a physics simulator, and the policy is optimized by Proximal Policy Optimization (PPO) [168].

In real-world experiments, the policy demonstrates considerable robustness and sophisticated recoveries. For example, in a recovery case demonstrated in [114], the robot almost falls down and deviates a lot from the nominal walking states, but the controller is still able to regulate the system without letting the states go unbounded and triggering instability. Although it is very hard to show the explicit stability and RoA of a controller represented by a high dimensional nonlinear neural network, it is possible that the robustness of this RL-based controller is due to its stability while having a relatively large RoA, or ideally, being globally stable.

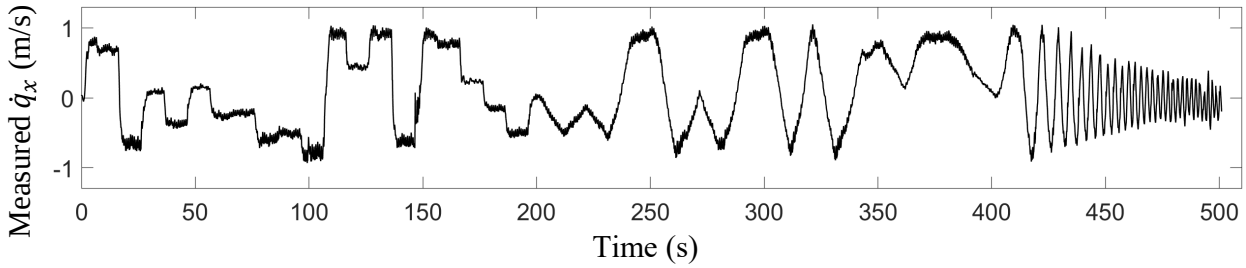
Therefore, in this work, we utilize two types of RL-based controllers based on [114]: (1) the same multi-layer perceptrons (MLP) introduced in [114], (2) the same MLP but with an additional encoder represented by a 2-layer convolutional neural network (CNN) to record longer history of robot states and actions that last 2 seconds (66 timesteps).

10.2.2 System Identification

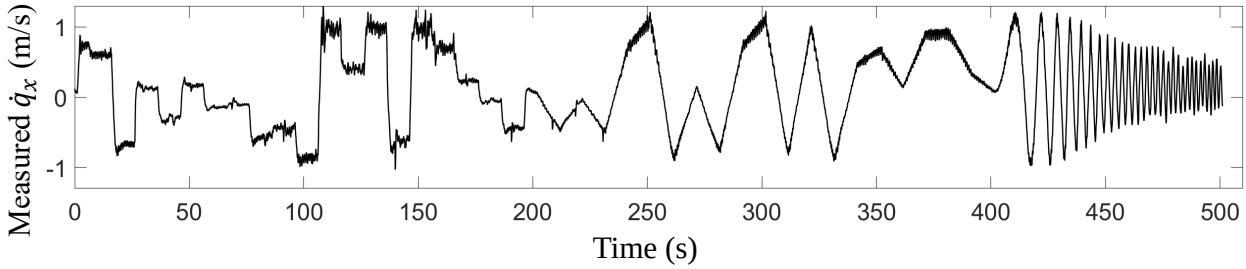
In order to understand the dynamics of the RL-based controller, we choose to identify the entire closed-loop system comprising of Cassie being controlled by the walking controllers obtained through RL. The input $\mathbf{u}$ to this closed-loop system has only 4 dimensions, which is



(a) An Example of System Inputs: Desired Saggital Walking Velocity



(b) System Outputs using MLP: Measured Saggital Walking Velocity



(c) System Outputs using CNN: Measured Saggital Walking Velocity

Figure 10.3: An example of input-output pair used in this chapter to identify the dynamics of Cassie being controlled by the RL-based walking controllers.

the control reference $[\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\phi^d]^T$. The output of this system is the observed robot walking velocities and walking height, i.e. $\mathbf{y} = [\hat{\dot{q}}_x, \hat{\dot{q}}_x, \hat{q}_z, \hat{\dot{q}}_\phi]^T$. We then collect input-output pairs of this system in a high-fidelity simulator of Cassie built on MATLAB Simulink. Please note that the control policy is trained on Mujoco and has no access to the data from Simulink during training and has also been shown to work in Simulink.

One example of the input-output signal of the saggital walking velocity dimension is shown in Figure 10.3. There are three types of the input signals: 1) random step signal as shown in 0-200 s in Figure 10.3a, 2) random ramp signal as demonstrated in 200-400 s in Figure 10.3a, and 3) swept frequency sine wave (*chirp*) whose frequency linearly expands from 0 Hz to 1 Hz in 400-500 s in Figure 10.3a. The resulting robot outputs represent the step response, ramp response, and frequency response of this system, respectively, as illustrated

in the corresponding time span in Figure 10.3b and 10.3c. After selecting a model structure, the parameters of the model can be fitted by combining the input signals and the output signals measured from the Cassie driven by the RL-based controller. As it is unsafe for the person-sized bipedal robot to experimentally follow those random input signals in real-life, Simulink provides a safe validation domain in this chapter.

10.3 Linearity

In this section, we use a linear model structure, *i.e.*, $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, to fit the input-output pairs of the system which is Cassie being controlled by the RL-based controller. Linear models show reasonably good fitting accuracy across all four dimensions which are saggital walking velocity $\dot{q}_x$, lateral walking velocity $\dot{q}_y$, walking height q_z, and turning yaw rate $\dot{q}_\phi$. The fitted linear systems show that they are stable systems while some of them having non-minimum phase. Moreover, the results also show that the input frequency should stay below certain threshold to preserve the linearity of the system.

10.3.1 Linear Systems

The fitting results using linear models for the four dimension outputs of the system which is Cassie being controlled by the CNN-based RL policy for walking controller are demonstrated in Figure 10.4. The fitting results of MLP-based RL walking controller are presented in Figure 10.5. During the fitting phase of each dimension, the measured input-output data of the system are used to find the parameters of the selected linear model in order to minimize the difference between the predicted and measured system outputs. For different dimensions of the system, the dynamic model may be varied. Therefore, the structure of the linear model, *i.e.*, numbers of zeros and poles, are searched in order to maximize fitting accuracy while having the simplest structure which has the least number of zeros and poles. During the validation phase, the fitted model is utilized to predict next 5 step system output using the system input and 1 step measured output data, the prediction results are shown as the blue lines in Figure 10.4.

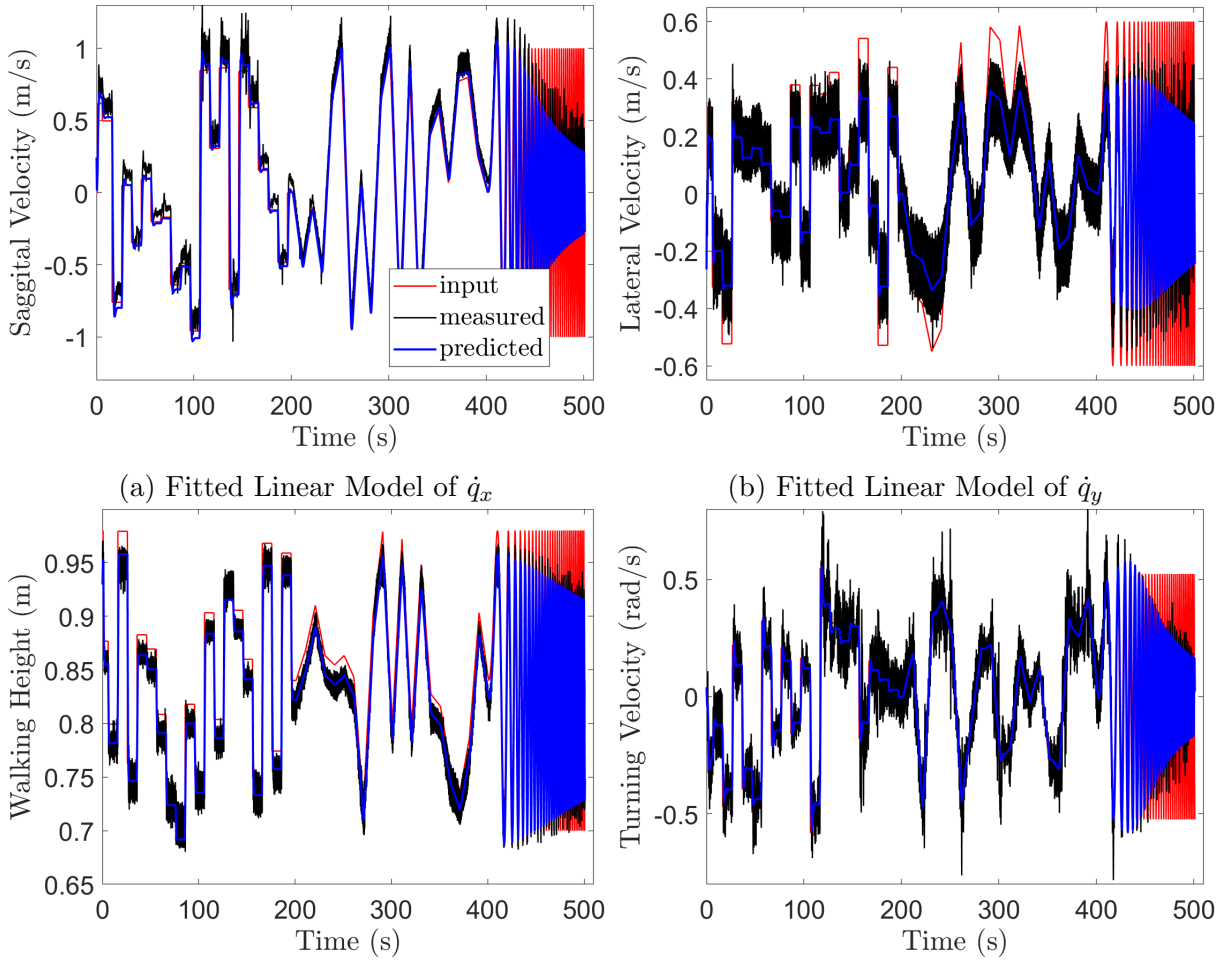
The fitted dynamics of saggital walking velocity $f_{\dot{x}}(\mathbf{x}_{\dot{x}}, u_{\dot{x}})$ is given by:

$$Y_{\dot{x}}(s) = \frac{0.4694s^2 + 6.089s + 8.697}{s^3 + 6.432s^2 + 11.03s + 8.274} U_{\dot{x}}(s) \tag{10.1}$$

with a prediction accuracy of 79.67% when compared with the ground truth measured data, as shown in Figure 10.4a. The dynamics of the lateral walking velocity $f_{\dot{y}}(\mathbf{x}_{\dot{y}}, u_{\dot{y}})$ is obtained by using a linear system of 3 poles and 1 zeros, and it can be written as:

$$Y_{\dot{y}(s)} = \frac{13.59s + 24.56}{s^3 + 11.48s^2 + 32.5s + 40.13} U_{\dot{y}}(s) \tag{10.2}$$

which has a prediction accuracy of 55.87%. However, as shown in Figure 10.4b, there is a large oscillation of the robot base lateral direction when Cassie is walking. After applying a



(a) Fitted Linear Model of $\dot{q}_x$

(b) Fitted Linear Model of $\dot{q}_y$

(c) Fitted Linear Model of q_z

(d) Fitted Linear Model of $\dot{q}_\phi$

Figure 10.4: Fitting results using linear models for all four dimensions of Cassie being controlled by a CNN-based RL policy.

