Correct-By-Construction Control Synthesis for Systems with Disturbance and Uncertainty

by

Yuxiao Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2018

Doctoral Committee:
Professor Huei Peng, Co-Chair
Professor Jessy Grizzle, Co-Chair
Assistant Professor Necmiye Ozay
Assistant Professor Ram Vasudevan

Yuxiao Chen

chenyx@umich.edu

ORCID iD: 0000-0001-5276-7156

Acknowledgments

First and foremost, I would like to thank my two advisors, Professor Huei Peng and Professor Jessy W. Grizzle, for their kind help and support throughout my graduate life. They brought me on board and guided me through my five-year journey as a Ph.D. student with their wisdom and enthusiasm. I am indeed fortunate to have had the opportunity to work with them.

I would like to thank Professor Necmiye Ozay and Professor Ram Vasudevan, with whom I collaborated while working on my Ph.D. The monthly meeting of the CPS group was an amazing source of information and ideas and I received a great deal of help from the members.

I would like to thank the National Science Foundation for funding my research, and more importantly, giving me the opportunity to meet awesome people outside the University of Michigan such as Professor Aaron Ames and Professor Paulo Tabuada. Collaborating with them and their students truly opened my eyes to many interesting tools and topics about CPS.

My Ph.D. life would not have been as smooth as it has been without the great friends I made at U-M. From my lab mates Chiao-ting, Byungjoo, Will, Xiaowu, Tianyou, Ding, Ziheng, Steve, Xianan, Su-yang, Geunsob, Songan and Nauman, to my teammates in the U-M fencing team such as Nitty and David, they all helped enliven my life and helped me in both my academic and personal life. My special thanks go to Shaobing and Minghan, who helped me with the experiment at Mcity.

I would also like to thank my dear girlfriend, Grace Mo, for the abundant love and joy she has given me. You are the apple of my eye.

Finally, my greatest appreciation goes to my parents, Jianping Chen and Guiying Chen, for their unconditional love and support. I consider myself carrying out part of their dreams, which makes me feel that I am never alone, even in the tough times.

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Abbreviations

ACC	**A**daptive **C**ruise **C**ontrol
ADMM	**A**lternating **D**irection **M**ethod of **M**ultipliers
CBF	**C**ontrol **B**arrier **F**unction
CLF	**C**ontrol **L**yapunov **F**unction
GPS	**G**lobal **P**ositioning **S**ystem
LK	**L**ane **K**eeping
LMI	**L**inear **M**atrix **I**nequality
LP	**L**inear **P**rogramming
MIP	**M**ixed **I**nteger **P**rogramming
mRCI	**m**inimum **R**obust **C**ontrol **I**nvariant set
ODE	**O**rdinary **D**ifferential **E**quation
PDE	**P**artial **D**ifferential **E**quation
RCI	**R**obust **C**ontrol **I**nvariant set
RTK	**R**eal-**T**ime **K**inematic
SDP	**S**emi**d**efinite **P**rogramming
SOS	**S**um **o**f **S**quares

List of Symbols

Spaces		**Sets**	
Set of real number	$\mathbb{R}$	Continuous state space	$\mathcal{X}$
n-dimensional Euclidean space	$\mathbb{R}^n$	Continuous input space	$\mathcal{U}$
open (closed) positive orthant of $\mathbb{R}^n$	$\mathbb{R}^n_{>0}(\mathbb{R}^n_{\geq 0})$	Continuous disturbance space	$\mathcal{D}$
Set of integers	$\mathbb{Z}$	Subset of state space	$\mathcal{S}$
Set of nonnegative integers	$\mathbb{Z}_{\geq 0}$		
Set of real polynomials of x	$\mathbb{R}[x]$		
Set of SOS polynomials of x	$\Sigma[x]$		
Symbols in dynamic systems			
State	x	Input	u
disturbance	d	Output	y or z
Time	t	Sampling time	T_s
Lyapunov function	V		
Math operators and symbols			
Convex hull	$Conv$	Euclidean inner product	$\langle \cdot, \cdot \rangle$
Indicator function	$\mathbf{1}(\cdot)$	Lie derivative of h w.r.t. f	$\mathcal{L}_f h = \frac{\partial h}{\partial x} \cdot f$
Quadratic form of polynomial f	Q_f	Minimum eigenvalue	$\lambda_{\min}$
High degree Lie derivative	$\mathcal{L}_f^n h = \mathcal{L}_f \mathcal{L}_f^{n-1} h$		
Basic logic and Linear Temporal Logic			
Logical conjunction	$\wedge$	Logic disjunction	$\vee$
Logic negation	$\neg$	Logic implication	$\rightarrow$
Temporal next	$\bigcirc$	Temporal always	$\Box$
Temporal eventually	$\Diamond$	Temporal until	**U**

Abstract

This dissertation focuses on correct-by-construction control synthesis for Cyber-Physical Systems (CPS) under model uncertainty and disturbance. CPSs are systems that interact with the physical world and perform complicated dynamic tasks where safety is often the overriding factor. Correct-by-construction control synthesis is a concept that provides formal performance guarantees to closed-loop systems by rigorous mathematic reasoning. Since CPSs interact with the environment, disturbance and modeling uncertainty are critical to the success of the control synthesis. Disturbance and uncertainty may come from a variety of sources, such as exogenous disturbance, the disturbance caused by co-existing controllers and modeling uncertainty. To better accommodate the different types of disturbance and uncertainty, the verification and control synthesis methods must be chosen accordingly. Four approaches are included in this dissertation. First, to deal with exogenous disturbance, a polar algorithm is developed to compute an avoidable set for obstacle avoidance. Second, a supervised learning based method is proposed to design a good student controller that has safety built-in and rarely triggers the intervention of the supervisory controller, thus targeting the design of the student controller. Third, to deal with the disturbance caused by co-existing controllers, a Lyapunov verification method is proposed to formally verify the safety of coexisting controllers while respecting the confidentiality requirement. Finally, a data-driven approach is proposed to deal with model uncertainty. A minimal robust control invariant set is computed for an uncertain dynamic system without a given model by first identifying the set of admissible models and then simultaneously computing the invariant set while selecting the optimal model. The proposed methods are applicable to many real-world applications and reflect the notion of using the structure of the system to achieve performance guarantees without being overly conservative.

Chapter 1 Introduction

We live in a world that is increasingly cybernetic and automatic. Computers have become an essential part of our daily lives, with more automatic features being developed to free humans from tedious and repetitive labor. Examples include autonomous vehicles, humanoid robots, exoskeletons, and automated assembly lines. However, people tend not to accept or trust these systems. Accidents caused by software errors in these systems have attracted a great deal of attention and aggravated people's worry (viz. the accidents by Tesla's autopilot and the recent tragic death of a pedestrian caused by Uber's autonomous vehicle experiment). These worries are not unjustified since there is typically no formal guarantee that the software (the control algorithm, in particular) will meet the specifications, even for safety. The difficulty stems from the fact that the software designed must interact with the physical world, which is difficult to model and has a great deal of uncertainty. Systems that involve computation, communication, and interaction with the physical world are usually called Cyber-Physical Systems (CPSs). As pointed out in [1], CPS requires control/computing co-design. Two significant problems arise: tools are needed to verify whether the design satisfies the specification; tools are needed to synthesize a controller that satisfies the given specification.

In the current state of the art,control of CPS is commonly obtained by extensive experience; various ad-hoc control algorithms such as PID, Linear Quadratic Regulator (LQR), Model Predictive Control (MPC); and laborious trial and error. However, closed-loop performance lacks formal guarantees of meeting the specifications. In search of a solution, researchers have turned their attention to the formal methods, which were proposed to provide formal performance guarantees in the computer science community targeting discrete transition systems. While introducing these formal methods and ideas to control synthesis, several issues must be resolved. First, control systems typically operate in the continuous time and state space, while states in transition systems are typically discrete. Second, the control systems need to deal with disturbance from the environment and model uncertainty. Although techniques exist in formal methods that are able to deal with uncertainty, they typically model the uncertainty as nondeterministic

transitions, thus not accurately capturing the true nature of the disturbance and uncertainty faced by control synthesis. Using the general framework of nondeterministic transitions, therefore, result in unnecessary conservativeness.

To fill this gap, the specific structure of the disturbance and uncertainty should be utilized. This dissertation presents several methods that handle different types of disturbance and uncertainty. Different as they may seem, they all abide by the same fundamental notion, which is to take advantage of the specific structure of the disturbance and uncertainty and provide a guarantee of closed-loop performance without being overly conservative. In particular, this dissertation focuses on the verification and synthesis of safety specifications for CPSs under model uncertainty and disturbance, which is the simplest and yet most fundamental type of specification.

1.1 Literature review

Control theories have been developing for decades, providing tools such as the Proportional–Integral–Derivative controller (PID) [2], the Linear Quadratic Regulators (LQR) [3] and Model Predictive Control (MPC) [4], yet most of the existing methods focus on optimality or some narrowly-defined closed-loop performance such as bandwidth and steady-state error. For safety-critical systems, however, a rigorous analysis of the time domain performance of the closed-loop system is required. The concept of correct-by-construction has therefore been adopted from the domain of computer science. In the realm of software engineering, tools for verification of the software have been developed since the 1980s, and these methods are usually referred to as formal methods [5]. A typical formal method approach modeling the systems as transition systems with discrete states and the state transitions are triggered by certain actions as inputs [6]. To express specifications that concern the system evolution over time, various temporal logics were developed, among which Linear Temporal Logic (LTL) is the most popular one. It is capable of expressing a complicated specification for the system by using not only logic symbols such as $\wedge$ (and), $\vee$ (or) and $\neg$ (not), but also temporal logic symbols such as $\Box$ (always), $\Diamond$ (eventually) and $\mathbf{U}$ (until) [7, 8]. With the transition system to describe the system and LTL to express the specifications, model checking tools are then developed to verify whether a transition system always satisfies a specification [9, 10]. With the capability of describing a complicated specification about the system behavior, and verifying whether a system satisfies the specification, the system verification process for transition systems then becomes rigorous, exhaustive and

automated. The promise is fascinating: a model-checking tool either returns a formal guarantee that the system always satisfies the specification or a counter-example showing how, in certain circumstances, the system fails the specification. Based on verification, there are tools that can synthesize policies (sometimes also referred to as symbolic controllers) that satisfy a given specification in an automated fashion [11]. An important milestone in the development of synthesis tools is the synthesis protocol for generalized reactivity(1) (GR(1)) specifications [12]. A GR(1) specification has the following form:

$$\bigwedge_{i=1}^{n} \square\diamond J_i^1 \rightarrow \bigwedge_{j=1}^{m} \square\diamond J_j^2, \tag{1.1}$$

where J^1 are specifications for the environment and J^2 are specifications for the system. GR(1) specifications include a wide range of specifications encountered in engineering applications and can be efficiently synthesized with cubic complexity. Toolboxes such as TuLiP [13] were developed for verification and synthesis of transition systems. The creation of similar tools for the control synthesis would be of great benefit to the community.

Some attempts have been made to introduce the formal verification and synthesis procedure into the control design process. One major difference between software design and control synthesis is the form of state space. Typically, a physics-driven dynamic system is described by ordinary differential equations (ODE) or difference equations, and has continuous states, whereas software design typically deals with discrete state space. For some applications, a hierarchy can be constructed that divides the tasks of the system into high-level tasks and low-level tasks, where the high-level controller treats the system as discrete transition systems, and the low-level controller is designed using traditional feedback control techniques to execute the low-level tasks assigned by the high-level controller [14, 15]. This hierarchical approach is limited, however, in that traditional feedback techniques typically do not provide performance guarantees for the low-level execution, except for stability.

To extend the formal methods designed for discrete transition systems to dynamic systems, many approaches have been proposed, including timed automata [6] and hybrid automaton [16]. A hybrid automaton is an extension of a finite state machine which includes a mode that ranges over finitely many discrete values and a finite set of real-valued variables. The evolution of the continuous variables is specified for each discrete mode, and edges between modes are annotated with guards and updates that specify discrete transitions [17]. However, as pointed out in [18], the

reachability problem is undecidable even for some very simple hybrid automata. While there have been efforts in the reachability computation for hybrid automata, such as over-approximations using zonotope [19] and support functions [20], these analyses are highly conservative and are limited to linear dynamics.

In order to adopt the tools developed for transition systems, a more brute force approach would be to decompose the state space into a finite collection of subsets. This process is called abstraction or bisimulation [21-23]. The discrete transition system is constructed as a bisimilar or approximately bisimilar transition system of the continuous system via reachability computation. If the continuous system happens to be incrementally stable, the reachability computation can be simplified and the approach can be less conservative in the sense that fewer nondeterministic transitions are needed. Since the dynamic system is now described as a transition system, both the temporal logic specifications and the model checking and reactive synthesis tools can be adopted [24]. This method has the advantage of being able to handle complicated specifications with the help of temporal logic tools, but it does not scale well since the abstraction step discretizes the whole state space and requires potentially difficult reachability computation, which has at least exponential complexity w.r.t. the state dimension. Moreover, if the system is not incrementally stable, the abstraction might generate many nondeterministic transitions, which is not the nature of the original continuous dynamics.

In addition to directly compute the reachable set, inductive invariance is widely used for verification of CPS. The idea is to find a set that contains the set of the initial states, satisfies the safety condition and is forward invariant, i.e., if the initial state is inside the set, it will remain in the set for the future evolution. Methods that use inductive invariance include the barrier certificate [25], inductive verification with polynomial templates [26], and efforts to unify continuous invariants and discrete invariants [27, 28]. The advantage of the inductive invariance type methods is that they do not require reachability computation. However, one disadvantage is that the supported specification is limited, typically only for safety specifications.

Methods have also been developed within the control community for reachability and the forward invariance analysis of dynamic systems, such as the barrier certificate [29], control invariant set with control barrier function [30-32], Hamilton Jacobi Partial Differential Equation (PDE) [33] and the occupation measure [34].

The barrier certificate method constructs a continuous function that is positive in the safe set, and negative in the danger set. In addition, for every point on the boundary of its 0-level set, the vector field is pointing towards the positive side of the boundary; therefore, the boundary of the 0-level set serves as a barrier, and the function serves as a barrier that proves the safety of the system. There exists a result of converse barrier certificate stating that a barrier certificate exists for any safe system with some mild assumptions [35].

A control invariant set is a set that for any state within the set, there exists a control strategy that keeps the future evolution of the state in the set. Control barrier function is then constructed based on the control invariant set, which works with other control strategies and serves as a supervisor to guarantee safety.

The Hamilton Jacobi (HJ) method formulates the problem in the form of a Hamilton-Jacobi-Isaac PDE that describes the optimal solution to a zero-sum differential game between the disturbance and control inputs. The 0-level set of the value function of such PDE represents the winning set of the control (or disturbance, depending on the setting). If the state is not inside the winning set of the disturbance, then for any disturbance allowed, there exists a control input trajectory that prevents the state from reaching the danger set within the horizon T. The HJ method is applicable to a wide range of applications since only mild restrictions on the form of dynamics are imposed. However, the PDE is solved numerically using the level set method, which is essentially dynamic programming; it does not scale well with the state dimension.

The occupation measure approach formulates an infinite-dimensional linear programming, which in theory calculates the reachable set of a system, then uses Semidefinite Programming (SDP) to approximate the solution. Its advantage lies in the fact that it can transform a nonconvex problem to a convex one by working on the measure space rather than the function space.

Many of the above-mentioned methods suffer from high complexity, and do not scale well. To reduce the size of the problem, compositional verification has been proposed. A typical one is the assume-guarantee approach, which views the interaction between subsystems as a disturbance to one another [36]. The protocol works as follows: for each subsystem, assuming that the disturbance from other subsystems are bounded by certain bounds, the disturbance from this subsystem to other subsystems can be bounded within certain sets, and the bound conditions check out, that is, the bound guaranteed by the synthesis is no larger than the bound assumed by the synthesis, then the

whole system satisfies the specification. Separable control invariant set [31], dissipative system verification [37] and some decentralized control synthesis methods are all based on this idea.

Another significant difference between software design and control synthesis is disturbance and model uncertainty. In software design, as complicated as it may be, designers deal with digits and logic. Though non-deterministic transitions are allowed and supported by many model checking tools, these non-deterministic transitions are well defined and finite. In real physical systems, however, designers must deal with real physical dynamics that typically are not modeled perfectly, and disturbance may come from a variety of sources. Treating everything as a non-deterministic transition does not capture the structure of the potential disturbances in control systems and typically leads to unnecessary conservativeness.

The lesson I learned while treating different types of disturbance and uncertainty is to take advantage of their specific structure. As the famous "no free lunch" theorem states, there is no such method that outperforms other methods in every situation. The best way to treat modeling uncertainty and disturbance depends on the specific situation.

1.2 Dissertation organization

The remainder of the dissertation is organized as follows, important tools used throughout this dissertation are reviewed in Chapter 2. Then four subproblems are discussed to illustrate different approaches for handling disturbance and uncertainty to guarantee safety. The first two subproblems are built around the concept of control barrier functions. First, in Chapter 3, the focus is on the exogenous disturbance, a polar method is proposed that computes an avoidable set and constructs a control barrier function (CBF) for moving obstacle avoidance of low-speed autonomous vehicles. The CBF serves as a supervisory controller that watches over whatever navigation controller the vehicle uses, and guarantees safety by intervening only when imminent danger is detected.

After discussing the supervisor, in Chapter 4, the second subproblem concerns the design of a student controller that works with a CBF as the supervisor. The CBF is combined with supervised learning to train a student controller that has safety built in and rarely triggers intervention from the CBF.

In Chapter 5, the third subproblem deals with the verification and synthesis of co-existing but non-cooperative controllers for a single dynamic system. In this case, the control actions from other controllers act as disturbances. A Lyapunov function and dual decomposition based method is developed to verify the composition of multiple controllers without exposing the control algorithms of each controller.

Finally, in Chapter 6, the last subproblem deals with modeling uncertainty. A model of an uncertain dynamic system consists of the nominal model and the uncertainty characterization. A data-driven method is proposed that approximates a minimal robust control invariant set. First, the set of all admissible models, that is, models that explains a finite sequence of measurement data, is identified from the measurement data. Then an iterative algorithm is developed to approximate a minimal robust control invariant set by simultaneously selecting the optimal model from the set of admissible models and minimizing the size of the invariant set. This method is able to approximate a minimal robust invariant set without a model being given and leverages the tradeoff between additive uncertainty, multiplicative uncertainty, and the nominal model. Some experimental work of the proposed methods is presented in Chapter 7.

Chapter 2 Review of important tools

Some important tools are used throughout this dissertation, including Sum of Squares programming and Control Barrier Function. A brief review of them is given in this chapter.

2.1 Sum of squares programming

First, a review of Sum of Squares (SOS) programming is presented in this section, which is used to construct CBF in Chapter 4 and the Lyapunov certificate of performance guarantee in Chapter 5. The application of SOS includes searching for a Lyapunov function to prove stability [38], constructing a CBF [39], synthesizing a nonlinear controller with safety guarantee [40], and calculating funnels around trajectories in motion planning [41].

A real coefficient polynomial p of x is a sum of squares if there exist polynomials $f_1,\ldots, f_m$ such that

$$p(x)=\sum_{i=1}^{m} f_i^2(x). \tag{2.1}$$

The set of polynomials with real coefficients in x is denoted as $\mathbb{R}[x]$, and the set of all sum of squares polynomials in x is denoted as $\Sigma[x]$. The problem of searching for the SOS decomposition of a polynomial can be formulated as semidefinite programming and thus solved efficiently by SDP solvers, Tools such as SOSTOOLS [42], YALMIP [43], and SPOTLESS [44] automatically calculate the associated parametric matrices and convert an SOS problem to an SDP, which in turn provides a sufficient condition for a polynomial to be nonnegative. In general, SOS is a sufficient but not necessary condition for the positive definiteness of polynomials, with the exception of a few known special cases [45]. Moreover, there exists a denseness result: every non-negative polynomial is almost an SOS; namely, it can be approximated by a sequence of SOS polynomials, see Theorem 4.1 in [46]. The verification of the SOS condition is solved by SDP in the space of symmetric real matrices.

Thanks to results from algebraic geometry, most notably Putinar's and Stengle's Positivstellensatz, SOS is extended from verifying the global non-negativity of a polynomial to verifying the non-negativity of a polynomial on a specific semialgebraic set, see page 2 of [46], page 13 of [45], and [47]. The formal Positivstellensatz states the following: consider a cone P of $\mathbb{R}[x]$ that satisfies

$$\begin{aligned} &a,b\in P\rightarrow a+b\in P,\\ &a,b\in P\rightarrow a\cdot b\in P,\\ &a\in\mathbb{R}[x]\rightarrow a^2\in P, \end{aligned} \tag{2.2}$$

which immediately implies $\Sigma[x]\subseteq P$.

Lemma 2.1: *Let* $(f_j)_{j=1,\ldots s},(g_k)_{i=1,\ldots t},(h_l)_{l=1,\ldots u}$ *be finite families of polynomials in* $\mathbb{R}[x]$. *Denote by* P *the cone generated by* $(f_i)_{i=1,\ldots s}$, M *the multiplicative monoid generated by* $(g_k)_{i=1,\ldots t}$, *and* I *the ideal generated by* $(h_l)_{l=1,\ldots u}$, *then the following properties are equivalent:*

1. *The set* $\left\{x\in\mathbb{R}^n \mid f_j(x)\geq 0, j=1,\ldots s, f_k(x)\neq 0, k=1,\ldots t, h_l(x)=0, l=1,\ldots u\right\}$ *is empty*
2. *There exists* $f\in P, g\in M, h\in I$ *such that* $f+g^2+h=0$.

This is Theorem 4.6 in [45]; see the proof therein.

Now suppose the goal is to verify whether $q(x)\geq 0\rightarrow p(x)\geq 0$ is true. This condition is equivalent to verifying whether the set $\left\{x\mid p(x)\leq 0, q(x)\geq 0, p(x)\neq 0\right\}$ is empty. Applying *Lemma 2.1*, a necessary and sufficient condition is to find $s_i\in\Sigma[x]$ and $r\in\mathbb{Z}_{\geq 0}$ such that

$$s_0-s_1p+s_2q-s_3pq+p^r=0. \tag{2.3}$$

Sufficiency is easily verified. However, this condition, in general, is hard to verify, especially when part of the coefficients of p is yet to be determined. A stronger condition uses fewer SOS multipliers: if there exists $s_0,s_1\in\Sigma[x]$ such that $s_0-p+sq=0$, then $q(x)\geq 0\rightarrow p(x)\geq 0$. Indeed, since $s_1(x)$ is nonnegative for all x and $p(x)-s_1(x)q(x)=s_0(x)\geq 0$, the conclusion follows. This argument provides a sufficient condition for a polynomial to be nonnegative inside a semialgebraic set. However, since it is not the complete form of the PositivStallensatz, and in practice, the order of the SOS multipliers' order is limited by the computation power, the condition

is merely sufficient but not necessary. For a review of the Positivestallensatz result, see Chapter 2 of [48] for reference.

2.2 Control barrier functions

2.2.1 Overview of control barrier functions

Control barrier functions can be used to provide a safety guarantee to CPSs and generate a supervisory control structure that works in a plug-and-play fashion with any existing legacy controller. The supervisory controller is referred to as the "teacher", while the legacy controller is treated as the "student". The teacher does not intervene until an imminent threat to safety is detected, at which point the teacher uses minimum intervention to guarantee safety. It has a simple form yet can work as an add-on safety feature to most existing methods. The barrier certificate was developed to verify the safety features of a system without input [29, 49]. A CBF incorporates the control action and serves as a supervisor to guarantee safety, which was first proposed by Ames et al. in [30]. The original CBF takes the reciprocal form inspired by the logarithmic barrier in the interior point method developed for optimization. Then a zeroing CBF was proposed in [50], inheriting all the good properties of the reciprocal CBF and adding additional robustness to the method. The CBF was applied in areas such as ground vehicle control [50], biped robot walking [51] and swarm control of multiple robots [52]. A typical supervisory control structure with CBF is depicted below:

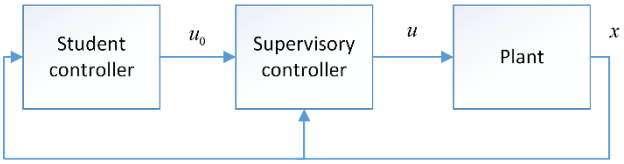

Figure 2.1 Supervisory control structure

The student controller can be any legacy controller, with its control command denoted as u_0; the CBF based supervisory controller solves for the final control input u to the system with the following optimization:

$$\begin{aligned} &\min \|u-u_0\| \\ &s.t.\, \mathcal{C}(u) \geq 0, \end{aligned} \tag{2.4}$$

where $\mathcal{C}(u)$ is the CBF condition, to be defined later.

2.2.2 Implementation of CBF

As mentioned earlier, there are two common types of control barrier function—reciprocal barrier, and zeroing barrier. The reciprocal barrier goes to infinity when the state approaches the boundary of the constraint, and is not defined when the constraint is not satisfied; the zeroing barrier is positive when the constraint is satisfied, negative when the constraint is not satisfied and zero at the boundary.

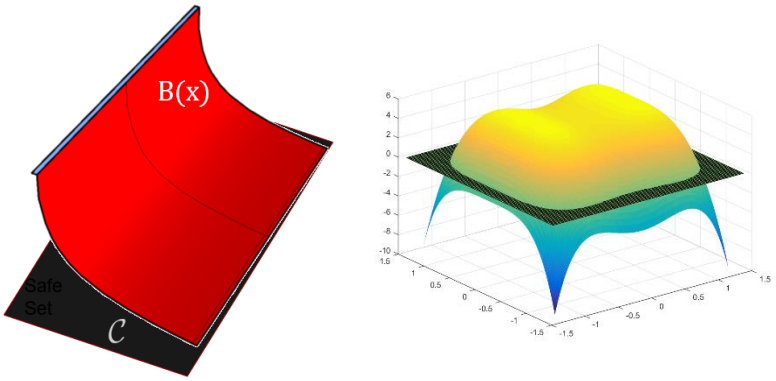

Figure 2.2 Reciprocal barrier and zeroing barrier

Although the reciprocal barrier is very similar to the logarithmic barrier used in optimization, as it is not defined on the other side of the boundary, the supervisory control fails when the constraint is violated. Two typical reciprocal barrier functions are

$$B(x) = -\log\frac{b(x)}{1+b(x)}, \qquad B(x) = \frac{1}{b(x)}, \tag{2.5}$$

where $b(x) > 0$ is the original constraint. In order to keep the constraint satisfied, the CBF should be finite; therefore, the barrier condition can be defined as

$$\dot{B} \le \gamma / B(x), \tag{2.6}$$

where γ is a positive constant. This condition gets tighter as the barrier function becomes larger. At the extreme, the RHS of the inequality drops to 0, restricting the CBF from growing.

The zeroing barrier, on the other hand, can be simply defined with $B(x) = b(x)$. The barrier condition can be set as

$$\dot{b} \ge -\gamma\alpha(b(x)), \tag{2.7}$$

where γ is a positive constant α is a class $\mathcal{K}$ function. A typical choice for α is simply the identify function. In this case, the barrier condition requires that the barrier derivative be non-negative at the boundary $(b(x)=0)$, and enforce exponential convergence to the set $\{x \mid b(x) \geq 0\}$ when the barrier constraint is violated. This setup provides some robustness to the formulation so that the supervisory structure tries to force the state to return to the safe set when the barrier is breached.

Remark 2.1: *The barrier condition is not unique; it is only required that* α *be a class* $\mathcal{K}$ *function, then the set invariance can be proved, (see Proposition 3 in [50] for detail). By tuning* γ*, the level of caution of the supervisory controller can be tuned. With a large* γ*, the barrier function will not be activated until* $b(x)$ *is very small; a smaller* γ *will make the supervisory control more cautious. However, a larger* γ *requires a faster convergence rate when* $b(x)<0$.

Remark 2.2: *In theory, when the supervisory control structure is enforced, the safety constraint will never be violated. However, this result is under the assumption that the model is perfect and there is no unknown disturbance. With modeling uncertainty, the barrier condition might be violated, but the CBF will try to let the state converge to the safe set exponentially.*

There seems to be some natural link between the Control Lyapunov Function (CLF) and the CBF. The CLF focuses on the convergence of the system to the equilibrium point while the CBF focuses on preventing the state from leaving the safe set. The CLF condition typically appears as

$$\mathcal{L}_f V + \mathcal{L}_g V u \leq 0, \tag{2.8}$$

where $V: \mathbb{R}^n \rightarrow \mathbb{R}$ is the CLF for a system with dynamics $\dot{x} = f(x) + g(x)u$, $\mathcal{L}_f V$ and $\mathcal{L}_g V$ are the Lie derivatives defined as

$$\begin{aligned} \mathcal{L}_f V &= \frac{\partial V}{\partial x} f(x), \\ \mathcal{L}_g V &= \frac{\partial V}{\partial x} g(x). \end{aligned} \tag{2.9}$$

Comparing (2.7) and (2.8), one may discover the difference between CLF and CBF. The CLF enforces the Lyapunov derivative to be negative or zero at any time, which can be overly restrictive for safety specifications. In contrast, the barrier requirement is loose when the state is far away from the boundary of the danger set, and gets tighter as the state approaches the boundary, giving the student controller some freedom when danger is not imminent.

2.2.3 Synthesis of control barrier functions

The synthesis of the CBF is not a trivial problem, and is closely related to the computation of the control invariant set. It should be noted that the barrier function, or control invariant set idea, is rather general. Depending on the specific properties of the problem, the formulation and computation methods may be quite different. HJ [33] is a powerful tool for computing the safe set and it can be extended to construct the CBF, see Appendix K, for example. For some systems, the CBF can be constructed using analytical methods, such as kinematic analysis [52-54]. The polar method is introduced in Chapter 3, which constructs a "polytopic" CBF. In this section, an SOS approach to constructing a CBF is reviewed, which is then used in Chapter 4. Only a synthesis procedure of the zeroing CBF will be discussed in this section as the zeroing CBF enjoys better properties compared to the reciprocal CBF, as stated in [50].

Let us consider a continuous dynamic system described by the following ODE:

$$\dot{x} = f(x) + g_u(x)u + g_d(x)d, \tag{2.10}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathcal{U}$ is the input and $d \in \mathcal{D}$ is the disturbance. Both u and d are bounded within some semialgebraic set.

Remark 2.3: *To make a distinction between the value of the control input and the mapping that determines the control input, u denotes the value of control input, and $u(\cdot)$ denotes the input as a function of other variables, such as the state and disturbance.*

There are a variety of settings for the synthesis; as an example, I choose a setting that enforces the CBF condition in (2.7) for all x inside $\{x \mid p(x) \geq 0\}$, a superset of $\{x \mid b(x) \geq 0\}$, and all $d \in \mathcal{D}$. If a larger level set of $b(x)$, say $\{x \mid b(x) \geq -c\}$ with $c > 0$, is contained inside $\{x \mid p(x) \geq 0\}$, then not only is $\{x \mid b(x) \geq 0\}$ robustly control invariant, it is also guaranteed that for any x inside $\{x \mid b(x) \geq -c\}$, there exists a control input sequence that robustly drives the state back to $\{x \mid b(x) \geq 0\}$ exponentially. For a given danger set X_d, and an initial set X_0, represented as semialgebraic sets: $X_0 = \{x \mid h_{x0}^i(x) \geq 0\}$, $X_d = \{x \mid h_{xd}^i(x) \geq 0\}$, the CBF b must satisfy the following:

$$
\begin{aligned}
&\forall x \in X_0, b(x) \geq 0, \\
&\forall x \in X_d, b(x) \leq 0, \\
&\forall x \in \{x \mid p(x) \geq 0\}, \forall d \in \mathcal{D}, \exists u \in \mathcal{U}, s.t. \\
&\frac{db}{dx}\left(f(x) + g_u(x)u + g_d(x)d\right) + \gamma\alpha\left(b(x)\right) \geq 0.
\end{aligned}
\tag{2.11}
$$

The first two conditions enforce the CBF to be positive in the initial set, and negative in the danger set; the last condition states that, within the 0-level set of p, for any possible disturbance, there always exists a feasible control input that renders the CBF condition satisfied. For simplicity, in the remainder of this section, α is chosen as the identity function. It is further assumed that the system is a polynomial dynamic system, that is, f, g_u and g_d are polynomials, $\mathcal{U}$ and $\mathcal{D}$ are semialgebraic sets. In particular, $\mathcal{U} = \{u \mid h_u(u) \geq 0\}$, $\mathcal{D} = \{d \mid h_d(d) \geq 0\}$.

Remark 2.4: *Here it is assumed that $\mathcal{U}$ and $\mathcal{D}$ do not depend on x, but this restriction can be relaxed by adding x to the semialgebraic set definition.*

The problem is not yet solvable by SOS because:

1. *the existence condition of the control input*
2. *the coupling between the control input and the CBF candidate.*

To resolve the first issue, the controller is restricted to be a polynomial controller $u(\cdot)$, which is a function of the state and measured disturbance. With this restriction, the second issue arises as the coupling between the controller and the CBF candidate generates a bilinear term of the decision variables, thus making the problem nonconvex. To resolve this issue, bilinear alternation is used, separating the problem into two parts: specifying a feedback controller and searching for a barrier function. Then the bilinear alternation process proceeds by alternating between two steps:

1. *fix the CBF candidate and search for a feedback controller,*
2. *fix the controller and search for a better CBF candidate.*

Given a barrier candidate b, the following SOS programming searches for a feedback controller that satisfies the input constraint and "almost" satisfies the barrier condition.

"Almost" means that it satisfies a relaxed CBF condition, and the SOS programming is minimizing the relaxation needed:

$$\begin{aligned}
&\min_{e,u(\cdot),s_1,s_2,s_{d1},s_{d2}} \quad e \quad s.t. \\
&h_u\left(u\left(x,d\right)\right)-s_1\left(x,d\right)p\left(x\right)-\sum_i s_{d1}^i\left(x,d\right)h_d^i\left(d\right)\in\Sigma\left[x,d\right], \\
&\frac{db}{dx}\left(f\left(x\right)+g_u\left(x\right)u\left(x,d\right)+g_d\left(x\right)d\right)+\gamma b\left(x\right)+s_2\left(x,d\right)p\left(x\right) \\
&-\sum_i s_{d2}^i\left(x,d\right)h_d^i\left(d\right)+ex^Tx\in\Sigma\left[x,d\right], \\
&s_1,s_2,s_{d1},s_{d2}\in\Sigma\left[x,d\right],
\end{aligned} \tag{2.12}$$

where $u\left(\cdot\right)$ is the polynomial control law depending on x and d. Whether it depends on d or a subset of d depends on whether some disturbance is measured and can be used as feedforward. s_1, s_2, s_{d1} and s_{d2} are the multipliers, SOS polynomial of x and d. e is the relaxation variable.