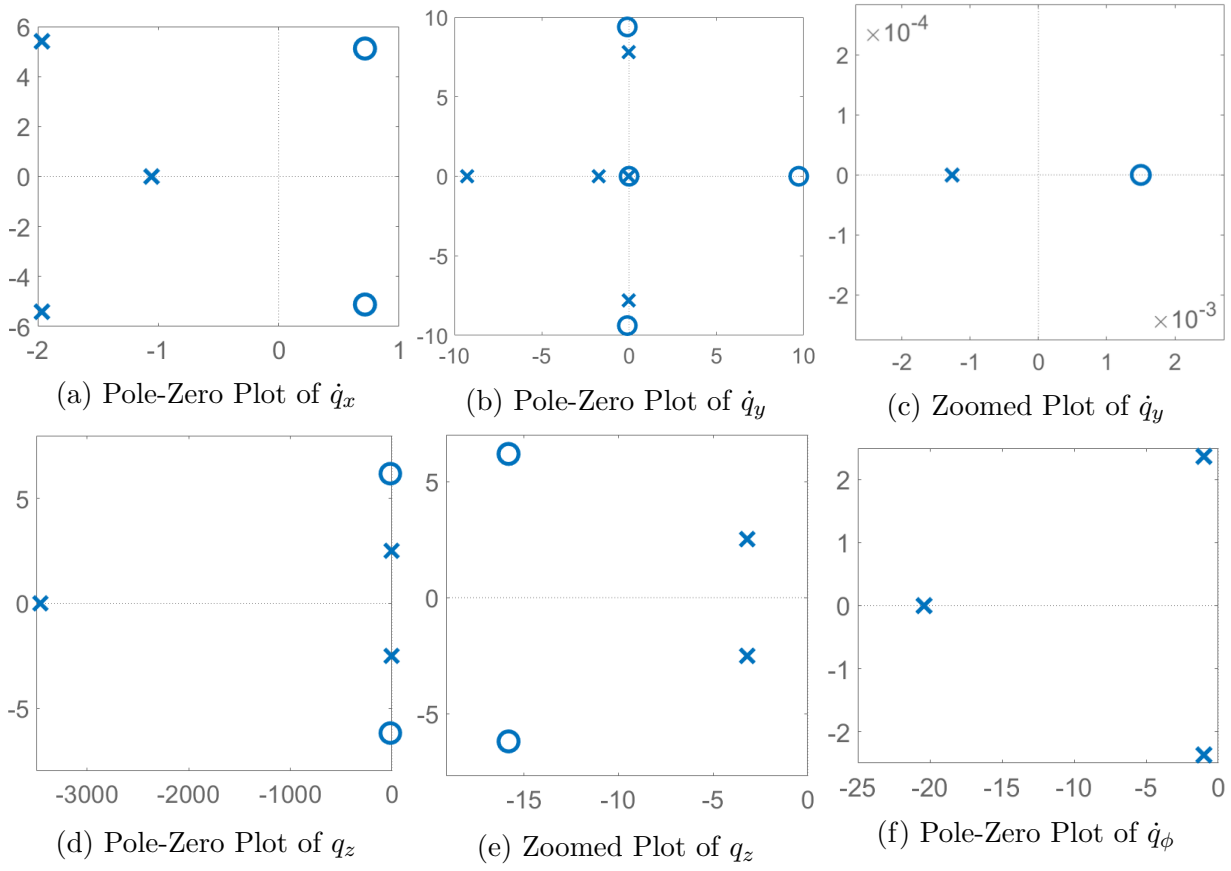
low-pass filter with a cut-off frequency of 5 Hz on the measured lateral velocity, the prediction accuracy on the flitted signal is 83.03% using the same fitted model via (10.2).

The dynamics of walking height $f_z(\mathbf{x}_z, u_z)$ and the dynamics of turning yaw velocity $f_{\dot{\phi}}(\mathbf{x}_{\dot{\phi}}, u_{\dot{\phi}})$ are fitted as:

$$Y_z(s) = \frac{145.9s + 37.55}{s^3 + 46.43s^2 + 161s + 38.38} U_z(s) \tag{10.3}$$

and

$$Y_{\dot{\phi}}(s) = \frac{0.3078s^2 + 5.267s + 5.553}{s^3 + 4.595s^2 + 7.528s + 6.045} U_{\dot{\phi}}(s) \tag{10.4}$$



(a) Pole-Zero Plot of $\dot{q}_x$ (b) Pole-Zero Plot of $\dot{q}_y$ (c) Zoomed Plot of $\dot{q}_y$

(d) Pole-Zero Plot of q_z (e) Zoomed Plot of q_z (f) Pole-Zero Plot of $\dot{q}_\phi$

Figure 10.5: Pole-Zero plots of the identified linear systems of MLP controller for saggital walking velocity, lateral walking velocity, walking height, and turning yaw rate.

and with prediction accuracy of 85.67% in Figure 10.4c and 65.58% (81.77% after filtering the measured output) in Figure 10.4d, respectively.

According to the prediction performance as demonstrated in Figure 10.4, all of the four dimensions show reasonably good linearity as the fitting accuracy of them are all around 80%. Therefore, we can come to a conclusion that all of these 4 dimensions of Cassie during walking shows linearity when the robot is controlled by the RL policy.

10.3.2 Stability

Stability analysis can be applied on each dimension after obtaining the low-dimensional linear dynamics model of the closed-loop system. Since the dynamics of each dimension is able to be explicitly expressed by a transfer function, a commonly used stability criterion is the position of the poles of the system. Please note that during the model identification in Sec. 10.3.1, we allow the algorithm to find unstable models as long as the fitting results are

better than stable models, *i.e.*, we don't impose stable model constraint during fitting.

The plots of poles and zeros of all four dimension of the system using the CNN policy are shown in Figure 10.6. As illustrated in Figure 10.6c, there is a very close pole-zero pair of the identified q_z dynamics. However, that pole cannot be cancelled otherwise it will lead to worse fitting accuracy.

According to the plot, all of the identified linear systems are asymptotically stable as all the poles are on the Left Half Plane (LHP) [56]. Moreover, all zeros are on the LHP as well, therefore, all dimensions are minimum phase. This shows that the system controlled by CNN-based RL policy has a very nice property that the closed-loop system with RL-based policy can be represented through a linear model which is also stable.

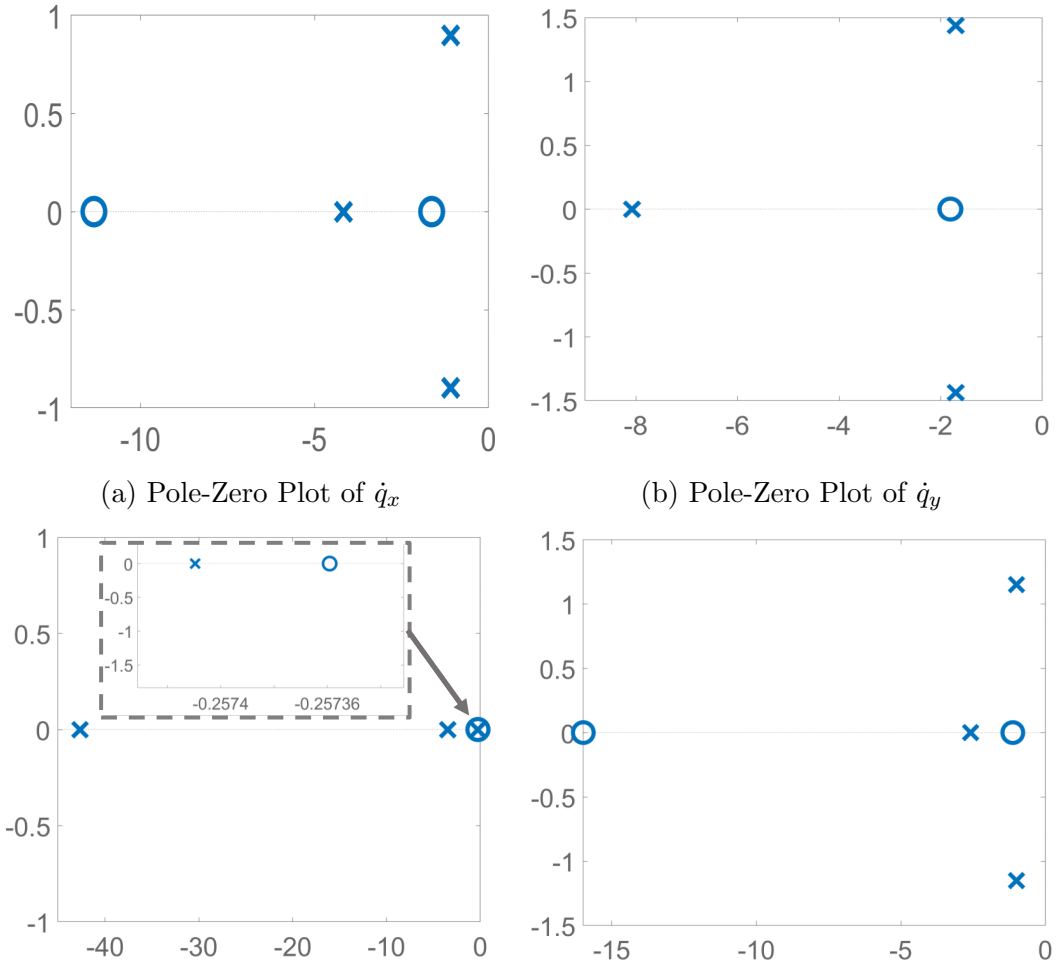
Remark 10.1. *All poles of the linear system obtained through system identification of the closed-loop system under MLP-based RL policy (pole-zero plots are in Figure 10.5) are asymptotically stable, however, the system driven by MLP is non-minimum phase in dimensions of the saggital and lateral walking velocity. Therefore, when choosing to utilize the robot driven by such an RL policy, the high-level controller/planner should be carefully designed. Such stability analysis provides a guidance to utilize the system controlled by RL policies.*

10.3.3 Criterion for Linearity

Although Cassie driven by the RL policy for walking controller shows reasonably good linearity as shown in Fig 10.4, there is a criterion for the existence of linearity. That is, the input frequency cannot exceed a threshold which is around $f_c = 0.6$ Hz. The frequency 0.6 Hz is an empiric value and this is found when the input tends to excite nonlinearity of the system if the input frequency exceed that value during system identification. The fitted linear model shows worse fitting accuracy if the chirp signal enters the region where frequency is larger than 0.6 Hz in Figure 10.4. This may be related to the stepping frequency of Cassie. Given the fact that Cassie steps on each feet at a stepping rate of 1.25 Hz, $f_c = 0.6$ Hz is in one half of it. The existence of this cutoff frequency may be due to fact that the RL policy is unable to change the walking velocities, such as $\dot{q}_x$, $\dot{q}_y$, $\dot{q}_\phi$, faster than the time Cassie takes to complete one walking step (comprising of two steps: left foot stance followed by right foot stance). Another evidence of this validity of this relationship is that there is no obvious linearity lost after this frequency threshold in the identified q_z dynamics. This is because changing the walking height doesn't need to wait until completion of one walking step, *e.g.*, robot can change the walking height by changing the length of the stance leg at any time. The phenomenon of loosing linearity after this frequency threshold will frequently appear in the later part of this chapter.

Table 10.1: Benchmark of fitting accuracy using the identified models on different input-output Pairs.

	$\dot{\mathbf{q}}_\mathbf{x}$ **chirp**	$\dot{\mathbf{q}}_\mathbf{y}$ **chirp**	$\mathbf{q}_\mathbf{z}$ **chirp**	$\dot{\mathbf{q}}_\phi$ **chirp**
$\dot{\mathbf{q}}_\mathbf{x}$ **chirp**	82.86%, Figure 10.7a	78.81%, , Figure 10.7b	66.17%, Figure 10.7c	57.98%(67.88%*), Figure 10.7d
$\dot{\mathbf{q}}_\mathbf{y}$ **chirp**	45.65%(69.36%*), Figure 10.7f	80.08%, Figure 10.7e	80.65%, Figure 10.7g	77.76%, Figure 10.7h
$\mathbf{q}_\mathbf{z}$ **chirp**	47.85%(54.29%*), Figure 10.7j	81.12%, Figure 10.7k	83.19%, Figure 10.7i	80.11%, Figure 10.7l
$\dot{\mathbf{q}}_\phi$ **chirp**	53.87%(61.75%*), Figure 10.7n	79.39%, Figure 10.7n	74.5%, Figure 10.7p	83.6%, Figure 10.7m



(a) Pole-Zero Plot of $\dot{q}_x$

(b) Pole-Zero Plot of $\dot{q}_y$

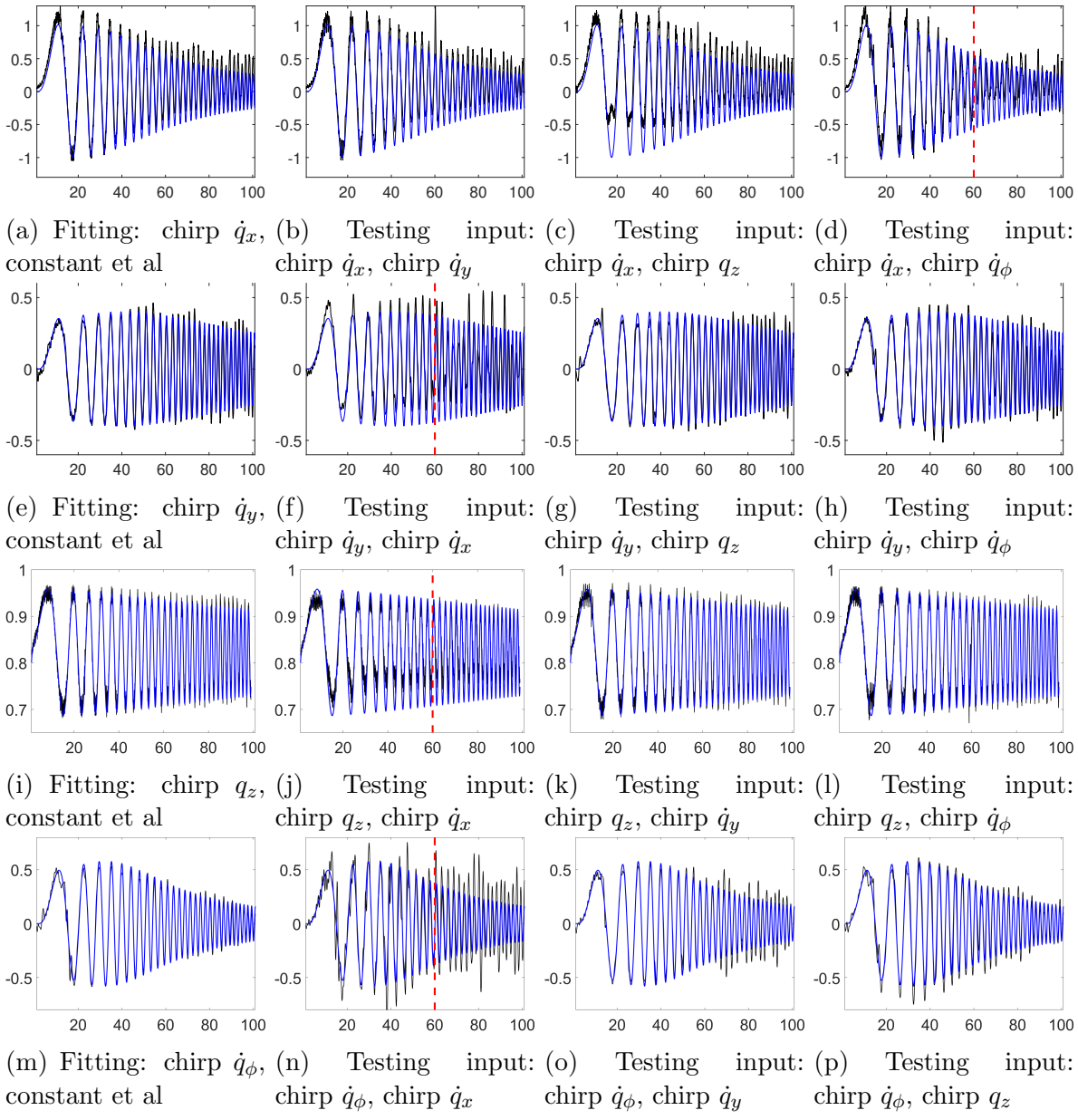
(c) Pole-Zero Plot of q_z

(d) Pole-Zero Plot of $\dot{q}_\phi$

Figure 10.6: Pole-Zero plots of the identified linear systems that uses CNN policy for saggital walking velocity, lateral walking velocity, walking height, and turning yaw rate.

10.4 Decoupling System

The system, Cassie driven by the RL policy as a walking controller, is a Multiple-Input Multiple-Output (MIMO) system, and there are four inputs, $\mathbf{u} = [\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\phi^d]^T \in \mathbb{R}^4$ and four outputs, $\mathbf{y} = [\hat{\dot{q}}_x, \hat{\dot{q}}_y, \hat{q}_z, \hat{\dot{q}}_\phi]^T \in \mathbb{R}^4$. As demonstrated in an existing design for navigation autonomy on Cassie proposed in [115], these four dimensions are tightly coupled using a model-based controller on Cassie and a lot of effort is exerted to consider such couplings for the planning purpose otherwise the planned output may cause a walking failure [115]. In this section, we will show that the closed-loop Cassie system with the RL policy as the walking controller has a four dimensional dynamics, i.e., $f_{\dot{x}}(\mathbf{x}_{\dot{x}}, u_{\dot{x}})$, $f_{\dot{y}}(\mathbf{x}_{\dot{y}}, u_{\dot{y}})$, $f_z(\mathbf{x}_z, u_z)$,



(a) Fitting: chirp $\dot{q}_x$, constant et al (b) Testing input: chirp $\dot{q}_x$, chirp $\dot{q}_y$ (c) Testing input: chirp $\dot{q}_x$, chirp q_z (d) Testing input: chirp $\dot{q}_x$, chirp $\dot{q}_\phi$

(e) Fitting: chirp $\dot{q}_y$, constant et al (f) Testing input: chirp $\dot{q}_y$, chirp $\dot{q}_x$ (g) Testing input: chirp $\dot{q}_y$, chirp q_z (h) Testing input: chirp $\dot{q}_y$, chirp $\dot{q}_\phi$

(i) Fitting: chirp q_z, constant et al (j) Testing input: chirp q_z, chirp $\dot{q}_x$ (k) Testing input: chirp q_z, chirp $\dot{q}_y$ (l) Testing input: chirp q_z, chirp $\dot{q}_\phi$

(m) Fitting: chirp $\dot{q}_\phi$, constant et al (n) Testing input: chirp $\dot{q}_\phi$, chirp $\dot{q}_x$ (o) Testing input: chirp $\dot{q}_\phi$, chirp $\dot{q}_y$ (p) Testing input: chirp $\dot{q}_\phi$, chirp q_z

Figure 10.7: Decoupling test.

and $f_{\dot{\phi}}(\mathbf{x}_{\dot{\phi}}, u_{\dot{\phi}})$, obtained in Sec.10.3, and these are all decoupled.

In order to test a system M that is independent of another system N, we applied four steps:

1. Collect the first measured input-output pair where the input to M is a swept-frequency signal, chirp, while the input to N is a fixed value.
2. Identify a model for M using the first input-output pair.
3. Obtain the second input-out pair where the input to M is a chirp while the input to N is also a chirp.
4. Test the prediction accuracy using the identified model from step 2 on the second pair.

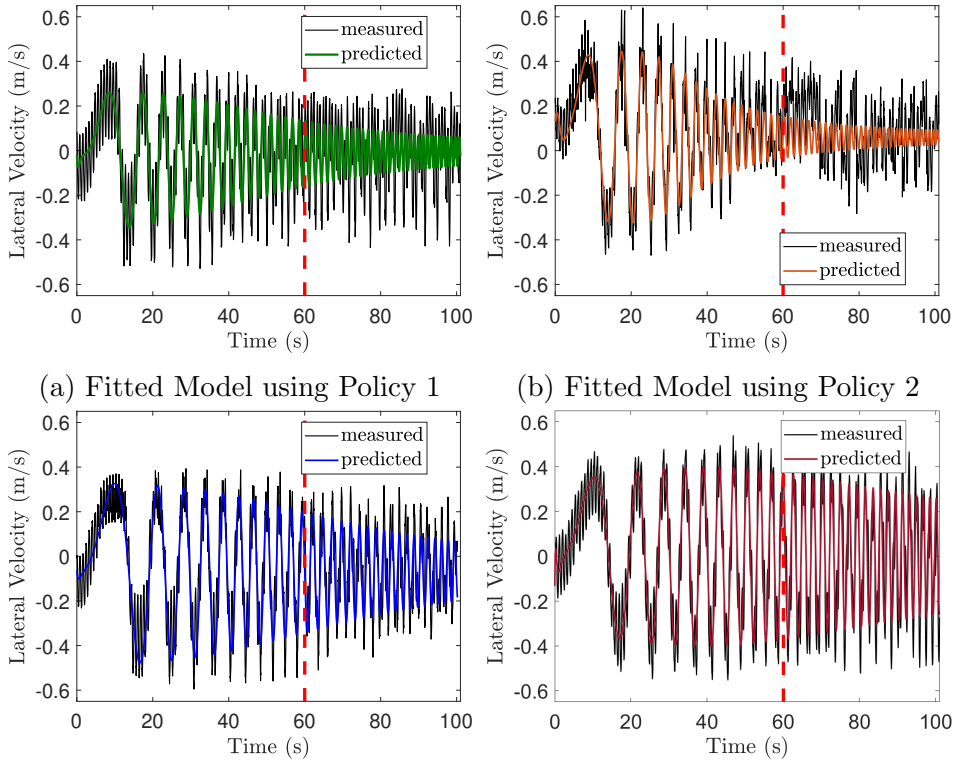
If the testing accuracy shows no significant loss than fitting accuracy, M is independent of N, otherwise, M and N are coupled. This is because a fixed value (zero frequency) doesn't contain any information of a swept-frequency wave (chirp) signal. As a result, if the model for M identified by data where the input to N is a fixed value is able to accurately predict a measured input-output data when the input to N is a chirp, this could show that the change in N will not excite the dynamics in M.

Let us consider the test on the dependency between saggital walking velocity $\dot{q}_x$ and lateral walking velocity $\dot{q}_y$ using the CNN-based RL policy as an example. During fitting the model for $\dot{q}_x$ in Sec. 10.3, the walking velocity command $\dot{q}_y$ is fixed at 0 m/s, as shown in Figure 10.7a. The fitting accuracy is 82.86%. This fitted model is directly applied to predict the measured output of $\dot{q}_x$ when the input to $\dot{q}_y$ is also a chirp. As shown in Figure 10.7b, the identified model still shows reasonably good prediction accuracy on this input-output pair, resulting in the accuracy of 78.81%. Therefore, we can draw a conclusion that saggital walking velocity $\dot{q}_x$ is independent of lateral walking velocity $\dot{q}_y$. Same decoupling tests are applied on all of the combinations of the four dimensions using the CNN-based RL policy, as shown in Figure 10.7, and the quantitative accuracy data is recorded in Table 10.1.