Remark 2.5: *The relaxation term should depend on the degree of the original polynomial in the SOS constraint. In this case, x^Tx is used as an example, but in some cases, a higher even order polynomial is needed to make the problem feasible.*

Given a control law, one can find a barrier certificate with the following SOS programming:

$$\begin{aligned}
&\min_{e,b,s_1,s_2,s_3,s_{d1},s_{d2}} \quad e \; s.t. \\
&b\left(x\right)-\sum_i s_1^i\left(x\right)h_{x0}^i\left(x\right)\in\Sigma\left[x\right], \\
&-b\left(x\right)-\sum_i s_2^i\left(x\right)h_{xd}^i\left(x\right)\in\Sigma\left[x\right], \\
&\frac{db}{dx}\left(f\left(x\right)+g_u\left(x\right)u\left(x,d\right)+g_d\left(x\right)d\right)+\gamma b\left(x\right)+ex^Tx \\
&-\sum_i s_{d1}^i\left(x,d\right)h_d^i\left(d\right)+s_{d2}\left(x,d\right)p\left(x\right)\in\Sigma\left[x,d\right], \\
&p\left(x\right)-s_3\left(x\right)b\left(x\right)\in\Sigma\left[x\right], \\
&s_1,s_2,s_3\in\Sigma\left[x\right],s_{d1},s_{d2}\in\Sigma\left[x,d\right].
\end{aligned} \tag{2.13}$$

The first two constraints restrict the CBF to be positive in X_0, and negative in X_d; the third constraint is the relaxed CBF condition; and the fourth constraint restricts the CBF 0-level set to be contained by the 0-level set of p. The SOS optimization may seem complicated, but essentially the SOS programming is structured based on Positivstellensatz and uses SOS multipliers to enforce non-negativeness constraints on different semialgebraic sets (check [45, 55] for more detail). One

can also vary X_0 and gradually increase the volume of X_0 until the problem becomes infeasible so that the volume of the 0-level set of b is maximized indirectly, as presented in [39]. The overall algorithm alternates between updating the controller and updating the CBF candidate until the CBF condition is satisfied without relaxation or no progress can be made. The SOS based algorithm finds a conservative CBF that guarantees the robust feasibility of the barrier condition. Due to the computation limitation, the order of the CBF candidate, the controller and the SOS multipliers is limited. By increasing the order of the polynomials, the method may find a less conservative CBF. Another limitation is that, due to the bilinear alternation, global optimality is not guaranteed. See [39] for more detail.

As mentioned previously, the format of the CBF varies in different applications. The SOS based method is applicable to polynomial systems, and results in a polynomial CBF, making the 0-level set a semialgebraic set.

Chapter 3 Polar method and obstacle avoidance

This chapter presents a method for solving the obstacle avoidance problem for low-speed autonomous vehicles. This method, referred to as the polar method, computes a polytopic avoidable set, whose complement is control invariant, and guarantees collision avoidance with a supervisory control structure. The supervisory control structure is implemented with the CBF, which works in a plug-and-play fashion with any existing navigation algorithm via a mixed integer program. The main difficulty being addressed in this problem is from exogenous disturbance, more specifically, the motion of the moving obstacles.

3.1 Introduction and motivation

Automotive companies are actively pursuing autonomous vehicles (AVs) to realize their potential for improved safety and mobility. Some of the efforts target high-speed applications, while others focus on low-speed applications, such as airport transport, driverless pods on urban streets, museum tours. In 2013, Hitachi announced their development of a single-passenger mobility-support robot "ROPITS", which uses stereo cameras and multiple laser radar sensors to navigate [56]. In Britain, LUTZ Pathfinder has tested a driverless "pod" vehicle [57]. Indoor autonomous robots have also been tested, including museum guiding robots Minerva [58] and KAPROS [59].

Low-speed autonomous vehicles differ from high-speed ones in two ways:

1. *there are no lane boundaries;*
2. *they share space with multiple pedestrians and stationary obstacles.*

A basic problem is to navigate the AV from an initial point to a target point within a reasonable amount of time while avoiding collision with obstacles. Although several methods have been proposed for obstacle avoidance of high-speed autonomous vehicles, such as Model Predictive Control [60, 61], Fuzzy logic [62], and the motion primitive method [63], they generally cannot guarantee safety.

On the other hand, when the operating speed is low, the problem of robot navigation with static or moving obstacles has been studied extensively. Cell decomposition was used in Minerva and tested with real tourists interacting with the robot [58]. Bis et al. extended the cell decomposition concept to deal with moving obstacle with known speed [64]. The potential field method, originally developed for stationary obstacles, was also extended to moving obstacles [65, 66]. However, none of these methods provides a safety guarantee. Van den Berg et al. proposed reciprocal collision avoidance, which provides a collision avoidance guarantee under the assumption that the vehicle speed can be controlled instantaneously [67]. However, this assumption is usually not valid in the real world since acceleration can typically be controlled directly, but not the speed. The Dynamic Window Approach (DWA) was proposed in [68], which guarantees that the vehicle will not collide with static obstacles. The DWA idea was further developed in [69] and in [70] for moving obstacles, with a heavy use of braking. The proposed method can guarantee the safety of low-speed autonomous vehicles by steering and braking, with steering preferred if braking is not necessary.

The safety assurance is rooted in the concept of control invariant set. The key challenge of this concept is the computation of the reachable set based on the system dynamics. Multiple methods of computing or approximating a control invariant set have been proposed in the literature, including barrier certificate [49], Hamilton-Jacobi method [33] and occupation measure [34, 71], as reviewed in Section 1.1.

The polar algorithm proposed in this chapter computes a polytopic avoidable set whose complement is an outer approximation of the reachable set, and control invariant. The gap between the reachable set and the avoidable set serves as a safety buffer. Then a control barrier function is constructed based on the avoidable set and implemented as a supervisory controller using Mixed Integer Programming (MIP). The remainder of the chapter is organized as follows. Section 3.2 presents the dynamic model and the problem formulation, Section 3.3 introduces the supervisory control structure, Section 3.4 presents the polar algorithm, Section 3.5 discusses the application of the polar algorithm on obstacle avoidance for autonomous vehicles, Section 3.6 shows the results and conclusion is drawn in Section 3.7.

3.2 Dynamic models and problem formulation

3.2.1 Dynamic models

Two dynamic models are used in this study. The first model describes vehicle motion in the Earth-fixed coordinates; the second model records the relative motion between the vehicle and a moving obstacle. When there are multiple obstacles, a copy of the second model can be created for each obstacle. For each type of obstacles, a maximum velocity and a geometric size are defined.

A unicycle model is used to represent the dynamics of the autonomous vehicle:

$$\begin{cases} \dot{X} = v\cos\psi & \dot{v} = a \\ \dot{Y} = v\sin\psi & \dot{\psi} = r \end{cases}, \tag{3.1}$$

where X and Y are the global Cartesian coordinates, v denotes the vehicle velocity and ψ is the heading angle. The inputs to the vehicle are acceleration a and yaw rate r.

$$u = (a, r). \tag{3.2}$$

The following constraints are assumed to hold:

$$\begin{aligned} v &\in [0, v_{\max}], \qquad r \in [-r_{\max}, r_{\max}], \\ a &\in [-a_{\max}, a_{\max}], \quad a^2 + v^2 r^2 \le \mu^2 g^2. \end{aligned} \tag{3.3}$$

where μ is the friction coefficient and g is the gravitational constant.

The motion relative to a moving obstacle is described with four states, as shown in Figure 3.1.

$$x = [\Delta X \quad \Delta Y \quad v \quad \theta]^T, \tag{3.4}$$

where ΔX and ΔY denote the relative position of the obstacle with respect to the vehicle in the global coordinates.

$$\begin{cases} \Delta X = X_d - X \\ \Delta Y = Y_d - Y \end{cases}, \begin{cases} \dot{X}_d = v_{dx} \\ \dot{Y}_d = v_{dy} \end{cases}, \tag{3.5}$$

where X_d and Y_d are the coordinates of the obstacle. The velocity of the obstacle v_d is a disturbance input and v_{dx}, v_{dy} are its components on X and Y directions. Relative heading angle θ is the difference between the AV heading angle and the yaw angle of the obstacle:

$$\theta = \psi - \tan^{-1}\left(\frac{\Delta Y}{\Delta X}\right). \tag{3.6}$$

The dynamic equations are as follows:

$$\begin{bmatrix} \Delta\dot{X} \\ \Delta\dot{Y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v_{dx} - v\cos\left(\theta + \tan^{-1}\left(\frac{\Delta Y}{\Delta X}\right)\right) \\ v_{dy} - v\sin\left(\theta + \tan^{-1}\left(\frac{\Delta Y}{\Delta X}\right)\right) \\ a \\ r + \frac{\sin\theta}{\Delta X^2 + \Delta Y^2} - \frac{v_{dx}\Delta Y - v_{dy}\Delta X}{\Delta X^2 + \Delta Y^2} \end{bmatrix}. \tag{3.7}$$

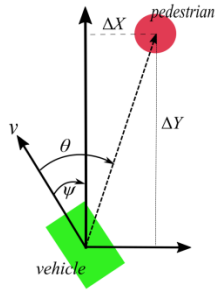

Figure 3.1 Coordinate system of the relative dynamic model

Both θ and ψ are restricted between $-\pi$ and π. In the remainder of this chapter, the dynamic system in (3.7) is denoted as Σ_0, and the set of state, input and disturbance are denoted as $\mathcal{S}_0$, $\mathcal{U}_0$ and $\mathcal{D}_0$, respectively.

3.2.2 Problem formulation

The goal of the AV is to reach a destination without colliding with any obstacle. Strictly speaking, under some circumstances, collision is inevitable, for example, when the AV is surrounded by hostile pursuers. In order to define a reasonable problem to solve, the concept of "passive friendly safety" proposed by Macek et al. in [72] is adopted and extended to multiple moving obstacles. The following rules are asserted:

(1)When the AV is stopped, any conflict is not considered a collision caused by the AV.

(2)When an obstacle runs into the AV from behind, it is not considered a collision caused by the AV.

The definition of collision from behind is

$$|\theta| \geq \frac{\pi}{2}. \tag{3.8}$$

The second rule is not needed for the single obstacle case since the vehicle can accelerate and avoid the collision. However, in the case of multiple moving obstacles, when an AV is threatened by an approaching obstacle from behind, neither accelerating (risk to others in front) nor slowing down (escalating the situation) is safe. These two rules are applied for all simulations in this chapter.

For simplicity, it is assumed that all obstacles are pedestrians, and both the vehicle and pedestrians are assumed to have a round shape with radius R_v and R_p, respectively. In simulations, collision is detected by the following rule:

$$\left(\sqrt{\Delta X^2 + \Delta Y^2} \leq R_v + R_p\right) \wedge (v > 0) \wedge \left(|\theta| \leq \frac{\pi}{2}\right). \tag{3.9}$$

In this study, the onboard sensors are assumed to measure all states accurately. The speed of the pedestrians is assumed to be bounded, and the bound is known:

$$\|v_d\| = \sqrt{v_{dx}{}^2 + v_{dy}{}^2} \leq v_{d\max}. \tag{3.10}$$

3.3 Supervisory control and avoidable set

The goal of the supervisory control is to avoid collision under all possible disturbance input, i.e., (3.9) is not violated by any obstacle at any time. First, based on the supervisory structure, the following three subsets of the state space $\mathcal{S}$ are defined, as shown in Figure 3.2.

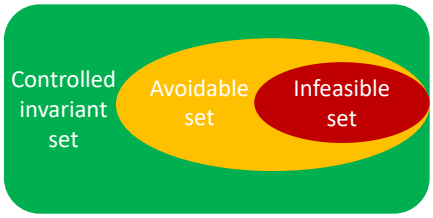

Figure 3.2 Three sets defining different stages of obstacle avoidance

(1) **Infeasible set:** As shown in Figure 3.2, the infeasible set is the region where a collision is not always preventable, that is, there exists a disturbance under which a collision will occur for any control input. However, note that if the pedestrian is neither hostile nor shrewd enough to choose this disturbance value, a collision may not occur. Once the AV is inside the infeasible

set, it should slow down and stop immediately. The infeasible set is denoted as X_{In} in the remainder of the chapter.

(2) **Avoidable set:** To guarantee that the AV never enters the infeasible set, a superset of the infeasible set is constructed, and its complement is control invariant. For a state in the avoidable set and outside of the infeasible set, a collision can be avoided by taking emergency action, to be explained later. If possible, the state should be driven out away from the avoidable set.

(3) **The control invariant set:** The complement of the avoidable set. Because it is control invariant, any state outside of the avoidable set can stay outside under all possible disturbances.

Denote the system dynamics as

$$\dot{x} = f(x,u,d), \tag{3.11}$$

where $x \in \mathcal{S}$, $u \in \mathcal{U}$ and $d \in \mathcal{D}$ are the state, control input, and the disturbance input, respectively. The mathematical condition for a set P_B to be avoidable is then

$$\begin{gathered} \forall x(0) \notin P_B, \forall t > 0, \forall s \in [0,t], \forall d(s) \in \mathcal{D}, \\ \exists u(s) \in \mathcal{U}, s.t.\ x(t) \notin P_B. \end{gathered} \tag{3.12}$$

If the set P_B is the zero level set of a real-valued function $b: \mathcal{S} \to \mathbb{R}$, i.e., P_B is characterized as $\{x |\ b(x) < 0\}$, then the set invariance condition becomes a boundary condition:

$$\begin{aligned} &\forall x \notin P_B, b(x) \geq 0; \forall x \in P_B, b(x) < 0 \\ &\forall x, b(x) = 0 \to \forall d \in \mathcal{D}, \exists u \in \mathcal{U}, \\ &s.t.\ \ \dot{b}(x) = (\nabla_x b)^T f(x,u,d) > 0. \end{aligned} \tag{3.13}$$

Suppose at a given point x_0 is on ∂P_B, i.e., the boundary of P_B, the normal vector exists. Denote the normal vector pointing outwards from P_B as $\vec{n}_{x_0}$, then the geometric condition of (3.13) is equivalent to

$$\forall d \in \mathcal{D}, \exists u \in \mathcal{U}\ s.t. \left\langle f(x_0,u,d), \vec{n}_{x_0} \right\rangle \geq 0. \tag{3.14}$$

Note the strict inequality is changed to non-strict inequality to simplify the computation. This can be done by introducing some small slack constant.

3.4 Polar algorithm

The key challenge of the supervisory control is to find an avoidable set that contains the infeasible set. To solve this problem, a polar algorithm is proposed. The infeasible set is represented by a bounded polytope containing the collision set as described in (3.9), then the polar algorithm solves for another polytope that contains the infeasible set and satisfies the boundary condition introduced in (3.12). The polar algorithm is applicable to a dynamic model in the following form

$$\dot{x} = Eu + Gd, x \in \mathcal{S}, u \in \mathcal{U}, d \in \mathcal{D}, \tag{3.15}$$

where $\mathcal{S}$, $\mathcal{U}$ and $\mathcal{D}$ are polytope; E and G are constant matrices. Note that there is no state dependent term in the state derivative. The dynamic model in Section 3.2 is simplified to this form by viewing all state dependent terms as disturbance, which is explained in detail in Section 3.5.2.

3.4.1 Polar of a polytope

In this section, the polar algorithm is introduced, which is built based on the dual property of polytope. A polytope P has two important elements: vertices and facets. For simplicity, only bounded polytopes with a finite number of vertices and bounding hyperplanes are considered. A bounded polytope $P \subseteq \mathbb{R}^n$ with the origin in its interior can be represented as a set of the convex combination of its vertices:

$$P = \left\{ x \mid x = \sum_i \lambda_i v_i, \sum_i \lambda_i = 1, \lambda_i \geq 0, \forall i \right\}, \tag{3.16}$$

where $\{v_i\} = P.V$ is the set of vertices of P. It can also be represented as an intersection of finitely many closed half spaces:

$$P = \bigcap_{H_i} \left\{ x \mid H_i^T x \leq 1 \right\}, \tag{3.17}$$

where $\left\{ x \mid H_i^T x \leq 1 \right\}$ are the bounding half spaces. $\{H_i\} \subset X^{\#}$ denotes the set of linear functionals that corresponds to the half spaces:

$$H_i(x) = H_i^T x, \tag{3.18}$$

where $X^{\#} = \{ f : X \to \mathbb{R} \mid f \text{ linear} \}$ is the algebraic dual of the vector space X. Since the algebraic dual of a Euclidean space is also a Euclidean space, the linear functionals in $X^{\#}$ are treated as vectors in the dual space and the set $\{H_i\}$ is denoted as $P.H$. In addition, half-spaces and

polytopes can also be defined in $X^{\#}$. The bounding hyperplane corresponding to a bounding half-space $\{x \mid H_i^T x \le 1\}$ is defined as

$$\{x \mid H_i^T x = 1\}, \tag{3.19}$$

and the corresponding facet is defined as

$$F(H_i) = P \cap \{x \mid H_i^T x = 1\}. \tag{3.20}$$

For polytopes, the normal vector of the facet $F(H_i)$ is simply H_i, which simplifies the condition in (3.14). The polar of P is a polytope in the dual space $X^{\#}$ defined as

$$P^{\Delta} = \{H \mid \forall x \in P, H^T x \le 1\}. \tag{3.21}$$

Because of the convexity and linearity of polytopes, a simpler definition is

$$P^{\Delta} = \{H \mid \forall v_i \in P.V, H^T v_i \le 1\}. \tag{3.22}$$

The vertices of P are mapped to the facets of P^{Δ}, and the facets of the P are mapped to the vertices of P^{Δ}, as shown in the following example.

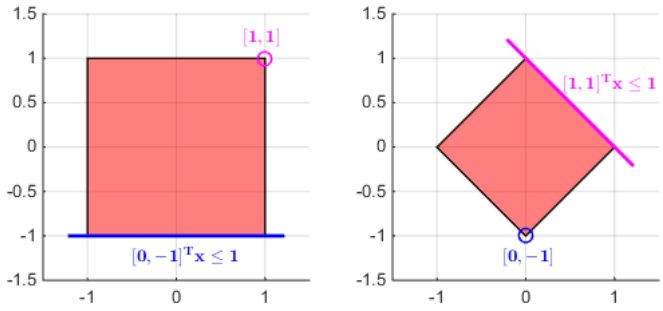

Figure 3.3 Example of polar of polytopes

For a bounded polytope P with the origin in its interior, the following hold:

(1) The polar of P, denoted as P^{Δ}, is a bounded polytope with the same dimension as P and containing the origin in the interior.

(2) The polar of P^{Δ}, denoted as, is the original polytope, i.e., $P^{\Delta\Delta} = P$.

(3) For any $H \in P^{\Delta}$, P is completely contained in the half-space $\{x \mid H^T x \le 1\}$, i.e.,

$$\forall H \in P^{\Delta}, \forall x \in P, H^T x \le 1. \tag{3.23}$$

(4) For any point H outside P^{Δ}, P is not completely contained in the half-space $\{x \mid H^T x \le 1\}$

$$\forall H \notin P^{\Delta}, \exists x \in P, H^T x > 1. \tag{3.24}$$

Because of the properties above, the polar concept provides a clear condition for polytope inclusion:

$$P_1 \subseteq P_2 \leftrightarrow P_2.H \subseteq P_1^{\Delta}. \tag{3.25}$$

This property is one of the building blocks of the polar algorithm. For more detailed properties of polar, see the textbook [73].

When finding an avoidable set that contains the infeasible set X_{In}, the set inclusion condition is enforced by the following constraint:

$$P_B.H \in X_{In}^{\Delta}. \tag{3.26}$$

3.4.2 Hyperplane orientation and boundary condition

The boundary condition of the avoidable set is interpreted as an orientation condition for the bounding hyperplanes. For each facet of a polytope P, suppose that the corresponding bounding hyperplane is $H^T x = 1$, the normal vector that points outwards from P is simply H.

For the dynamic system in (3.15), the avoidable set boundary condition in (3.14) becomes

$$\begin{aligned} &\forall H \in P.H, \forall d \in \mathcal{D}, \\ &\exists u \in \mathcal{U} \; s.t. \langle Eu + Gd, H \rangle \geq 0. \end{aligned} \tag{3.27}$$

In order to find all bounding hyperplane orientations that are valid for an avoidable set, input u is fixed first. For a bounding hyperplane $H^T x = 1$ to satisfy the boundary condition of the avoidable set with a fixed input u, the following inequality must hold:

$$\forall d \in \mathcal{D}, H^T \left(Eu + Gd \right) \geq 0. \tag{3.28}$$

Since $\mathcal{D}$ is a polytope and the system dynamics is linear, (3.28) is simplified to checking only its vertices:

$$\forall d \in \mathcal{D}.V, H^T \left(Eu + Gd \right) \geq 0. \tag{3.29}$$

Condition (3.29) defines a polytope in $X^{\#}$ and it is easy to check that $P_{Hs}{}^{u} = \left\{ H \mid \forall d \in \mathcal{D}.V, \left(Eu + Gd \right)^T H \geq 0 \right\}$ contains all the functionals corresponding to the bounding hyperplanes valid under input u. Define

$$P_{Hs} = \bigcup_{\forall u \in \mathcal{U}.V} P_{Hs}^{u}, \tag{3.30}$$

this union may not be convex, but using the linearity of the dynamics, the following theorem is true:

Theorem 3.1: *P_{Hs} is the maximal set of linear functionals that satisfies the boundary condition of an avoidable set for the dynamic system shown in (3.15).*

See Appendix A for proof.

Recall that there are two requirements for the avoidable set: set inclusion and boundary conditions. The set inclusion requirement is simplified to selecting bounding hyperplanes from the polar of the infeasible set X_{In}; the boundary conditions requirement is simplified to selecting bounding hyperplanes from P_{Hs}. Therefore, it is natural to intersect the two sets. Define

$$P_H = P_{Hs} \cap X_{In}^{\Delta}. \tag{3.31}$$

This set may not be convex since it is the intersection of a convex polytope and a nonconvex union of polytopes. Further, define

$$P_B = Conv\left(P_H\right)^{\Delta}, \tag{3.32}$$

where $Conv(\cdot)$ denotes the convex hull of a polytope. Then P_B possess the following properties:

Theorem 3.2: P_B is an avoidable set that contains X_{In}.

Theorem 3.3: P_B is the minimal avoidable set.

Theorem 3.4: *If the origin is in the interior of X_{In}, and $\overline{X_{In}} = X_{In} + c$, where c is a constant shifting vector, then $\overline{P_B} = P_B + c$ where $\overline{P_B}$ is the constructed avoidable set based on $\overline{X_{In}}$.*

The above claims mean that P_B is the minimal avoidable set that satisfies both set inclusion condition and boundary condition, and it is invariant with respect to the change of origin position.

See Appendix B, Appendix C, and Appendix D for proofs.

3.5 Avoidable set for low-speed autonomous vehicles

In this section, the avoidable set for a single moving obstacle is solved using the polar algorithm.

3.5.1 Infeasible set

The obstacle avoidance problem with a single moving obstacle can be formulated as a pursuit-evasion problem. In general, the vehicle can use both steering and braking to avoid collision (and both are used in the control implementation). In the computation of the infeasible set, however, only braking is used, which allows an easy extension to multiple pedestrian cases.

For a given initial condition, an obstacle's future position is bounded by:

$$\left(X_d(t)-X_d(0)\right)^2+\left(Y_d(t)-Y_d(0)\right)^2 \le \left(v_{d\max}t\right)^2, \tag{3.33}$$

then the vehicle's position is calculated by applying the maximum brake. By checking whether a collision occurs before the vehicle stops, the points in the state space can be then identified as either feasible (safety is guaranteed) or infeasible (safety cannot be guaranteed). A grid is selected with certain resolution in the state space and it is computed for each grid point whether or not a collision will occur. A polytope X_{In} is then found that contains all the infeasible grid points by computing their convex hull.

3.5.2 Avoidable set for the autonomous vehicle

Consider the relative dynamics shown in (3.7). It is rewritten as:

$$\begin{bmatrix} \Delta\dot{X} \\ \Delta\dot{Y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \bar{d}_1 \\ \bar{d}_2 \\ a \\ r+\dfrac{\sin\theta}{\Delta X^2+\Delta Y^2}+\bar{d}_3 \end{bmatrix}, \tag{3.34}$$

where

$$\begin{aligned} \bar{d}_1 &= v_{dx}-v\cos\left(\theta+\tan^{-1}\left(\Delta Y/\Delta X\right)\right), \\ \bar{d}_2 &= v_{dy}-v\sin\left(\theta+\tan^{-1}\left(\Delta Y/\Delta X\right)\right), \\ \bar{d}_3 &= \frac{\Delta X v_{dy}-\Delta Y v_{dx}}{\Delta X^2+\Delta Y^2}. \end{aligned} \tag{3.35}$$

It is then easy to find the bounds of the disturbance:

$$\begin{aligned} \bar{d}_1^{\,2}+\bar{d}_2^{\,2} &\le \left(v_{d\max}+v_{\max}\right)^2, \\ \left|\bar{d}_3\right| &\le \frac{v_{d\max}}{\sqrt{\Delta X^2+\Delta Y^2}}. \end{aligned} \tag{3.36}$$

From the collision avoidance condition,

$$\left(\sqrt{\Delta X^2+\Delta Y^2} \ge R_v+R_p\right) \Rightarrow \left(\left|\bar{d}_3\right| \le \frac{v_{d\max}}{R_v+R_p}\right). \tag{3.37}$$

Therefore, the dynamic is simplified as

$$\dot{x} = Eu + G\bar{d} + k,$$
$$k = \begin{bmatrix} 0 & 0 & 0 & \dfrac{\sin\theta}{\Delta X^2 + \Delta Y^2} \end{bmatrix}. \tag{3.38}$$

where

$$\bar{d}_1^{\,2} + \bar{d}_2^{\,2} \le \left(v_{d\max} + v_{\max}\right)^2 + v_{\max}^{\,2},$$
$$\left|\bar{d}_3\right| \le \frac{v_{d\max}}{R_v + R_p}. \tag{3.39}$$

Although the constraint for $\bar{d}_1$ and $\bar{d}_2$ is a circle and nonlinear, it can be outer approximated by a polygon. As the number of vertices grows, the approximation becomes more accurate. Therefore, the disturbance $\bar{d} = \begin{bmatrix} \bar{d}_1 & \bar{d}_2 & \bar{d}_3 \end{bmatrix}^T$ is bounded by a polytope. The input constraints are

$$a \in \left[-a_{\max}, a_{\max}\right],$$
$$r \in \left[-r_{\max}, r_{\max}\right], \tag{3.40}$$
$$a^2 + v_{\max}^{\,2} r^2 \le \left(\mu g\right)^2.$$

where μ is the tire-road friction coefficient, and g is the gravitational constant. A polygon is again used to approximate the circular constraint so that the set to constrain $\mathcal{U}$ is also approximated by a polytope. Denote the dynamic system in (3.38) by $\bar{\Sigma}$, and the set of disturbance is denoted by $\bar{\mathcal{D}}$.

Theorem 3.5: *If a set P is an avoidable set for the system $\bar{\Sigma}$, then it is an avoidable set for Σ_0 in (3.7).*

See Appendix E for proof.

The last term k in (3.38) provides an additional safety margin. When $\theta > 0$, the obstacle is on the right-hand side of the vehicle. To avoid a collision, θ must increase. Similarly, when $\theta < 0$, θ must decrease for safety. Both of these conditions are helped by k. Denote the dynamic system that ignores the last term k as $\tilde{\Sigma} : \dot{x} = E_1 u + G_1 \bar{d}$.

Theorem 3.6: *Suppose a polytope P is an avoidable set for $\tilde{\Sigma}$. Additionally, for any point x on the boundary of P where $x_4 = \theta > 0$, the bounding hyperplane $H^T x = \begin{bmatrix} H_1 & H_2 & H_3 & H_4 \end{bmatrix} x = 1$ at that point satisfies $H_4 \ge 0$, and for any point where $x_4 = \theta < 0$, $H_4 \le 0$ holds, then P is an avoidable set for $\bar{\Sigma}$.*

See Appendix F for proof.

With Theorem 3., the problem is simplified to finding an avoidable set for $\tilde{\Sigma}$, and check whether the condition in Theorem 3. holds.

Applying the procedure described in Section 3.4, the avoidable set P_B is obtained and shown in Figure 3.4. Because the dimension of the state space is four, P_B is projected to three 3-dimensional plots. The parameters used in generating the avoidable set can be found in Table H.1 in Appendix H.

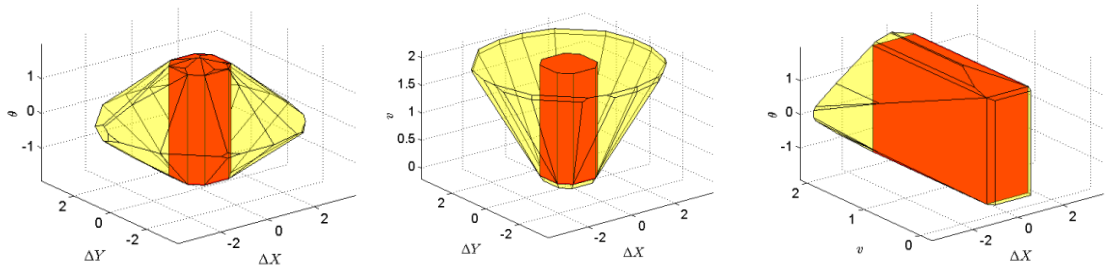

Figure 3.4 The avoidable set (yellow) and the infeasible set (red)

3.5.3 Supervisory control with control barrier functions

The concept of CBF is used to formulate the supervisory structure. First, a control barrier function is applied to a single half-space generated from a facet of the avoidable set, a Mixed Integer Program is used to deal with multiple facets.

Consider one of the half-space of the polytope of the avoidable set. If the state must stay outside of the half-space $\{x \mid H^T x \le 1\}$ to ensure safety, the control barrier function is defined as

$$b(x) = H^T x - 1, \;\; B(x) = -\log \frac{b(x)}{1+b(x)}. \tag{3.41}$$

It can be seen that $B(x)$ goes to infinity as x approaches the bounding hyperplane. When x is far from the boundary, $B(x)$ is positive but small. If the derivative of $B(x)$ is always finite, then $B(x)$ is finite, and x remains outside of the half-space. In particular, the following constraint is enforced by the supervisory controller:

$$\dot{B} \le \gamma / B(x), \tag{3.42}$$

where γ is a positive constant. This constraint is loose when x is far away from the boundary, and becomes tighter as x approaches the half-space. Consider the influence of sampling, the condition is modified to

$$H^T f(x,u,d) \geq -\frac{\gamma b}{B(x)+\gamma T_s}, \tag{3.43}$$

where T_s is the sampling time. This is the discrete version of (3.42) derived from Taylor expansion. See Appendix G for the derivation of (3.43).

3.5.4 Mixed integer program

For each bounding hyperplane H of P_B, if the state satisfies $H^T x \leq 1$, then H is said to be active, otherwise, it is said to be inactive. Since the avoidable set P_B is the intersection of all bounding half-spaces, the state x is outside P_B if and only if there exists at least one active bounding hyperplane, i.e.,

$$x \notin P_B \Leftrightarrow \exists H_0 \in P_B.H, {H_0}^T x > 1. \tag{3.44}$$

When $x(t)$ is outside the avoidable set, each active bounding hyperplane generates a linear constraint, in the form of (3.43), and at least one constraint must be satisfied. Since this is a logic disjunction, combining the supervisory control structure presented in Section 2.2, an MIP with slack variables is used to solve for u:

$$\begin{gathered} minimize \left\| u-u_0 \right\|_Q^2, s.t. \\ \forall H_j \in H_a, {H_j}^T f(x,u,d) \geq \frac{-\gamma b_{H_j}}{B_{H_j}(x)+\gamma T_s} - s_j \beta, \\ \sum_{j=1}^{N_P} s_j \leq N_P - 1, s_j \in \{0,1\}, \end{gathered} \tag{3.45}$$

where $\{s_j\}$ are slack variables, and β is a large positive number. When $s_j = 1$, the corresponding inequality is automatically satisfied; when $s_j = 0$, the original barrier inequality is enforced. H_a is the set of all active bounding hyperplanes and N_P is its cardinality. The condition $\sum_{j=1}^{N_P} s_j \leq N_P - 1$ ensures that at least one of the original barrier inequalities is satisfied. $\|\cdot\|_Q$ is the 2-norm induced by a positive definite matrix Q, $\|x\|_Q^2 = x^T Q x$, It is used to adjust the weight on each component of the input interference.

3.5.5 From single obstacle to multiple obstacles

When there are multiple moving obstacles, one can follow the same approach to calculate the avoidable set by expanding the state space to

$$x = \begin{bmatrix} \Delta X_1 & \Delta Y_1 & \theta_1 & \dots & \Delta X_n & \Delta Y_n & \theta_n & v \end{bmatrix}^T . \tag{3. 46}$$

Because of the limitation on computation complexity, and the number of pedestrians is not known in general, this naive approach is not scalable to a large number of pedestrians. A simpler alternative is used. The key innovation is that the AV is only allowed to brake when calculating the infeasible set, as mentioned in Section 3.5.1. Under this assumption, the infeasible set can be computed for one obstacle and then applied to multiple obstacles, since the emergency action to avoid all obstacles are the same. If steering is allowed, this will no longer be the case.

Recall the concept of responsibility in Section 3.2.2. Let X_{In} be the set outside of which the vehicle can come to a full stop before hitting the obstacle for all obstacle movement. In the multiple-obstacle case, as long as all obstacles are outside of X_{In} , the vehicle is always able to stop before hitting any one of the obstacles.

For each pedestrian, there is a set of states as described in (3.4), representing the relative dynamics between the pedestrian and the AV. Now the task is to keep multiple states outside of a single avoidable set.