According to Figure 10.7 and Table 10.1, for the dynamics on each dimension, the model fitted on the data when the inputs on other dimensions are fixed values can well describe the dynamics when the input to one other dimension is a chirp, such as $\dot{q}_y \leftrightarrow q_z$ pair, $q_z \leftrightarrow \dot{q}_\phi$ pair, and $\dot{q}_y \leftrightarrow \dot{q}_\phi$ pair. Although fitting accuracy for the $\dot{q}_y$, q_z and $\dot{q}_\phi$ decreases under the existence of a chirp $\dot{q}_x$, the accuracy can be still over 50% if we only look at the measured output before the cut-off frequency $f_c = 0.6$ Hz introduced in Sec. 10.3.3. In conclusion, such a system can be considered as decoupled under a low-frequency input, and higher frequency may excite the coupled dynamics along some dimensions.

In order to show the generality of the phenomenon that when Cassie being controlled by a RL walking policy tends to show linearity, we test the measured input-output pairs obtained from using different RL policies. We test 4 policies:

1. A MLP-based RL policy is well trained with around 400 million samples from scratch using reward formulation.



(a) Fitted Model using Policy 1

(b) Fitted Model using Policy 2

(c) Fitted Model using Policy 3

(d) Fitted Model using Policy 4

Figure 10.8: Cross policy validation using lateral walking velocity $\dot{q}_y$ as an example.

2. A MLP-based RL policy uses a different reward form but is in the middle of training after 30 million samples using Policy 1 as a warm start.

3. A MLP-based RL Policy keeps training on Policy 2 using the same reward and is trained with 100 million samples.

4. A CNN-based RL policy (different neural network structure) and different reward, and well trained after 400 M samples from scratch.

We use the dynamics of lateral walking velocity $\dot{q}_y$ as an example. The system identification is applied on Cassie driven by these 4 policies, respectively, and the results are shown in Figure 10.8. If the measured input-output data can be well fitted by a linear model, we can tell that Cassie being controlled by each of these policies shows linearity.

According to Figure 10.8, since Policy 2 is not well trained, the system driven by this policy shows the worst linearity, *i.e.*, a linear model cannot well predict the system output by the input. Specifically, the nonlinearity appears after the cut-off frequency f_c. After being trained with ample samples, Policy 3 which uses the same reward and neural network structure, shows a clear existence of the linearity in Figure 10.8c. Moreover, the measured

input-output pairs from the other policies, like Policy 1 and 4, demonstrate reasonably good fitting accuracy before cutoff frequency $f_c = 0.6$ Hz.

From all the discussion above, we can draw a conclusion that linearity is not always guaranteed. If a RL policy is not well-trained, such as Policy 2, even if we "force" the system identification to find a linear structure, the fitting result could be poor because of the existence of strong nonlinearity in the system. However, if the learning of the policies has converged, the linearity appears, such as Figure 10.8a, 10.8c, 10.8d, and this is validated across different RL policies with different rewards or neural network structure.

Remark 10.2. *We note that the proposed linearity analysis could potentially be a metric for the convergence of the learning of a RL policy on a dynamic system.*

Therefore, we can summarize the low-dimensional representation of Cassie being controlled by a well-trained RL policy using a linear model as below:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{10.5}$$

where the states $\mathbf{x} = [\mathbf{x}_{\dot{x}}^T, \mathbf{x}_{\dot{y}}^T, \mathbf{x}_z^T, \mathbf{x}_{\dot{\theta}}^T]^T$, output $\mathbf{y}$ is $[y_{\dot{x}}, y_{\dot{y}}, y_z, y_{\dot{\theta}}]^T = [\dot{q}_x, \dot{q}_y, q_z, \dot{q}_\theta]^T$, and input $\mathbf{u}$ is $[u_{\dot{x}}, u_{\dot{y}}, u_z, u_{\dot{\theta}}]^T = [\dot{q}_x^d, \dot{q}_y^d, q_z^d, \dot{q}_\theta^d]^T$, and matrices $A = \text{diag}(A_{\dot{x}}, A_{\dot{y}}, A_z, A_{\dot{\theta}})$, B and C are proper stack of $B_{\dot{x}}, B_{\dot{y}}, B_z, B_{\dot{\theta}}$ and $C_{\dot{x}}, C_{\dot{y}}, C_z, C_{\dot{\theta}}$, respectively. Hence the linearized dynamics is as follows,

$$\begin{bmatrix} \dot{\mathbf{x}}_{\dot{x},\dot{y},z,\dot{\phi}} \\ \mathbf{y}_{\dot{x},\dot{y},z,\dot{\phi}} \end{bmatrix} = \begin{bmatrix} A_{\dot{x},\dot{y},z,\dot{\phi}} & B_{\dot{x},\dot{y},z,\dot{\phi}} \\ C_{\dot{x},\dot{y},z,\dot{\phi}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\dot{x},\dot{y},z,\dot{\phi}} \\ \mathbf{u}_{\dot{x},\dot{y},z,\dot{\phi}} \end{bmatrix} \tag{10.6}$$

And these models are the control canonical forms of (10.1), (10.2), (10.3), and (10.4) showed in Sec. 10.3, respectively.

10.4.1 Safety-critical Navigation Framework

In this section, we demonstrate an application of the proposed methodology, that is, utilizing the identified simple system to provide safety guarantee on the nonlinear system driven by its RL policy.

The linear system representation expressed in (10.5) can be applied to solve a safe navigation problem for Cassie. The navigation autonomy framework proposed in this chapter can be found in Figure 10.9 which is based on [115] to navigate unknown environments with height-constraints using Cassie. In this autonomy, after being given a goal location, a global planner firstly finds a collision-free path on the map from the robot's current position, followed by two local planners using nonlinear optimization to track the global path. The locomotion controller used in the previous work to enable Cassie to follow planning results is based on Hybrid Zero Dynamics (HZD) and is developed in [112] based on [68]. A detailed introduction of this autonomy can be found in [115].

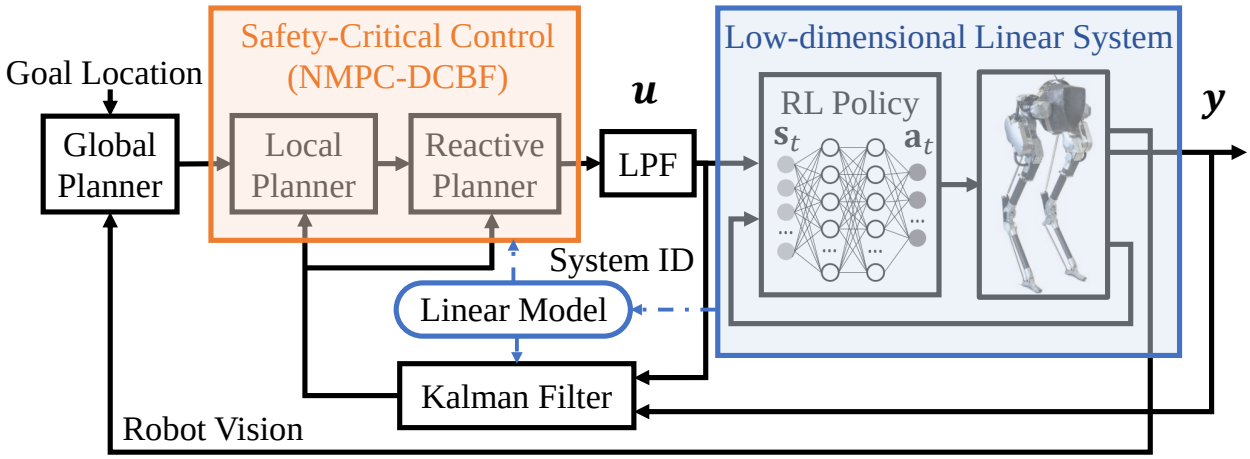


Figure 10.9: Safe navigation autonomy using Cassie driven by a RL walking policy and the identified linear system.

In this work, we use the CNN-based RL locomotion policy on Cassie. Instead of using the nonlinear reduced-order dynamics model in [115] during local planning by collocation, we adopt its identified linear system in (10.5) and safety-critical control framework that combines nonlinear model predictive control (NMPC) and discrete-time control barrier function (DCBF) introduced in [223].

To formulate the NMPC-DCBF problem, we over-approximate the geometries of Cassie and the obstacles by cylinders. The distance between robot and each obstacle o_i can be computed analytically as follows,

$$d_k^{o_i} = (x_k^g - x^{o_i})^2 + (y_k^g - y^{o_i})^2 - (R^{robot} + R^{o_i})^2, \tag{10.7}$$

where x_k^g y_k^g are the global x-y position of the robot at time k, (x^{o_i}, y^{o_i}) is the position of the obstacle o_i, R^{robot} is the radius of the cylindrical approximation of Cassie, and R^{o_i} is the radius of the obstacle. Obstacle avoidance between the robot and the obstacle can then be enforced by constraining $d_k^{o_i} > 0, \forall k$. However, over short planning horizons this constraint can fail to ensure long-term obstacle avoidance. Therefore, we use DCBF constraints, which can provide long-term obstacle avoidance, even on short planning horizons [224]. The DCBF constraint is as follows,

$$d_{k+1}^{o_i} \geq \omega_k \alpha_{\text{DCBF}} d_k^{o_i}, 0 \leq \alpha_{\text{DCBF}} \leq 1, \omega_k \geq 0 \tag{10.8}$$

where α_{DCBF} represents the maximum decay-rate at which $h_k^{o_i}$ can converge to zero and ω_k is a slack variable which enhances feasibility and safety [223]. Specifically, the trajectory

generation problem with NMPC-DCBF is formulated as follows,

$$\min_{\mathbf{x},\mathbf{u},\boldsymbol{\delta}} \; J(\mathbf{x},\mathbf{u},\boldsymbol{\delta},\boldsymbol{\omega}), \tag{10.9a}$$

$$\text{s.t.} \;\; \mathbf{x}_{k+1} = A^d\mathbf{x}_k + B^d\mathbf{u}_k \tag{10.9b}$$

$$\mathbf{x}_0 = \mathbf{x}_{init} \tag{10.9c}$$

$$\mathbf{x}_k \in \mathcal{X}_{adm}, \mathbf{u}_k \in \mathcal{U}_{adm}, \tag{10.9d}$$

$$x^g_{k+1} = x^g_k + C^d_{\dot{x}}\mathbf{x}_{\dot{x},k}\cos(C^d_{\dot{\phi}}\mathbf{x}_{\phi,k})dt, \tag{10.9e}$$

$$y^g_{k+1} = y^g_k + C^d_{\dot{y}}\mathbf{x}_{\dot{y},k}\sin(C^d_{\dot{\phi}}\mathbf{x}_{\phi,k})dt, \tag{10.9f}$$

$$d^{o_i}_{k+1} \geq \omega_k\alpha_{\text{DCBF}}d^{o_i}_k, \quad \omega_k \geq 0, \tag{10.9g}$$

$$[x^g_N, y^g_N, \mathbf{x}_{\phi,N}]^T = [x^g_f, y^g_f, \mathbf{x}_{\phi,f}]^T + \boldsymbol{\delta} \tag{10.9h}$$

where A^d, B^d, C^d are coefficients of discrete-time dynamics transferred from the proposed continuous linear dynamics in (10.5). The cost function is designed as below,

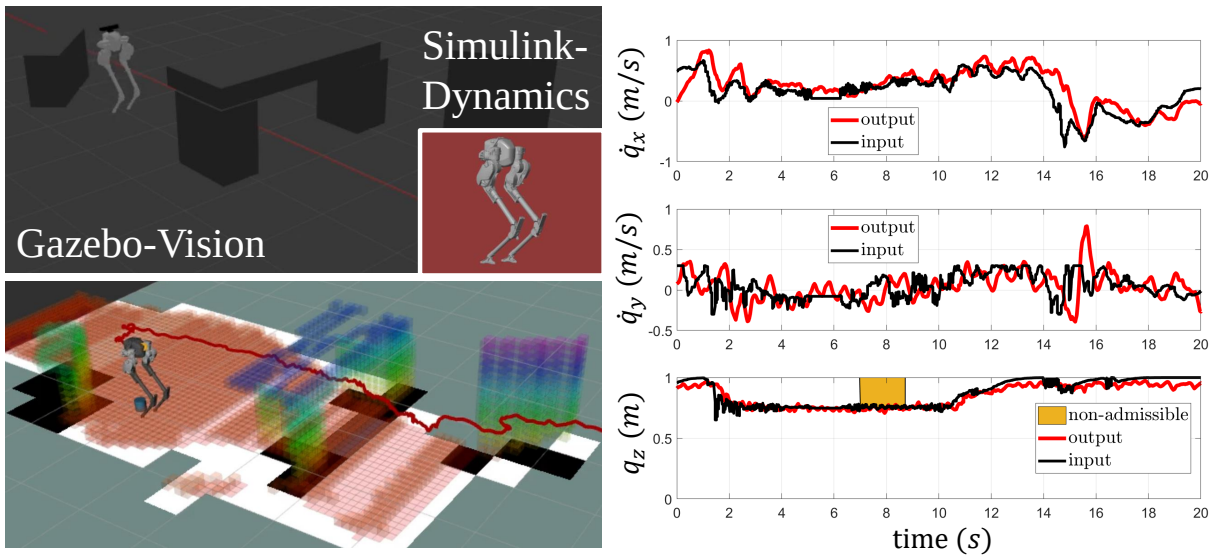
$$J(\mathbf{x},\mathbf{u},\boldsymbol{\delta},\boldsymbol{\omega}) = \sum_{i=1}^{N-1}(||\mathbf{x}_i||^2_Q + ||\mathbf{u}_i||^2_R + ||\mathbf{x}_{i+1}-\mathbf{x}_i||^2_{dQ} + \rho(1-\omega_k)^2) + ||\boldsymbol{\delta}||^2_K. \tag{10.10}$$

The discrete-time dynamics (10.9b), together with kinematics along x and y axis in (10.9e) and (10.9e) formulate a trajectory optimization problem, where the robot's orientation is considered to transfer the robot's velocity in the robot's frame into the world frame. The initial condition, state and input constraint can be found in (10.9c), (10.9d). Specifically, $\mathcal{X}_{adm}$ represents the combination of state constraint together with safety constraint to avoid nearby obstacles defined by distance functions, presented in [115]. $\mathcal{U}_{adm}$ is the input constraint. $\boldsymbol{\omega} = [\omega_1, ..., \omega_{N-1}]^T$ represents the relaxation variables of decay rate α_{DCBF} for DCBF constraints. We also notice that ρ in (10.10) shall be chosen as a relatively large scalar such that the DCBF constraints wouldn't be over-relaxed [226]. Moreover, $\boldsymbol{\delta}$ represents the slack variable for terminal constraint on desired final state $[x^g_f, y^g_f, \mathbf{x}_{\phi,f}]^T$ which is minimized with a quadratic term in the cost function.

The output from the reactive local planner is the real-time desired sagittal and lateral walking velocities $\dot{q}^d_{x,y}$, walking height q^d_z, and turning yaw rate $\dot{q}^d_\phi$. This is the input $\mathbf{u}$ to the linear system that was identified to describe the closed-loop dynamics of Cassie driven by its RL locomotion policy. Moreover, $\mathbf{u}$ is firstly passed through a low pass filter with cutoff frequency of 0.5 Hz which is below the threshold of the linearity criterion found in Sec. 10.3.3 to prevent exciting nonlinearity of the system.

10.4.2 Autonomy Validation

To validate the proposed autonomy that combines NMPC-DCBF with CNN-based RL locomotion policy, the entire algorithm is tested in a joint simulation on Cassie, as shown in Figure 10.10a In this test, a congested space with two obstacles and a height-constrained



(a) Joint simulation of vision and dynamics using the safe navigation autonomy in a congested space

(b) The input (desired commands) and output (robot measured states) during navigation

Figure 10.10: Validation of the proposed safety-critical navigation autonomy in the joint simulation.

space (arch) in between is built on Gazebo where the robot depth camera reading is simulated. The robot dynamics driven by its RL policy is computed on MATLAB Simulink which has high-fidelity for the dynamics computation. These two simulators are synchronized and the result is illustrated in Figure 10.10.

During the simulation, the robot successfully achieves accelerating to full speed (around 1 m/s) when there are no obstacles while quickly pulling back when obstacles are in range. Moreover, the robot shows the capacity to crouch down to travel underneath the arch with a relative high speed. By using the linear model in the optimization, the two-second local-term trajectory can be solved around $0.1-0.2$ s which is at least 5 times faster than prior approach using a nonlinear model [115]. Additionally, it turns out that there are reduced deadlocks and the whole navigation task can be finished around 20 s which is almost twice faster than the existing work with HZD-based controller [115] which results in conservative planned speed (it finishs the same trial in 50 s). For the controller performance, the RL-based policy for walking controller is also more robust and agile than the HZD-based walking controller, such as the coupled walking dynamics are cancelled by RL so the robot can quickly lift its body up after passing through an arch while accelerating to its goal without falling over, as shown in Figure 10.10a.

By using nonlinear model predictive control with control barrier functions, we provide a formal safety guarantee on such a high order nonlinear complex system. All this allows

Cassie controlled by RL locomotion policy during navigation while enabling the safety-critical control-planners to exploit the agility brought by the RL policy. We remark that our implementation is the first work to use a nonlinear model predictive control formulation to achieve safety-critical navigation in height-constrained environments within a joint simulation of vision and high-fidelity dynamics.

10.5 Chapter Summary

In this chapter, we demonstrate evidence that a reinforcement learning based controller acting on a highly nonlinear dynamical system, such as a bipedal robot, may tend to linearize it such that the entire closed-loop system can be represented by a low-dimensional linear system. This is an interesting finding since a high order nonlinear system controlled by a high dimensional nonlinear neural network policy behaves like a linear system. Based on this observation, we propose a new direction to bridge safety-critical control and model-free RL by finding certifications for stability and safety on the low-dimensional model identified on the complex system driven by its RL policy. We validate such new methodology on a bipedal robot Cassie controlled by its RL locomotion policy. We apply the first attempt to do system identification on this closed-loop system to find a linear model, and later we find that such a system demonstrates reasonably good linearity. Moreover, the fitted multiple-input multiple-output linear model is decoupled, minimum phase and asymptotically stable in all dimensions. We also provide a criterion for the linearity existence, that is, the input frequency should be under a given threshold. By analyzing the fitted linear model, we are able to explain the robustness using RL-based controller demonstrated in previous work [114] may come from the guaranteed stability of the closed-loop system. This actually shows a new method to provide stability analysis over a controller represented by high dimensional neural network optimized by reinforcement learning. By cross comparing such linearity on different RL policies, we show that linearity is preserved across different well-trained policies, but linearity is not guaranteed if the learning of the RL policy has not converged.

For application, the fitted linear model is later utilized in a safety-critical navigation framework using NMPC-DCBF which utilizes the RL policy as a robust and powerful low-level walking controller on Cassie. In this application, we note that while the RL policy for the walking controller is able to address the highly nonlinear dynamics of Cassie, the identified closed-loop dynamics represented in a linear model is able to provide guarantees of stability while also introducing safety through control barrier functions. Moreover, such a method is a more practical approach to bridge model-based safety and mode-free reinforcement learning. Because RL has shown the capacity to well control high-dimensional nonlinear systems in the real world by exploiting the system full-order dynamics and using a low-dimensional linear dynamic model for safety-critical control is more realistic in real time.

After having presented the technical work in this thesis, we will draw a brief conclusion in the next chapter.

Chapter 11

Conclusion

11.1 Conclusion

This thesis addresses several existing challenges about control barrier functions in optimization, control, planning and navigation.

In Part I, we address the feasibility problem in optimal control for CLF-CBF-QP under input constraints in Chapter 3. We also notice that the potential conflict between input constraints and safety constraints also exists in the discrete-time domain together with model predictive control, presented in Chapter 4. This conflict is later formally identified in reachability analysis and enhanced in proposed formulations discussed in Chapter 5.

In Part II, we focus on the tight-fitting obstacle avoidance for control and trajectory generation, specifically for polytopic obstacle avoidance. The formulations for obstacle avoidance in continuous domain and discrete-time domain are discussed in Chapter 6 and Chapter 7, respectively.

In Part III, we discuss about the limited performance and deadlock issues for trajectory generation with finite state machines and computationally-fast trajectory generation for high-dimensional nonlinear systems, together with applications about autonomous driving, autonomous racing and legged robots in Chapter 8, 9 and 10 respectively.

11.2 Future Work

Regarding the challenging problems of control barrier functions, there are several interesting topics for future work.

11.2.1 Persistent Feasibility for General Nonlinear Systems

Related to optimization feasibility about CBFs, a general approach of guaranteeing feasibility for both one and high relative-degree nonlinear systems would be an interesting topic based on the results of Chapter 3. In Chapter 3, we discussed a relaxing approach over the decay

rate of the control barrier function, which can be regarded as a linear relaxation, i.e., single variable relaxing. Future work can implement general relaxation for the functional space of the class $\mathcal{K}$ $\alpha(.)$ function with relaxing on multiple variables. We also notice that the relation between control barrier functions and viability kernel remains a challenging problem, while Chapter 3 focused on the point-wise feasibility problem. Based on a more general relaxing technique, we hope the persistent feasibility can be tackled, which will finally bridge the control barrier function and viability kernel [165]. This problem could be challenging as viability kernel for high-dimensional systems are not easy to be calculated and approximated. However, if this relation has been figured out, we will be able to walk around the calculation of viability kernel in later research, which is equivalent to a synthesis of high-dimensional set calculation with a low-dimensional representation of a multivariable relaxing function.

11.2.2 Linearity and Convexification of DCBF in Trajectory Generation

Related to trajectory generation with CBFs in discrete-time domain, we have seen that current work is usually related to nonlinear optimization with non-convex obstacle avoidance constraints, shown in Chpaters 4, 7, 9, 10. Although the complexity could be more or less optimized by warm starting (Chapter 7), reference trajectories (Chapter 9), or local planner guidance (Chapter 10), we still suffer from the non-convexity or the non-linearity of obstacle avoidance. We expect future work about DCBF would be on how to linearize the obstacles locally or globally in order to convert the trajectory generation problem into a convex optimization. This direction would be relatively hard but iterative [100] or sequential [84] approaches could be potential tools to handle these problems, but it involved intensive engineering. An alternative approach would be converting nonlinear control barrier functions into some linear ones in a set of local convex regions in the state space and this would result the problem into a mixed-integer programming.

11.2.3 Relation between Continuous-time and Discrete-time domain CBFs

Besides the problems mentioned above, another interesting topic would be on how to bridge the safety between the continuous-time domain and the discrete-time domain. In the continuous-time domain, researchers are required to describe the system in the control affine form, while lacks prediction behavior, while are also required to do nmpc-like optimization for the trajectory generation problem, while lacks of convexity or linearity. A two-layer approach [163] has been tried to address this challenge and safety can also be enhanced through predictive terms [195]. Future interesting work could be a direct combination between these two parts while offering advantages in both continuous-time and discrete-time domains. This could be potentially achieved by reinforcement learning over $h(.)$ and $\alpha(.)$ and $\gamma(.)$ in the continuous-time domain, even the system dynamics $f(.)$ and $g(.)$, shown in (3.9). The learning errors

can be treated as robust term in the discrete-time domain. However, these learning approach could be hard as we try to find some sparse features in the latent space for the trajectory generation.