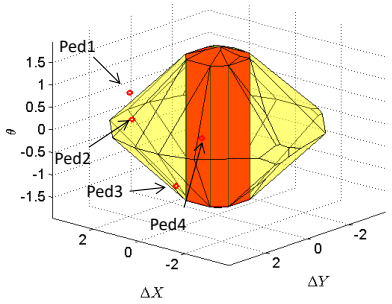

Figure 3.5 Multiple-pedestrian case with a single infeasible set and avoidable set

With multiple moving obstacles, the avoidable set is not always avoidable. Nevertheless, the supervisory control developed for a single pedestrian can still be used with the following modifications: (i) the constraint in (3.45) must be checked for each of the pedestrians; and (ii) when any obstacle breaches the "avoidable set," braking is applied until either the states for all pedestrians are outside the avoidable set, or the AV comes to a complete stop. Although braking is used to guarantee safety in the multiple pedestrian case, the simulation shows that the

supervisory control does not rely on heavily braking. The vehicle comes to a full stop only when it is trapped by multiple pedestrians.

3.6 Simulation results

3.6.1 Simulation setup and result

The goal of the AV is to reach a destination from a fixed starting point without colliding with any pedestrian. The initial positions and velocities of the pedestrians are random, and they walk randomly. All objects stay in a predefined rectangular region $[-X_{\lim}, X_{\lim}]\times[-Y_{\lim}, Y_{\lim}]$. The random walk is generated with Gaussian distributed acceleration in X and Y directions:

$$\begin{aligned} &\dot{v}_{px} = a_{px}, \qquad \dot{v}_{py} = a_{py}, \\ &a_{px} \sim N(0, \sigma_{ax}), \quad a_{py} \sim N(0, \sigma_{ay}). \end{aligned} \tag{3.47}$$

The velocity must also satisfy the boundedness constraint in (3.10). To keep the pedestrians inside the rectangular region, the following (reflection) rule is used

$$\begin{aligned} &v_{px} = -|v_{px}|, \; if \; X_p \geq X_{\lim}; \\ &v_{px} = |v_{px}|, \; if \; X_p \leq -X_{\lim}; \\ &v_{py} = -|v_{py}|, \; if \; Y_p \geq Y_{\lim}; \\ &v_{py} = |v_{py}|, \; if \; Y_p \leq -Y_{\lim}. \end{aligned} \tag{3.48}$$

A greedy Model Predictive Controller is used as the navigation controller. The detail of MPC is shown in Appendix I. The control structure is shown below:

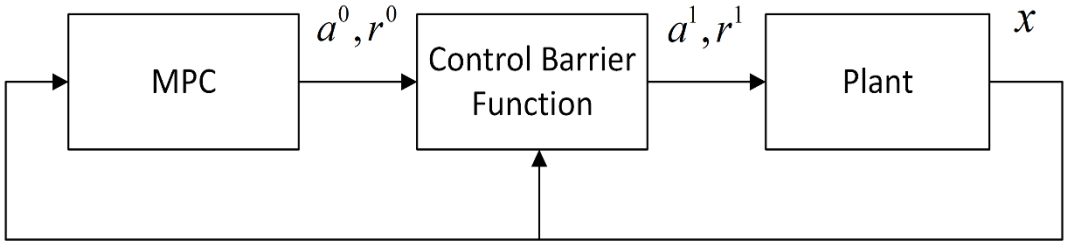

Figure 3.6 Control structure for simulation

The simulation parameters are listed in Table 3.1.

Table 3.1 Settings of the simulation runs

Number of pedestrians	7
The region of pedestrian movement	$[-5,5]\times[-5,5]$
Initial Position of the vehicle	(5,5)

Initial velocity of the vehicle	2m/s
Initial yaw angle	$\pi/2$
Destination	(-5,-5)

A sample simulation is illustrated in Figure 3.7. The blue circles are snapshots of the position of the vehicle; the green circles show the positions of the pedestrian and turn red when getting close to the vehicle, and the red square is the destination. One snapshot is taken every 0.5 seconds. The color of the snapshots changes from light to thick as time passes.

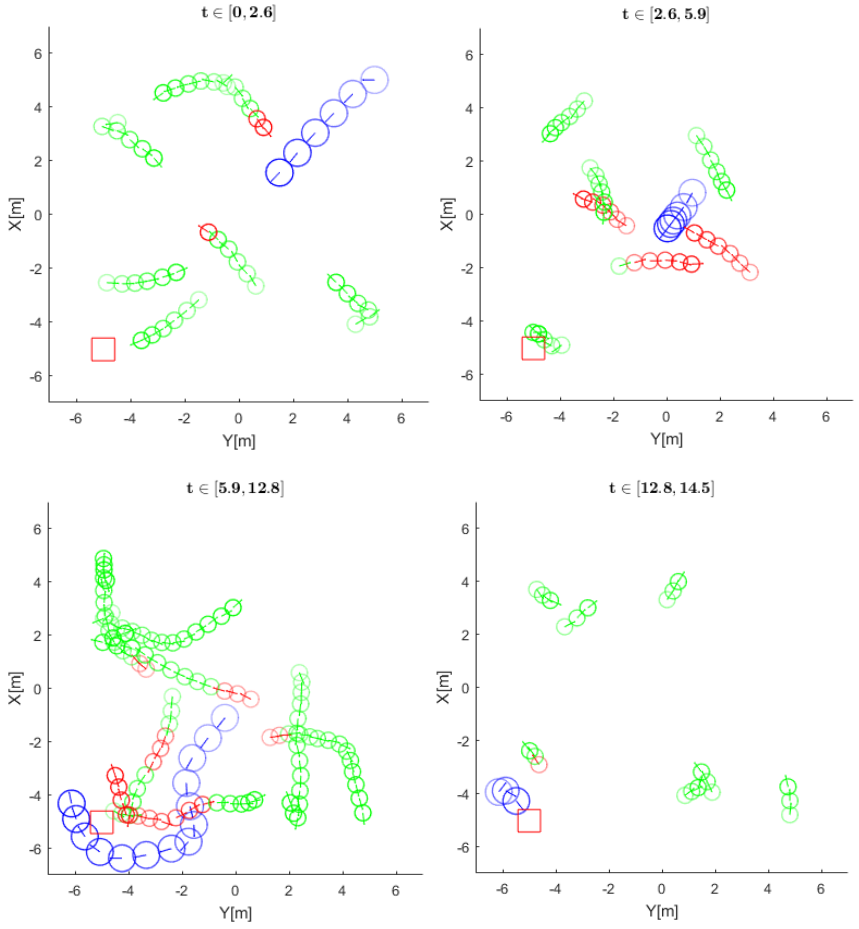

Figure 3.7 Sample simulation results

The control inputs from the navigation controller and the supervisory controller are shown in the first two subplots in Figure 3.8, where a^0 and r^0 are the acceleration and yaw rate command from the navigation controller, and are the command from the supervisory controller. $\mathbf{d}_{\max}$ denotes the distance from the state to the avoidable set, and was plotted in the third subplot. Different colors correspond to $\mathbf{d}_{\max}$ for different pedestrians.

$$\mathbf{d}_{\max} = \max_{H \in P.H} \frac{H^T x - 1}{\|H\|}. \tag{3.49}$$

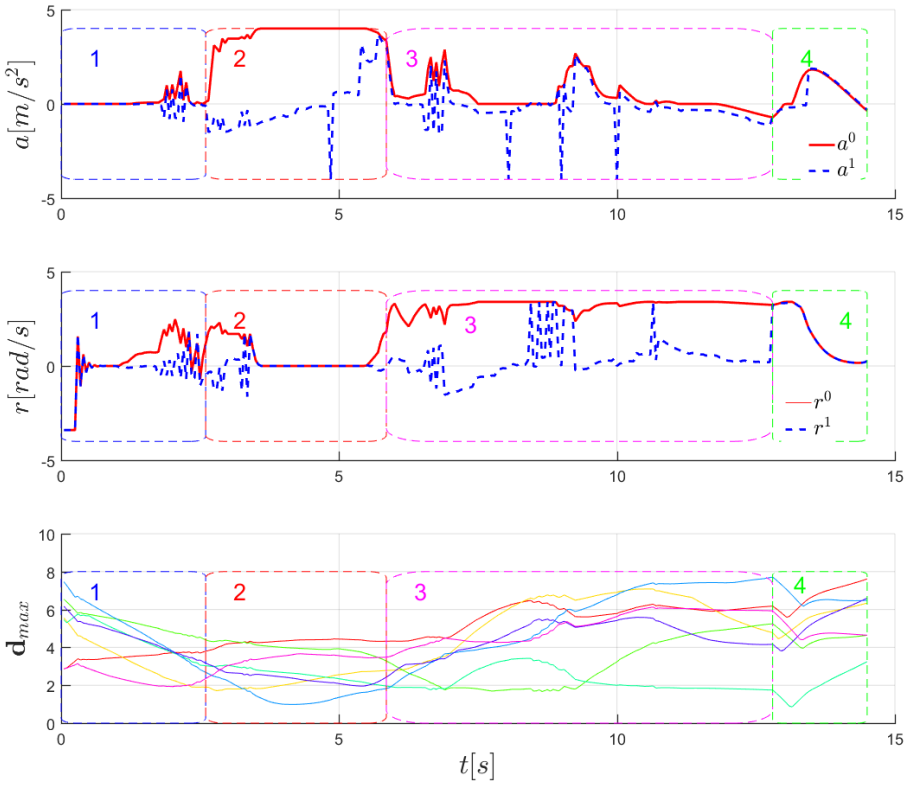

Figure 3.8 Control input and minimum distance to avoidable set

Based on the difference between MPC input and Barrier input, the simulation was divided into five stages. In stages 3 and 5, the supervisory controller detects little danger, so the two control inputs stay close; in stage 1 and 4, the supervisory controller manages to follow a similar acceleration command, but uses a different yaw rate to avoid a collision. This preference is because more weight is put on the acceleration difference than on the yaw rate difference when defining Q matrix in (3.45). In stage 2, both acceleration and steering are changed by the supervisory control to ensure safety. Compare the input plot and $\mathbf{d}_{\max}$ plot, the time when supervisory control changes the MPC input corresponds to the time when the smallest $\mathbf{d}_{\max}$ among seven pedestrians is small, which indicates danger.

3.6.2 Comparison to two benchmark methods

The performance of the polar algorithm is compared to two benchmark methods: the potential field method [66] and the Hamilton Jacobi method [33]. The details of both benchmark methods are included in Appendix J and Appendix K.

In order to compare the performance, the simulation is repeated 1000 times with the same setting shown in Table 3.1. In each trial, the vehicle starts at the same location and tries to reach the same destination, while the pedestrians appear at random positions and do random walking. A simulation trial is marked as "stuck" if the AV failed to reach the destination within 25 seconds, and a trial is marked as "crash" if (3.9) is satisfied at any time.

Table 3.2 Key performance indices of the three methods in 1000 simulation trials

Method	Average time	Collision	Stuck trips
Polar Method	10.88s	0	25
Hamilton Jacobi	14.93s	0	171
Potential Field	8.47s	436	0

The statistics of the 1000 trials are shown in Table 3.2. Both the polar method and the Hamilton Jacobi method can ensure safety, that is, no crash, while the potential field method crashes in about half of the trials. The Polar method reaches the destination in a much shorter time, and with fewer "stuck" cases compared with the Hamilton Jacobi method, indicating that the proposed method is as safe as, but much less conservative than, the Hamilton Jacobi method.

3.7 Conclusion and discussion

This chapter presents a polar algorithm to design collision avoidance algorithms for low-speed autonomous vehicles. The concept is based on the construction of an "avoidable set," which is an extension of the commonly used concept of a control invariant set. A Mixed Integer Programming based supervisory control structure is proposed to implement this algorithm. Safety can be guaranteed for both single moving obstacle and multiple moving obstacles while liveness is maintained. The safety guarantee was verified with simulations.

The polar method is suitable for models that are similar to integrators. Given the way disturbance is treated, the method maybe conservative when applied to a system with strong drift

dynamics. However, since many of the vehicle kinetic models are similar to integrators, the polar method is a powerful method for constructing supervisory controller for vehicle motion control.

In the simulation, the MIP runs faster than real-time with an avoidable set that has 70 facets. It is reasonable to claim that the real-time implementation is feasible for models of similar complexity. For higher dimensional models, one can make an over-approximation of the avoidable set by eliminating some vertices that are too close to others in P_H , which results in an over-approximation of P_B that is also avoidable.

Chapter 4 Supervised learning based design for safe controllers

As reviewed in Section 2.2 and demonstrated in Chapter 3, CBF is a powerful tool that can provide safety guarantee to existing controllers (namely, the student controller) in a supervisory structure. Typically, CBF works as a supervisor that intervenes when danger is detected. However, when intervention happens, the action may be harsh and performance may be compromised. Moreover, since the CBF condition is only enforced in the 0-level set of the CBF, the convergence from initial states outside the 0-level set of the CBF is not guaranteed. In this chapter, the focus is on the other side of the coin: the design of the student controller. A supervised learning based method is proposed to design a proper student controller that mitigates the influence of CBF and is capable of driving bad initial condition back to the safe region. A training set is generated by trajectory optimization for multiple initial conditions that incorporate the CBF constraint. Then a policy is trained via supervised learning that maps the feature representing the initial condition to a parameterized desired trajectory. Finally, the learning based controller is used as the student controller, and a CBF based supervisory controller on top of that guarantees safety. A case study of lane keeping for articulated trucks shows that the student controller trained by the supervised learning "inherits" the good properties of the training set and the CBF is rarely triggered.

4.1 Introduction and motivation

Control Barrier Function was developed to impose safety guarantee on control systems [30, 39]. The key idea is to compute a forward invariant set that contains the safe set and excludes the danger set. CBF is typically implemented in a supervisory control structure to guarantee safety with minimum interventions. As shown in Figure 2.1, u_0 denotes the control input from the student controller, which can be designed with any existing method, and u denotes the input signal after the intervention of the supervisory controller. If u_0 respects the safety constraint, $u = u_0$; otherwise

a minimum intervention is applied. Depending on the form of the barrier function, the implementation can be quadratic programming [30, 51], mixed integer programming [74] or in other forms.

While safety is assured independently of the choice of student controller, if the student controller is not properly designed, or is designed in a way that is not compatible with the CBF, the CBF may be triggered frequently, leading to undesirable closed-loop performance. In [54], when working with a student controller for Adaptive Cruise Control (ACC) that is not properly designed, the CBF causes spikes on the input when activated. In [74], when the student controller is designed without considering obstacle avoidance, the CBF has to intervene frequently and severely to ensure obstacle avoidance.

On the other hand, machine learning has been used extensively in dynamic control. Supervised learning has been used to learn a control policy with structure [75, 76], deep learning recently was used to generate end-to-end Lane Keeping (LK) policy, i.e., a mapping directly from the camera pixels to the steering input [77], and reinforcement learning can be used to generate a control policy in an "explore and evaluate" manner [78-80]. However, one major deficit of machine learning is its extreme difficulty for analysis. The number of parameters contained in a neural network can easily reach several thousand, even millions, which makes it practically impossible to analyze. Therefore, the safety of a learning-based controller should rely on other tools, such as reachable sets and barrier functions. In this sense, machine learning and CBFs complement one other.

Existing methods that combine learning with safety guarantee include reachable-set-based learning scheme that can guarantee safety for online learning of a control policy [81, 82] and Gaussian process learning [83]. Unlike approaches that aim at guaranteeing safety with learning, such as [81-83], the method proposed in this chapter separates safety from the performance. The safety guarantee is provided by a CBF, and supervised learning is used to improve the performance considering the influence of the CBF as a supervisor.

The proposed method starts by performing trajectory optimization offline, generating a library consisting of trajectories with good properties, such as stabilizing an equilibrium and attenuating disturbances, as well as the satisfaction of the CBF condition. Then supervised learning is used to train a student controller that inherits the properties of the trajectory library. In addition, CBF acts

as a supervisor on top of the learning-based controller, as shown in Figure 4.1. Since the CBF condition is enforced in the training set, an intervention by the supervisor is rarely triggered. It should be emphasized that the safety is still guaranteed by the CBF, the supervised learning only aims to improve performance.

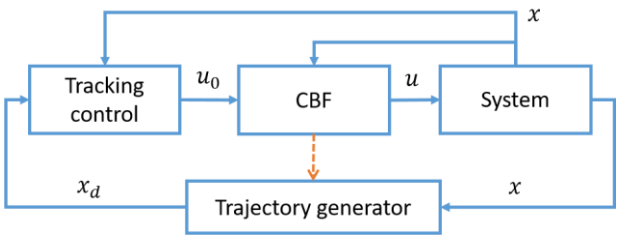

Figure 4.1 Block diagram of the supervisory control

The main contributions of this chapter are the following two points. First, a supervised learning based method is proposed to design a student controller that takes the CBF condition into account, which is applicable to a large region of initial conditions, and rarely triggers an intervention from the supervisory controller. With supervised learning, the design of a safe student controller is transformed into the design of safe trajectories, which is much easier, as conceptually shown in Figure 4.2.

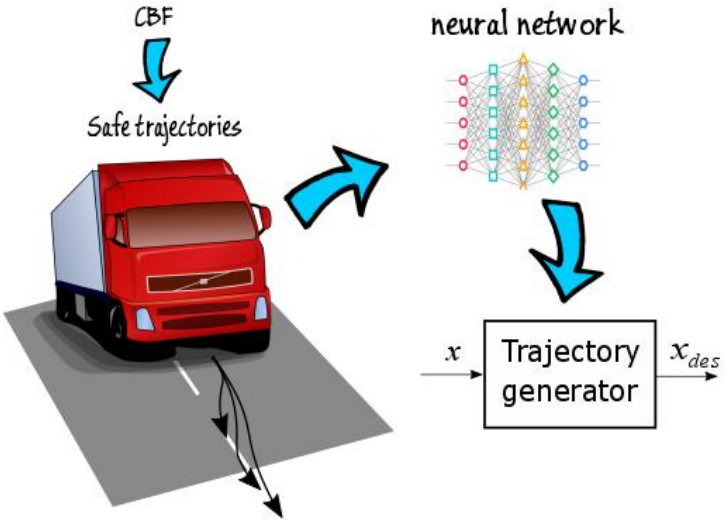

Figure 4.2 Learning based trajectory generator

Second, a stability and set invariance analysis is provided for the learning-based controller under the framework of continuous hold (CH) feedback control. Applying the proposed method, a safety guarantee is provided for the lane keeping control (LK) of an articulated truck, while achieving good ride comfort.

4.2 Dynamic model and virtual constraint

In this chapter, a control affine nonlinear model is considered:

$$\dot{x} = f(x) + g(x)u + g_{d1}(x)d_1 + g_{d2}(x)d_2, \tag{4.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}$, $d_1 \in \mathcal{D}_1 \subseteq \mathbb{R}^{l1}$, and $d_2 \in \mathcal{D}_2 \subseteq \mathbb{R}^{l2}$ represent the state, input, measured disturbance, and unmeasured disturbance, respectively.

Remark 4.1: *The unmeasured disturbance d_2 will be countered with the feedback control. Therefore, it is assumed that $d_2 = 0$ for the following analysis of feedback linearization.*

4.2.1 Model assumptions

The results in this chapter are developed under four key assumptions:

Assumption 4.1: *It is assumed that d_1 changes slowly comparing to the system dynamics. Therefore, d_1 is treated as constant in the following analysis.*

Assumption 4.2: *There exists an output $z = h(x)$ for x within an open subset $\mathcal{S} \in \mathbb{R}^n$, such that for all $d_1 \in \mathcal{D}_1$, z has a relative degree ρ, where the relative degree is defined as the integer such that $\forall x \in \mathcal{S}$,*

$$\begin{aligned} &\mathcal{L}_g \mathcal{L}_{\bar{f}}^{i-1} h(x) = 0, i = 1, 2, ..., \rho - 1, \\ &\mathcal{L}_g \mathcal{L}_{\bar{f}}^{\rho-1} h(x) \neq 0. \end{aligned} \tag{4.2}$$

Where

$$\bar{f}(x) = f(x) + g_{d1}(x)d_1, \tag{4.3}$$

Assumption 4.3: *It is assumed that when $d_2 = 0$, for all $d_1 \in \mathcal{D}_1$, there exists a unique $\eta_u(d_1) \in \mathcal{U}$ that maintains a unique equilibrium point $x_e \in \mathbb{R}^n$ with $h(x_e) = 0$, denoted as $x_e = \eta_x(d_1)$:*

$$\begin{aligned} &f(x_e) + g(x_e)\eta_u(d_1) + g_{d1}(x_e)d_1 = 0, \\ &h(x_e) = 0. \end{aligned} \tag{4.4}$$

Then from feedback linearization, there exists a state transformation:

$$\begin{bmatrix} \sigma \\ \xi \end{bmatrix} = \begin{bmatrix} T_1(x) \\ T_2(x) \end{bmatrix} = T(x), \sigma \in \mathbb{R}^{n-\rho}, \xi = \begin{bmatrix} h(x) \\ \vdots \\ \mathcal{L}_{\bar{f}}^{\rho-1} h(x) \end{bmatrix} = \begin{bmatrix} z \\ \vdots \\ z^{(\rho-1)} \end{bmatrix} \in \mathbb{R}^{\rho}, \tag{4.5}$$

where T is a bijective diffeomorphism over $\mathcal{S}$, and the transformation satisfies $\frac{\partial T_1}{\partial x} g(x) = 0$. Therefore, the dynamics of the "hidden" states σ is represented as

$$\dot{\sigma} = \Gamma(\sigma, \xi). \tag{4.6}$$

In particular, $\dot{\sigma} = \Gamma(\sigma, 0)$ is the zero dynamics of the system with output z, and there exists a smooth surface $\mathcal{Z} \subset \mathcal{S}$ defined by $\mathcal{Z} := \{x \in \mathcal{S} | \ \xi = 0\}$, which is the zero dynamics manifold.

Assumption 4.4: *It is assumed that the zero dynamics of the system under output z is exponentially stable within* $\mathcal{S}$.

Then by Theorem 11.2.3 in [84], the following feedback linearization controller constructed from z and its derivatives stabilizes the equilibrium x_e:

$$u = -\frac{1}{\mathcal{L}_g \mathcal{L}_f^{\rho-1} h(x)} \left[k_0 \xi_1 + \ldots + k_{\rho-1} \xi_\rho + \mathcal{L}_f^\rho h(x) \right], \tag{4.7}$$

where $\{k_i\}$ is a set of exponentially stabilizing gains in the sense that the following characteristic equation

$$\lambda^\rho + k_{\rho-1} \lambda^{\rho-1} + \ldots + k_0 = 0 \tag{4.8}$$

has all of its roots are in the open left half plane of the complex plane. See e.g. [85] for reference on feedback linearization and zero dynamics.

4.2.2 Virtual constraint and tracking control

To let the system track a desired trajectory of z, the virtual constraint method is used, which was originally developed in the robotics literature [86-88], and now appearing more widely. Suppose system need to track the following trajectory:

$$z = h_{des}(t), \tag{4.9}$$

where $h_{des} : [0, \infty) \to \mathbb{R}$ is a ρ times continuously differentiable function. Differentiate (4.9) $\rho - 1$ times and define the error states:

$$\begin{aligned} e_1 &= z - h_{des} \\ e_2 &= \dot{z} - \dot{h}_{des} \\ &\vdots \\ e_\rho &= z^{(\rho-1)} - h_{des}^{\rho-1}. \end{aligned} \tag{4.10}$$

Then select $\{k_i\}$ to be a set of stabilizing gains as described in (4.8), and let

$$u = -\frac{1}{\mathcal{L}_g \mathcal{L}_f^{\rho-1} h(x)} \left[k_0 e_1 + \ldots + k_{\rho-1} e_\rho + \mathcal{L}_f^\rho h(x) \right]. \tag{4.11}$$

When h_{des} is ρ times continuously differentiable and its derivatives are bounded, the feedback linearization control can locally track h_{des} imposed as a virtual constraint of z [89].

The benefit of the virtual constraint approach is that it gives a simple means of parameterizing the desired evolution of the vehicle. Instead of all the states, the desired trajectory is parameterized only by an output z satisfying Assumption 4.2 and Assumption 4.4. Later, trajectory optimization is used to determine the existence of a set of interesting trajectories that can be tracked by considering the full dynamics and the feedback structure.

4.2.3 Tractor-semitrailer models

In this work, two models are used: a design model and a validation model. For validation, a TruckSim model is used with its impressive 312 states. The literature contains a range of less detailed models that could be considered for control design, ranging from the nonlinear 37-state, physics-based model in [22], to linear models. To demonstrate the fundamental robustness of the proposed approach, the control design is based on a low-complexity model for an articulated truck adapted from [90] and [91], namely a 4 DOF linear model with 8 states:

$$x = \begin{bmatrix} y & v_y & \psi & r & \psi_a & r_s & \phi & p \end{bmatrix}^T, \tag{4.12}$$

where y is the lateral deviation from the lane center to the tractor Center of Gravity (CG), v_y is the lateral sideslip velocity of the tractor, ψ is the heading angle of the tractor, r is the yaw rate of the tractor, ψ_a is the articulation angle on the fifth wheel (the joint between the tractor and semitrailer), r_s is the yaw rate of the semitrailer, ϕ is the roll angle and p is the roll rate, as shown in Figure 4.3.

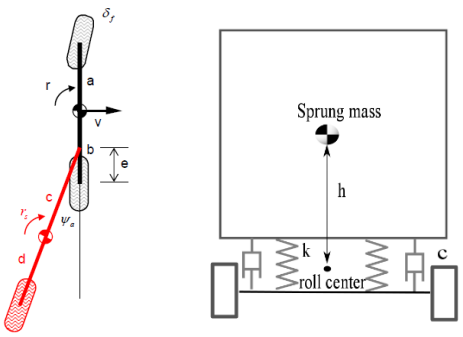

Figure 4.3 Lateral-yaw-roll model of articulated truck

The linear model is expressed in the form of (4.1) for consistency,

$$\begin{aligned}\dot{x} &= f(x) + g(x)\delta_f + g_{d1}(x)r_d + g_{d2}(x)F_y \\ &= Ax + B\delta_f + E_1 r_d + E_2 F_y .\end{aligned} \tag{4.13}$$

The input to the system is the steering angle δ_f of the tractor front axle and the disturbances are road curvature r_d and side wind F_y, where r_d is the measured disturbance, namely, d_1 in (4.1) and F_y is the unmeasured disturbance, namely, d_2 in (4.1).

A priori, the above linear model is only valid under the following assumptions:

- The longitudinal speed v_x of the truck has small variation;
- Due to the stiff connection on the roll dimension, the roll angle of the tractor and semitrailer are the same;
- The pitch and vertical motion are weakly coupled with the lateral, yaw and roll motion, and are ignored in the model;
- The angles are small, therefore the model can be approximated by a linear dynamic model.

The simulations performed later in TruckSim support that these assumptions are satisfied in a highway lane keeping scenario.

4.2.4 The virtual constraint for the truck model

The lateral displacement with preview is selected as the output for feedback linearization:

$$z = h(x) := y + T_0 v_x \psi , \tag{4.14}$$

with T_0 is the preview time, as shown in Figure 4.4.

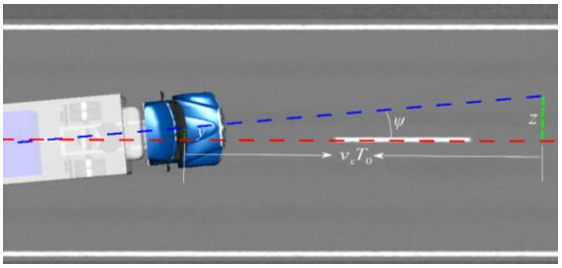

Figure 4.4 Preview of truck lateral dynamics

The output z so-defined has relative degree 2 for any r_d, i.e.,

$$\mathcal{L}_g h = 0, \mathcal{L}_g \mathcal{L}_{f+g_{d1} r_d} h \neq 0 . \tag{4.15}$$

To be more specific, the output dynamics is

$$\begin{aligned} z &= h(x), \\ \dot{z} &= \mathcal{L}_f h + \mathcal{L}_{g_{d1}} h \cdot r_d, \\ \ddot{z} &= \mathcal{L}_f^2 h + \mathcal{L}_g \mathcal{L}_f h \cdot u + \mathcal{L}_f \mathcal{L}_{g_{d1}} h \cdot r_d . \end{aligned} \tag{4.16}$$

By Assumption 4.1, r_d changes slowly compared to the dynamics, therefore, $\dot{r}_d$ is omitted. Since there are eight states but only z and $\dot{z}$ are used in the feedback linearization, six dimensions of the state space are hidden. It is shown that the zero dynamics of the system is exponentially stable, see Appendix L. Since $\rho = 2$, the feedback structure in (4.11) is essentially a PD controller:

$$u = -\frac{1}{\mathcal{L}_g \mathcal{L}_f h(x)} \begin{bmatrix} K_p (z - h_{des}) + K_d (\dot{z} - \dot{h}_{des}) + \\ \mathcal{L}_f^2 h(x) + \mathcal{L}_f \mathcal{L}_{g_{d1}} h(x) r_d \end{bmatrix}, \tag{4.17}$$

where K_p and K_d are the PD gains.

At this point, specifying the desired performance of the truck is simplified to designing h_{des}, the desired trajectory of the output z, which is discussed in Section 4.3.

Remark 4.2: *If smooth steering angles are desired, the control design model can be augmented with an integrator appended to u. In this case, the system has relative degree three and the control design is nearly the same.*

4.3 Trajectory optimization

A CBF is constructed following the procedure described in Section 2.2.3. Some of the key parameters for the truck lane keeping problem are listed in Table 4.1.

Table 4.1 List of parameters

v_x	$20m/s$
Bound on y	$\pm 0.3m$
Bound on ϕ	$\pm 0.1rad$
Bound on r_d	$\pm 0.02rad/s$ (turning radius of 1000m)
Bound on F_y	$\pm 2000N$
Bound on δ_f	$\pm 0.2rad$

Although CBF gives safety guarantee to the system, the performance could be compromised should the student controller be not properly designed. For example, a student controller designed with LQR is demonstrated in Figure 4.13, which causes severe intervention from the CBF and thus leads to bad ride comfort. In this section, the optimization procedure is presented that incorporates the CBF condition, which is then used to train a student controller that is compatible with the CBF. In addition to the CBF condition, other constraints are needed to ensure the stability of the continuous hold controller, as introduced later in Section 4.4.1.

4.3.1 Direct collocation for trajectory optimization

As discussed in Section 4.2, the trajectory optimization is boiled down to the optimization of h_{des}, the desired trajectory of the output z. Direct collocation is used to generate the trajectory of the states and h_{des}, while h_{des} is imposed as the virtual constraint.

Direct collocation is widely employed in trajectory optimization problems due to its effectiveness and robustness. It is thus chosen to optimize the trajectory while enforcing the virtual constraint. It works by replacing the explicit forward integration of the dynamical systems with a series of defect constraints via implicit Runge-Kutta methods, which provides better convergence and stability properties particularly for highly underactuated dynamical systems. The result is a nonlinear programming problem (NLP) [92].

In this problem, a modified Hermite-Simpson scheme based direct collocation trajectory optimization method is utilized [93]. Particularly, the flow (a.k.a. trajectory), $x(t)$, of the continuous dynamical system in (4.13) is approximated by discrete value x^i at uniformly distributed discrete time instant $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$ with $N > 0$ being the number of discrete intervals. Let x^i and $\dot{x}^i$ be the approximated states and first order derivatives at the node

i, they must satisfy the system dynamics equation given in (4.13). Further, if these discrete states satisfy the following defect constraints at all interior $i \in [1,3,\ldots,\mathrm{N}-1]$,

$$\begin{aligned} \zeta^i &:= \dot{x}^i - \frac{3N}{2T}\left(x^{i+1} - x^{i-1}\right) + \frac{1}{4}\left(\dot{x}^{i-1} + \dot{x}^{i+1}\right) = 0, \\ \delta^i &:= x^i - \frac{1}{2}\left(x^{i+1} + x^{i-1}\right) - \frac{T}{8N}\left(\dot{x}^{i-1} - \dot{x}^{i+1}\right) = 0. \end{aligned} \tag{4.18}$$

Then, they are accurate approximations of the given continuous dynamics. (4.18) defines the modified Hermite-Simpson conditions for the direct collocation trajectory optimization [93].

Based on the above formulation, now one can construct a constrained nonlinear programming problem to solve for the virtual constraint excited trajectory optimization of the articulated truck model. To incorporate the virtual constraints based feedback control with the trajectory optimization, the output dynamics equation given in (4.16) is enforced at each node. Then the control input u^i will be implicitly determined via this constraint without explicitly enforcing it as in (4.16). Further, the output z and its derivative $\dot{z}$ should equal to the desired trajectory h_{des} at $t = 0$ to ensure that the system lies on the zero dynamics manifold $\forall t \in [0,t]$.

The desired trajectory h_{des} is parameterized as Bezier curve, which is widely used in computer graphics and related fields. A Bezier curve of order *m* is defined on $[0,1]$ as

$$\mathbf{B}(s) = \sum_{i=0}^{m} \alpha_i \binom{m}{i} s^i (1-s)^{m-i}, \tag{4.19}$$

where α_i are the Bezier coefficients. It is a parameterized polynomial of order *m*. Compared to the parameterization with standard monomial bases($[1, s, s^2, \ldots]$), Bezier curve has the following nice properties:

- *The representation of initial and final values are simple:* $\mathbf{B}(0) = \alpha_0, \mathbf{B}(1) = \alpha_m$,
- *The derivative of Bezier curve is also a Bezier curve, and the dependence of the new Bezier coefficient on the original Bezier coefficient is sparse.*

Though the input for Bezier curve is between 0 and 1, Bezier curve can parameterize trajectories of any length. Suppose the horizon of h_d is T, then the input is defined as $s = \frac{t}{T}$.

Let $\mathcal{J}(\cdot)$ be the cost function to be minimized, the trajectory optimization problem can be stated as:

$$
\begin{aligned}
&\arg\min \ \mathcal{J}(\cdot) \quad s.t. \\
&\zeta^i = 0, \delta^i = 0, \\
&\dot{x}^i = f\left(x^i\right)x^i + g\left(x^i\right)u^i + g_{d1}\left(x^i\right)d_1, \\
&\ddot{z}^i - \ddot{h}_{des}\left(t_i\right) + K_p\left(z^i - h_{des}\left(t_i\right)\right) + K_d\left(\dot{z} - \dot{h}_{des}\left(t_i\right)\right) = 0, \\
&x\left(t_0\right) = x^0, \\
&z^0 - h_{des}\left(t_0\right) = 0, \\
&\dot{z}^0 - \dot{h}_{des}\left(t_0\right) = 0, \\
&-u_{\max} \le u^i \le u_{\max}, \\
&\dot{b}\left(x^i, \dot{x}^i\right) + \kappa\frac{e^{b(x^i)} - 1}{e^{b(x^i)} + 1} \ge 0, \\
&V\left(x(T) - \eta_x\left(d_1\right)\right) \le c_1 V\left(x^0 - \eta_x\left(d_1\right)\right), \\
&\left\|x_2(T) - \gamma(x)(T)\right\| \le c_3,
\end{aligned}
\tag{4.20}
$$

where $z^i = z\left(x^i\right)$, $\dot{z}^i = \dot{z}\left(x^i\right)$, and $\ddot{z}^i = \ddot{z}\left(x^i, \dot{x}^i\right)$, respectively. The first three lines of constraints correspond to the colocation constraint; 4th line specifies the initial states; 5th and 6th line correspond to the virtual constraint; the 7th line is the input constraint; the 8th line is the CBF constraint, the last two constraints are needed to guarantee stability of the continuous hold controller, which will be explained in Appendix M.

Remark 4.3: *The CBF condition is modified based on (2.7). Since* $\frac{e^b - 1}{e^b + 1}$ *is bounded within* $[-1,1]$, *when* $b(x)$ *is small, the lower bound for* $\dot{b}$ *saturates at 1, instead of growing linearly as* $-\gamma b$, *which may be too difficult to satisfy. Besides, when* $b(x) = 0$, $\frac{e^b - 1}{e^b + 1} = 0$, *which resembles the original barrier condition in (2.7). Since* $\frac{e^b - 1}{e^b + 1}$ *is still an extended class* $\mathcal{K}$ *function, by Proposition 1 in [50],* $\{x \mid b(x) \ge 0\}$ *is still invariant under the modified constraint.*

The cost function in (4.20) is a weighted sum of multiple cost functions, consisting of the following terms:

- *Final value cost* $V\left(x(T)-\eta_x(r_d)\right)$, *where V is the same Lyapunov function appeared in the constraint.*
- $\int z^2 dt$, *the square integral of* z;
- $\int \ddot{z}^2 dt$, *the square integral of jerk;*
- $\|y\|_\infty$, *the maximum deviation from road center;*
- $\|r\|_\infty$, *maximum yaw rate;*
- $\int u^2 dt$, *the square integral of the input;*
- $|\alpha_m|$, *penalty on the last Bezier coefficient (facilitating convergence of the Bezier curve).*

The terms that consist of function integrals are approximately computed using the Simpson's quadrature rule [94].

The setting of the constraints and costs seem complicated, they are the result of repeated trial and tuning. It should be emphasized that CBF constraint is enforced in the trajectory optimization. The motivation is that by enforcing CBF condition on the training set, the policy generated by supervised learning may inherit this property.

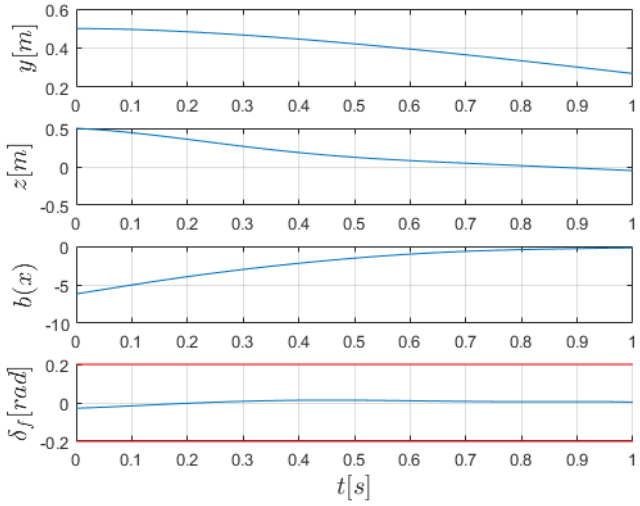

Figure 4.5 Example of trajectory optimization result

Figure 4.5 shows an example trajectory with initial lateral deviation $y_0 = 0.5m$ and road yaw rate $r_d = 0.02rad/s$. The plot of y and the Bezier output z shows that the trajectory is converging to the lane center. The plot of the CBF value and the control input shows that the trajectory generated by direct collocation satisfies the input and CBF constraints.

The trajectory optimization is solved with FROST, which uses a symbolic calculation to boost the nonlinear optimization [95]. The trajectory optimization for each initial condition can be finished within ten seconds.

4.3.2 Generating the training set

It is impossible to perform trajectory optimization for all the initial conditions offline, so instead, supervised learning is used to train the mapping from initial conditions to desired trajectories with a finite trajectory library, which is generated by the above-described trajectory optimization process.

By varying the initial conditions and generating the corresponding trajectories with direct collocation, the neural network trained with the training set is expected to generate good trajectories for various initial conditions. The inputs to the neural network are called features, denoted as Φ; in the training of the trajaectory generator, they are variables that describe the initial condition. The output of the neural network is a vector of control parameters, denoted as $\mathcal{A}$, in this case, the Bezier coefficients.

$$\pi : \Phi \to \mathcal{A} . \tag{4.21}$$

The selection of initial conditions is done in a grid fashion. A grid on the feature space is defined and trajectory optimization is performed on each of the grid points. Since the zero dynamics is stable, $z(t) \to h_{des}(t)$ for $h_{des} \in C^2$ implies $x(t) \to x_{des}(t)$, where x_{des} is the desired state trajectory corresponding to h_{des}. This implies that only two states are needed to determine the asymptotic behavior of the system, but not necessarily the transient behavior. In practice, the more states are used to parameterize the initial condition, the finer the trajectory library will be.

However, under a grid fashion of drawing samples, the number of samples needed grows exponentially with the state dimension. Therefore, the dimension of Φ is limited by available computation power. Φ is chosen to consist of six features, including five states and r_d:

$$\Phi = \left[y, \psi, r, \psi_a, v_y, r_d \right]. \tag{4.22}$$

Under this setup, the computation needed to generate the trajectory library is manageable (about 20 hours on a desktop). With more computation power, a higher dimensional Φ can lead to a finer trajectory library.

Even though most driving behavior is mild, it is important that the controller be able to handle bad initial conditions. Two training sets are therefore generated, denoted as S_1 and S_2, where S_1 consists of trajectories defined for a duration of 1 second, and the features of the trajectories have a wider span and S_2 consists of trajectories defined over a 3-second window, with the features more concentrated around the origin. S_1 is used to train a mapping for severe initial conditions and transients, and S_2 is used to train a mapping for mild situations and normal driving. Some of the initial conditions might render the trajectory optimization infeasible, therefore only the feasible cases are included in the training sets. In the implementation, the CH controller will choose which mapping to use based on the severity of the situation.

The parameters for the training are presented in Table 4.2.

Table 4.2 Training set parameter setting

Feature	S_1	S_2
y range	$[-0.5, 0.5]$[m]	$[-0.3, 0.3]$[m]
ψ range	$[-0.04, 0.04][rad]$	$[-0.04, 0.04][rad]$
r range	$[-0.06, 0.06][rad/s]$	$[-0.03, 0.03][rad/s]$
r_d range	$[-0.03, 0.03][rad/s]$	$[-0.025, 0.025][rad/s]$
ψ_a range	$[-0.04, 0.04][rad]$	$[-0.04, 0.04][rad]$

In total, there are 62825 trajectories in S_1, and 29300 trajectories in S_2.

4.3.3 Supervised learning

With the training set ready, there are several choices for the supervised learning, such as linear regression, Gaussian process regression, and neural networks. In the truck problem, since there is no structural information about the trajectory generator and strong expressive power is needed to capture the potentially complicated mapping from the initial condition to the desired trajectory, neural network is chosen for its strong expressive power.

A neural network that has six hidden layers with 200 neurons in each layer and uses the ReLU function as the rectifier is trained. The training is performed using Tensorflow [96]. 85% of the data is used for training and 15% is used for testing. Table 4.3 shows the mean squared error (MSE) of the training result.

Table 4.3 Training result

	S_1	S_2
MSE of training data	0.13	0.0023
MSE of testing data	0.16	0.0024

4.4 Implementation of learning based controller

4.4.1 Continuous hold feedback control

Once the trajectory generator is trained, it can generate a finite horizon desired trajectory for a given initial condition. In order to piece together the finite horizon trajectories and synthesize a controller from the trajectory generator, a continuous hold (CH) controller is employed. The name continuous hold comes from the analogy with a zero-order hold and an *n-th* order hold. While an *n-th* order hold approximates the segment between two consecutive sampling times with an *n-th* order polynomial, continuous hold executes a predefined continuous trajectory.

The idea of continuous hold is not claimed to be novel; a motion primitive is a special type of continuous hold [63]. The trajectory is updated in an event-triggered fashion, which will be discussed in detail in Section 4.4.2. While event-triggered finite-horizon control is studied in [97], in the CH setting, it should be noted that the control action between triggering events is a continuous function of time and states instead of being a constant.

For the truck example, the basic continuous hold controller [98] must be extended to systems with exogenous disturbance. The stability and set invariance property of the CH controller are proved, including the analysis for the case when only a subset of the state is used for feedback, in Appendix M.

4.4.2 Event-triggered update of the CH controller

The CH controller uses the mapping trained by supervised learning to generate a desired trajectory h_{des} for the output z based on the current state and r_d, then track the desired trajectory with the control law in (4.17). The desired trajectory will be updated under three circumstances:

- *The Bezier curve is executed to the end;*
- *A significant change in* r_d *;*
- *The trajectory tracking error is too large.*

In the first case, since the trajectory optimization has a limited horizon (1s or 3s), the neural network will use the current value of the features to generate a new desired trajectory. In the second case, if r_d differs much from the r_d used to generate the current desired trajectory, the trajectory should be updated since r_d is assumed to be constant during the entire horizon of the trajectory. The rest of the features are simply initial conditions, so their change does not trigger an update of the desired trajectory. In the third case, when the trajectory deviates too far from the desired trajectory, re-planning is called for. This is likely to be caused by an unexpected disturbance, such as wind gust.

When switching from one trajectory to the next, smoothing is performed to make sure that h_{des} is twice differentiable, which ensures that the control signal is continuous. The smoothing process is explained in the Appendix N.

4.4.3 CBF as a supervisory controller

Even though the CBF condition is enforced in the trajectory optimization for the training set, after supervised learning, there is no guarantee that the trajectory generated by the neural network will always satisfy the CBF condition. Therefore, CBF is still implemented as a supervisory controller on top of the learning based controller. The CBF solves the following optimization:

$$\begin{aligned} &\min\ w_1 \|\Delta u\|^2 + w_2 \|\Delta u - \Delta u_{old}\|^2 \ s.t. \\ &\mathcal{C}(u_0 + \Delta u) \geq 0,\ u_0 + \Delta u \in \mathcal{U}, \end{aligned} \tag{4.23}$$

where Δu is the intervention of the CBF, Δu_{old} is the intervention of the previous time instant, $\mathcal{C}(\cdot)$ is the barrier condition and w_1, w_2 are the weights. The reason for the second penalty term is to prevent chattering if intervention is necessary.

The barrier condition is defined as

$$\begin{cases} \dot{b} + \gamma b \geq 0, & if\ b(x) \geq 0 \\ \dot{b} + \gamma \dfrac{e^b - 1}{e^b + 1} \geq 0, & if\ b(x) \leq 0 \end{cases}, \tag{4.24}$$

where the transition at $b(x) = 0$ is continuous, i.e. the two constraint coincides at $b(x) = 0$, as shown in Figure 4.6.

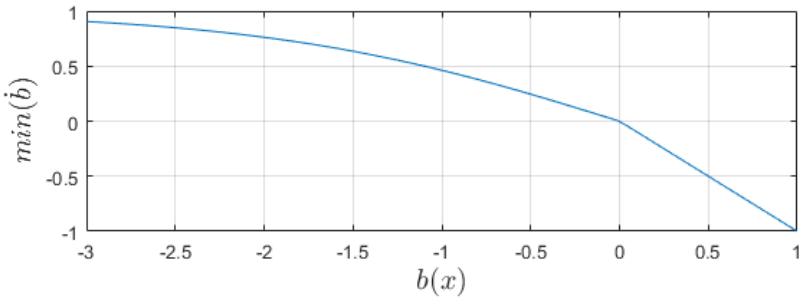

Figure 4.6 Lower bound for $\dot{b}$

Remark 4.4: *When* $b(x) \geq 0$, *the existence of* Δu *is guaranteed by the construction of the CBF; when* $b(x) \leq 0$, *there is no guarantee of feasibility. When (4.23) is infeasible, the input is saturated by* $\mathcal{U}$.

4.5 Simulation result

The proposed control design is validated on TruckSim, a high fidelity physics based simulation software that is widely acknowledged by the industry. The model selected for simulation has 312 states and is a tractor-semitrailer with heavy cargo in the trailer, weighing 35 tons in total, as shown in Figure 4.7.

Figure 4.7 Animation with a 312 state model in TruckSim

The truck is asked to drive on a road with minimum turning radius 1000m at 20m/s, and sidewind is simulated with lateral force and roll moment to the truck. Because of the cargo, the truck has high CG, the roll motion in the simulation is significant and the maneuver is aggressive.

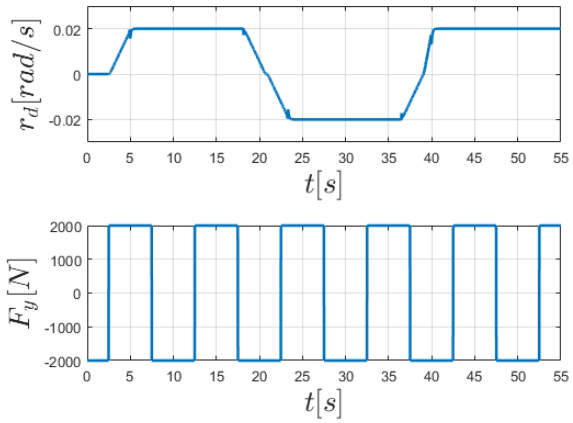

Figure 4.8 Disturbance to the system in simulation

As shown in Figure 4.8, the road profile consists of segments with different curvatures, and the sidewind is a square wave with maximum allowed magnitude.

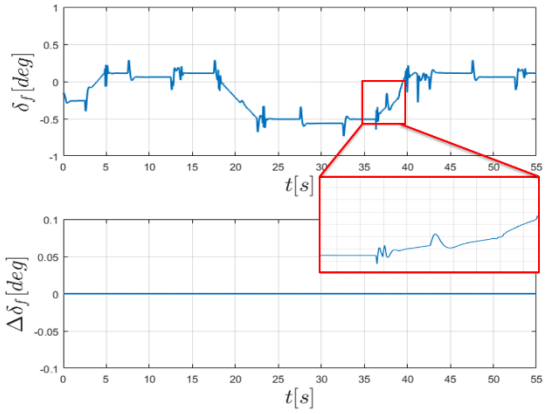

Figure 4.9 Input and intervention of CBF during simulation

The steering input trajectory is shown in Figure 4.9. A zoom-in view is presented to show a five seconds period of input. The input is actually quite smooth. The little bumps are necessary to counter the sidewind when it changes direction. The lower plot shows $\Delta\delta_f$, which indicates that no intervention from CBF happened.

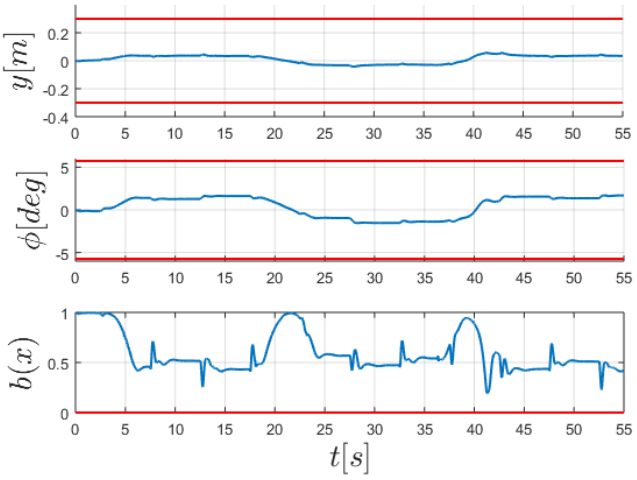

Figure 4.10 Value of CBF and key states during simulation

Figure 4.10 shows the value of CBF and two key states. Lateral deviation y and roll angle ϕ never exceed the limit (plotted in red) and $b(x)$ is always positive, showing that the CBF bound was never breached. This is under the CH controller without any help from the CBF.

To demonstrate the controller's ability to handle bad initial conditions, the lateral deviation was perturbed with a square wave, simulating the situation when the initial position is 0.5m from the lane center, as shown in Figure 4.11 and Figure 4.12.

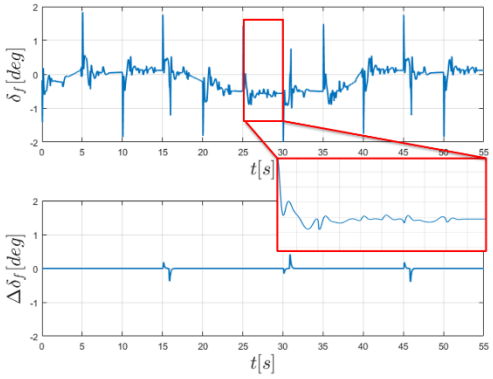

Figure 4.11 Input and intervention of CBF with large initial deviation

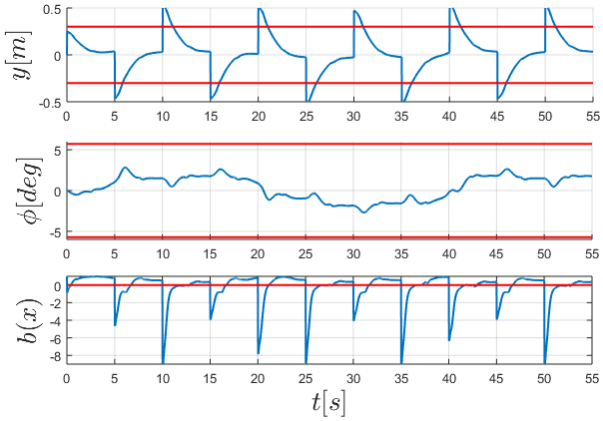

Figure 4.12 Value of CBF and key states with large initial deviation

Figure 4.11 shows the input under large deviation. The CBF intervention occurred three times, but they are very mild compared to the size of u_0. The learning based controller was able to handle the bad initial conditions most of the time. When $b(x)$ was below zero, the learned controller was able to drive the system back to the safe set without the intervention of CBF.

As a comparison, an LQR controller is tuned with feedforward control of r_d, and it performs very well under normal driving conditions. However, when the initial condition is bad (under the same setting as the previous simulation), the LQR controller triggered intervention from the CBF multiple times (11 times) and the jerk was severe, as shown in Figure 4.13.

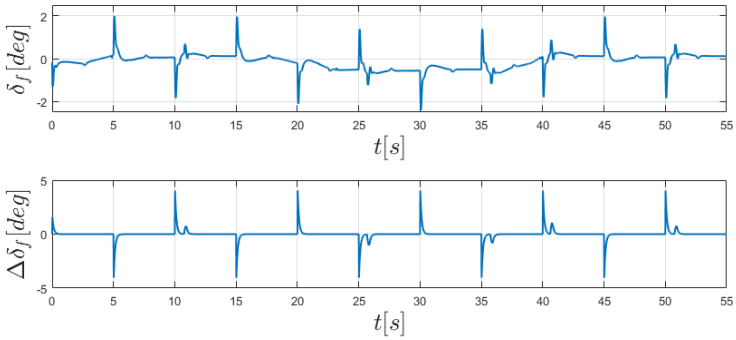

Figure 4.13 Simulation result with LQR as student controller

Though the LQR controller was fine-tuned, it triggered severe intervention from the CBF frequently. On the other hand, none or very mild interventions was triggered from the CBF under the learning based controller in all trial simulations when the states are within the training set span. When the states are outside the span of the training set, the neural network is asked to extrapolate instead of interpolating, therefore the performance is not guaranteed.

4.6 Conclusion and discussion

The focus of this chapter is on the synthesis of the student controller. When a CBF is synthesized as a supervisor, it poses additional requirement to the student controller, that is, the student controller should be compatible with the CBF supervisor and not causing frequent intervention from the CBF. A supervised learning based method is proposed to enforce this requirement. The idea is to use trajectory optimization technique to generate a training set consisting of trajectories that satisfy the CBF constraint, supervised learning is then used to learn the mapping from the initial condition to a parameterized desired trajectory. The policy generated with supervised learning will inherit the good properties of the training set, though not rigorously. On top of that, safety guarantee is formally imposed with CBF as the supervisory controller. The simulation shows that the proposed approach is able to reduce the intervention of the CBF and therefore improve the performance while guaranteeing safety.

The supervised learning is set up to learn the mapping from an initial condition to the desired output trajectory, instead of the mapping from the initial condition to the desired input trajectory. The stability is guaranteed under the CH framework, whereas there is no guarantee that following the desired input trajectory will lead to desired performance. The proposed CH control structure is able to transform the synthesis problem into a trajectory optimization problem, which may be much simpler for complicated nonlinear systems such as robots [98].

There are problems to be solved for the proposed method. First, when the initial condition is not contained inside the feature range of the training set, i.e. when the neural network is doing extrapolation rather than interpolation, the performance is poor. In other words, to get good performance in a wide range, one needs to have training data with enough coverage. Second, when training data from a large range of features are stacked together for the training, the regression accuracy drops and the performance is bad. To solve this, one might need more complicated neural network structure, or use multiple neural networks for different situations.

Chapter 5 Lyapunov approach for validation of non-cooperative control designs

In this chapter, a decentralized procedure to verify the closed-loop performance of a system with multiple controllers designed by non-cooperative agents is presented. The main problem to address is the disturbance coming from coexisting controllers on the same dynamic system. The goal is to verify whether performance specifications are met when multiple controllers that have been designed separately are applied to the same system. The key concept is to use Lyapunov functions as measures of the influence of each controller on the system. Sum of Squares programming is used to verify Lyapunov derivative conditions, thus providing a sufficient condition for the specification. Dual decomposition is then used to guarantee that the decentralized verification is equivalent to the centralized verification. Moreover, the verification procedure is extended to decentralized controller synthesis to satisfy safety specifications.

5.1 Introduction and motivation

As modern vehicles become more automated, they contain an increasing number of controllers, with more to be added. NHTSA mandated that Electronic Stability Control (ESC) be installed on all light vehicles manufactured after 2011. Adaptive Cruise Control (ACC) and Lane Keeping (LK) are available as convenience features on both sedans and trucks. Moreover, systems such as active suspension and rollover prevention are installed to enhance the riding comfort and safety of trucks and buses. A crucial issue is whether these features will cause problems to one another when put together, since the controllers are all coupled through the vehicle's dynamics. The concept of integrated chassis control has been proposed to take into account all actuators that influence vehicle chassis dynamics, and it aims at better performance, including safety, comfort, and drivability. The design was first carried out in a centralized manner, assuming one agent is in charge of designing all the controllers involved [99-102]. The benefit of centralization is obvious: the designer can fully consider the coupling in the dynamics and design controllers accordingly. However, the centralized approach is not always feasible in real engineering situations. The Original Equipment

Manufacturer (OEM) usually cooperates with multiple suppliers that specialize in the different areas of chassis control, and the controllers are designed by non-cooperative agents (suppliers) who keep their control algorithms confidential.

There are several approaches to address the coupling and potential conflicts between different chassis controllers. One is supervisory control, which treats the local controllers as servo-loop controllers and relies on a supervisor to give set points to the local controllers [103]. The coordinator approach uses worst-case analysis and "tunes down" the control signal when it detects that failure might occur because of the coupling [104]. However, as stated in [105], without global structure, a decentralized control structure (at least the traditional ones) is not suitable for safety-critical tasks. In addition to the design of the controllers, verification is another important issue. Ground testing can be performed by the OEMs, but it is notorious for its high cost of money and time because of the curse of dimensionality. Despite great efforts to expose the system to various scenarios in both the ground test approach and simulation approach, verification is never complete, meaning that there are always scenarios that are omitted during experimentation. In this chapter, a novel Lyapunov approach is proposed that provides safety guarantees to a decentralized design of a chassis control system. The proposed method does not suffer from the curse of dimensionality, and it provides verification of the closed-loop system. In addition, it satisfies the requirement of decentralized design by independent suppliers by keeping the control algorithms confidential.

The proposed approach requires that all control laws be explicit analytic functions of the states and the measured disturbance. This requirement might limit the applicability of the proposed approach, but it already covers a large portion of the existing control techniques, and in future work, the proposed method may be extended to implicit control algorithms with methods such as system identification.

5.2 Problem formulation and major tools

5.2.1 Problem formulation

In this section, the case where two controllers are acting on one system is considered. The system dynamics is assumed to have the form

$$\dot{x} = f(x) + g_1(x)u_1(x,d) + g_2(x)u_2(x,d) + g_d(x)d \tag{5.1}$$

where $x \in \mathbb{R}^n$ is the state, $u_1 \in \mathbb{R}^{m_1}, u_2 \in \mathbb{R}^{m_2}$ are inputs from two coexisting controllers, $d \in \mathbb{R}^l$ is the exogenous disturbance; $f(x)$ is the intrinsic (or drift) dynamics, $g_1(x)$ is the input dynamics for u_1, $g_2(x)$ is the input dynamics for u_2, and $g_d(x)$ is the exogenous disturbance dynamics, for both measured and unmeasured disturbance. The control inputs u_1 and u_2 are assumed to be functions of x and d, but the disturbance feedforward is limited to measured disturbance.

It is assumed that the system dynamics can be described by a polynomial (can be extended to rational functions by the simple transformation of multiplying through by the denominator). For non-polynomial systems, Taylor expansion can be used to obtain a polynomial approximation, and then the proposed method can apply.

Now recall the Lyapunov theory. For a continuous dynamic system described by an ordinary differential equation:

$$\dot{x} = f(x). \tag{5.2}$$

Consider a continuously differentiable Lyapunov function candidate $V : \mathbb{R}^n \to \mathbb{R}$ that satisfies

$$V(0) = 0, \forall x \neq 0, V(x) > 0. \tag{5.3}$$

If it can be shown that

$$\begin{aligned} &\forall x \in \mathbb{R}^n, \dot{V}(x) = \frac{dV}{dx} f(x) \leq 0, \\ &\forall x \in \mathbb{R}^n, x \neq 0, \dot{V}(x) = \frac{dV}{dx} f(x) < 0. \end{aligned} \tag{5.4}$$

Then, the origin is asymptotically stable; if it can be shown that

$$\forall x \in \{x | V(x) \geq 1\}, \dot{V}(x) \leq 0, \tag{5.5}$$

the set $\{x | V(x) \leq 1\}$ is invariant. The former is considered a special case of the latter; therefore, in the remainder of the chapter, only the set invariance verification will be discussed. Note that the Lyapunov function can always be scaled to adjust for the size of the 1-level set; thus, discussing the 1-level set is sufficient. Given a Lyapunov function candidate V and the dynamic system in (5.1), the Lyapunov derivative is

$$\dot{V}(x,d) = \frac{dV}{dx}\left(f(x) + g_1(x)u_1(x,d) + g_2(x)u_2(x,d) + g_d(x)d \right). \tag{5.6}$$

In real-world control problems, the exogenous disturbance is usually bounded; It is assumed that the bound is known and is represented as a semialgebraic set $\mathcal{D}$. The standard verification

setup considered in this chapter is that, given the system dynamics, with the control inputs as functions of the state and measured disturbance, and the Lyapunov candidate, whether the following condition holds:

$$\forall x \in \{x \mid V(x) \le 1\}, \forall d \in \mathcal{D}, \dot{V}(x) \le 0. \tag{5.7}$$

Above is a sufficient condition for the set $\{x \mid V(x) \le 1\}$ to be invariant under all disturbance in $\mathcal{D}$.

5.3 Verification using Lyapunov functions

In this section, a verification procedure using Lyapunov functions is introduced. Since the procedure checks the positive definiteness of polynomials, some basic techniques of SOS are introduced.

5.3.1 SOS verification for polynomial dynamic systems

SOS has been used for verification of dynamic systems in a variety of ways. For a given control law, SOS is used to find a Lyapunov function to prove stability [38, 106], calculate the approximate region of attraction [107], and guarantee that the states never enter an unsafe set [41, 49]. SOS is also used to directly synthesize controllers [40]. A review of SOS was given in Section 2.1. The Lyapunov verification problem can be formulated as the problem of verifying positive definiteness of a polynomial inside a semialgebraic set, of which a sufficient condition can be given using the SOS multipliers. The Lyapunov derivative condition is enforced in the following semialgebraic set:

$$\{x, d \mid h_i(x, d) \ge 0\} = \{x \mid V(x) \ge 1\} \times \mathcal{D}, \tag{5.8}$$

and the condition to be verified becomes

$$-\dot{V}(x, d) - \sum_i a_i(x, d) h_i(x, d) \in \Sigma[x, d], \tag{5.9}$$

where $\{a_i\}$ is the set of SOS multipliers with arguments x and d. This direct SOS approach is referred to as the **centralized verification**. The centralized verification requires that there be an agent who knows the control algorithms of both controllers.

5.3.2 Decomposition of Lyapunov derivative

As stated in Section 5.1, in a decentralized design process, since the control laws are kept confidential by the agents, centralized verification is not applicable.

In this situation, the Lyapunov derivative is decomposed into two parts, one part containing u_1 only, and the other part containing u_2 only. In this way, the two agents are able to find polynomial upper bounds for the two parts, respectively. A sample decomposition of the Lyapunov derivative is shown below:

$$\dot{V} = \left[\frac{dV}{dx} \left(f(x) + g_1(x) u_1(x,d) + g_d(x) d \right) \right] + \left[\frac{dV}{dx} g_2(x) u_2(x,d) \right]. \tag{5.10}$$

Agents 1 and 2 can find upper bounds for the two parts, α_1 and α_2, respectively.

$$\begin{aligned} &\textit{Agent } 1: \forall x \in \{x | V(x) \geq 1\}, \forall d \in \mathcal{D}, \\ &\frac{dV}{dx} \left(f(x) + g_1(x) u_1(x,d) + g_d(x) d \right) \leq \alpha_1(x,d), \\ &\textit{Agent } 2: \forall x \in \{x | V(x) \geq 1\}, \forall d \in \mathcal{D}, \\ &\frac{dV}{dx} g_2(x) u_2(x,d) \leq \alpha_2(x,d). \end{aligned} \tag{5.11}$$

Remark 5.1: *There is freedom in decomposing the Lyapunov derivative.* $g_1(x)u_1$ *and* $g_2(x)u_2$ *should always be put into two parts. In addition to this rule, a good decomposition may simplify the computation. For example, if* u_2 *does not depend on* d, α_2 *only depends on* x. *See Appendix O for more detail.*

Since the first part of the Lyapunov derivative is completely known to agent 1, and the second part is completely known to agent 2, the upper bound can be found using SOS programming. If

$$\alpha_1 + \alpha_2 \leq 0, \tag{5.12}$$

the verification is successful. The above-mentioned condition is sufficient but not necessary. Next, it is shown that the decentralized verification is equivalent to the centralized verification, that is, if a system can be verified by centralized verification, the decentralized verification returns the same result.

First, the existence of such α_1 and α_2 is discussed. If the system satisfies the set invariance condition to be verified, that is,

$$\forall x \in \{x | V(x) \geq 1\}, \forall d \in \mathcal{D}, \dot{V}(x,d) \leq 0, \tag{5.13}$$

then the decentralized verification aims to find α_1 and α_2 that satisfy (5.11) and (5.12).

Theorem 5.1 (Equivalency of centralized verification and decentralized verification):

Suppose $\mathcal{D}$ is a semialgebraic set, V is a polynomial candidate Lyapunov function in x, then

- *If (5.13) is satisfied, there exist α_1 and α_2 that satisfy (5.11) and (5.12).*
- *If (5.13) can be verified with SOS, there exist α_1 and α_2 such that (5.11) and (5.12) can be verified with SOS.*

Furthermore, with some additional conditions, Theorem 5.1 can be extended to cases where α_1 and α_2 are functions of only x. The proof is given in Appendix O.

With existence guaranteed, the next important question is how to find α_1 and α_2. The dual decomposition technique is used to solve this coupled problem.

5.4 Dual decomposition for verification

In this section, the dual decomposition procedure for decentralized verification is presented.

5.4.1 Dual decomposition for Lyapunov verification

Dual decomposition has been widely used in convex optimization recently [108, 109]. The main benefit is its ability to decompose a problem into smaller ones. The procedure in [109] is modified to suit the purpose of decentralized verification. The standard setup is as follows:

$$\begin{aligned} &\min_{x,z} f_1(x) + f_2(z) \; s.t. \\ &Ax + Bz - c = 0, \end{aligned} \tag{5.14}$$

where f_1 and f_2 are convex objective functions. The variables x and z are coupled by an equality constraint. The Lagrangian is written as

$$L(x, y, z) = f_1(x) + f_2(z) + y^T (Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|^2, \tag{5.15}$$

where y is the dual variable, and the last term is a quadratic penalty term. Then the dual ascent method is used to search for the optimum:

$$\begin{aligned} x^{k+1} &= \arg\min_x L(x, y^k, z^k), \\ z^{k+1} &= \arg\min_z L(x^{k+1}, y^k, z), \\ y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \end{aligned} \tag{5.16}$$

The algorithm known as alternating direction method of multipliers (ADMM) iterates until no improvement can be made.

Remark 5.2: *Additional constraint is allowed for x and z, but there should be no coupling, i.e., the update of x should be an optimization of only x and the same is required for the update of z.*

The problem of decentralized verification differs from the original ADMM setup mainly in that the coupling constraint is a semidefinite constraint, not a linear equality constraint. To simplify the representation, define two indicator functions as

$$\begin{aligned}\mathcal{C}_1(\alpha_1) &= \mathbf{1}\left\{\forall x \in \{x|\, V(x) \geq 1\}, d \in \mathcal{D}, \frac{\partial V}{\partial x}(f(x) + g_1(x)u_1(x,d) + g_d(x)d) \leq \alpha_1(x,d)\right\}, \\ \mathcal{C}_2(\alpha_2) &= \mathbf{1}\left\{\forall x \in \{x|\, V(x) \geq 1\}, d \in \mathcal{D}, \frac{\partial V}{\partial x} g_2(x) u_2(x,d) \leq \alpha_2(x,d)\right\}.\end{aligned} \tag{5.17}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Further, define $\mathcal{F}_1(\alpha_1)$ and $\mathcal{F}_2(\alpha_2)$ as

$$\begin{aligned}\mathcal{F}_1(\alpha_1) &= \begin{cases} 0, & \mathcal{C}_1(\alpha_1) = 1 \\ \infty, & \mathcal{C}_1(\alpha_1) = 0 \end{cases}, \\ \mathcal{F}_2(\alpha_2) &= \begin{cases} 0, & \mathcal{C}_2(\alpha_2) = 1 \\ \infty, & \mathcal{C}_2(\alpha_2) = 0 \end{cases}.\end{aligned} \tag{5.18}$$

Assumption 5.1: *The set of α_1 and α_2 that satisfy (5.11)* is nonempty.