In summary, there are still numerous opportunities for improving performance of optimization with control barrier functions, and this thesis only explores a few aspects of them. There are also likely many new ideas to explore for the convex optimization, differential geometry related to the safety term using control barrier functions, which will help them become more widely used in real-life applications.

Bibliography

[1] Fayçal Ben Adda and Jacky Cresson. "About non-differentiable functions". In: *Journal of Mathematical Analysis and Applications* 263.2 (2001), pp. 721–737.

[2] Ayush Agrawal and Koushil Sreenath. "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation". In: *Proceedings of Robotics: Science and Systems*. 2017.

[3] Devansh R. Agrawal and Dimitra Panagou. "Safe Control Synthesis via Input Constrained Control Barrier Functions". In: *IEEE Conference on Decision and Control*. 2021, pp. 6113–6118.

[4] Eugenio Alcalá et al. "Autonomous racing using linear parameter varying-model predictive control (lpv-mpc)". In: *Control Engineering Practice* 95 (2020), p. 104270.

[5] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 242–252.

[6] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*. Vol. 26. Birkhäuser, 2012.

[7] Hassan Almubarak et al. "Safety Embedded Differential Dynamic Programming Using Discrete Barrier States". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2755–2762.

[8] Mohammed Alshiekh et al. "Safe reinforcement learning via shielding". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[9] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. "Control barrier function based quadratic programs with application to adaptive cruise control". In: *IEEE International Conference on Decision and Control*. 2014.

[10] Aaron D Ames and Matthew Powell. "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs". In: *Control of Cyber-Physical Systems*. Springer, 2013, pp. 219–240.

[11] Aaron D Ames et al. "Control barrier function based quadratic programs for safety critical systems". In: *IEEE Transactions on Automatic Control* (2016).

[12] Aaron D Ames et al. "Control barrier functions: Theory and applications". In: *European Control Conference*. 2019.

[13] Aaron D. Ames et al. "Rapidly Exponentially Stabilizing Control Lyapunov Functions and Hybrid Zero Dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[14] Joel AE Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[15] Nikolaos Athanasopoulos, George Bitsoris, and Mircea Lazar. "Construction of invariant polytopic sets with specified complexity". In: *International Journal of Control* 87.8 (2014), pp. 1681–1693.

[16] Sangjae Bae et al. "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network". In: *American Control Conference*. 2020, pp. 1209–1216.

[17] Somil Bansal and Claire 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.

[18] Somil Bansal et al. "Hamilton-Jacobi reachability: A brief overview and recent advances". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2242–2253.

[19] Robert G Bartle and Donald R Sherbert. *Introduction to real analysis*. Vol. 2. Wiley New York, 2000.

[20] Michael J Best and Nilotpal Chakravarti. "Stability of linearly constrained convex quadratic programs". In: *Journal of Optimization Theory and Applications* 64.1 (1990), pp. 43–53.

[21] Jayanth Bhargav et al. "Track based Offline Policy Learning for Overtaking Maneuvers with Autonomous Racecars". In: *2021 IEEE International Conference on Robotics and Automation (ICRA) Workshop for "Opportunities and Challenges with Autonomous Racing"*. 2021.

[22] Lorenz T Biegler and Victor M Zavala. "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization". In: *Computers & Chemical Engineering* (2009).

[23] Lars Blackmore, Masahiro Ono, and Brian C. Williams. "Chance-Constrained Optimal Path Planning With Obstacles". In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.

[24] R. Bohlin and L.E. Kavraki. "Path planning using lazy PRM". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. 2000, 521–528 vol.1.

[25] Saeed Amirfarhangi Bonab and Ali Emadi. "Optimization-based Path Planning for an Autonomous Vehicle in a Racing Track". In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. 2019, pp. 3823–3828.

[26] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[27] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[28] Joseph Breeden and Dimitra Panagou. "High Relative Degree Control Barrier Functions Under Input Constraints". In: *IEEE Conference on Decision and Control*. 2021, pp. 6119–6124.

[29] Tim Brüdigam et al. "Gaussian Process-based Stochastic Model Predictive Control for Overtaking in Autonomous Racing". In: *2021 IEEE International Conference on Robotics and Automation (ICRA) Workshop for "Opportunities and Challenges with Autonomous Racing"*. 2021.

[30] Monimoy Bujarbaruah et al. "Adaptive MPC for Iterative Tasks". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 6322–6327.

[31] Danilo Caporale et al. "Towards the Design of Robotic Drivers for Full-Scale Self-Driving Racing Cars". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 5643–5649.

[32] Guillermo A. Castillo et al. "Robust Feedback Motion Policy Design Using Reinforcement Learning on a 3D Digit Bipedal Robot". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 5136–5143.

[33] Kwan Ho Ryan Chan et al. "Deep Networks from the Principle of Rate Reduction". In: *2021 International Conference on Learning Representations*. 2021.

[34] Jiajia Chen et al. "Lane change path planning based on piecewise Bezier curve for autonomous vehicle". In: *Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety*. 2013, pp. 17–22.

[35] Mo Chen et al. "FaSTrack:A Modular Framework for Real-Time Motion Planning and Guaranteed Safe Tracking". In: *IEEE Transactions on Automatic Control* 66.12 (2021), pp. 5861–5876.

[36] Xiang Chen and Wenhao Zhang. "Lane Change Control for Self-Driving Vehicle Based on Model Predictive Control Considering the Instability of Sensor Detection". In: *International Conference on Mechanical, Control and Computer Engineering*. 2019, pp. 346–3465.

[37] Yimin Chen, Chuan Hu, and Junmin Wang. "Impaired Driver Assistance Control With Gain-Scheduling Composite Nonlinear Feedback for Vehicle Trajectory Tracking". In: *Journal of Dynamic Systems, Measurement, and Control* 142.7 (2020).

[38] Yuxiao Chen, Huei Peng, and Jessy Grizzle. "Obstacle avoidance for low-speed autonomous vehicles with barrier function". In: *IEEE Transactions on Control Systems Technology* 26.1 (2017), pp. 194–206.

[39] Yuxiao Chen et al. "Enhancing the performance of a safe controller via supervised learning for truck lateral control". In: *Journal of Dynamic Systems, Measurement, and Control* 141.10 (2019).

[40] Richard Cheng et al. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3387–3395.

[41] Jason Choi et al. "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions". In: *Proceedings of Robotics: Science and Systems*. 2020.

[42] Yinlam Chow et al. "A Lyapunov-Based Approach to Safe Reinforcement Learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 8103–8112.

[43] Yinlam Chow et al. "Risk-constrained reinforcement learning with percentile risk criteria". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6070–6120.

[44] Max H. Cohen and Calin Belta. "Approximate Optimal Control for Safety-Critical Systems with Control Barrier Functions". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 2062–2067.

[45] Jorge Cortes. "Discontinuous dynamical systems". In: *IEEE Control Systems Magazine* 28.3 (2008), pp. 36–73.

[46] Hongkai Dai et al. "Lyapunov-stable neural-network control". In: *Proceedings of Robotics: Science and Systems*. 2021.

[47] Robin Deits and Russ Tedrake. "Computing large convex regions of obstacle-free space through semidefinite programming". In: *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 109–124.

[48] Robin Deits and Russ Tedrake. "Efficient mixed-integer planning for UAVs in cluttered environments". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 42–49.

[49] František Duchoň et al. "Path planning with modified a star algorithm for a mobile robot". In: *Procedia Engineering* 96 (2014), pp. 59–69.

[50] David D Fan, Ali-akbar Agha-mohammadi, and Evangelos A Theodorou. "Deep learning tubes for tube mpc". In: *Proceedings of Robotics: Science and Systems*. 2020.

[51] Huckleberry Febbo et al. "Moving obstacle avoidance for large, high-speed autonomous ground vehicles". In: *American Control Conference (ACC)*. 2017, pp. 5568–5573.

[52] Federica Ferraguti et al. "A Control Barrier Function Approach for Maximizing Performance While Fulfilling to ISO/TS 15066 Regulations". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5921–5928.

[53] Roya Firoozi, Xiaojing Zhang, and Francesco Borrelli. "Formation and reconfiguration of tight multi-lane platoons". In: *Control Engineering Practice* 108 (2021), p. 104714.

[54] Roya Firoozi et al. "A distributed multi-robot coordination algorithm for navigation in tight environments". In: *arXiv preprint arXiv:2006.11492* (2020).

[55] Jaime F. Fisac et al. "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8550–8556.

[56] Gene F Franklin et al. *Feedback control of dynamic systems*. Vol. 4. Prentice hall Upper Saddle River, NJ, 2002.

[57] Janick V Frasch et al. "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles". In: *European Control Conference (ECC)*. 2013, pp. 4136–4141.

[58] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[59] Florian Fuchs et al. "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4257–4264.

[60] Aditya Gahlawat et al. "L1-GP: L1 adaptive control with Bayesian learning". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 826–837.

[61] Kevin Galloway et al. "Torque Saturation in Bipedal Robotic Walking Through Control Lyapunov Function-Based Quadratic Programs". In: *IEEE Access* 3 (2015), pp. 323–332.

[62] E.G. Gilbert and C.-P. Foo. "Computing the distance between general convex objects in three-dimensional space". In: *IEEE Transactions on Robotics and Automation* 6.1 (1990), pp. 53–61.

[63] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE J. Robot. Autom.* 4.2 (1988), pp. 193–203.

[64] Scott Gilroy et al. "Autonomous Navigation for Quadrupedal Robots with Optimized Jumping through Constrained Obstacles". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. 2021, pp. 2132–2139.

[65] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. "Boolean Composability of Constraints and Control Synthesis for Multi-Robot Systems via Nonsmooth Control Barrier Functions". In: *IEEE Conference on Control Technology and Applications*. 2018, pp. 897–902.

[66] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. "Nonsmooth Barrier Functions With Applications to Multi-Robot Systems". In: *IEEE Control Systems Letters* 1.2 (2017), pp. 310–315.

[67] Jonathan Y. Goh, Tushar Goel, and J. Christian Gerdes. "Toward Automated Vehicle Control Beyond the Stability Limits: Drifting Along a General Path". In: *Journal of Dynamic Systems, Measurement, and Control* 142.2 (Nov. 2019).

[68] Yukai Gong et al. "Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway". In: *American Control Conference (ACC)*. 2019, pp. 4559–4566.

[69] Ruben Grandia et al. "Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control". In: *2021 IEEE International Conference on Robotics and Automation*. 2021.

[70] Ruben Grandia et al. "Nonlinear Model Predictive Control of Robotic Systems with Control Lyapunov Functions". In: *Proceedings of Robotics: Science and Systems*. 2020.

[71] Ignacio E Grossmann. "Review of nonlinear mixed-integer and disjunctive programming techniques". In: *Optimization and engineering* 3.3 (2002), pp. 227–252.

[72] Long Han et al. "Bézier curve based path planning for autonomous vehicle in urban environment". In: *2010 IEEE Intelligent Vehicles Symposium*. 2010, pp. 1036–1042.

[73] Suiyi He, Jun Zeng, and Koushil Sreenath. "Autonomous Racing with Multiple Vehicles using a Parallelized Optimization with Safety Guarantee using Control Barrier Functions". In: *IEEE International Conference on Robotics and Automation*. 2022.

[74] Suiyi He et al. "Rule-Based Safety-Critical Control Design using Control Barrier Functions with Application to Autonomous Lane Change". In: *2021 American Control Conference*. 2021, pp. 178–185.

[75] Alexander Heilmeier et al. "Minimum curvature trajectory planning and control for an autonomous race car". In: *Vehicle System Dynamics* (2019).

[76] Sylvia L. Herbert et al. "FaSTrack: A modular framework for fast and guaranteed safe motion planning". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 1517–1522.