Then $\mathcal{F}_1(\alpha_1)$ and $\mathcal{F}_2(\alpha_2)$ are closed, proper and convex. The verification problem then becomes

$$\min_{\alpha_1, \alpha_2} \mathcal{F}_1(\alpha_1) + \mathcal{F}_2(\alpha_2) \, s.t. \, \alpha_1 + \alpha_2 \leq 0, \tag{5.19}$$

where α_1 and α_2 are polynomials of x and d, and the inequality sign enforces negative definiteness.

Then, the Lagrangian needs to be formulated. The SDP literature suggests two common ways to formulate the Lagrangian: the conic representation [110] and the minimum eigenvalue representation [111]. Although the former is more commonly used in SDP solvers, the latter formulation is used primarily because it generates only one dual variable for each SDP constraint, thus simplifying the dual decomposition process. The Lagrangian is formulated as

$$L(\alpha_1, \alpha_2, \lambda) = \mathcal{F}_1(\alpha_1) + \mathcal{F}_2(\alpha_2) - \lambda \lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right), \tag{5.20}$$

where Q_{α_1} and Q_{α_2} are the matrix representations of the polynomials α_1 and α_2; $\lambda > 0$ is the dual variable corresponding to the semidefinite constraint:

$$\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right) \geq 0. \tag{5.21}$$

With this Lagrangian formulation, the dual ascent algorithm appears as follows:

$$\begin{aligned} \alpha_1^{k+1} &= \arg\min_{\alpha_1} L\left(\alpha_1, \alpha_2^k, \lambda^k\right), \\ \alpha_2^{k+1} &= \arg\min_{\alpha_2} L\left(\alpha_1^{k+1}, \alpha_2, \lambda^k\right), \\ \lambda^{k+1} &= \lambda^k - \gamma\lambda_{\min}\left(-Q_{\alpha_1^{k+1}} - Q_{\alpha_2^{k+1}}\right), \end{aligned} \tag{5.22}$$

where γ is a constant representing the step size. Notice that the update for α_1 is

$$\alpha_1^{k+1} = arg\min_{\alpha_1} \mathcal{F}_1\left(\alpha_1\right) + \mathcal{F}_2\left(\alpha_2^k\right) - \lambda\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right). \tag{5.23}$$

If $\mathcal{F}_2(\alpha_2^k) \neq \infty$, then

$$\mathcal{F}_1\left(\alpha_1\right) + \mathcal{F}_2\left(\alpha_2^k\right) + \lambda\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right) = \begin{cases} \lambda\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right), & \mathcal{C}_1\left(\alpha_1\right) = 1 \\ \infty, & \mathcal{C}_1\left(\alpha_1\right) = 0 \end{cases}. \tag{5.24}$$

Assume there exists an α_1 such that $\mathcal{C}_1(\alpha_1) = 1$, the update for α_1 is

$$\alpha_1^{k+1} = arg\min_{\alpha_1} -\lambda\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2^k}\right) \; s.t. \mathcal{C}_1(\alpha_1) = 1. \tag{5.25}$$

Therefore, λ can be normalized to be 1. Similarly, in the update of α_2, λ is also normalized to 1, which renders the dual update step unnecessary. The procedure for dual decomposition is simplified to

$$\begin{aligned} \alpha_1^{k+1} &= arg\min_{\alpha_1} -\lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2^k}\right) \; s.t. \mathcal{C}_1(\alpha_1) = 1, \\ \alpha_2^{k+1} &= arg\min_{\alpha_2} -\lambda_{\min}\left(-Q_{\alpha_1^{k+1}} - Q_{\alpha_2}\right) \; s.t. \mathcal{C}_2(\alpha_2) = 1. \end{aligned} \tag{5.26}$$

The purpose of dual decomposition, in this case, is not to reduce the size of the optimization but rather to allow the agents to keep the control algorithms confidential. As such, dual decomposition suits this purpose perfectly. With this procedure, the two agents are asked to exchange polynomials α_1 and α_2, and no further information about the control algorithms.

5.4.2 Convergence of decentralized verification

Next, let's discuss the convergence of decentralized verification. First, the following theorem is presented.

Theorem 5.1: *Suppose there exist* $\alpha_1^\star$ *and* $\alpha_2^\star$ *that satisfy (5.11) and (5.12), then* $[\alpha_1^\star, -\alpha_1^\star]$ *and* $[-\alpha_2^\star, \alpha_2^\star]$ *also satisfy (5.11) and (5.12).*

Proof: Add (5.12) to (5.11):

$$\begin{aligned} \frac{dV}{dx}\left(f(x) + g_1(x)u_1(x,d) + g_d(x)d \right) &\le -\alpha_2^\star(x,d), \\ \frac{dV}{dx} g_2(x)u_2(x,d) &\le -\alpha_1^\star(x,d), \end{aligned} \tag{5.27}$$

which proves that $[\alpha_1^\star, -\alpha_1^\star]$ *and* $[-\alpha_2^\star, \alpha_2^\star]$ *satisfy (5.11). It is also straightforward to check that* $[\alpha_1^\star, -\alpha_1^\star]$ *and* $[-\alpha_2^\star, \alpha_2^\star]$ *satisfy (5.12).*

■

With Theorem 5.1, (5.12) can be replaced with the following equality constraint

$$\alpha_1 + \alpha_2 = 0, \tag{5.28}$$

which can be transformed to a linear equality constraint on the coefficients. Decentralized verification, therefore, follows the standard form of ADMM in [109]. With Assumption 5.1, $\mathcal{F}_1(\alpha_1)$ and $\mathcal{F}_2(\alpha_2)$ are closed, proper and convex functions of α_1 and α_2, and therefore Theorem 3.2 in [109] proves convergence of the dual decomposition process.

Remark 5.3: *In practice, it is found that compared to dual decomposition with the equality constraint in (5.28), the convergence is actually faster if the inequality constraint in (5.12) is used and the Lagrangian is formulated as in (5.20). However, there is no proof of convergence.*

5.4.3 Verification for systems with piecewise dynamics

As mentioned in the problem formulation section, this method requires an explicit analytic form of the control algorithm. This restriction can be relaxed to a piecewise analytic form of the closed-loop dynamics. Suppose there exists some piecewise structure in the control algorithm such as

$$u_1(x,d)=\begin{cases} u_1^1(x,d),\ if\ [x,d]\in\{x,d|\ G_1(x,d)\geq 0\} \\ \vdots \\ u_1^n(x,d),\ if\ [x,d]\in\{x,d|\ G_n(x,d)\geq 0\} \end{cases}, \tag{5.29}$$

where $\{u_1^i(\cdot)\}$ are analytic functions of x and d, $\{\{x,d|\ G_i(x,d)\geq 0\}\}$ are disjoint semialgebraic sets of x and d that satisfies

$$\bigcup_{i=1\ldots n}\{x,d|\ G_i(x,d)\geq 0\}=\mathcal{S}\times\mathcal{D}, \tag{5.30}$$

where $\mathcal{S}$ is the domain of x, $\mathcal{S}\times\mathcal{D}$ is the set on which $u_i(\cdot)$ are defined. Then the condition for α_1 is modified to

$$\begin{aligned} &\forall[x,d]\in\left(\{x|\ V(x)\geq 1\}\times\mathcal{U}\right)\cap\{x,d|\ G_i(x,d)\geq 0\},i=1,\ldots n, \\ &\frac{dV}{dx}\left(f(x)+g_1(x)u_1^i(x,d)+g_d(x)d\right)\leq\alpha_1(x,d). \end{aligned} \tag{5.31}$$

The rest of the verification procedure stays the same.

5.4.4 Extension to control synthesis

The proposed dual decomposition method can be used for both the verification of existing control design and synthesis of controllers that satisfy the specification. Consider the update procedure in (5.22). Suppose one or both controllers need to be synthesized. Then the Lagrangian would depend on the control algorithms:

$$L(\alpha_1,\alpha_2,\lambda,u_1(\cdot),u_2(\cdot))=\mathcal{F}_1(\alpha_1,u_1(\cdot))+\mathcal{F}_2(\alpha_2,u_2(\cdot))-\lambda\lambda_{\min}\left(-Q_{\alpha_1}-Q_{\alpha_2}\right). \tag{5.32}$$

Since $u_1(\cdot)$ and $u_2(\cdot)$ are part of the optimization variables, the two functions $\mathcal{F}_1(\alpha_1,u_1(\cdot))$ and $\mathcal{F}_2(\alpha_2,u_2(\cdot))$ depend on the control algorithms as well.

The dual decomposition procedure is used to solve the problem with the dual ascent method. In addition to the Lyapunov condition, cost functions and constraints such as the control input bounds may be added to the synthesis process. An example where the tradeoff between different control objectives is resolved with this synthesis procedure is presented in Section 5.6.2.4.

Remark 5.4: *It is important that the proposed verification method does not expose the control algorithm of each agent. Note that*

- *the polynomial upper bound of Lyapunov derivative being exchanged between agents may not be tight, so the control law that achieves it is not unique;*
- *the two agents do not exchange the form of their controller (e.g. linear, quadratic or piecewise linear);*
- *the algorithm relies on SOS multipliers, which are not exchanged between agents (especially when the controller has piecewise structure).*

Due to the above three reasons, it is not possible for one agent to identify the control algorithm of the other agent in most cases.

5.5 Improving the Lyapunov function candidate

The above-mentioned dual decomposition is only guaranteed to succeed if the original centralized verification is feasible, which is strongly limited by the choice of the Lyapunov function candidate. A poorly chosen Lyapunov function may fail the verification even though the system satisfies the specification.

The problem of finding a Lyapunov function with its 1-level set being invariant is, however, bilinear in the Lyapunov function candidate and the SOS multipliers, and is thus nonconvex. Methods for solving Bilinear Matrix Inequality (BMI) include bilinear alternation [112], which is local, and branch-and-cut, which gives a global solution to a relaxed convex problem [113].

In the Lyapunov verification problem, besides the bilinearity that arises from the invariant set setup, the need to decompose and decentralize the algorithm should also be considered due to the confidentiality requirement. A method that uses perturbation to resolve the bilinearity is proposed. The idea is simple. Starting with an initial Lyapunov function candidate, the algorithm refines the Lyapunov function candidate and seeks to find a tighter lower bound of the Lyapunov derivative by adding a small perturbation to the Lyapunov function. Since the perturbation is small, the level set does not change much, so the condition is verified on the level set of the original Lyapunov function, thus eliminating the bilinear term.

5.5.1 Centralized Lyapunov perturbation

First, the perturbation algorithm for centralized verification is presented. Suppose the specification requires the state to stay inside a set $X_{safe} \in \mathbb{R}^n$, and the Lyapunov candidate satisfies

$\{x|\ V(x)\le 1\}\in X_{safe}$. Then an additional slack variable e is introduced to make the problem always feasible:

$$\begin{aligned}&\min_{V,e} e\ \ s.t.\\&\forall x\in\{x|\ V(x)\ge 1\},\forall d\in D,\\&\frac{dV}{dx}\left(f(x)+g_1(x)u_1(x,d)+g_2(x)u_2(x,d)+g_d(x)d\right)\le ex^Tx.\end{aligned}\tag{5.33}$$

If $e<0$, the original centralized verification is successful. Using perturbation, the bilinear term is eliminated:

$$\begin{aligned}&\min_{e,\Delta V} e\ \ s.t.\{x|\ V_0(x)+\Delta V(x)\le 1\}\in X_{safe},\\&\forall x\in\{x|\ V_0(x)\le 1\},\forall d\in\mathcal{D},\\&\frac{d(V_0+\Delta V)}{dx}\dot{x}\le ex^Tx,\|Q_{\Delta V}\|\le\epsilon\|Q_{V_0}\|,\end{aligned}\tag{5.34}$$

where ΔV is the perturbation and V_0 is the original Lyapunov candidate. The last constraint limits the size of the perturbation with a small constant $\epsilon>0$. The inner product between matrices is defined as

$$\langle A,B\rangle=Tr(A^TB),\tag{5.35}$$

and the matrix norm is induced from this inner product. The above optimization iterates until no further improvement can be made and $\|Q_{\Delta V}\|\to 0$.

5.5.2 Decentralized Lyapunov perturbation

The extension of Lyapunov perturbation from centralized verification to decentralized verification is not straightforward. The "local copy" idea is adopted, similar to that in [114]. In the search for ΔV, the two agents will each keep a local copy of ΔV, and the two local copies should converge to be identical. Dual decomposition will again be used to guarantee convergence. Consider the following decentralized Lyapunov perturbation:

$$
\begin{aligned}
&\min_{e,\Delta V_1,\Delta V_2} e \ \ s.t.\ \ \forall x \in \{x|\ V_0 \geq 1\}, d \in \mathcal{D},\\
&\frac{d\left(V_0+\Delta V_1\right)}{dx}\left(f\left(x\right)+g_d\left(x\right)d+g_1\left(x\right)u_1\left(x,d\right)\right) \leq \alpha_1\left(x,d\right),\\
&\frac{d\left(V_0+\Delta V_2\right)}{dx}g_2\left(x\right)u_2\left(x,d\right) \leq \alpha_2\left(x,d\right),\\
&\alpha_1\left(x,d\right)+\alpha_2\left(x,d\right) \leq ex^T x,\\
&\Delta V_1 = \Delta V_2, \{x|\ V_0+\Delta V_1 \leq 1\} \in X_{safe}, \left\|Q_{\Delta V_1}\right\| \leq \epsilon \left\|Q_{V_0}\right\|.
\end{aligned}
\tag{5.36}
$$

Define the indicator functions $\bar{\mathcal{C}}_1$ and $\bar{\mathcal{C}}_2$ as

$$
\begin{aligned}
\bar{\mathcal{C}}_1\left(\alpha_1,\Delta V_1\right) &= \mathbf{1}\left\{\begin{array}{l}\forall x \in \{x|\ V_0 \geq 1\}, \forall d \in \mathcal{D},\\ \dfrac{d\left(V_0+\Delta V_1\right)}{dx}\left(f\left(x\right)+g_d\left(x\right)d+g_1\left(x\right)u_1\left(x,d\right)\right) \leq \alpha_1\left(x,d\right),\\ \{x|\ V_0+\Delta V_1 \leq 1\} \in X_{safe}, \left\|Q_{\Delta V_1}\right\| \leq \epsilon \left\|Q_{V_0}\right\|\end{array}\right\},\\
\bar{\mathcal{C}}_2\left(\alpha_2,\Delta V_2\right) &= \mathbf{1}\left\{\begin{array}{l}\forall x \in \{x|\ V_0 \geq 1\}, \forall d \in \mathcal{D},\\ \dfrac{d\left(V_0+\Delta V_2\right)}{dx}g_2\left(x\right)u_2\left(x,d\right) \leq \alpha_2\left(x,d\right),\\ \{x|\ V_0+\Delta V_2 \leq 1\} \in X_{safe}, \left\|Q_{\Delta V_2}\right\| \leq \epsilon \left\|Q_{V_0}\right\|\end{array}\right\}.
\end{aligned}
\tag{5.37}
$$

Then define feasibility functions $\mathcal{F}_1(\alpha_1,\Delta V_1)$ and $\mathcal{F}_2(\alpha_2,\Delta V_2)$ following the convention in (5.18) based on the indicator function $\bar{\mathcal{C}}_1$ and $\bar{\mathcal{C}}_2$. The Lagrangian is then formulated as

$$
\begin{aligned}
L\left(\alpha_1,\alpha_2,\Delta V_1,\Delta V_2,\lambda,e\right) = e &+ \left\langle \lambda, Q_{\Delta V_1}-Q_{\Delta V_2}\right\rangle + \frac{\rho}{2}\left\|Q_{\Delta V_1}-Q_{\Delta V_2}\right\|^2\\
&+\mathcal{F}_1\left(\alpha_1,\Delta V_1\right)+\mathcal{F}_1\left(\alpha_2,\Delta V_2\right).
\end{aligned}
\tag{5.38}
$$

Here it is assumed that an admissible perturbation exists that renders $\bar{\mathcal{C}}_1$ and $\bar{\mathcal{C}}_2$ equal to 1. This process is a direct extension of the standard dual decomposition setup, and the dual variable λ is a real symmetric matrix of the same size as ΔV. The dual ascent update is

$$
\begin{aligned}
\left(\alpha_1^{k+1},\Delta V_1^{k+1}\right) &= \arg\min_{\alpha_1,\Delta V_1} L\left(\alpha_1,\alpha_2^k,\Delta V_1,\Delta V_2^k,\lambda^k,e\right),\\
\left(\alpha_2^{k+1},\Delta V_2^{k+1}\right) &= \arg\min_{\alpha_2,\Delta V_2} L\left(\alpha_1^{k+1},\alpha_2,\Delta V_1^{k+1},\Delta V_2,\lambda^k,e\right),\\
\lambda^{k+1} &= \lambda^k + \rho\left(\Delta V_1^{k+1}-\Delta V_2^{k+1}\right).
\end{aligned}
\tag{5.39}
$$

The perturbation algorithm is demonstrated in Figure 5.1. Two loops in the Lyapunov perturbation algorithm allow for decentralized verification: an inner loop that converges to ΔV

and an outer loop that iteratively perturbs the current Lyapunov function until no improvement can be made.

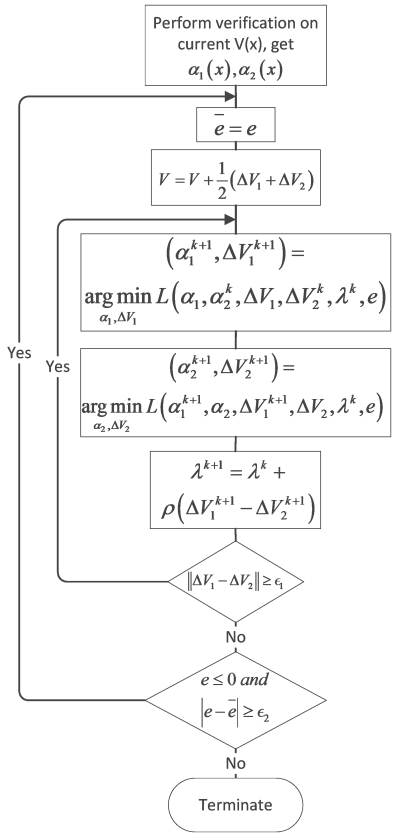

Figure 5.1 Lyapunov perturbation procedure

5.6 Case studies

The proposed methodology is applied to two examples: a toy problem of an inverted pendulum and a practical vehicle chassis control problem.

5.6.1 Inverted pendulum

As shown in Figure 5.2, the two control inputs are the horizontal force F on the cart and the torque T on the hinge. The cart has mass M, the ball on the tip of the pendulum has mass m, the pendulum has length L and no mass. The two degrees of freedom (DOF) are displacement of cart S and angle of the pendulum ϕ. In addition to the actuation, a disturbance force d_1 acts on the cart and a disturbance torque d_2 acts on the pendulum.

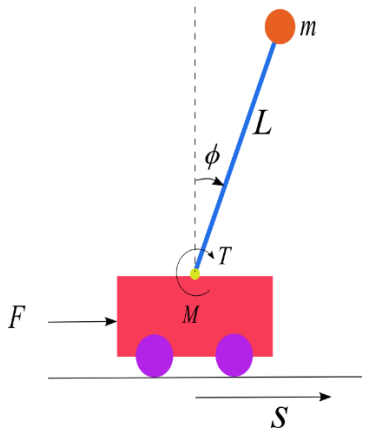

Figure 5.2 Lyapunov perturbation procedure

The dynamic equations are obtained using the Lagrangian method:

$$\begin{aligned}\frac{d^2\phi}{dt^2} &= \frac{(M+m)mgL\sin(\phi)-(mL\dot{\phi})^2\sin(\phi)\cos(\phi)}{(M+m)mL^2-(mL)^2\cos^2(\phi)} \\ &+\frac{-(mL\cos(\phi))F+(m+M)T}{(M+m)mL^2-(mL)^2\cos^2(\phi)}+d_1, \\ \frac{d^2S}{dt^2} &= \frac{mL^2mL\dot{\phi}^2\sin(\phi)-(mL)^2g\sin(\phi)\cos(\phi)}{(M+m)mL^2-(mL)^2\cos^2(\phi)} \\ &+\frac{mL^2F-mL\cos(\phi)T}{(M+m)mL^2-(mL)^2\cos^2(\phi)}+d_2.\end{aligned} \tag{5.40}$$

Two simplified models are considered:

1. *Simplified nonlinear model:*

Assume $\phi \ll 1$, therefore $sin(\phi) \approx \phi$ and $\cos(\phi) \approx 1$.

$$\begin{aligned}\frac{d^2\phi}{dt^2} &= \frac{(M+m)mgL\phi-(mL\dot{\phi})^2\phi-mLF+(m+M)T}{MmL^2}+d_1, \\ \frac{d^2S}{dt^2} &= \frac{mL^2mL\dot{\phi}^2\phi-(mL)^2g\phi+mL^2F-mLT}{MmL^2}+d_2.\end{aligned} \tag{5.41}$$

2. *Linear model:*

In addition to the small angle assumption, ignore the higher order terms of ϕ and $\dot{\phi}$. Then the model becomes linear:

$$\dot{x} = Ax + B_1u_1 + B_2u_2 + B_dd, \tag{5.42}$$

where

$$x = \begin{bmatrix} S & \dot{S} & \phi & \dot{\phi} \end{bmatrix}, u_1 = F, u_2 = T. \tag{5.43}$$

5.6.1.1 Lyapunov verification for inverted pendulum

The specification is assumed to impose bounds on all four states:

$$|x_i| \le x_{\max^i}, \ i = 1,2,3,4. \tag{5.44}$$

A hyperbox is constructed from the specification. The Lyapunov function is obtained by solving the LQR cost-to-go function. Then the cost-to-go function is scaled such that its 1-level set is contained in the hyperbox constructed from the specification. Bounds are put on the disturbance terms.

To demonstrate the verification process, a set of sane controllers is synthesized via SOS programming that satisfies the Lyapunov condition. The order of α_1 and α_2 are quadratic. While to guarantee the existence of a solution, the order of α_1 and α_2 must be as high as the original polynomial to be verified, in practice, using lower order polynomials is often found to be sufficient. The order of the multipliers is selected according to the computation capability of existing SDP solvers, in this case, 4. α_1 and α_2 are initialized to be 0.

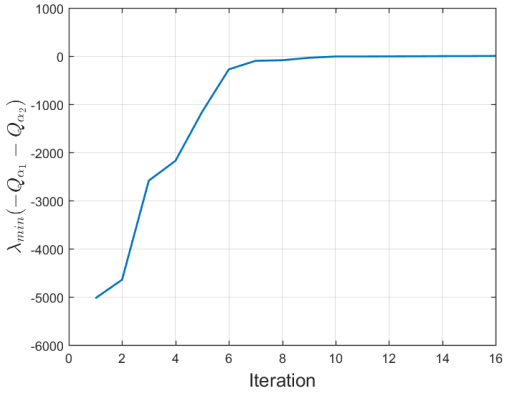

Figure 5.3 Verification of the centralized synthesized controllers for inverted pendulum

Figure 5.3 shows that $\lambda_{min}(-Q_{\alpha_1} - Q_{\alpha_2})$ monotonically increases over 16 iterations. Eventually, it becomes positive, proving that the Lyapunov derivative is negative definite outside the 1-level set of the Lyapunov function and the verification is successful.

5.6.1.2 Lyapunov perturbation for inverted pendulum

In order to demonstrate the Lyapunov perturbation process, the Lyapunov function is altered so that the derivative condition is not met. With the altered Lyapunov function, the verification process converges to α_1 and α_2, whose summation is not negative definite:

$$\lambda_{\min}\left(-Q_{\alpha_1}-Q_{\alpha_2}\right)=-451. \tag{5.45}$$

Then using the perturbation algorithm in Figure 5.1, the Lyapunov function is changed to improve the result of verification. After four iterations, a Lyapunov function that renders the verification successful is obtained.

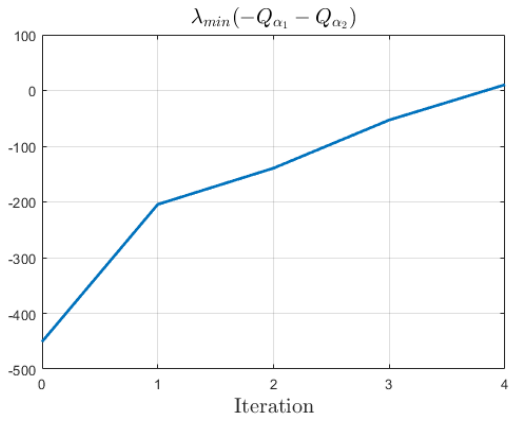

Figure 5.4 Lyapunov perturbation process

5.6.2 Vehicle chassis control

The second case study is the motivating engineering application: vehicle chassis control. Two controllers that have strong coupling are selected: LK and ESC. The dynamic model for the vehicle lateral dynamics is the linear lateral-yaw model:

$$
\begin{aligned}
\begin{bmatrix} \dot{y} \\ \dot{v}_y \\ \dot{\psi} \\ \dot{r} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & v_x & 0 \\ 0 & -\dfrac{C_{\alpha f}+C_{\alpha r}}{m v_x} & 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{m v_x}-v_x \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{I_z v_x} & 0 & -\dfrac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z v_x} \end{bmatrix} \begin{bmatrix} y \\ v_y \\ \psi \\ r \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ \dfrac{C_{\alpha f}}{m} \\ 0 \\ \dfrac{aC_{\alpha f}}{I_z} \end{bmatrix} \delta_f + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} T + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} d \\
&= Ax + B_1 u_1 + B_2 u_2 + B_d d,
\end{aligned}
\tag{5.46}
$$

where the four states are the lateral displacement w.r.t. the road center y, sideslip velocity of the chassis v_y, yaw angle ψ and the yaw rate r. The inputs are the steering angle δ_f and yaw moment T generated by differential braking. m and I_z are the mass and moment of inertia of the vehicle, $C_{\alpha f}$ and $C_{\alpha r}$ are the cornering stiffness of the two axles, a and b are the distance from vehicle CG (center of gravity) to the front and rear axle. These parameters are constant. v_x is the longitudinal speed, if it stays constant, the dynamic is linear. Disturbance d comes in with three channels: $d = \begin{bmatrix} F_{dy} & r_d & T_d \end{bmatrix}^T$: the external lateral force F_{dy} affects v_y, road curvature r_d affects yaw angle, and the external yaw moment T_d affects the yaw rate.

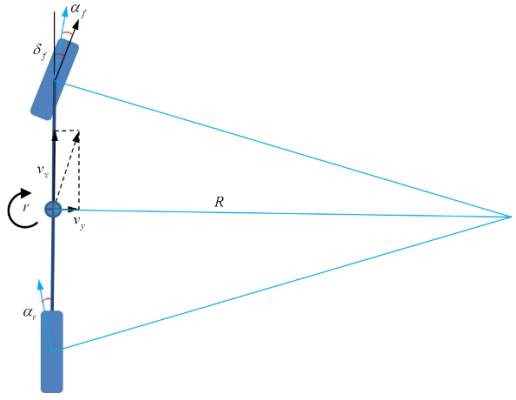

Figure 5.5 Lateral yaw model

5.6.2.1 Lyapunov function setting for vehicle chassis control

It is assumed that the Lyapunov function has the following form:

$$V(x) = \min\{V_1(x), V_2(x)\}. \tag{5.47}$$

This form arises from the assumption that each supplier may have its own focus on the performance. The two Lyapunov functions are assumed to represent a different emphasis on performance.

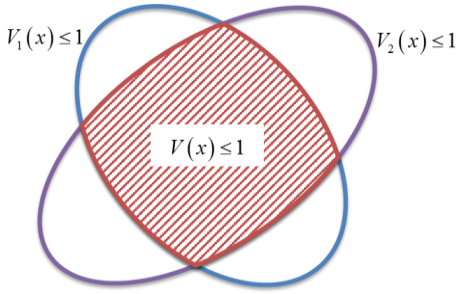

Figure 5.6 Level set of the Lyapunov function for vehicle chassis control

The goal of verification is to prove that the 1-level set of V is invariant, as shown in Figure 5.6. For simplicity, it is required that both the 1-level set of V_1 and that of V_2 are invariant, which is a sufficient condition of $\{x|\, V(x) \le 1\}$ being invariant.

The Lyapunov derivative condition then becomes

$$\begin{aligned} &\forall x \in \{x|\, V_1(x) \ge 1\}, d \in \mathcal{D}, \dot{V}_1 \le 0, \\ &\forall x \in \{x|\, V_2(x) \ge 1\}, d \in \mathcal{D}, \dot{V}_2 \le 0. \end{aligned} \tag{5.48}$$

It is assumed that the LK controller u_1 depends on state x and the road curvature r_d, which is a measured disturbance, and the ESC controller u_2 uses only state feedback. The decentralized verification aims to find polynomials α_1, α_2, β_1 and β_2 that satisfy

$$\begin{aligned} &\forall x \in \{x|\, V_1(x) \ge 1\}, d \in \mathcal{D}, \\ &\frac{dV_1}{dx}\left(Ax + B_1 u_1(x,d) + B_d d\right) \le \alpha_1(x); \frac{dV_1}{dx} B_2 u_2(x) \le \alpha_2(x), \\ &\forall x \in \{x|\, V_2(x) \ge 1\}, d \in \mathcal{D}, \\ &\frac{dV_2}{dx}\left(Ax + B_1 u_1(x,d) + B_d d\right) \le \beta_1(x); \frac{dV_2}{dx} B_2 u_2(x) \le \beta_2(x), \end{aligned} \tag{5.49}$$

where α_1 and α_2 are the lower bounds of the two parts of $\dot{V}_1$, β_1 and β_2 are the lower bounds of the two parts of $\dot{V}_2$. Since u_2 does not depend on d, the polynomial bounds depend only on x. If

$\alpha_1 + \alpha_2 \le 0$ and $\beta_1 + \beta_2 \le 0$, the verification is successful. Note that the verification of the two Lyapunov derivative conditions is decoupled, so the verification follows the same procedure, except that it should be done on both $\dot{V}_1$ and $\dot{V}_2$.

Define the following:

$$\lambda_{min}^{\alpha} = \lambda_{\min}\left(-Q_{\alpha_1} - Q_{\alpha_2}\right),$$
$$\lambda_{min}^{\beta} = \lambda_{\min}\left(-Q_{\beta_1} - Q_{\beta_2}\right). \tag{5.50}$$

If both λ_{min}^{α} and λ_{min}^{β} are positive, the verification is successful.

5.6.2.2 Dealing with varying v_x

One major challenge for this dynamic system is the varying v_x. The lateral dynamic model is linear only when the longitudinal speed v_x is constant. In reality, v_x may vary, for example, due to differential braking. Thus the verification must be done for v_x in a range. A convex hull method adopted from [115] is used to solve this problem. The verification is performed on several linear systems. Note that v_x affects only the A matrix in (5.46); furthermore, the entries of A matrix are linear functions of v_x and $1/v_x$.

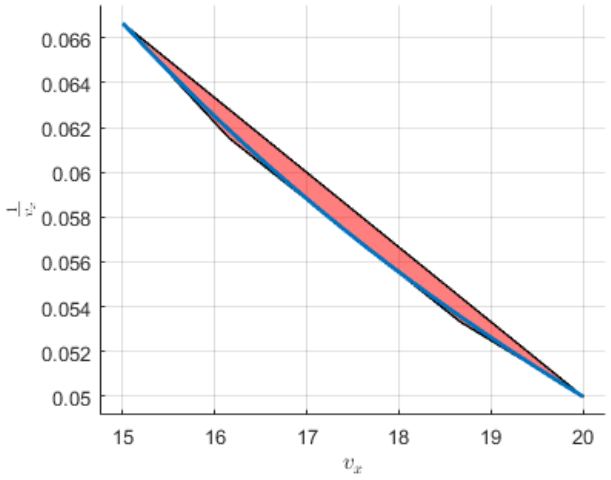

Figure 5.7 Convex hull $v_x - 1/v_x$ of the curve

Based on the linear relationship between v_x, $1/v_x$ and the A matrix, two variables p and q are introduced. In Figure 5.7, each point on the p-q plane corresponds to an A matrix by the following mapping:

$$A(p,q)=\begin{bmatrix} 0 & 1 & p & 0 \\ 0 & -\dfrac{C_{\alpha f}+C_{\alpha r}}{m}q & 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{m}q-p \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{I_z}q & 0 & -\dfrac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z}q \end{bmatrix}. \tag{5.51}$$

Each point on the $p\cdot q=1$ curve corresponds to an A matrix for the vehicle lateral dynamic model in (5.46) under the speed $v_x=p$, plotted as the blue curve in Figure 5.7. Then a polytope is constructed as a convex hull of the curve, plotted as the red polytope in Figure 5.7. A finite set of A matrices is constructed by taking the vertices of the polytope, denoted as $\mathcal{W}$. Since the mapping from *p* and *q* to Lyapunov derivative is linear if the Lyapunov derivative condition is satisfied by each of the linear models in $\mathcal{W}$, by convex inclusion, all models corresponding to the points inside the polytope satisfy the Lyapunov derivative condition, including the models with v_x varying between $v_{\min}$ and $v_{\max}$.

For simplicity, taking the verification of $\dot{V}_1$ for example (the verification of $\dot{V}_2$ follows the same procedure), the Lyapunov condition becomes

$$\begin{aligned} &\forall x\in\{x|\ V_1(x)\geq 1\},\forall d\in\mathcal{D},\forall A_i\in\mathcal{W}, \\ &\frac{dV_1}{dx}\left(A_ix+B_1u_1(x,d)+B_2u_2(x)+B_dd\right)\leq 0. \end{aligned} \tag{5.52}$$

The decentralized verification is to find α_1 and α_2 such that

$$\begin{aligned} &\forall x\in\{x|\ V_1(x)\geq 1\},\forall d\in\mathcal{D},\forall A_i\in\mathcal{W}, \\ &\frac{dV_1}{dx}\left(A_ix+B_1u_1(x,d)+B_dd\right)\leq\alpha_1(x), \\ &\frac{dV_1}{dx}B_2u_2(x)\leq\alpha_2(x);\ \alpha_1(x)+\alpha_2(x)\leq 0. \end{aligned} \tag{5.53}$$

In practice, the optimization in (5.53) has a heavy computation load due to a large number of the constraints. An alternative (relaxed) problem can be solved instead:

$$
\begin{aligned}
&\forall x \in \left\{x \middle| V_1(x) \geq 1\right\}, \forall d \in \mathcal{D}, \forall A_i \in \mathcal{W}, \\
&\frac{dV_1}{dx}\left(A_i x + B_1 u_1(x,d) + B_d d\right) \leq \alpha_1^i(x), \\
&\frac{dV_1}{dx} B_2 u_2(x) \leq \alpha_2(x); \\
&\forall i, \alpha_1^i \leq \alpha_1; \alpha_1(x) + \alpha_2(x) \leq 0.
\end{aligned}
\tag{5.54}
$$

This relaxation is a conservative approximation of the original search, as it is the sufficient condition, but it allows separate search of α_1^i for each A_i, and then search for a common upper bound for all the α_1^i. It accelerates the search, but λ_{min}^{α} and λ_{min}^{β} may not decrease monotonically, as shown in Figure 5.8.

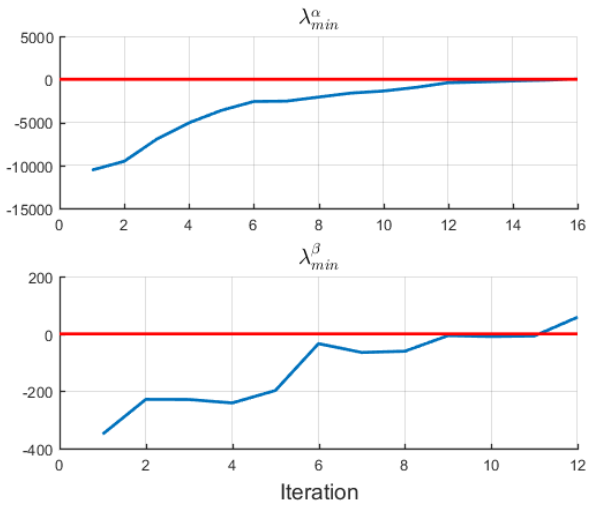

Figure 5.8 Verification of ESC+LK with the convex hull

5.6.2.3 Verification of controllers with piecewise structure

Next, the case where the chassis controllers are piecewise functions of the states is considered. An example is presented to demonstrate the capability of the proposed method to deal with piecewise controllers. In this example, $u_1(\cdot)$ is assumed to have the following form:

$$
u_1 = K_{ff} d + \begin{cases} g(x), & V_1(x) \leq 1.1 \\ sat_{u_{\max}}\left(k_1^T x\right), & 1.1 \leq V_1(x) \leq 1.2 \end{cases},
\tag{5.55}
$$

where g is a quadratic function of x, u_{max} is a constant, representing the maximum allowed feedback input, and $sat_{u_{max}}$ is the saturation function that saturates the input at $\pm u_{max}$. In effect, when the state is close to the origin ($\left\{x \middle| V_1(x) \leq 1.1\right\}$), a mild nonlinear control law is used; as the

state moves far from the origin ($\{x|\ 1.2 \geq V_1(x) \leq 1.1\}$), a more aggressive linear control law is used; when the states are too far away from the origin ($\{x|\ V_1(x) \geq 1.2\}$), the control law is not defined. Moreover, due to actuator capacity, the control input saturates at $\pm u_{max}$.

While a real controller used in the industry may be much more complicated than the controller described above, this controller is a good example of controllers with piecewise structures.

The controller is synthesized so that it satisfies the Lyapunov condition.

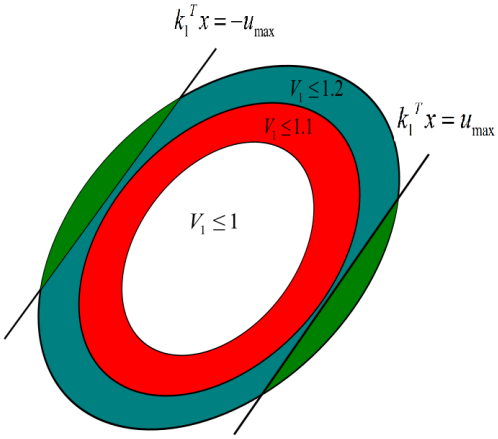

Figure 5.9 Piecewise control structure for u_1

Since $u_2(\cdot)$ has a single analytical form, the search for α_2 remains the same as the original algorithm in (5.26); the search for α_1 uses the following procedure:

$$\begin{aligned}
&\alpha_1 = \arg\max_{\alpha_1} \lambda_{\min}\left(-Q_{\alpha_2} - Q_{\alpha_1}\right) s.t. \\
&\forall x \in \{x|\ 1 \leq V_1(x) \leq 1.1\}, d \in \mathcal{D}, \frac{dV_1}{dx}\left(Ax + B_1 g(x) + B_d d\right) \leq \alpha_1(x), \\
&\forall x \in \{x|\ 1.1 \leq V_1(x) \leq 1.2\} \cap \{x|\ |k_1^T x| \leq u_{\max}\}, d \in \mathcal{D}, \frac{dV_1}{dx}\left(Ax + B_1 k_1^T x + B_d d\right) \leq \alpha_1(x), \\
&\forall x \in \{x|\ 1.1 \leq V_1(x) \leq 1.2\} \cap \{x|\ k_1^T x \geq u_{\max}\}, d \in \mathcal{D}, \frac{dV_1}{dx}\left(Ax + B_1 u_{\max} + B_d d\right) \leq \alpha_1(x).
\end{aligned} \tag{5.56}$$

One can use the relaxation trick in (5.54) to make the computation complexity grow linearly with the number of regions in the piecewise controller. This is also true with multiple piecewise controllers (e.g., both $u_1(\cdot)$ and $u_2(\cdot)$ are piecewise functions), since the search for α_1 and α_2 are performed separately.

5.6.2.4 Synthesis of vehicle chassis control

Unlike verification, synthesis with this Lyapunov function is more complicated. In (5.49), if $u_1(\cdot)$ and $u_2(\cdot)$ are part of the optimization variables, the verification of the two Lyapunov derivative conditions become coupled, since changing the control algorithm to decrease $\dot{V}_1$ could end up increasing $\dot{V}_2$, and vice versa. Because of this coupling, the dual variable cannot be normalized to 1, as was the case in (5.26).

To simplify the expression, the feasibility functions $\mathcal{F}_1^{\alpha}(\alpha_1, u_1(\cdot))$, $\mathcal{F}_2^{\alpha}(\alpha_2, u_2(\cdot))$, $\mathcal{F}_1^{\beta}(\beta_1, u_1(\cdot))$ and $\mathcal{F}_2^{\beta}(\beta_2, u_2(\cdot))$ are defined following the convention in (5.18). They take zero if the conditions in (5.49) are met; otherwise, infinity. Note that the indicator functions depend on the control algorithms as well. The Lagrangian is defined as

$$\begin{aligned} &L\left(\alpha_1, \alpha_2, \beta_1, \beta_2, \lambda_1, \lambda_2, u_1(\cdot), u_2(\cdot)\right) = J_1\left(u_1(\cdot)\right) + J_2\left(u_2(\cdot)\right) - \lambda_1 \lambda_{\min^{\alpha}} - \lambda_2 \lambda_{min}^{\beta} \\ &+\mathcal{F}_1^{\alpha}\left(\alpha_1, u_1(\cdot)\right) + \mathcal{F}_2^{\alpha}\left(\alpha_2, u_2(\cdot)\right) + \mathcal{F}_1^{\beta}\left(\beta_1, u_1(\cdot)\right) + \mathcal{F}_2^{\beta}\left(\beta_2, u_2(\cdot)\right), \end{aligned} \tag{5.57}$$

where $J_1\left(u_1(\cdot)\right)$ and $J_2\left(u_2(\cdot)\right)$ are cost functions of the control algorithms for performance purposes, λ_1 and λ_2 are the dual variables corresponding to the following two constraints, respectively:

$$\alpha_1 + \alpha_2 \le 0, \ \beta_1 + \beta_2 \le 0. \tag{5.58}$$

The dual ascent update is then

$$\begin{aligned} &\left[\alpha_1^{k+1}, \beta_1^{k+1}, u_1^{k+1}(\cdot)\right] = \underset{\alpha_1, \beta_1, u_1(\cdot)}{arg\min} L\left(\alpha_1, \alpha_2^k, \beta_1, \beta_2^k, \lambda_1^k, \lambda_2^k, u_1(\cdot), u_2^k(\cdot)\right), \\ &\left[\alpha_2^{k+1}, \beta_2^{k+1}, u_2^{k+1}(\cdot)\right] = \underset{\alpha_2, \beta_2, u_2(\cdot)}{arg\min} L\left(\alpha_1^{k+1}, \alpha_2, \beta_1^{k+1}, \beta_2, \lambda_1^k, \lambda_2^k, u_1^{k+1}(\cdot), u_2(\cdot)\right), \\ &\lambda_1^{k+1} = \max\left\{\lambda_1^k - \gamma \lambda_{\min}\left(-\alpha_1 - \alpha_2\right), 0\right\}, \\ &\lambda_2^{k+1} = \max\left\{\lambda_2^k - \gamma \lambda_{\min}\left(-\beta_1 - \beta_2\right), 0\right\}, \end{aligned} \tag{5.59}$$

where γ is the step size to update the dual variables. If λ_{min}^{α} and λ_{min}^{β} are both positive, the synthesis is successful; otherwise, continue the dual decomposition iteration.

In practice, it is found that performing a verification step before updating the dual variable, that is, verifying the system with the current controllers as described in Section 5.4, is beneficial to the

convergence. The verification process searches for an α and β that achieve the maximum λ_{min}^{α} and λ_{min}^{β}, thus making the synthesis converge faster.

In the ESC+LK case, the search of controllers is restricted to linear controllers, with the LK using road curvature to construct the feedforward control:

$$u_1(x,d) = k_1^T x + K_{ff} d,\ u_2(x) = k_2^T x. \tag{5.60}$$

The cost functions are defined as

$$J_1(u_1(\cdot)) = \|k_1\|_\infty,\ J_2(u_2(\cdot)) = \|k_2\|_\infty. \tag{5.61}$$

The synthesis takes five iterations to complete:

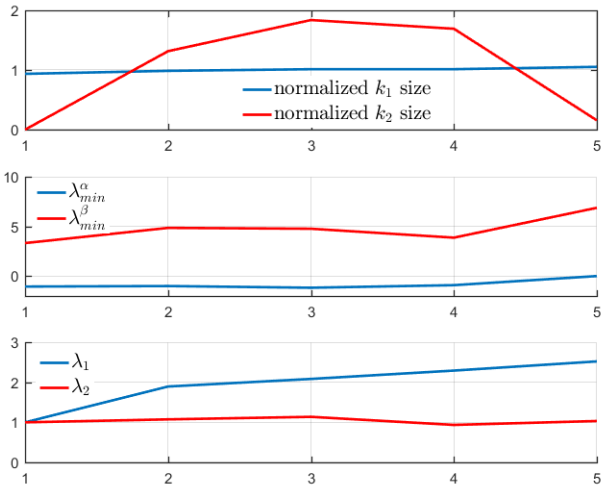

Figure 5.10 Synthesis of ESC+LK

Figure 5.10 shows the change of $\|k_1\|$ and $\|k_2\|$, λ_{min}^{α} and λ_{min}^{β}, λ_1 and λ_2. The plot shows how the synthesis process handles the tradeoff between the two Lyapunov conditions. At the beginning of synthesis, since $\lambda_{min}^{\alpha} \le 0$, λ_1 kept increasing. Since k_2 was very small after the first iteration, the solver realized that increasing k_2 was beneficial to increasing both λ_{min}^{α} and λ_{min}^{β}. As the size of k_2 grew, its influence on $\dot{V}_1$ was too large, which made $\dot{V}_1$ grow, and as λ_1 grew, the size of k_2 decreased and eventually the algorithm found a design that made both λ_{min}^{α} and λ_{min}^{β} positive.

5.7 Conclusion

This chapter discusses the situation where coexisting controllers become the disturbance. The key feature of this situation is that the disturbance is not necessarily hostile, but the control law must not be exposed due to the confidentiality requirement. A Lyapunov function based method is proposed for verifying of the closed loop performance with multiple coexisting controllers. This method uses dual decomposition to allow non-cooperative agents to exchange information in the form of polynomials, which does not expose their control algorithms. The decentralized verification is proved to be equivalent to the centralized verification, which would require exposure of the control algorithm. Moreover, the method also provides information regarding how to modify the existing controllers if the verification fails, and it is capable of directly synthesizing controllers given the specifications. The proposed method is applicable to controllers with analytic expressions, and an example application with a piecewise analytic controller is shown.

Chapter 6 Data-driven computation of minimal robust control invariant set

A data-driven framework to compute an approximation of a minimal robust control invariant set (mRCI) for an uncertain dynamical system is proposed in this chapter, which demonstrates the importance of properly treating the modeling uncertainty of the system in correct-by-construction control synthesis. In the previous chapters, the dynamic models are assumed to be known. In fact, most of the correct-by-construction synthesis literature assumes that the model is known, including the nominal model and the characterization of uncertainty, which is not an assumption that is easily satisfied. In this chapter, the tradeoff between the nominal model and different types of uncertainty and the relationship between the model chosen and the synthesis result are discussed. Measurement data is used to identify the set of admissible models and a robust optimization approach is proposed to select the optimal admissible model for the purpose of synthesis.

6.1 Background and motivation

A fundamental concept related to safety specifications is robust control invariant sets (RCI). If an initial condition lies in an RCI, then there exist control inputs to guarantee that the trajectory of the system remains in the set indefinitely, despite all possible uncertainties. In addition to providing a safety certificate, set invariance can be used in a supervisory control structure, which guarantees safety with minimal intervention on top of an existing controller [31, 74].

Existing methods for computing invariant sets include, on the one hand, LMI-based Lyapunov type analysis [116], sum of squares programming [107], and occupation measures [117], which result in invariant sets with a smooth boundary, and on the other hand, Minkowski type methods [118-121], polytopic projection [31, 32] and linear programming [122], which result in polytopic invariant sets. Although polytopes are commonly used to represent invariant sets for discrete-time dynamical systems, the complexity of the polytope representing the invariant set grows quickly within iterative algorithms. To overcome this challenge, several techniques for the computation of

low-complexity robust invariant sets are proposed (see, e.g., [122-124]). In this chapter, inspired by the one-shot approach proposed in [122] for low-complexity invariant set computation for autonomous systems (i.e., systems without a control input), an iterative algorithm is proposed to compute an RCI with constant representation complexity where one can leverage the available control authority for enforcing invariance.

Depending on the control problem at hand, either maximal or minimal control invariant sets can be relevant. A maximal control invariant set can describe the region of attraction with limited control authority. It is formally defined in [125], and the definition of the maximum is in the set inclusion sense, that is, every control invariant set within a compact subset of the state space is a subset of the maximal control invariant set. The existence and uniqueness of the maximal control invariant set can be guaranteed with some mild assumptions. For linear discrete-time systems, the maximal control invariant set can be computed via polytopic projection [31, 126]. On the other hand, an mRCI describes how small a robust invariant set can be under disturbance and uncertainty. An mRCI is useful when the objective is to limit the deviations from a desired operating point. For instance, when assume-guarantee reasoning [115, 127] is adopted, an mRCI is useful since it minimizes the bound on the states of one part of the system, which is then treated as uncertainty bound in the synthesis of other parts of the system. However, in general, there does not exist a unique mRCI that is a subset of every RCI.

The computation of invariant sets depends on a model of the system, and uncertainty characterization is critical for the computation of RCI. Among the above-mentioned methods, while some can handle modeling uncertainty and exogenous disturbance, but they all assume that the model is given, including both the nominal model and the uncertainty characterization. However, the a priori uncertainty characterization might be too loose or too tight, leading to control designs that may fail to satisfy the specifications due to under-estimating the uncertainty, or designs that are too conservative due to over-estimating the uncertainty. In addition, if the environment or the dynamical system itself is changing, the assumption about the bound of uncertainty should be large enough to cover all the possible changes, which may cause the synthesized controllers to be unnecessarily conservative, or even infeasible.

In real engineering practice, the model is often the result of a system identification step. Even for physics-based models, the uncertainty is often characterized through experiments. The most

classical system identification method is the least square regression, including many extensions that incorporate various filtering structures [128]. Another class of identification methods are the set membership methods, which identify the set of admissible model parameters via set intersections [129]. Since the 1980s, control relevant identification has been studied, including H_∞ identification [130], generalized predictive control [131], and stochastic embedding [132]. However, the H_∞ identification and stochastic embedding approaches are for model identification in the frequency domain, and the generalized predictive control focuses on optimality rather than robustness. In terms of the identification of a model for an uncertain system that suits the need for correct-by-construction control synthesis, there is a gap to be filled. Sadraddini et al. provide an identification method that uses mixed integer programming to identify a piecewise linear model with a bound on the disturbance for formal synthesis [133]. Although with assumptions on the knowledge of Lipschitz constants on the system, the approach can provide safety guarantees also for unseen data, the system identification and control synthesis are done separately and the identified model is not necessarily "optimal" for the control synthesis task. Besides, the resulting mixed integer program is very large scale, making it unsuitable for onboard identification.

The main contribution of this chapter is a robust linear programming (LP) framework for approximating a minimal robust control invariant set while simultaneously selecting the optimal admissible uncertain model. An admissible model, which is not unique, is defined as a model that explains a finite measurement history. The novelty of the proposed framework lies in two aspects. First, it treats the uncertainty bound as part of the identification parameters, which leads to a non-unique model characterization and allows for trading off between different types of uncertainty. Second, the non-uniqueness of the model is exploited by the mRCI algorithm and an optimal model for computing an mRCI is selected concurrently. The proposed method is demonstrated with a lane keeping problem for road vehicles. The lateral dynamics of a vehicle is nonlinear, but typically it is approximated by a linear model, therefore modeling uncertainty is introduced. Moreover, the nominal model, as well as the modeling uncertainty, varies with road conditions, vehicle properties such as mass and tire properties, which might not be known exactly a priori. The objective of a lane keeping controller is to keep the vehicle inside the lane boundary, therefore an mRCI is useful for bounding the lateral deviation.

The remainder of the chapter is organized as follows. First, the system identification framework that identifies the set of admissible models for systems with uncertainty is presented in Section 6.2 and 6.3. Then a robust LP algorithm that computes an approximation of an mRCI by selecting a model from the set of admissible models is presented in Section 6.4. The whole process of approximating an mRCI from data is demonstrated on a lane keeping problem in Section 6.5 and finally, conclusion is drawn in Section 6.6.

Nomenclature

In this chapter, for two vectors $x, y \in \mathbb{R}^n$, the inequality is defined in the element-wise sense: $x \le y \Leftrightarrow y - x \in \mathbb{R}^n_{\ge 0}$. For a matrix A, A_i denotes its *i-th* row, A^j denotes its *j-th* column and A_{ij} denotes the entry on the *i-th* row, *j-th* column. $x(t_1 : t_2)$ denotes a sequence of vectors, labeled by time, starting from $t_1 \in \mathbb{Z}$ and ending at $t_2 \in \mathbb{Z}$. For simplicity, $\mathcal{P}(P, q)$ denotes the polyhedron $\{x |\ Px \le q\}$.

6.2 Linear parametrization with uncertainty

The following discrete-time linear model with uncertainty is considered:

$$x^+ = \hat{A}x + \hat{B}u + \hat{E}d + \tilde{A}x + \tilde{E}d + e, \tag{6.1}$$

where $x \in \mathbb{R}^n$ is the state of the system, x^+ is the state of the next sampling time, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input and $d \in \mathcal{D} \subseteq \mathbb{R}^l$ is the exogenous measured disturbance. $\hat{A}, \hat{B}, \hat{E}$ are the nominal model matrices, $\tilde{A}, \tilde{E}$ are the matrices for the multiplicative uncertainty and $e \in \mathbb{R}^n$ is the additive uncertainty. In fact, this is simply *n* uncertain linear parametrizations stacked together. Take the *i-th* dimension as an example, the linear parametrization appears as:

$$z_i = \varphi_i^T \hat{\theta}_i + \varphi_i^T \tilde{\theta}_i + e_i, \tag{6.2}$$

by taking

$$\begin{aligned} \varphi_i &= \begin{bmatrix} x^T & u^T & d^T \end{bmatrix}^T \in \mathbb{R}^{n+m+l}, z_i = x_i^+ \in \mathbb{R}, \\ \hat{\theta}_i &= \begin{bmatrix} \hat{A}_i & \hat{B}_i & \hat{E}_i \end{bmatrix}^T \in \mathbb{R}^{n+m+l}, \\ \tilde{\theta}_i &= \begin{bmatrix} \tilde{A}_i & \tilde{B}_i & \tilde{E}_i \end{bmatrix}^T \in \mathbb{R}^{n+m+l}. \end{aligned} \tag{6.3}$$

It is assumed that the uncertainties are bounded in hyper-boxes:

$$|\tilde{\theta}_i| \le \Omega_M^i, \quad |e_i| \le \Omega_A^i. \tag{6.4}$$

However, unlike most set membership approaches which assume a fixed bound on the uncertainty, the bounds Ω_M and Ω_A are not given and are included as part of the identification process. It is assumed that u and d are bounded in polytopes and the bound is known a priori:

$$\begin{aligned} d &\in \mathcal{D} = \{d \mid Gd \le g\}, \\ u &\in \mathcal{U} = \{u \mid Ru \le r\}. \end{aligned} \tag{6.5}$$

This assumption is satisfied by many engineering problems since the bound for u and d are often determined by system specifications or physics.

An uncertain linear model is determined by the value of $[\hat{\theta}, \Omega_M, \Omega_A]$, which contains the information of both the nominal model and the uncertainty characterization.

In many cases, the model to be identified has additional structure. For example, due to underlying physics, some of the model parameters may be known to be zero or some entries of the system matrices may be linearly dependent. In order to incorporate such structure, it is assumed that the model parameters are affinely parameterized by a hyperparameter π. That is, $\hat{\theta} = \hat{\mathbf{\Theta}}(\pi)$, $\Omega_M = \mathbf{\Omega_M}(\pi)$, and $\Omega_A = \mathbf{\Omega_A}(\pi)$, where the **bold font** is used to denote the known affine mapping from π to the model parameters, e.g., $\hat{A} = \hat{\mathbf{A}}(\pi)$. Since these mappings are affine, the overall parametrization of the model is also affine in π. Moreover, when no structural information is available, these mappings can be taken to be the trivial ones.

6.3 Admissible model for measurements

Given a sequence of measurement $x(1:T+1)$, the output and regressor for time step t is defined as

$$z_i(t) = x_i(t+1), \varphi_i(t) = [x(t)^T, u(t)^T, d(t)^T]^T. \tag{6.6}$$

A model $[\hat{\theta}, \Omega_M, \Omega_A]$ is called admissible if for $t = 1, 2, ..., T$,

$$\begin{aligned} &\exists e(t), \tilde{\theta}(t), s.t. |e(t)| \le \Omega_A, |\tilde{\theta}(t)| \le \Omega_M, \\ &z(t) = \left(\hat{\theta} + \tilde{\theta}(t)\right)^T \varphi(t) + e(t). \end{aligned} \tag{6.7}$$

In fact, for each time step, the measurement introduces a linear constraint to the model parameters:

$$\left|z(t)-\varphi(t)^T\hat{\boldsymbol{\Theta}}(\pi)\right| \le \left|\varphi(t)\right|^T \boldsymbol{\Omega}_{\mathbf{M}}(\pi)+\boldsymbol{\Omega}_{\mathbf{A}}(\pi). \tag{6.8}$$

Since the π-parametrization of $[\hat{\theta},\Omega_M,\Omega_A]$ is affine, the above condition is a linear inequality constraint on π. The set of admissible models is then parameterized by π constrained inside a polyhedron, with the following representation:

$$\Sigma=\left\{\pi|\ \forall t\in\mathbb{Z}_{1:N},|\ z(t)-\varphi(t)^T\hat{\boldsymbol{\Theta}}(\pi)|\le\left|\varphi(t)\right|^T\boldsymbol{\Omega}_{\mathbf{M}}(\pi)+\boldsymbol{\Omega}_{\mathbf{A}}(\pi)\right\}. \tag{6.9}$$

Figure 6.1 shows the comparison between the proposed uncertain model structure and linear regression. The dots represent the measurement data and the center purple line represents the nominal model. In addition to the nominal model, the uncertain model on the right introduces the bound on additive uncertainty, represented as the parallel red dashed lines, and the bound on multiplicative uncertainty, represented as the green radiating dashed lines. With additive and multiplicative uncertainty, the model on the right covers all data points and therefore is an admissible model.

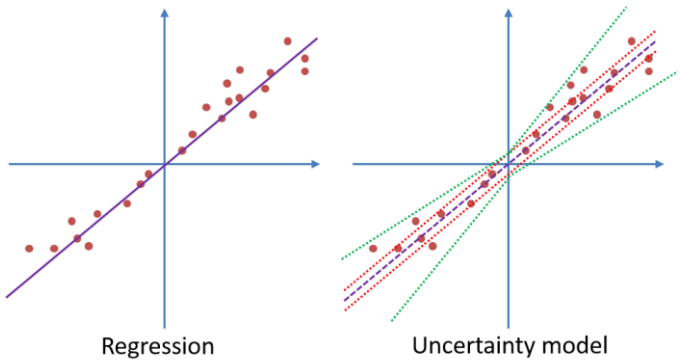

Figure 6.1 Comparison of regression and uncertainty models

If Σ is nonempty, then all models in Σ explain the measurement data. In fact, under mild assumptions, Σ is guaranteed to be non-empty.

Theorem 6.1: *If parametrization of additive uncertainty is full rank, i.e. the image of* $\boldsymbol{\Omega}_{\mathbf{A}}(\cdot)$ *under* π *spans* $\mathbb{R}^n$*, and the parametrization of* Ω_A *is decoupled from* $\hat{\theta}$*, then* Σ *is nonempty.*

Proof: For any $\hat{\theta}$*, since the parametrization of* $\hat{\theta}$ *and* Ω_A *are decoupled,* Ω_A *can be chosen independently. Let*

$$\Omega_A=\max_t\left|z(t)-\varphi(t)^T\hat{\theta}\right|. \tag{6.10}$$

Then the model is admissible for any Ω_M.

■

This shows that when Ω_A is sufficiently large, the model becomes admissible. However, a model with a large uncertainty bound may be useless in control synthesis.

In addition, there is a trade-off between different types of uncertainty, as shown in Figure 6.2. When the additive uncertainty bound is large, the bound on multiplicative uncertainty can be smaller, and vice versa. This is the direct result of (6.8).

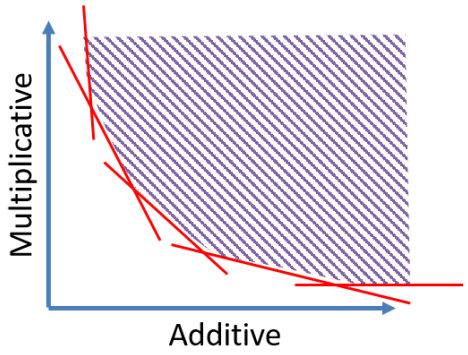

Figure 6.2 The tradeoff between uncertainty bounds

The set of admissible models Σ gives the domain from which the model should be selected. Among the admissible models, which one is "the best" depends on how this model is to be used. If the goal is to find a model with the least squared error, then the least squares regression gives the best model, with corresponding uncertainty characterization. However, since the goal is to compute an mRCI, the incorporation of model selection process into the mRCI computation, as shown in the next section, may result in a more desirable invariant set.

6.4 Robust LP algorithm for mRCI

6.4.1 One-step propagation

In this section, a robust LP algorithm that simultaneously selects the optimal model and computes the mRCI is proposed.

Definition 6.1: *A set* $\mathcal{S} \subseteq \mathbb{R}^n$ *is called robust control invariant for the system described by* (6.1), (6.4) *and* (6.5) *if there exists a control strategy* $\mu : \mathbb{R}^n \times \mathcal{D} \to \mathcal{U}$, *satisfying,* $\forall x \in \mathcal{S}, d \in \mathcal{D}, \mu(x,d) \in \mathcal{U}$ *and for all* $x \in \mathcal{S}$, *with* $u = \mu(x,d)$, $x^+ \in \mathcal{S}$ *under all possible uncertainty given by* (6.4).

As mentioned in Section 6.1, the existence of a minimal control invariant set in the set inclusion sense is not guaranteed. The definition of an mRCI in a weak set inclusion sense is given:

Definition 6.2: *A robust control invariant set $\mathcal{S}$ is a minimal robust control invariant set if $\nexists \mathcal{S}' \subsetneq \mathcal{S}, s.t.\ \mathcal{S}'$ is a robust control invariant set.*

However, even with this definition, finding an mRCI is non-trivial. Typically, one tries to find an (approximate) mRCI by minimizing a certain measure of size, such as volume [121, 123]. The mRCI algorithm proposed in this chapter computes a polytopic RCI that minimizes a linear objective function.

The proposed method draws inspiration from [122], where the author proposed a one-step LP approach to compute a robust invariant set for a system without control. The key idea is to fix the orientation of the separating hyperplanes defining a polytopic invariant set. However, the method in [122] can only deal with autonomous systems. For systems that are not open loop stable, the problem becomes infeasible. One can design a feedback controller to make the system autonomous, but there is no guarantee that the feedback input respects the input bounds for all states inside the RCI. Besides, the uncertainty is assumed to be purely additive in [122] and no measured disturbance is included. Borrowing the idea of fixing the hyperplane orientation, an iterative approach based on a robust LP is proposed.

The method starts by choosing a set of L hyperplanes with fixed orientation P_i and varying offset q_i, $i = 1,...,L$. Let $P = [P_1^T, P_2^T, ..., P_L^T]^T$, $q = [q_1, ..., q_L]^T$. Without loss of generality, assume $\|P_i\| = 1$. If $\mathcal{S} = \mathcal{P}(P, q)$ has a nonempty interior, then P_i is the normalized normal vector pointing outwards the corresponding separating hyperplane.

Assumption 6.1: *The hyperplanes are chosen such that $\{x |\ Px \le \mathbf{1}_L\}$ is a compact set, where $\mathbf{1}_L \in \mathbb{R}^L$ denotes the column vector consisting of all ones.*

Given a polytope $\mathcal{S} = \mathcal{P}(P, q)$, the following one-step propagation is considered which searches for $\mathcal{S}^+$ that contains all possible x^+:

$$\begin{aligned} &\min_{q^+} c^T q^+ \ s.t.\ \forall x \in \mathcal{P}(P, q), \forall d \in \mathcal{D}, \\ &\forall |e| \le \Omega_A, \forall |\tilde{\theta}| \le \Omega_M, \exists u \in \mathcal{U}, s.t. x^+ \in \mathcal{P}(P, q^+). \end{aligned} \tag{6.11}$$

The set $S^+ = \mathcal{P}(P, q^+)$ satisfies the following condition: for any $x \in \mathcal{S}$, $d \in \mathcal{D}$, there exists $u \in \mathcal{U}$ such that all possible x^+ under u is contained in $\mathcal{S}^+$. It is clear that if $\mathcal{S}^+ \subseteq \mathcal{S}$, $\mathcal{S}$ is control invariant. By minimizing $c^T q^+$, an $\mathcal{S}^+$ that is as small as possible is preferred.

Next, a few simplification steps are presented to make the one-step propagation solvable by convex optimization. First, as mentioned at the beginning of this section, there is no minimum RCI that is the subset of every RCI. Therefore the RCI obtained depends on a specific control strategy. For the linear discrete-time system discussed in this chapter, the following control structure is imposed:

$$u = K_{ff}^T d + K_{fb}^T x, \tag{6.12}$$

where K_{ff} and K_{fb} are constant matrices, representing the feedforward and feedback gain respectively.

Second, it is assumed that $\hat{B}$ is given so that the cross product terms between K_{ff}, K_{fb} and $\hat{B}$ vanish.

Third, the one-step propagation should be robust against the model uncertainty, i.e., $\mathcal{S}^+$ should contain all possible x^+ under the uncertain model. This is enforced by considering the worst case uncertainty, captured by the "for all" quantifiers for e and $\tilde{\theta}$ in (6.11). Since the uncertainty bounds are assumed to be hyper-boxes, the following is true:

$$\begin{aligned}
&\max_{|\tilde{A}| \le \Omega_{\tilde{A}}} P_i \tilde{A} x = \max_{|\tilde{A}| \le \Omega_{\tilde{A}}} Tr\left(|xP_i|\,|\tilde{A}|\right) = |P_i \Omega_{\tilde{A}}|\,|x|, \\
&\max_{|\tilde{E}| \le \Omega_{\tilde{E}}} P_i \tilde{E} d = \max_{|\tilde{E}| \le \Omega_{\tilde{E}}} Tr\left(|dP_i|\,|\tilde{E}|\right) = |P_i \Omega_{\tilde{E}}|\,|d|, \\
&\max_{|e| \le \Omega_A} P_i e = |P_i| \Omega_A,
\end{aligned} \tag{6.13}$$

where $\Omega_{\tilde{A}} \in \mathbb{R}^{n \times n}$ and $\Omega_{\tilde{E}} \in \mathbb{R}^{n \times l}$ are the bounds on $|\tilde{A}|$ and $|\tilde{E}|$ induced from Ω_M respectively. P_i is the *i-th* row of P. Note that the expressions in (6.13) are not yet linear in x and d due to the absolute value. Absolute value constraints can be converted to linear constraints using standard LP techniques but, for the sake of keeping the notation simple, the absolute value form is used for the remainder of the chapter. Now the one-step propagation problem is formulated as the following robust optimization problem:

$$\min_{K_{ff},K_{fb},\pi,q^+} c^T q^+ \ s.t.\ \pi \in \Sigma, \forall x \in \mathcal{P}(P,q), \forall d \in \mathcal{D},$$
$$P\left(\hat{\mathbf{A}}(\pi)x + \hat{B}\left(K_{ff}^T d + K_{fb}^T x\right) + \hat{\mathbf{E}}(\pi)d\right) + \left|P\mathbf{\Omega}_{\hat{\mathbf{A}}}(\pi)\right||x| + \left|P\mathbf{\Omega}_{\hat{\mathbf{E}}}(\pi)\right||d| + |P|\mathbf{\Omega}_{\mathbf{A}}(\pi) \le q^+, \quad (6.14)$$
$$K_{ff}^T d + K_{fb}^T x \in \mathcal{U}.$$

The above optimization simultaneously searches for 1) an admissible model, 2) a linear controller that satisfies the input bound constraint, and 3) $\mathcal{S}^+ = \mathcal{P}(P, q^+)$, the polytopic set containing all possible x^+ under the selected model and controller. It is a robust LP in the sense that the constraints have to be satisfied for all $x \in \mathcal{S}$ and all $d \in \mathcal{D}$. The robust LP is solved via dualization [134].