[77] Martin Herceg et al. "Multi-Parametric Toolbox 3.0". In: *European Control Conference (ECC)*. 2013, pp. 502–510.

[78] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars". In: *European Control Conference (ECC)*. 2018, pp. 1341–1348.

[79] Francois R Hogan and Alberto Rodriguez. "Reactive planar non-prehensile manipulation with hybrid model predictive control". In: *The International Journal of Robotics Research* (2020).

[80] William E Howden. "The sofa problem". In: *The computer journal* 11.3 (1968), pp. 299–301.

[81] Shao-Chen Hsu, Xiangru Xu, and Aaron D. Ames. "Control barrier function based quadratic programs with application to bipedal robotic walking". In: *2015 American Control Conference (ACC)*. 2015, pp. 4542–4548.

[82] Yeping Hu, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic prediction of vehicle semantic intention and motion". In: *IEEE Intelligent Vehicles Symposium*. 2018, pp. 307–313.

[83] Yiwen Huang and Yan Chen. "Switched Control Barrier Function With Applications to Vehicle Safety Control". In: *ASME Dynamic Systems and Control Conference*. 2020.

[84] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *International conference on learning and intelligent optimization*. Springer. 2011, pp. 507–523.

[85] Y.K. Hwang and N. Ahuja. "A potential field approach to path planning". In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 23–32.

[86] Jemin Hwangbo et al. "Control of a Quadrotor With Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103.

[87] Fahad Islam et al. "RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution". In: *2012 IEEE International Conference on Mechatronics and Automation*. 2012, pp. 1651–1656.

[88] Boris Ivanovic et al. "BaRC: Backward Reachability Curriculum for Robotic Reinforcement Learning". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 15–21.

[89] Achin Jain and Manfred Morari. "Computing the racing line using Bayesian optimization". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 6192–6197.

[90] Mrdjan Jankovic. "Robust control barrier functions for constrained stabilization of nonlinear systems". In: *Automatica* 96 (2018), pp. 359–367.

[91] Ming Jin and Javad Lavaei. "Stability-Certified Reinforcement Learning: A Control-Theoretic Perspective". In: *IEEE Access* 8 (2020), pp. 229086–229100.

[92] Ajinkya A Joshi, Diane L Peters, and Jennifer M Bastiaan. *Autonomous Lane Change Control Using Proportional-Integral-Derivative Controller and Bicycle Model*. Tech. rep. SAE Technical Paper, 2020.

[93] Girish Joshi and Girish Chowdhary. "Deep Model Reference Adaptive Control". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 4601–4608.

[94] Chanyoung Jung et al. "Game-Theoretic Model Predictive Control with Data-Driven Identification of Vehicle Model for Head-to-Head Autonomous Racing". In: *2021 IEEE International Conference on Robotics and Automation (ICRA) Workshop for "Opportunities and Challenges with Autonomous Racing"*. 2021.

[95] Juraj Kabzan et al. "Learning-Based Model Predictive Control for Autonomous Racing". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3363–3370.

[96] Sanket Kamthe and Marc Deisenroth. "Data-efficient reinforcement learning with probabilistic model predictive control". In: *International conference on artificial intelligence and statistics*. PMLR. 2018, pp. 1701–1710.

[97] Nitin R Kapania, John Subosits, and J Christian Gerdes. "A sequential two-step algorithm for fast generation of vehicle racing trajectories". In: *Journal of Dynamic Systems, Measurement, and Control* 138.9 (2016).

[98] Nitin R. Kapania and J. Christian Gerdes. "Learning at the Racetrack: Data-Driven Methods to Improve Racing Performance Over Multiple Laps". In: *IEEE Transactions on Vehicular Technology* 69.8 (2020), pp. 8232–8242.

[99] Alexander Katriniok. "Control-sharing Control Barrier Functions for Intersection Automation under Input Constraints". In: *European Control Conference (ECC)*. 2022.

[100] Carl T Kelley. *Iterative methods for optimization*. SIAM, 1999.

[101] III Kennedy Monroe et al. "Optimal Paths for Polygonal Robots in SE(2)". In: *Journal of Mechanisms and Robotics* 10.2 (Feb. 2018).

[102] Pradeep Khosla and Richard Volpe. "Superquadric artificial potentials for obstacle avoidance and approach". In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. 1988, pp. 1778–1784.

[103] Diethard Klatte and Bernd Kummer. "Stability properties of infima and optimal solutions of parametric optimization problems". In: *Nondifferentiable Optimization: Motivations and Applications*. Springer, 1985, pp. 215–229.

[104] Johannes Köhler et al. "A robust adaptive model predictive control framework for nonlinear uncertain systems". In: *International Journal of Robust and Nonlinear Control* 31.18 (2021), pp. 8725–8749.

[105] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *IEEE Intelligent Vehicles Symposium*. 2015, pp. 1094–1099.

[106] Hanna Krasowski, Xiao Wang, and Matthias Althoff. "Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7.

[107] Miroslav Krstic and Matt Bement. "Nonovershooting Control of Strict-Feedback Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 51.12 (2006), pp. 1938–1943.

[108] Grüne Lars and P Jürgen. *Nonlinear model predictive control theory and algorithms*. Springer, 2011.

[109] Joonho Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science Robotics* 5.47 (2020), eabc5986.

[110] Bai Li and Zhijiang Shao. "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles". In: *Knowledge-Based Systems* 86 (2015), pp. 11–20.

[111] Nan Li et al. *Autonomous racecar control in head-to-head competition using Mixed-Integer Quadratic Programming*. Tech. rep. Institut Polytechnique de Paris, 2021.

[112] Zhongyu Li, Christine Cummings, and Koushil Sreenath. "Animated Cassie: A Dynamic Relatable Robotic Character". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 3739–3746.

[113] Zhongyu Li et al. "Bridging Model-based Safety and Model-free Reinforcement Learning through System Identification of Low Dimensional Linear Models". In: *Proceedings of Robotics: Science and Systems*. 2022.

[114] Zhongyu Li et al. "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2811–2817.

[115] Zhongyu Li et al. "Vision-Aided Autonomous Navigation of Underactuated Bipedal Robots in Height-Constrained Environments". In: *arXiv preprint arXiv:2109.05714* (2021).

[116] Lars Lindemann et al. "Learning Hybrid Control Barrier Functions from Data". In: *Conference on Robot Learning*. PMLR. 2021, pp. 1351–1370.

[117] Alexander Liniger, Alexander Domahidi, and Manfred Morari. "Optimization-based autonomous racing of 1: 43 scale RC cars". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.

[118] Alexander Liniger and John Lygeros. "A Noncooperative Game Approach to Autonomous Racing". In: *IEEE Transactions on Control Systems Technology* 28.3 (2020), pp. 884–897.

[119] Hairong Liu, Wenyu Liu, and Longin Jan Latecki. "Convex shape decomposition". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 97–104.

[120] Johan Lofberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *2004 IEEE International Conference on Robotics and Automation*. 2004, pp. 284–289.

[121] Daniele Loiacono et al. "The 2009 Simulated Car Racing Championship". In: *IEEE Transactions on Computational Intelligence and AI in Games* 2.2 (2010), pp. 131–147.

[122] Brett T. Lopez, Jean-Jacques E. Slotine, and Jonathan P. How. "Robust Adaptive Control Barrier Functions: An Adaptive and Data-Driven Approach to Safety". In: *IEEE Control Systems Letters* 5.3 (2021), pp. 1031–1036.

[123] Yiwei Lyu, Wenhao Luo, and John M. Dolan. "Probabilistic Safety-Assured Adaptive Merging Control for Autonomous Vehicles". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 10764–10770.

[124] Haitong Ma et al. "Feasibility Enhancement of Constrained Receding Horizon Control Using Generalized Control Barrier Function". In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. 2021, pp. 551–557.

[125] Haitong Ma et al. "Model-based Constrained Reinforcement Learning using Generalized Control Barrier Function". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 4552–4559.

[126] Mathias Marley, Roger Skjetne, and Andrew R. Teel. "Synergistic control barrier functions with application to obstacle avoidance for nonholonomic vehicles". In: *2021 American Control Conference*. 2021, pp. 243–249.

[127] Mathias Marley et al. "Maneuvering with safety guarantees using control barrier functions". In: *13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2021*. Vol. 54. 16. 2021, pp. 370–377.

[128] Zahra Marvi and Bahare Kiumarsi. "Safe reinforcement learning: A control barrier function optimization approach". In: *International Journal of Robust and Nonlinear Control* 31.6 (2021), pp. 1923–1940.

[129] Aakar Mehra et al. "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars". In: *American Control Conference (ACC)*. 2015, pp. 1411–1418.

[130] Teodor Mihai Moldovan and Pieter Abbeel. "Safe exploration in markov decision processes". In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*. 2012, pp. 1451–1458.

[131] Michael Montemerlo et al. "Junior: The stanford entry in the urban challenge". In: *Journal of field Robotics* 25.9 (2008), pp. 569–597.

[132] Alex Nash et al. "Theta^*: Any-angle path planning on grids". In: *AAAI*. Vol. 7. 2007, pp. 1177–1183.

[133] Quan Nguyen and Koushil Sreenath. "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints". In: *American Control Conference*. 2016, pp. 322–328.

[134] Quan Nguyen and Koushil Sreenath. "Optimal robust control for constrained nonlinear hybrid systems with application to bipedal locomotion". In: *2016 American Control Conference*. 2016, pp. 4807–4813.

[135] Quan Nguyen and Koushil Sreenath. "Safety-critical control for dynamical bipedal walking with precise footstep placement". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 147–154.

[136] Quan Nguyen et al. "3d dynamic walking on stepping stones with control barrier functions". In: *IEEE International Conference on Decision and Control*. 2016.

[137] Quan Nguyen et al. "Dynamic bipedal locomotion over stochastic discrete terrain". In: *The International Journal of Robotics Research* (2018).

[138] Quan T Nguyen. "Robust and Adaptive Dynamic Walking of Bipedal Robots". PhD thesis. Carnegie Mellon University, 2017.

[139] Hussein Obeid et al. "Barrier function-based adaptive sliding mode control". In: *Automatica* 93 (2018), pp. 540–544.

[140] Rafael Oliveira et al. "Learning to Race Through Coordinate Descent Bayesian Optimisation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6431–6438.

[141] Chris J. Ostafew, Angela P. Schoellig, and Timothy D. Barfoot. "Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking". In: *The International Journal of Robotics Research* 35.13 (2016), pp. 1547–1563.

[142] Rushen B Patel and Paul J Goulart. "Trajectory generation for aircraft avoidance maneuvers using online optimization". In: *Journal of guidance, control, and dynamics* 34.1 (2011), pp. 218–230.

[143] Xue Bin Peng et al. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: (July 2020).

[144] Xue Bin Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3803–3810.

[145] Alejandro Perez et al. "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2537–2542.

[146] Etienne Perot et al. "End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 474–475.

[147] Philip Polack et al. "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: *IEEE Intelligent Vehicles Symposium*. 2017, pp. 812–818.

[148] "Pushing the limits: From lanekeeping to autonomous racing". In: *Annual Reviews in Control* 35.1 (2011), pp. 137–148.

[149] Xiangjun Qian et al. "Motion planning for urban autonomous driving using Bézier curves and MPC". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 826–833.

[150] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[151] Saša V. Rakovic and Miroslav Baric. "Parameterized Robust Control Invariant Sets for Linear Systems: Theoretical Advances and Computational Remarks". In: *IEEE Transactions on Automatic Control* 55.7 (2010), pp. 1599–1614.

[152] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2017.

[153] Adrian Remonda et al. "Formula RL: Deep Reinforcement Learning for Autonomous Racing using Telemetry Data". In: *IJCAI Workshop on Scaling-Up Reinforcement Learning: SURL@IJCAI*. 2019.

[154] A. Richards and J.P. How. "Aircraft trajectory planning with collision avoidance using mixed integer linear programming". In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*. Vol. 3. 2002, 1936–1941 vol.3.

[155] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems". In: *Conference on Robot Learning*. PMLR. 2018, pp. 466–476.

[156] Alexander Robey et al. "Learning Control Barrier Functions from Expert Demonstrations". In: (2020), pp. 3717–3724.

[157] Alexander Robey et al. "Learning Robust Hybrid Control Barrier Functions for Uncertain Systems". In: *IFAC-PapersOnLine*. Vol. 54. 5. 7th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2021. 2021, pp. 1–6.

[158] Muhammad Zakiyullah Romdlony and Bayu Jayawardhana. "Stabilization with guaranteed safety using control Lyapunov–barrier function". In: *Automatica* 66 (2016), pp. 39–47.

[159] Ugo Rosolia and Aaron D. Ames. "Multi-Rate Control Design Leveraging Control Barrier Functions and Model Predictive Control Policies". In: *IEEE Control Systems Letters* 5.3 (2021), pp. 1007–1012.

[160] Ugo Rosolia and Francesco Borrelli. "Learning How to Autonomously Race a Car: A Predictive Control Approach". In: *IEEE Transactions on Control Systems Technology* 28.6 (2020), pp. 2713–2719.

[161] Ugo Rosolia and Francesco Borrelli. "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework". In: *IEEE Transactions on Automatic Control* 63.7 (2018), pp. 1883–1896.

[162] Ugo Rosolia, Stijn De Bruyne, and Andrew G. Alleyne. "Autonomous Vehicle Control: A Nonconvex Approach for Obstacle Avoidance". In: *IEEE Transactions on Control Systems Technology* 25.2 (2017), pp. 469–484.

[163] Ugo Rosolia, Andrew Singletary, and Aaron D Ames. "Unified Multi-Rate Control: from Low Level Actuation to High Level Planning". In: *arXiv preprint arXiv:2012.06558* (2020).

[164] Matthias Rungger and Paulo Tabuada. "Computing Robust Controlled Invariant Sets of Linear Systems". In: *IEEE Transactions on Automatic Control* 62.7 (2017), pp. 3665–3670.

[165] Patrick Saint-Pierre. "Approximation of the viability kernel". In: *Applied Mathematics and Optimization* 29.2 (1994), pp. 187–209.

[166] Manuel Schmidt et al. "An Interaction-Aware Lane Change Behavior Planner for Automated Vehicles on Highways Based on Polygon Clipping". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1876–1883.

[167] John Schulman et al. "Motion planning with sequential convex optimization and convex collision checking". In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[168] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[169] Nicola Scianca et al. "MPC for Humanoid Gait Generation: Stability and Feasibility". In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1171–1188.

[170] Basav Sen, John D Smith, Wassim G Najm, et al. *Analysis of lane change crashes*. Tech. rep. United States. National Highway Traffic Safety Administration, 2003.

[171] Xu Shen et al. "Collision Avoidance in Tightly-Constrained Environments without Coordination: a Hierarchical Control Approach". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2674–2680.

[172] Jonah Siekmann et al. "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning". In: *Proceedings of Robotics: Science and Systems*. 2021.

[173] Jonah Siekmann et al. "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 7309–7315.

[174] Andrew Singletary et al. "Safety-Critical Rapid Aerial Exploration of Unknown Environments". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10270–10276.

[175] Tong Duy Son and Quan Nguyen. "Safety-Critical Control for Non-affine Nonlinear Systems with Application on Autonomous Vehicle". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 7623–7628.

[176] Yunlong Song et al. "Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9403–9409.

[177] Eduardo D Sontag. "A Lyapunov-like characterization of asymptotic controllability". In: *SIAM journal on control and optimization* 21.3 (1983), pp. 462–471.

[178] Eric Squires, Pietro Pierpaoli, and Magnus Egerstedt. "Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance". In: *2018 IEEE Conference on Control Technology and Applications (CCTA)*. 2018, pp. 1656–1661.

[179] Mohit Srinivasan et al. "Synthesis of Control Barrier Functions Using a Supervised Machine Learning Approach". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 7139–7145.

[180] Sirish Srinivasan, Sebastian Nicolas Giles, and Alexander Liniger. "A Holistic Motion Planning and Control Solution to Challenge a Professional Racecar Driver". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7854–7860.

[181] Tim Stahl et al. "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 3149–3154.

[182] William J Stein and Timothy R Neuman. *Mitigation strategies for design exceptions*. Tech. rep. United States. Federal Highway Administration. Office of Safety, 2007.

[183] Yanan Sui et al. "Safe exploration for optimization with Gaussian processes". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 997–1005.

[184] Andrew Taylor et al. "Learning for safety-critical control with control barrier functions". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 708–717.

[185] Andrew J. Taylor and Aaron D. Ames. "Adaptive Safety with Control Barrier Functions". In: *2020 American Control Conference (ACC)*. 2020, pp. 1399–1405.

[186] Sangli Teng et al. "Toward Safety-Aware Informative Motion Planning for Legged Robots". In: (2021). arXiv: `2103.14252 [cs.RO]`.

[187] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. "Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions". In: *2022 American Control Conference (ACC)*. 2022.

[188] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. "Safety-Critical Control and Planning for Obstacle Avoidance between Polytopes with Control Barrier Functions". In: *IEEE International Conference on Robotics and Automation*. 2022.

[189] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[190] Valerio Turri et al. "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads". In: *International Conference on Intelligent Transportation Systems*. 2013.

[191] Charlott Vallon et al. "A machine learning approach for personalized autonomous lane change initiation and control". In: *IEEE Intelligent Vehicles Symposium*. 2017, pp. 1590–1595.

[192] José L. Vázquez et al. "Optimization-Based Hierarchical Motion Planning for Autonomous Racing". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2397–2403.

[193] Robin Verschueren et al. "Towards time-optimal race car driving using nonlinear MPC in real-time". In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 2505–2510.

[194] Eugene Vinitsky et al. "Robust reinforcement learning using adversarial populations". In: *arXiv preprint arXiv:2008.01825* (2020).

[195] Kim P Wabersich and Melanie N Zeilinger. "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control". In: *arXiv preprint arXiv:2105.10241* (2021).

[196] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1 (2006), pp. 25–57.

[197] Shakti N Wadekar et al. "Towards End-to-End Deep Learning for Autonomous Racing: On Data Collection and a Unified Architecture for Steering and Throttle Prediction". In: *arXiv preprint arXiv:2105.01799* (2021).

[198] Mingyu Wang et al. "Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios". In: *IEEE Transactions on Robotics* 37.4 (2021), pp. 1313–1325.

[199] Rongyao Wang, Yiqiang Han, and Umesh Vaidya. *Deep Koopman Data-Driven Optimal Control Framework for Autonomous Racing*. Tech. rep. EasyChair, 2021.

[200] Dustin J. Webb and Jur van den Berg. "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5054–5061.

[201] Colin Wei, Sham Kakade, and Tengyu Ma. "The implicit and explicit regularization effects of dropout". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 10181–10192.

[202] Moritz Werling et al. "Optimal trajectory generation for dynamic street scenarios in a frenet frame". In: *IEEE International Conference on Robotics and Automation*. 2010, pp. 987–993.

[203] Tyler Westenbroek et al. "Feedback Linearization for Uncertain Systems via Reinforcement Learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1364–1371.

[204] Tyler Westenbroek et al. "Learning Min-norm Stabilizing Control Laws for Systems with Unknown Dynamics". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 737–744.

[205] Peter Wieland and Frank Allgöwer. "Constructive safety using control barrier functions". In: *IFAC Proceedings Volumes* 40.12 (2007), pp. 462–467.

[206] Adrian G Wills and William P Heath. "Barrier function based model predictive control". In: *Automatica* 40.8 (2004), pp. 1415–1422.

[207] Christian Wissing et al. "Probabilistic time-to-lane-change prediction on highways". In: *IEEE Intelligent Vehicles Symposium*. 2017, pp. 1452–1457.

[208] John Wright and Yi Ma. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2021.

[209] Guofan Wu and Koushil Sreenath. "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds". In: *American Control Conference*. 2015, pp. 2038–2044.

[210] Guofan Wu and Koushil Sreenath. "Safety-Critical Control of a 3D Quadrotor With Range-Limited Sensing". In: *Dynamic Systems and Control Conference*. Oct. 2016.

[211] Xiangyu Wu et al. *Model-free online motion adaptation for energy efficient flights of multicopters*. 2021. arXiv: `2108.03807 [cs.RO]`.

[212] Zhe Wu et al. "Control Lyapunov-Barrier function-based model predictive control of nonlinear systems". In: *Automatica* 109 (2019), p. 108508.

[213] Anxing Xiao et al. "Robotic Guide Dog: Leading a Human with Leash-Guided Hybrid Physical Interaction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 11470–11476.

[214] Wei Xiao and Calin Belta. "Control barrier functions for systems with high relative degree". In: *IEEE Conference on Decision and Control*. 2019, pp. 474–479.

[215] Wei Xiao, Calin A Belta, and Christos G Cassandras. "Sufficient conditions for feasibility of optimal control problems using control barrier functions". In: *Automatica* 135 (2022), p. 109960.

[216] Zhaoming Xie et al. "Feedback Control For Cassie With Deep Reinforcement Learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1241–1246.

[217] Xiangru Xu et al. "Robustness of control barrier functions for safety critical control". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 54–61.

[218] Shakiba Yaghoubi, Georgios Fainekos, and Sriram Sankaranarayanan. "Training Neural Network Controllers Using Control Barrier Functions in the Presence of Disturbances". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6.

[219] Lizhi Yang et al. "Bayesian Optimization Meets Hybrid Zero Dynamics: Safe Parameter Learning for Bipedal Locomotion Control". In: *IEEE International Conference on Robotics and Automation*. 2022.

[220] Wen Yao et al. "Learning lane change trajectories from on-road driving data". In: *2012 IEEE Intelligent Vehicles Symposium*. 2012, pp. 885–890.

[221] Yongsoon Yoon et al. "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles". In: *Control Engineering Practice* 17.7 (2009), pp. 741–750.

[222] Ming Yue et al. "Quintic polynomial-based obstacle avoidance trajectory planning and tracking control framework for tractor-trailer system". In: *International Journal of Control, Automation and Systems* 17.10 (2019), pp. 2634–2646.

[223] Jun Zeng, Zhongyu Li, and Koushil Sreenath. "Enhancing Feasibility and Safety of Nonlinear Model Predictive Control with Discrete-Time Control Barrier Functions". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021, pp. 6137–6144.

[224] Jun Zeng, Bike Zhang, and Koushil Sreenath. "Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function". In: *American Control Conference (ACC)*. 2021, pp. 3882–3889.

[225] Jun Zeng et al. "Differential Flatness Based Path Planning With Direct Collocation on Hybrid Modes for a Quadrotor With a Cable-Suspended Payload". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3074–3081.

[226] Jun Zeng et al. "Safety-Critical Control using Optimal-decay Control Barrier Function with Guaranteed Point-wise Feasibility". In: *American Control Conference (ACC)*. 2021, pp. 3856–3863.

[227] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. "Optimization-Based Collision Avoidance". In: *IEEE Transactions on Control Systems Technology* 29.3 (2021), pp. 972–983.

[228] Xiaojing Zhang et al. "Autonomous Parking Using Optimization-Based Collision Avoidance". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 4327–4332.

[229] Huarong Zheng et al. "Model predictive control for intelligent vehicle lane change". In: *ICTIS: Improving Multimodal Transportation Systems-Information, Safety, and Integration*. 2013, pp. 265–276.