Lemma 6.1: *Consider the following robust LP problem:*

$$\begin{aligned} &\min_{\alpha} c^T \alpha \ s.t. \forall \beta \in \mathcal{P}(F, f), \\ &H_1^i \beta + \alpha^T H_2^i \beta + H_3^i \alpha \le h^i, i = 1,...M, \end{aligned} \quad (6.15)$$

where α is the decision variable, β is the uncertain variable, and $\mathcal{P}(F, f)$ is the bound for uncertainty. It is equivalent to an LP as follows:

$$\begin{aligned} &\min_{\alpha,\lambda} c^T \alpha \ s.t.\ H_3^i \alpha + \left(\lambda^i\right)^T f \le h^i, \\ &H_1^i + \alpha^T H_2^i = \left(\lambda^i\right)^T F, \ \lambda^i \ge 0, \ i = 1,...M. \end{aligned} \quad (6.16)$$

Proof: See Appendix P.

Observe that the one-step propagation in (6.14) is in the robust LP form of (6.15), by taking $\alpha = \left[K_{ff}, K_{fb}, q^+, \pi\right]$ and $\beta = \left[x, d\right]$. Therefore, by Lemma 6.1, it can be transformed to a standard LP and can be solved efficiently.

In some cases, the mRCI is required to be contained inside a set Γ, assumed to be a polytope:

$$\Gamma := \mathcal{P}(M, m). \quad (6.17)$$

This constraint can be easily enforced by appending *M* to *P*, and setting an upper bound on the entries of the offset *q* corresponding to *M* based on *m*. Suppose the original *P* contains N_1 hyperplanes and *M* contains N_2 hyperplanes. Simply let

$$\mathcal{S} := \mathcal{P}\left(\bar{P}, \bar{q}\right), \bar{P} = \left[P; M\right], \quad (6.18)$$

and add the following upper bound on $\bar{q}^+$:

$$\bar{q}^{+}_{N_1+1:N_1+N_2} \le m\,. \tag{6.19}$$

6.4.2 Iterative algorithm

As mentioned previously, if $\mathcal{S}^{+} \subseteq \mathcal{S}$, $\mathcal{S}$ is a robust control invariant set. With the one-step propagation solvable as an LP, there are two typical iterative methods that solve for invariant sets, the outside-in method and the inside-out method [31, 121, 126]. In the proposed formulation, the inside-out algorithm is used to solve for an RCI, and the outside-in algorithm is used to shrink a known RCI to a smaller size.

6.4.2.1 Inside-out algorithm

The inside-out algorithm starts with a small initial $\mathcal{S}$, iteratively solves for $\mathcal{S}^{+}$ with the one-step propagation and replace $\mathcal{S}$ with $\mathcal{S}^{+}$, until $\mathcal{S}^{+} \subseteq \mathcal{S}$ is satisfied.

Algorithm 6.1 Inside-out algorithm for mRCI

procedure RCI-IO($\Sigma, P, q^0, \mathcal{D}, \mathcal{U}, \epsilon$)
  $q \leftarrow q^0$
  do
    Find $\left[q^{+}, \pi, K_{ff}, K_{fb}\right]$ s.t. $\pi \in \Sigma$,
    $\forall x \in \mathcal{P}(P,q), \forall d \in \mathcal{D}, K_{ff}d + K_{fb}x \in \mathcal{U}$,
    $x^{+} \in \mathcal{P}(P, q^{+} - \epsilon \mathbf{1}_L)$
    $q \leftarrow q^{+}$
  while $q^{+} \le q + \epsilon \mathbf{1}_L$
  return $[q, \pi, K_{ff}, K_{fb}]$
end procedure

The algorithm is shown in Algorithm 6.1, where $0 < \epsilon << 1$ is a small constant that helps accelerate the convergence.

Proposition 6.1: *If Algorithm 6.1 terminates, $\mathcal{S} = \mathcal{P}(P,q)$ is an RCI.*

Proof: By construction, $\mathcal{S}^{+} = \mathcal{P}(P, q^{+} - \epsilon \mathbf{1}_L)$ contains all possible x^{+} with $x \in \mathcal{S}$, $d \in \mathcal{D}$, and since $q^{+} \le q + \epsilon \mathbf{1}_L$, $\mathcal{S}^{+} \subseteq \mathcal{S}$, therefore $\mathcal{S}$ is an RCI.

Remark 6.1: *With $\epsilon > 0$, the algorithm searches for an $\mathcal{S}^+$ slightly larger than that in* (6.14)*, so that ϵ tolerance is allowed for the termination condition $\mathcal{S}^+ \subseteq \mathcal{S}$, which accelerates the convergence of the algorithm.*

Remark 6.2: *π changes in every iteration, which allows the algorithm to choose a model based on $\mathcal{S}$ in each iteration. For a small $\mathcal{S}$, more uncertainty may be lumped into multiplicative terms since $|x|$ is relatively small; for a large $\mathcal{S}$, a smaller Ω_M may be preferred.*

6.4.2.2 Inside-out algorithm

On the other hand, if an initial RCI is known for some admissible model, the outside-in algorithm can further shrink the initial RCI with a convergence guarantee. The algorithm iteratively solves for $\mathcal{S}^+ \subseteq \mathcal{S}$, and replace $\mathcal{S}$ with $\mathcal{S}^+$ until $\mathcal{S}$ can no longer be shrunk.

Algorithm 6.2 Outside-in algorithm for mRCI

procedure RCI-OI($\Sigma, P, q^0, \mathcal{D}, \mathcal{U}, \epsilon$)
  $q \leftarrow q^0$
  do
    Find $\left[q^+, \pi, K_{ff}, K_{fb}\right]$ s.t. $\pi \in \Sigma$, $q^+ \le q$
    $\forall x \in \mathcal{P}(P,q), \forall d \in \mathcal{D}, K_{ff}d + K_{fb}x \in \mathcal{U}$,
    $x^+ \in \mathcal{P}(P,q^+)$
    $q \leftarrow q^+$
  while $\left\|q^+ - q\right\| \ge \epsilon$
  return $[q, \pi, K_{ff}, K_{fb}]$
end procedure

Algorithm 6.2 shows the outside-in algorithm, which is very similar to the inside-out algorithm except for two differences. First, the one-step propagation has an additional constraint $q^+ \le q$, which ensures that $S^+ \subseteq S$. Second, the termination condition is on the norm of the difference between q and q^+.

Theorem 6.1: *If for a certain admissible model π^0, a polytopic RCI $\mathcal{P}(P, q^0)$ is known, then the outside-in algorithm is guaranteed to converge.*

Before proving the above theorem, the following lemma needs to be proved:

Lemma 6.2: *For a compact polytope* $\mathcal{P}(P,q)$*, suppose one moves the i-th hyperplane from* q_i *to* $\overline{q_i}$ *, and leaves the rest unchanged, resulting in the following polytope* $\mathcal{P}(P,q')$ *, where* $q' = [q_1, ...\overline{q_i}, q_{i+1}, ...q_L]^T$ *. Then there exists a constant* c_i *such that if* $\overline{q_i} < c_i$*,* $\mathcal{P}(P,q') = \varnothing$ *.*

Proof: Since $\mathcal{P}(P,q)$ *is compact,* $f(x) = P_i x$ *is a continuous function and it always achieves its minimum value on a compact set. Let* $c_i = \min_{x \in \mathcal{P}(P,q)} P_i x$*. Obviously* $c_i \le q_i$*. Note that if* $\overline{q_i} \le q_i$*,* $\mathcal{P}(P,q') = \mathcal{P}(P,q) \cap \left\{ x |\ P_i x \le \overline{q_i} \right\}$*. Therefore when* $\overline{q_i} < c_i$*,* $\mathcal{P}(P,q') = \varnothing$ *.*

■

Lemma 6.3: *Let* $\mathcal{P}(P,q)$ *be a compact polytope. Define* $c_i = \min_{x \in \mathcal{P}(P,q)} P_i x$*. For any* $q' \le q$*, if the set* $\mathcal{P}(P,q')$ *is nonempty,* $q' \ge c$ *.*

Proof: The proof follows from Lemma 6.2 *and the fact that* $c_i' = \min_{x \in \mathcal{P}(P,q')} P_i x \ge c_i$ *.*

■

Now let us prove Theorem 6.1.

Proof: First it is shown that the one-step propagation is always feasible, and every q^+ *during the iteration leads to an RCI. For clarity, denote the offset q found in the i-th iteration as* q^i *. This is shown by induction. For the first iteration, by assumption,* $\mathcal{P}(P,q^0)$ *is an RCI, so* $q^1 = q^0$ *is a feasible solution for the one-step propagation in the first iteration. Since it is an LP, thus convex, the LP solver always finds a feasible solution. Assume at the n-th iteration,* $\mathcal{P}(P,q^{n-1})$ *is an RCI. Then*

$$\begin{aligned} &\exists K_{ff}, K_{fb}, \pi, q^n \ s.t. \ \pi \in \Sigma, \\ &\forall x \in \mathcal{P}(P,q^{n-1}), \forall d \in \mathcal{D}, |\tilde{\theta}| \le \mathbf{\Omega_M}(\pi), |e| \le \mathbf{\Omega_A}(\pi), \\ &P\left(\hat{\mathbf{A}}(\pi)x + \hat{B}\left(K_{ff}^T d + K_{fb}^T x\right) + \hat{\mathbf{E}}(\pi)d + \tilde{E}d + \tilde{A}x + e\right) \le q^n, \\ &K_{ff}^T d + K_{fb}^T x \in \mathcal{U}, q^n \le q^{n-1}. \end{aligned} \tag{6.20}$$

This implies that the one-step propagation is feasible at the n-th iteration. Consider the one-step propagation in the n+1-th iteration. The only difference between the robust LP in the n-th and

n+1-th iteration is that the uncertainty set of x *changes from* $\mathcal{P}(P,q^{n-1})$ *to* $\mathcal{P}(P,q^n)$. *Since* $\mathcal{P}(P,q^n)\subseteq\mathcal{P}(P,q^{n-1})$, *the uncertainty set for the n+1-th iteration is a subset of that in the n-th iteration, therefore* q^n *is still a solution to the one-step propagation in the n+1-th iteration, and the one-step propagation is still feasible. By induction, the one-step propagation in the outside-in algorithm is always feasible, and for all* q^n, $\mathcal{P}(P,q^n)$ *is always an RCI.*

Next, let $c_i=\min_{x\in\mathcal{P}(P,q^0)} P_i x$. *Since* $\forall n\in\mathbb{Z}_{\geq 0}, q^n\leq q^0$, *by* Assumption 6.1, $\mathcal{P}(P,q^n)$ is compact, *then by Lemma 6.3,* $\forall n\in\mathbb{Z}_{\geq 0}, q^n\geq c$. *Since* q *is monotonically decreasing, and lower bounded by* c, *by bounded convergence theorem,* $\{q^n\}$ *eventually converges.*

■

Table 6.1 Comparison of the iterative algorithms

	Initialization	Convergence
Inside-out	Arbitrary	Not guaranteed
Outside-in	Need an RCI to start with	Guaranteed

In conclusion, the inside-out algorithm solves for an RCI, and the outside-in algorithm shrinks the size of a known RCI. The comparison of the iterative algorithms is shown in Table 6.1.

6.5 Application on lane keeping of ground vehicle

In this section, an application on the vehicle lane keeping problem is presented. The measurement comes from simulation data from CarSim, a commercial software highly acknowledged by the auto industry that runs simulations with high-fidelity physics-based vehicle models. The CarSim model used for collecting data has 113 states. A linear model with four states is used to approximate this detailed model.

6.5.1 Model structure

The model to be identified for the lateral dynamics of a ground vehicle is called lateral-yaw model, or bicycle model, which has four states:

$$x=\left[y,v_y,\psi,r\right]^T, \tag{6.21}$$

where y is the lateral displacement from the lane center, v_y is the sideslip velocity, ψ is the heading angle with respect to the lane direction and r is the yaw rate. The model is linear, yet the

coefficient may change with the forward speed v_x, road condition and vehicle condition such as mass, inertia and tire properties. A linear discrete model with uncertainty is used to describe the dynamics. The input is the steering angle on the front axle δ_f; the measured disturbance is road curvature r_d.

In order to reduce the complexity of the uncertainty characterization, certain structures are imposed on the model based on the properties of the dynamics. The dynamics for $\dot{y}$ and $\dot{\psi}$ are essentially integrators, therefore no multiplicative uncertainty is put on these two dimensions. Similarly, since $\dot{v}_y$ and $\dot{r}$ do not depend on y and ψ, no multiplicative uncertainty is put on the corresponding entries of $\tilde{A}$. The influence of r_d follows a simple kinetic equation, therefore, $\tilde{E}$ is set to 0. Thus, the model is in the following form:

$$\begin{bmatrix} \dot{y} \\ \dot{v}_y \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \hat{\mathbf{A}}(\pi) \begin{bmatrix} y \\ v_y \\ \psi \\ r \end{bmatrix} + \hat{B}\delta_f + \hat{\mathbf{E}}(\pi) r_d + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \tilde{A}_{22} & 0 & \tilde{A}_{24} \\ 0 & 0 & 0 & 0 \\ 0 & \tilde{A}_{42} & 0 & \tilde{A}_{44} \end{bmatrix} \begin{bmatrix} y \\ v_y \\ \psi \\ r \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}. \tag{6.22}$$

The bounds on uncertainty are

$$\left[|\tilde{A}_{22}| \quad |\tilde{A}_{24}| \quad |\tilde{A}_{42}| \quad |\tilde{A}_{44}| \right]^T \le \mathbf{\Omega_M}(\pi), |e| \le \mathbf{\Omega_A}(\pi). \tag{6.23}$$

If one were to fully parameterize the nominal model, 20 independent variables are needed, which may not be necessary since $\hat{A}$ and $\hat{E}$ only have several strongly varying dimensions. Instead, it is assumed that $\hat{A}$ and $\hat{E}$ are linearly parameterized by a set of bases:

$$\hat{A} = \sum_{i=1}^{n_1} \pi_1^i \bar{A}_i, \; \hat{E} = \sum_{i=n_1+1}^{n_1+n_2} \pi_1^i \bar{E}_i, \tag{6.24}$$

where $\{\bar{A}_i\}$ and $\{\bar{E}_i\}$ are the bases for $\hat{A}$, $\hat{E}$, with cardinality n_1 and n_2 respectively.

Remark 6.3: *Multiple simulations are run under different scenarios to obtain multiple sets of the nominal model with least square regression, and then Principle Component Analysis (PCA) is applied on the models obtained to extract bases for* $\hat{A}$, $\hat{E}$.

The overall π parametrization appears as

$$\hat{A}=\sum_{i=1}^{n_1}\pi_1^i\bar{A}_i,\ \hat{E}=\sum_{i=n_1+1}^{n_1+n_2}\pi_1^i\bar{E}_i,\ \Omega_M=\left[\pi_2^1,\pi_2^2,\pi_2^3,\pi_2^4\right]^T,$$
$$\Omega_A=\left[\pi_3^1,\pi_3^2,\pi_3^3,\pi_3^4\right]^T,\ \pi=\left[\pi_1^T,\pi_2^T,\pi_3^T\right]^T. \tag{6.25}$$

The parametrization is indeed affine.

6.5.2 Preparation for mRCI

First, data is collected from CarSim, in which the vehicle is equipped with a simple LK controller with some input noise and follows a prescribed route. Then the data is used to formulate the set of admissible models Σ following the procedure in (6.8). Σ is a polyhedron of π. n_1 is selected to be 6, and n_2 is selected to be 4, so $\pi\in\mathbb{R}^{18}$.

The selection of the hyperplane orientation is not the focus of this chapter since it is not clear yet how to optimally select L hyperplanes. In [122], the author shows a 2-dimensional example, where the hyperplanes are selected so that $\mathcal{P}\left(P,\mathbf{1}_L\right)$ is an L-sided regular polygon. This is not applicable to state space with a higher dimension. There exist strategies for choosing hyperplane orientations in higher dimensional space, e.g. using quaternions [135, 136], but uniform sampling is in general inefficient for computing RCI since the shape of RCI is strongly influenced by the dynamic system. A particle simulation approach is used to generate the hyperplane orientation for the lane keeping problem.

To be specific, an uncertain model is identified from the experiment data, with the nominal model being the least square regression result and bound on additive uncertainty:

$$x^+=Ax+Bu+Ed+e,|e|\le\Omega_A. \tag{6.26}$$

Note that when fixing the nominal model to be the least square regression result, the minimum Ω_A can be uniquely identified. A saturated LQR is then designed with the nominal model of the system:

$$u=\begin{cases}Kx & ,|Kx|\le u_{\max}\\ sign\left(Kx\right)u_{\max} & ,|Kx|>u_{\max}\end{cases}, \tag{6.27}$$

where $sign(\cdot)$ is the sign function and $\mathcal{U}=\left[-u_{\max},u_{\max}\right]$ is the bound on input. Then M particles with random initial state are simulated following this control law. The disturbance and additive uncertainty are chosen in the following fashion: with probability λ, d and e are chosen such that

$\|x^+\|$ is maximized; with probability $1-\lambda$, d and e are uniformly sampled among the extreme values of d and e.

After T sampling times, the convex hull of the M particles $\mathcal{H}$ is computed and the orientations of the bounding hyperplanes of the convex hull are used for the mRCI computation:

$$P = \mathcal{H}.H\,. \tag{6.28}$$

6.5.3 Result

The result of mRCI computation with the model structure in (6.25) is shown.

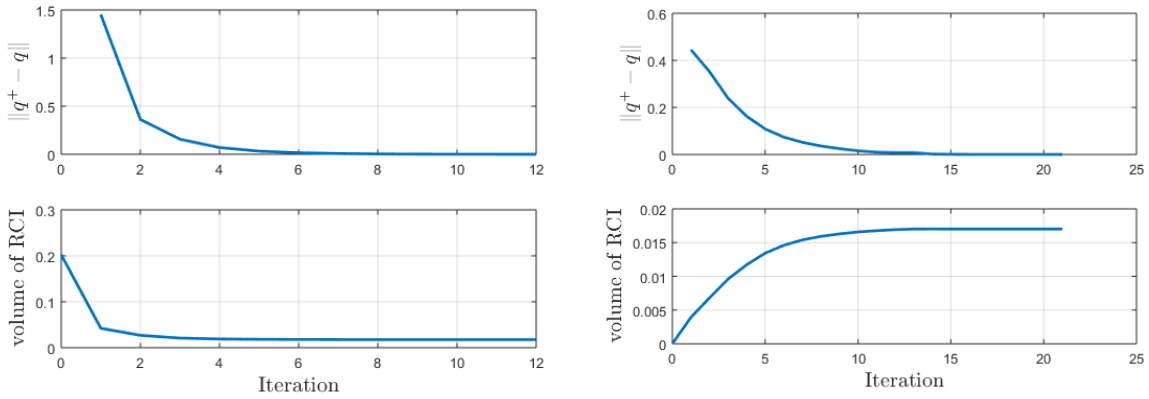

Figure 6.3 Convergence of the iterative algorithm

Figure 6.3 shows the convergence of the inside-out algorithm and outside-in algorithm. In fact, the inside-out and outside-in algorithms converge to the same mRCI in this case, but there is no guarantee that this will happen every time.

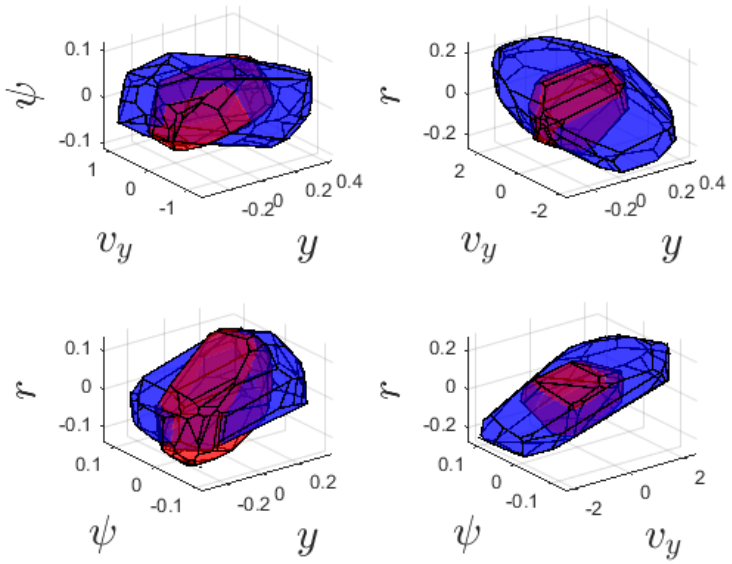

Figure 6.4 mRCI obtained with least square model and optimal nominal model

Figure 6.4 shows the computed RCI. For a benchmark, the least square regression result is used as the nominal model and the mRCI algorithm selects the optimal bound for multiplicative and additive uncertainty. In this case, π is simply $\left[\pi_2^T, \pi_3^T\right]^T$ in (6.25) and $\pi \in \mathbb{R}^8$. The RCI with the least square model is shown in blue, and the RCI under optimal nominal model is shown in red. The reduction in volume is more than 50%. Since the RCI is in $\mathbb{R}^4$, the plot shows the slices of the polytope by setting one dimension to 0.

Notice that the nominal model selected by the mRCI algorithm is only optimal for the purpose of computing an mRCI. In fact, under the least square model, the uncertainty bound found by the mRCI algorithm is as follows:

$$\begin{aligned} \Omega_M &= \begin{bmatrix} 0 & 0.0307 & 0 & 0.0022 \end{bmatrix}^T, \\ \Omega_A &= \begin{bmatrix} 0.0046 & 0.0283 & 0.0088 & 0.0116 \end{bmatrix}^T. \end{aligned} \tag{6.29}$$

For the optimal nominal model, the uncertainty bound becomes

$$\begin{aligned} \Omega_M &= \begin{bmatrix} 0 & 0.129 & 0 & 0.0112 \end{bmatrix}^T, \\ \Omega_A &= \begin{bmatrix} 0.0327 & 0.1643 & 0.0088 & 0.0236 \end{bmatrix}^T, \end{aligned} \tag{6.30}$$

which is significantly larger than the previous case. This shows that the least square model indeed fits the measurement better, but results in a larger mRCI. This again shows the necessity of performing model selection and mRCI computation simultaneously. Such co-optimization leads

to a model that better suits the mRCI computation. On the other hand, one does not need to worry about the correctness of the model, since it belongs to the admissible model set.

6.6 Conclusion

This chapter presents a novel data-driven algorithm to approximately compute a minimal robust invariant set by simultaneously selecting an admissible model and minimizing the size of the RCI. The algorithm has two steps: first, the set of all admissible models with uncertainty characterization is identified from the measurement data, then a robust LP is formulated to iteratively search for an mRCI. The robust LP-based algorithm is able to simultaneously select an optimal model, finding a good tradeoff between the nominal model and different types of uncertainties and minimize the size of an mRCI. A vehicle lane keeping example is used to demonstrate the method, and the result shows more than 50% reduction in the volume of the RCI computed compared to the benchmark.

Although the current algorithm provides a means to co-optimize invariant set size, the selection of invariance-inducing controllers, and the selection of models among the models consistent with the experimentally observed data, it does not necessarily provide invariance guarantees for unseen data in cases where the unseen data can reveal additional dynamics and can further shrink the admissible model set. Hence, another interesting direction is to extend the invariance guarantees to unseen data by incorporating a priori information on the dynamics as in [133].

Chapter 7 Experimental results

This chapter presents the experimental result of some of the methods proposed in this dissertation. The experiments were conducted on the OpenAV platform in Mcity, which is a test facility built by the University of Michigan. In particular, two algorithms were tested. First, a supervisory control structure with CBF as the supervisor for lane keeping was tested. A human driver was used to emulate the student controller, and the CBF's ability to guarantee safety was tested in a lane keeping scenario. Second, the data-driven algorithm for computing an mRCI was validated with experiments. Data were collected from experiments on the vehicle, then an mRCI was computed using the algorithm proposed in Chapter 6. The computed mRCI was then implemented on the same vehicle, showing that the state indeed stayed inside the computed mRCI.

The effectiveness of the proposed methods are demonstrated with the experiments, and the shortcomings of the theory and possible direction of future work to improve performance are discussed.

7.1 Hardware setup

The hardware platform is built based on a Lincoln MKZ sedan. Multiple sensors, such as Lidars, radars, cameras, and GPS units, are installed on the car, and the vehicle can be controlled by wire, including its throttle, braking, transmission shift, and steering.

Figure 7.1 The Mcity OpenAV platform

The sensors and actuators are accessible via the middleware called Polysync, designed specifically for autonomous vehicle operation [137]. Polysync is able to collect the sensor data packages, share among the nodes linked to the network, and send commands to actuation nodes such as steering and braking.

The major sensor used in the experiment is the Real Time Kinematic GPS. RTK is an advanced type of differential GPS that provides accurate measurement of position and velocity with the help of a base station. The model on board the MKZ is OXTS RT3003, which has the following specifications:

Table 7.1 Specifications of OXTS RT3003 RTK GPS

Position error	Velocity error	Pitch/roll error	Heading error
0.01m	0.05km/h	0.03deg	0.1deg

The test facility is at Mcity, which is built by the University of Michigan for the testing and validation of autonomous vehicles. Mcity is a closed test facility with urban roads as well as a segment of straight highway.

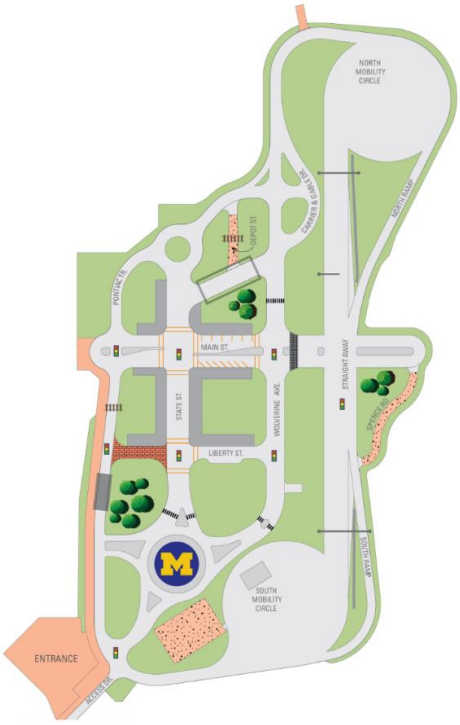

Figure 7.2 Map of the Mcity test facility

7.2 The experiment of CBF for lane keeping

The first experiment done on the MKZ platform was to test the performance of a CBF for lane keeping, where the safety specification is to bound the lateral deviation of the vehicle:

$$|y| \le y_{\max}, \tag{7.1}$$

where y is the lateral deviation from the lane center, set to $0.9m$. The CBF was constructed based on a linear lateral-yaw model of the test vehicle, which is a four-state linear model of the lateral dynamics of the vehicle:

$$\begin{aligned}\begin{bmatrix}\dot{y}\\ \dot{v}_y\\ \dot{\psi}\\ \dot{r}\end{bmatrix} &= \begin{bmatrix}0 & 1 & v_x & 0\\ 0 & -\dfrac{C_{\alpha f}+C_{\alpha r}}{mv_x} & 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{mv_x}-v_x\\ 0 & 0 & 0 & 1\\ 0 & \dfrac{bC_{\alpha r}-aC_{\alpha f}}{I_z v_x} & 0 & -\dfrac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z v_x}\end{bmatrix}\begin{bmatrix}y\\ v_y\\ \psi\\ r\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{C_{\alpha f}}{m}\\ 0\\ \dfrac{aC_{\alpha f}}{I_z}\end{bmatrix}\delta_f + \begin{bmatrix}0\\ 0\\ -1\\ 0\end{bmatrix} r_d \\ &= Ax + Bu + Ed.\end{aligned} \tag{7.2}$$

where the four states are the lateral deviation y, sideslip velocity of the chassis v_y, yaw angle ψ and the yaw rate r. The input u is the steering angle δ_f and the measured disturbance d is r_d. m and I_z are the mass and moment of inertia of the vehicle; $C_{\alpha f}$ and $C_{\alpha r}$ are the cornering stiffness of the two axles; a and b are the distance from vehicle CG (center of gravity) to the front and rear axles. v_x is the longitudinal speed; if it stays constant, the dynamics is linear. The parameters identified through testing are listed in Table 7.2.

Table 7.2 Model parameters of the test vehicle

m	I_z	a	b	$C_{\alpha f}$	$C_{\alpha f}$
$1800kg$	$3270kg\cdot m^2$	$1.65m$	$1.25m$	$140000N/rad$	$140000N/rad$

The CBF was constructed using SOS programming and followed the procedure described in Section 2.2.3. Bounds on all four states, the input, and the disturbance were enforced. Among the bounds, the bound on disturbance was an assumption on the environment; the bound on the input and states were the specifications to be satisfied. The parameters for the CBF construction are listed in Table 7.3.

Table 7.3 Parameters for the CBF construction

$y_{\max}$	$v_{y\max}$	$\psi_{\max}$	$r_{\max}$	$\delta_{f\max}$	$r_{d\max}$	v_x
$0.9m$	$1m/s$	$0.07rad$	$0.3rad/s$	$0.5rad/s$	$0.2rad/s$	$10m/s$

The order of the CBF was limited to quadratic. Moreover, since the LK problem is symmetric w.r.t. the origin, the CBF was in the following form:

$$b(x) = 1 - x^T Q x\,. \tag{7.3}$$

The following CBF condition was enforced:

$$\dot{b} + \gamma b = -2x^T Q(Ax + Bu + Ed) + \gamma b \geq 0\,, \tag{7.4}$$

which was a linear constraint for u, given x and d. The overall optimization solved on board was the following:

$$\begin{aligned} u = \arg\min_u\ & w_1 \|u - u_0\| + w_2 \|u - u_{pre}\| + w_3 s \quad s.t. \\ & -2x^T Q(Ax + Bu + Ed) + \gamma b + s \geq 0, \\ & u \in \mathcal{U},\ s \geq 0, \end{aligned} \tag{7.5}$$

where u_0 is the input from the student controller and u_{pre} is the input from the previous time instance. s is a relaxation term for the CBF condition, which is heavily penalized. The optimization was solved online with Gurobi C++ implementation [138].

The student controller was emulated by a human driver, which could behave like a well-designed student controller or a badly designed controller in the experiment. In order to keep the system safe, another set of driving-by-wire controllers was installed on the passenger seat using a gaming console, as shown in Figure 7.3. A safety driver would sit in the driver's seat watching over the experiment and would terminate the experiment should anything go wrong.

Figure 7.3 Human driver setup to implement the "student controller"

The command from the student controller (emulated by a human driver) u_0 was sent to the CBF; then the CBF would determine whether an intervention was needed via the optimization in (7.5) and send the command u to the drive-by-wire system of the test vehicle. One sample run is shown in Figure 7.4.

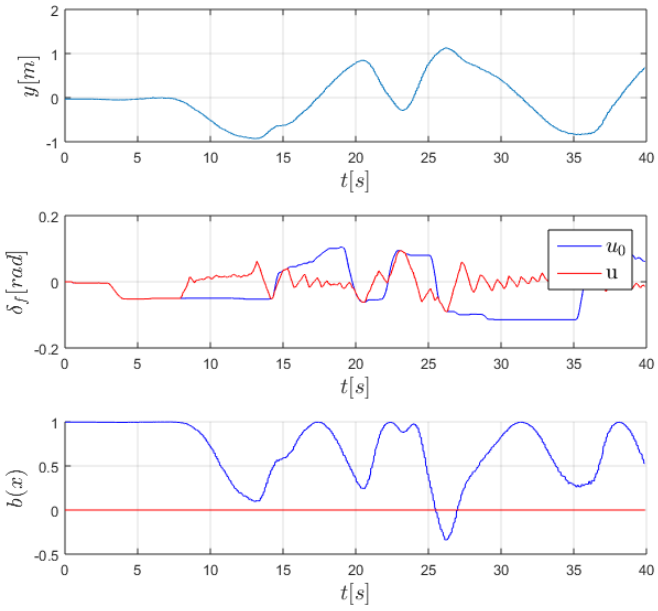

Figure 7.4 A sample run of the CBF experiments

During the experiment run shown in Figure 7.4, the human driver was trying to steer the vehicle out of the lane, and the CBF was able to keep the vehicle in the lane. u stayed the same as u_0 when no danger was detected and intervened when danger was detected. The experiment was not perfect, as shown in Figure 7.4: the CBF $b(x)$ dropped below 0 once, which was due to the unmodeled dynamics of the system, such as the delay and lag of the drive-by-wire system, and the unmodeled dynamics of the vehicle excited by the abrupt steering. The video of this experiment can be found at experiment video.

Multiple experimental runs were attempted, with some results worse than the one shown above. Typically, when u_0 changed direction quickly, or the vehicle was asked to bounce between the two lane boundaries, the CBF performed poorly. The major drawback of the current CBF theory is that it is not able to handle the change-rate limit of the input and the delay in the system. It assumes that u can change arbitrarily fast when the command is within the input limit. However, in experiments, when an abrupt change is commanded for δ_f, the servo motor cannot keep up with

the pace and the CBF may fail. In addition, a delay exists from the signal transmission on the CAN bus and the Ethernet.

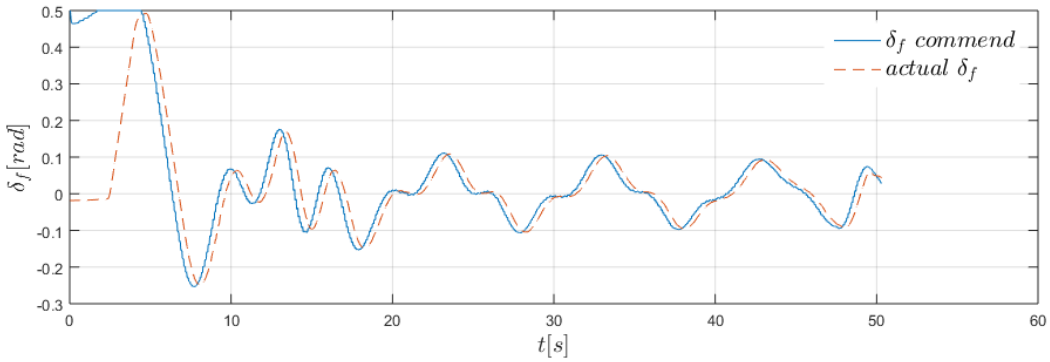

Figure 7.5 CBF Delay on the input

Figure 7.5 shows the comparison of the δ_f commanded and the actual δ_f. The delay is about 0.4 seconds.

In future work, how to incorporate the delay and the input rate bound into the construction of the CBF will be considered.

7.3 The experiment of a data-driven computation of an RCI

This section presents the experiment on the data-driven algorithm for computing a minimal robust control invariant set (mRCI), as proposed in Chapter 6. The same experiment platform—the MKZ in Mcity—was used.

First, data were collected by running the vehicle on a prescribed path. The path was a sinusoidal-like curve with a maximum curvature of around $0.02\,rad^{-1}$. The path was not perfectly sinusoidal since it was recorded by a human driver. A map representing the lane center was recorded in the form of trajectory points consisting of X, Y coordinates, heading angle and curvature.

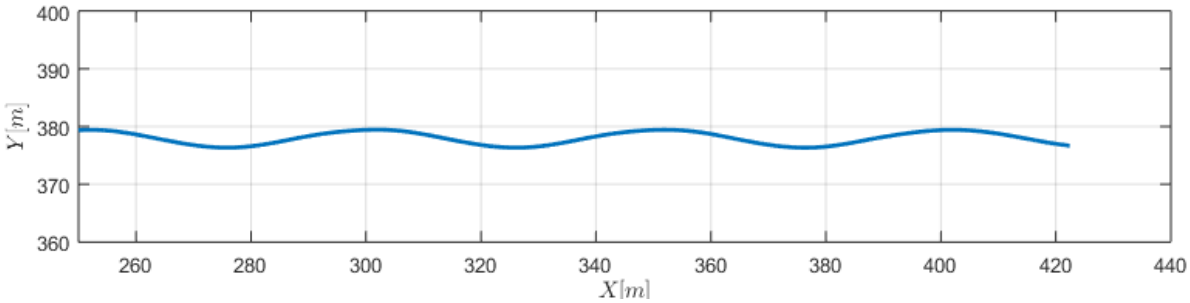

Figure 7.6 the route for collecting data

The vehicle was controlled by an LK controller designed with a preview control method, as introduced in [139]. Details of the preview control design are omitted here. The vehicle followed the prescribed path with a longitudinal velocity around 5m/s, which was enforced by a cruise control using PID. Among the states, v_y and r were directly measured with the RTK GPS, y and ψ were obtained by projecting the current GPS coordinates to the reference trajectory, that is, finding the closest point in the map file and computing the difference. With the projection, the curvature is also determined, along with r_d, which is the multiplication of longitudinal velocity and road curvature.

As presented in Section 6.5, the model for LK consists of four states, one input, and one measured disturbance:

$$x = \left[y, v_y, \psi, r \right]^T, u = \delta_f, d = r_d . \tag{7.6}$$

The measurements of x, u, d were recorded, But the sensor was not perfect, especially when measuring v_y and r. Therefore, a Kalman filter was used to filter out the noise. The Kalman filter was built based on a nominal model of the MKZ:

$$x^+ = A_d x + B_d u + E_d d , \tag{7.7}$$

which was obtained by discretizing the model shown in (7.2) with the parameters shown in Table 7.2. The output is the full state measurement, and the input is u and d. The process model is shown below:

$$\begin{aligned} x^+ &= A_d x + B_d u + E_d d + w, \\ z &= x^+ + v, \end{aligned} \tag{7.8}$$

where w and v are the process and measurement noise. The covariance for w and v were selected as

$$\begin{aligned} Q = \operatorname{cov}(w, w) &= \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix}, \\ R = \operatorname{cov}(v, v) &= \begin{bmatrix} 0.2 & 0 & 0.001 & 0 \\ 0 & 5 & 0 & 0 \\ 0.001 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix}. \end{aligned} \tag{7.9}$$

The covariance matrix for x was initialized with the identity matrix, and then the Kalman filter followed standard Kalman filter iteration as the experiment proceeded. See [140] and other textbooks for reference.

To this end, the model to be identified was actually the following sequence of blocks:

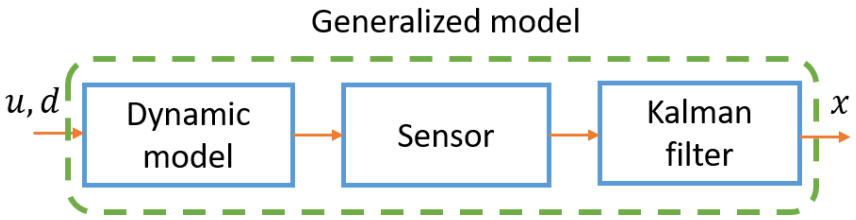

Figure 7.7 CBF Experiment result

The above sequence of blocks was viewed as a generalized model, as this was actually what the controller had to deal with in the experiment.

Since part of the LK dynamics is purely kinetic equations, additional structures were imposed on the model. Notice that the dynamics of y^+ and ψ^+ are determined purely by kinetics, which can be obtained by discretizing the following linear system:

$$\begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & v_x \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \psi \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r + \begin{bmatrix} 0 \\ -1 \end{bmatrix} r_d . \tag{7.10}$$

After discretizing the model in (7.10), the entries in the nominal models that correspond to the parameters from the kinetic model were fixed. As a result, all entries of $\hat{E}$ were now fixed, $\hat{A}$ was parameterized as follows:

$$\begin{aligned} &\hat{A} = \sum_{i=1}^{n_1} \pi_1^i \bar{A}_i + \hat{A}_0, \\ &\hat{A}_0 = \begin{bmatrix} 1 & 0 & v_x T_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 0 \end{bmatrix}, \bar{A}_i = \begin{bmatrix} 0 & * & 0 & * \\ * & * & * & * \\ 0 & 0 & 0 & 0 \\ * & * & * & * \end{bmatrix}. \end{aligned} \tag{7.11}$$

Remark 7.1: *The rest of the entries of $\hat{A}$ were parameterized by π, with linear bases generated via PCA, as described in Remark 6.3.*

Then following the procedure presented in Section 6.3, the set of admissible models Σ was identified. The model structure was the same as described in Section 6.5.1, and the procedure presented in Section 6.5.2 was used to generate the orientations of the hyperplanes of P. In

addition, the mRCI was enforced to be contained in $\{x \mid |x_1| \le 0.9\}$, which restricted the lateral deviation y to be less than 0.9m. The specifications for the mRCI computation are listed in Table 7.4.

Table 7.4 Setup of the computation of mRCI

Maximum steering	Maximum road curvature	Maximum deviation
0.2 rad	0.025 rad^{-1}	0.9 m

Since there was no known RCI to begin with, an mRCI was computed with the inside-out algorithm. At the first iteration, a high gain controller with a small RCI was obtained, but when put to the experiment, the MKZ steer-by-wire system was not able to keep up with the input command, and the controller performed poorly. Then the feedback gain was restricted and an RCI with moderate gains was obtained. The bound on the gain was set as follows:

$$\left|K_{fb}^{1}\right| \le 0.3, \ \left|K_{fb}^{3}\right| \le 1.2. \tag{7.12}$$

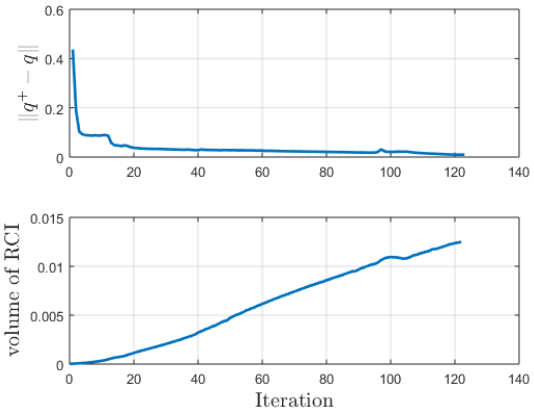

Figure 7.8 Inside-out algorithm to compute an mRCI

As shown in Figure 7.8, the algorithm terminated after 123 iterations, returning an RCI, an admissible model, and a control law consisting of feedback of x and feedforward of r_d:

$$u = [-0.30, -0.061, -0.85, -0.05]x + 0.03r_d. \tag{7.13}$$

Next the lane keeping controller was tested on a sinusoidal track with maximum allowed curvature with the above control law. The parameters of the desired path are listed below:

Table 7.5 Parameters of the sinusoidal desired path

Wavelength	Maximum road curvature	Amplitude
49 m	0.025 rad^{-1}	1.5 m

The same Kalman filter was used on board and the feedback was based on the filtered measurement of the states.

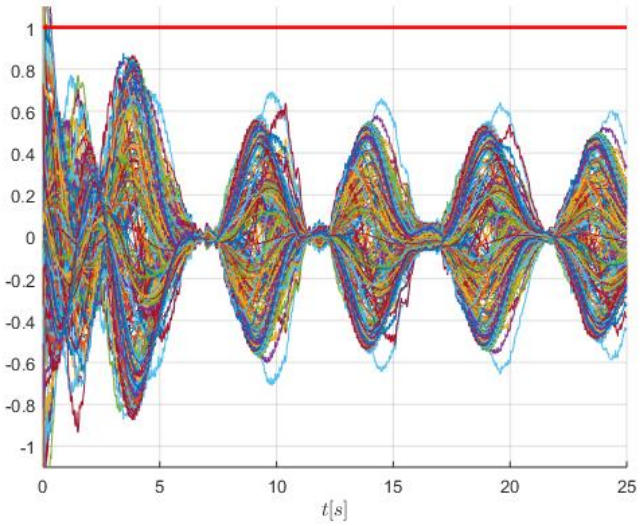

Figure 7.9 Relative position in the computed mRCI

Figure 7.9 plots $P_i x / q_i$ for every P_i in P. If x is contained inside $\mathcal{S} = \mathcal{P}(P, q)$, all plots should stay below 1. As shown in Figure 7.9, after converging from the initial condition, x stayed inside $\mathcal{S}$ during the experiment run.

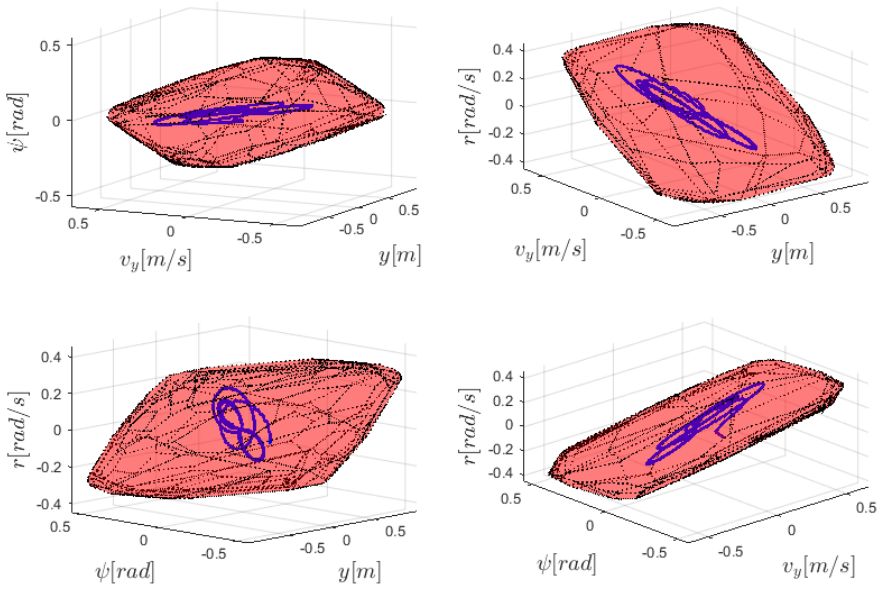

Figure 7.10 State trajectory and mRCI

Figure 7.10 shows the plot of state trajectory and the computed mRCI. Since the mRCI is in $\mathbb{R}^4$, both the state trajectory and the mRCI are projected into four 3-dimensional space. As shown in Figure 7.10, the state stayed within the computed mRCI.

Remark 7.2: *Experiments were conducted on paths with sharper (40m wavelength) and milder (80m wavelength) turns, and the state was contained in the mRCI in both cases. However, when the speed of the vehicle was changed, the performance deteriorated significantly. This result indicating that the proposed method is quite sensitive to* v_x. *Since* v_x *is an important parameter in the nominal model, the model may not be admissible once* v_x *is changed. One possible solution is to parameterize the model with* v_x, *but some modification is needed in the one-step propagation, as the model will no longer be linear.*

In Section 6.5.3, when the nominal model is fixed to be the least squares regression model, the RCI computed is larger in size compared to the RCI computed with the nominal model determined by the mRCI algorithm. However, when the undetermined entries of (7.11) were determined by linear regression with the measurements from the experiments, the inside-out algorithm did not terminate. The failure arose from the fact that the data from the real experiments were noisier than those from the CarSim simulation, meaning that a badly chosen nominal model may cause the iterative algorithm to fail.

7.4 Conclusion

Experiments were conducted on two methods—the CBF supervisory control method and the data-driven mRCI method. The CBF experiment shows the effect of the CBF on preserving safety, even with a student controller with bad intentions. However, a gap always exists between theory and reality, and the experiments on the CBF demonstrate some shortcomings of the theory. Because the CBF uses last-minute intervention to guarantee safety, which is drastic and sensitive to delay and to the limited change-rate of the input, the CBF must be improved upon.

On the other hand, the mRCI computed worked well with the imperfection of the model in the experiments. There may be three reasons that lead to the good performance. First, the model used for computing the mRCI is already an uncertain one that includes the system imperfection. Second, the mRCI is enforced with a linear controller with feedback and feedforward, which is simple yet

more robust against the delay. Third, bound on the feedback gain is enforced during the computation of the RCI, thus preventing the system from being high-gain.

One possible solution to the problem with the CBF method is to restrict the gain of the controller used to construct the CBF, as presented in Section 2.2.3, though it may lead to establishing a very conservative CBF which intervenes frequently. A CBF that intervenes even when safety is not endangered beats the purpose of using a CBF as a supervisor. Possible directions for improving the CBF method include:

1. *using an integrator to incorporate the change rate limit of the input;*
2. *including delay by considering the worst-case propagation of the system.*

In conclusion, one may achieve safety with a simple implementation, as is the case with the mRCI experiment, but it is restrictive, dictating the whole design of the controller. One may achieve safety with a more complicated structure, such as the supervisory structure in the CBF case, but it would require a better model and the theory is not yet mature.

Conclusion and future work

Conclusion

This dissertation considers the problem of correct-by-construction control synthesis and verification for dynamic systems. In order to guarantee safety for the system without being overly conservative, disturbance and model uncertainty must be accounted for properly. Several approaches are presented, tackling different types of disturbance and uncertainty, yet they all center around the idea of taking advantage of the structure of the problem, including the dynamic model, the disturbance structure, and the uncertainty characterization and reducing the level of conservativeness during the verification and control synthesis.

Chapter 3 presents the polar method, which solves the problem of moving obstacle avoidance for low-speed autonomous vehicles. The main issue addressed in this problem is the motion of moving obstacles, viewed as an exogenous disturbance. When no more information is available other than the maximum velocity of the moving obstacles, the control synthesis should prepare for all possible motion within the velocity limit. Safety is achieved with a supervisory control structure, where a CBF is constructed with the avoidable set solved with the polar method. The CBF supervisor would follow the student controller's command when no danger is detected, and intervene when the student controller's control input is leading to danger. The supervisor works with a student controller that may be designed with any existing methods, which really shows the method's ability to provide a guarantee of safety in a plug-and-play fashion. In the single obstacle case, the polar method is able to guarantee obstacle avoidance under all possible motion of the obstacle. In the multiple obstacles case, since collision is inevitable in some situations, the concept of responsibility of the collision is adopted to make the problem reasonable. The polar method is able to guarantee collision avoidance in the sense that the autonomous vehicle will never cause a collision for which it is responsible.

After discussing the design of a supervisor in Chapter 3, Chapter 4 switches gears to the other side of the coin—the design of a student controller. As powerful as the CBF may be, it focuses on safety rather than performance. When a student controller is not designed in a way that is compatible with the supervisor, intervention may be triggered frequently and the performance may be compromised. Therefore, a supervised learning based method is proposed that combines machine learning with CBF. First, trajectory optimization is used to generate a trajectory library consisting of trajectories that satisfy the CBF condition and will not trigger an intervention. These trajectories are then used to train a trajectory generator that will presumably inherit the good properties of the training set. The trajectory generator is implemented in the form of a Continuous Hold controller, while CBF guarantees safety on top of it. The proposed method is demonstrated on a truck lane keeping example, where the safety specification is bounding the lateral deviation and roll angle of the truck. Simulation is performed on TruckSim, a high-fidelity physics-based model. The result shows that the learning based controller is able to satisfy the safety specification without triggering the CBF supervisor, and the ride comfort of the truck is enhanced compared to that of the benchmark, which results in severe intervention from the CBF.

Next, Chapter 5 discusses a situation where the disturbance is not necessarily hostile. Previously, in the obstacle avoidance problem and the lane keeping problem, the disturbance was assumed hostile, that is, the control synthesis had to guarantee safety even for the worst-case realization of the disturbance. In the vehicle chassis control problem, however, the coexisting controllers designed by suppliers are not necessarily hostile, and considering the worst-case scenario would render the problem unsolvable. The main difficulty lies in the fact that the control algorithms are trade secrets of the suppliers, which means that it is impossible to ascertain the whole picture of the closed-loop system. To resolve this problem, the Lyapunov function is used as a measure of the influence of each controller, and dual decomposition is used to build a bridge of communication that preserves confidentiality. It is proved that the decentralized verification is equivalent to the centralized verification, thus obviating the need for the suppliers to expose their control algorithms.

Chapter 6 then focuses on modeling uncertainty. Most correct-by-construction methods, including the three methods previously mentioned in this dissertation, assume that the model of the dynamic system is known, including the nominal model and the uncertainty characterization. In practice, however, obtaining a model is not trivial. Moreover, if the model is the result of system

identification from measurement data, there are typically non-unique choices of the model parameters, and it is not clear how to choose one that would suit the control synthesis. A data-driven algorithm is proposed to approximate a minimal robust control invariant set. It begins by identifying the set of all admissible models, which include the nominal model and the uncertainty characterization. An admissible model is defined as a model that explains the measurement and is thus viewed as a "correct" model for control synthesis. Then a robust optimization based algorithm computes a minimal robust control invariant set while selecting the optimal admissible model that leads to the smallest invariant set. This work shows that when system identification and control synthesis are considered in an integrated way, the tradeoff between the nominal model and different types of modeling uncertainty may be dealt in a way that suits the control synthesis.

In conclusion, by utilizing tools in classic control theory, optimization and machine learning, several methods are proposed that give a safety guarantee to many interesting Cyber-Physical Systems. Disturbance and model uncertainty is an important issue for providing a safety guarantee to dynamic systems, and it is shown that when the structure of the problem is utilized in a smart way, conservativeness can be reduced. In some cases, a less conservative method means the difference between a solvable problem and a non-solvable one. Some of the proposed methods are tested on an autonomous vehicle experiment platform, demonstrating the capability of providing safety guarantees to a realistic system.

Future work

Some open problems and possible extensions of the proposed methods have not been addressed in this dissertation. Future work will include:

- *Improving the CBF theory:* The CBF theory has been found to be incomplete by the experiment due to a lack of consideration for delay and lag in the dynamics. Lag may be solved by increasing the order of the dynamic model; a good solution for the delay is not yet apparent.
- *More complicated specifications:* Most of the work covered by this dissertation concerns invariant set and safety constraints. Extensions of the proposed methods to more complicated specifications concerning liveness and reactiveness of the system are possible.

- *Extension to hybrid systems:* The systems considered in this dissertation consist of purely continuous states. The proposed methods may be extended to hybrid systems in which some states are discrete, such as traffic light signals.
- *Extension to stochastic settings:* In some cases, guaranteeing safety in a rigorous sense may be impossible. A lower bound of the probability of safety may be sought instead, such as the work on the Markov Decision Process [141, 142]. A potential direction of research is to extend the proposed methods to a stochastic setting.
- *Safe learning:* Although machine-learning techniques were used in Chapter 4, it was separated from the safety guarantee. Methods do exist that guarantee safety along with learning (viz. the Gaussian process approach [82, 83]). However, assuming the unknown function to be a Gaussian process is very general, and may hide some inner structure of the problem. A potential extension of the proposed methods is to combine learning with structural information of the system and provide a performance guarantee to a partially known system.

Appendices

Appendix A. Proof of Theorem 3.1

For any bounding half space $H^T x \le 1$ that satisfies (3.29), since $\mathcal{U}$ is a convex polytope, it can be rewritten as a convex combination of the vertices of $\mathcal{U}$:

$$\begin{aligned} &\forall d \in \mathcal{D}, \exists u = \sum \lambda_i u_i, u_i \in \mathcal{U}.V, \\ &\lambda_i \ge 0, \sum \lambda_i = 1 : H^T \left(Eu + Gd \right) \ge 0. \end{aligned} \tag{A.1}$$

Note that $\mathcal{U}.V$ is a set with finite elements, so $\max_{u_i \in \mathcal{U}.\mathrm{V}} \left\{ H^T E u_i \right\}$ exists, and

$$\max_{u_i \in \mathcal{U}.\mathrm{V}} \left\{ H^T E u_i \right\} \ge H^T E u \,. \tag{A.2}$$

Therefore, let $u_m = \arg\max_{u \in \mathcal{U}.\mathrm{V}} \left\{ H^T E u \right\}$,

$$H \in P_{Hs}^{u_m} \subseteq P_{Hs}. \tag{A.3}$$

(A.3) proves that P_{Hs} contains all linear functionals corresponding to bounding half spaces satisfying the boundary conditions.

■

Appendix B. Proof of Theorem 3.2

First, write $Conv\left(P_H \right)$ as

$$Conv\left(P_H \right) = Conv\left(P_{Hs} \cap X_{In}^{\Delta} \right) = Conv\left(\bigcup_{u \in \mathcal{U}.V} \left(P_{Hs}^{u} \cap X_{In}^{\Delta} \right) \right). \tag{B.1}$$

Recall the fact that for any union of polytopes $\bigcup_i P_i$, the vertices of its convex hull is a subset of the union of the vertices of all polytope components.

$$Conv\left(\bigcup_i P_i \right).V \subseteq \bigcup_i \left(P_i.V \right) \tag{B.2}$$

Therefore,

$$Conv(P_H).V \subseteq \bigcup_{u \in \mathcal{U}.V} \left(P_{Hs}^{u} \cap X_{In}^{\Delta}\right).V \,. \tag{B.3}$$

Recall the definition of a polar; it follows that

$$P_B.H = Conv(P_H).V \subseteq \bigcup_{u_i \in \mathcal{U}.V} \left(P_{Hs}^{u_i} \cap X_{In}^{\Delta}\right).V \,. \tag{B.4}$$

Therefore the bounding hyperplanes of P_B lie inside the intersection of X_{In}^{Δ} and P_{Hs}, and P_B is an avoidable set that contains X_{In}.

■

Appendix C. Proof of Theorem 3.3

First, it is shown that for all $H \notin P_H$, $H^T x = 1$ is not a valid bounding hyperplane for the avoidable set containing X_{In}. Then it is shown that for all $H \in P_H$, the half space $H^T x \le 1$ contains P_B. In other words, P_B is minimal.

For all $H \notin P_H$, by definition, $H \notin X_{In}^{\Delta} \vee H \notin P_{Hs}$. If $H \notin X_{In}^{\Delta}$, then from (3.24), $H^T x \le 1$ is a half space that does not contain X_d. If $H \notin P_{Hs}$, from (B.1), H does not satisfy the boundary conditions.

For all $H \in P_H$, $H \in P_H \Rightarrow H \in Conv(P_H)$, so

$$H = \sum \lambda_i H_i, \lambda_i \ge 0, \sum \lambda_i = 1, H_i \in Conv(P_H).V \,. \tag{C.1}$$

For all $x_0 \in P_B$, $H^T x_0 = \sum \lambda_i H_i{}^T x_0$. Since $x_0 \in P_B$, it follows that

$$\begin{aligned} &H_i{}^T x_0 \le 1, \forall H_i \in Conv(P_H).V \\ &\Rightarrow H^T x_0 = \sum \lambda_i H_i{}^T x_0 \le 1. \end{aligned} \tag{C.2}$$

(C.2) implies that adding $H^T x \le 1$ as a half space to P_B will not reduce any point from P_B, so P_B is minimal.

■

Appendix D. Proof of Theorem 3.4

Let P_H be the polytope consisting of all feasible bounding hyperplanes H for an avoidable set P_B. P_H may not contain the origin in its interior. Normalize the expression of its bounding hyperplanes to

$$\begin{cases} v_i^T H \le 1, i = 1, \dots N_v \\ \alpha_j^T H \le 0,\ j = 1, \dots N_\alpha \end{cases}, \tag{D.1}$$

where the first category denotes bounding hyperplanes corresponding to facets that do not contain the origin, and the second category denotes those facets that contain the origin. N_v and N_α are the number of linear constraints for each category.

Let $H_v = \{v_i\}$ and $H_\alpha = \{\alpha_i\}$ denote the two groups of bounding hyperplanes in (D.1). Since P_H is constructed by intersecting P_{Hs}, where every facet contains the origin, and X_{In}^{Δ}, where every facet does not contain the origin, it is clear that $H_v \subseteq X_{In}^{\Delta}.H = X_{In}.V$. Recall that shifting a polytope is equivalent to shifting all the vertices. It follows that

$$\overline{X_{In}}^{\Delta}.H = \{v + c, \forall v \in X_{In}.V\}. \tag{D.2}$$

Since the construction of P_{Hs} is an invariant of the shifting operation in the state space, P_{Hs} remains the same after shifting X_{In}. It then follows that $\overline{P_H}$ has the following bounding half spaces:

$$\begin{cases} (v + c)^T H \le 1, \forall v \in H_v \\ \alpha^T H \le 0, \forall \alpha \in H_\alpha \end{cases}. \tag{D.3}$$

For any vertex H_0 of $\overline{P_H}$, it is the intersection of *N* facets, where *N* is the dimension of the state space. Assume the linear equations corresponding to H_0 is

$$\begin{aligned} & AH_0 = a, \\ & A = \begin{bmatrix} v_{p1} & \dots & v_{pM} & \alpha_{q1} & \dots & \alpha_{qN-M} \end{bmatrix}^T, \\ & a = \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{bmatrix}^T. \end{aligned} \tag{D.4}$$

Since (D.4) has a unique solution H_0, A is invertible and $H_0 = A^{-1}a$. In addition, since $H_0 \in P_H$,

$$\begin{aligned} & \forall v \in H_v, v \ne v_{p1}, \dots v_{pM}, v^T H_0 < 1, \\ & \forall \alpha \in H_\alpha, \alpha \ne \alpha_{q1}, \dots \alpha_{qN-M}, \alpha^T H_0 < 0. \end{aligned} \tag{D.5}$$

Now consider the corresponding vertex $\overline{H_0}$ of $\overline{P_H}$. If it exists, it satisfies

$$\left(A+ac^T\right)\overline{H_0}=0. \tag{D.6}$$

If the matrix $\left(A+ac^T\right)$ is invertible and $\left(A+ac^T\right)^{-1}a$ satisfies all inequalities of $\overline{P_H}$, then $\overline{H_0}$ exists and is a vertex of $\overline{P_H}$.

From the matrix inversion lemma, if A and $1+c^TA^{-1}a$ are invertible, then $\left(A+ac^T\right)$ is invertible and

$$\left(A+ac^T\right)^{-1}=A^{-1}-A^{-1}a\left(1+c^TA^{-1}a\right)^{-1}c^TA^{-1}. \tag{D.7}$$

Clearly, A is invertible; $1+c^TA^{-1}a$ is a scalar. Since $\overline{X_{In}}=X_{In}+c$ contains the origin, $-c$ is inside X_{In}. H_0 is a vertex of P_H, so $H_0^T\left(-c\right)<1$. It follows that

$$1+c^TA^{-1}a=1+c^TH_0>0. \tag{D.8}$$

Therefore

$$\overline{H_0}=\left(1+c^TA^{-1}a\right)^{-1}A^{-1}a. \tag{D.9}$$

Next it is shown that $\overline{H_0}$ is inside $\overline{P_H}$. For all $v\in H_v, v\neq v_{p1},\ldots v_{pM}$:

$$\left(v+c\right)^T\overline{H_0}=\left(1+c^TA^{-1}a\right)^{-1}\left(v^TA^{-1}a+c^TA^{-1}a\right). \tag{D.10}$$

From (D.8), $v^TA^{-1}a=v^TH_0<1$. It follows that

$$\left(v+c\right)^T\overline{H_0}<\frac{1+c^TA^{-1}a}{1+c^TA^{-1}a}=1. \tag{D.11}$$

For all $\alpha\in H_\alpha, \alpha\neq\alpha_{q1},\ldots\alpha_{qN-M}$,

$$\alpha_i^T\overline{H_0}=\left(1+c^TA^{-1}a\right)^{-1}\alpha_i^TH_0<0. \tag{D.12}$$

Therefore $\overline{H_0}$ is a vertex of $\overline{P_H}$. Since the mapping from the vertices of P_H to the vertices of $\overline{P_H}$ is established, the only thing left to check is wether $\overline{P_B}=\overline{P_H}^{\Delta}=P_B+c$. Note that

$$\begin{aligned}
&\overline{H_0}^T\left(x+c\right)\le 1\\
&\Leftrightarrow\left(1+c^TA^{-1}a\right)^{-1}a^T\left(A^{-1}\right)^T\left(x+c\right)\le 1\\
&\Leftrightarrow\frac{a^T\left(A^{-1}\right)^Tx+a^T\left(A^{-1}\right)^Tc}{1+c^TA^{-1}a}\le 1\\
&\Leftrightarrow\frac{a^T\left(A^{-1}\right)^Tx}{1+c^TA^{-1}a}\le\frac{1}{1+c^TA^{-1}a}\\
&\Leftrightarrow H_0^Tx\le 1,
\end{aligned} \tag{D.13}$$

which proves that for each facet of P_B, a corresponding facet of $\overline{P_B}$ exists by shifting the facet with vector c, i.e., $\overline{P_B} = P_B + c$.

■

Appendix E. Proof of Theorem 3.5

$\bar{\Sigma}$ is actually a differential inclusion of the original system Σ_0, which implies:

$$\begin{aligned} &\forall x \in S_0, d \in \mathcal{D}_0, \forall u \in \mathcal{U}, \exists \bar{d} \in \overline{\mathcal{D}}, \\ &s.t.\ \dot{x} = f(x,u,d) = \bar{f}(x,u,\bar{d}). \end{aligned} \tag{E.1}$$

This means for any x,u,d in Σ_0, there exists $\bar{d} \in \overline{\mathcal{D}}$ that reproduce the state derivative in system $\bar{\Sigma}$ with the same state and input. Therefore, any avoidable set under dynamic $\bar{\Sigma}$ is also an avoidable set under dynamics Σ_0. It follows that P is also an avoidable set for Σ_0.

■

Appendix F. Proof of Theorem 3.6

Only the case when $\theta > 0$ is proved. The case when $\theta > 0$ follows the same reasoning. For any point x at the boundary of P, suppose $\theta > 0$, then $H_4 \geq 0$. P is an avoidable set w.r.t. the dynamic system $\tilde{\Sigma}$. Therefore,

$$\forall \bar{d} \in \overline{\mathcal{D}}, \exists u \in \mathcal{U} : H^T\left(E_1 u + G_1 \bar{d}\right) \geq 0. \tag{F.1}$$

Since

$$\begin{aligned} &H^T\left(E_1 u + G_1 \bar{d} + k\right) \\ &= H^T\left(E_1 u + G_1 \bar{d}\right) + \frac{H_4 \sin\theta}{\Delta X^2 + \Delta Y^2} \\ &\geq H^T\left(E_1 u + G_1 \bar{d}\right) \geq 0. \end{aligned} \tag{F.2}$$

It follows that P is also an avoidable set w.r.t. $\bar{\Sigma}$.

■

Appendix G. Derivation of (3.43)

From the definition of the barrier function in (3.41), the derivative of $B(x)$ is

$$\dot{B}(x) = \frac{\partial B}{\partial b}\frac{\partial b}{\partial x}\dot{x}. \tag{G.1}$$

Assuming that $\dot{x}$ is constant within the sampling time T_s,

$$\frac{d\left(\frac{\partial b}{\partial x}\dot{x}\right)}{dt} = \frac{d\left(H^T\dot{x}\right)}{dt} = 0. \tag{G.2}$$

It follows that

$$B^{(n)}(x)=\frac{\partial^n B}{\partial b^n}\left(H^T\dot{x}\right)^n. \tag{G.3}$$

For this certain function, $\frac{\partial^n B}{\partial b^n}$ has an explicit expression:

$$\frac{\partial^n B}{\partial b^n}=\frac{(-1)^n(n-1)!\left[(1+b)^n-b^n\right]}{(1+b)^n b^n}. \tag{G.4}$$

Given a sampling time T_s, the barrier function at the next time step can be calculated using the Taylor expansion:

$$B(t+T_s)=B(t)+\sum_{i=1}^{\infty}\frac{1}{n!}B^{(i)}(t)T_s{}^i=B(t)+\sum_{i=1}^{\infty}\frac{1}{n}\frac{(-1)^i\left[(1+b)^i-b^i\right]}{(1+b)^i b^i}\left(H^T\dot{x}\right)^i T_s{}^i. \tag{G.5}$$

It follows that

$$B(t+T_s)-B(t)\le\sum_{i=1}^{\infty}\left(\frac{-H^T\dot{x}T_s}{b}\right)^i=\frac{-H^T\dot{x}T_s}{b+H^T\dot{x}T_s}. \tag{G.6}$$

Recall (3.42),

$$\frac{-\frac{H^T\dot{x}T_s}{b}}{1+\frac{H^T\dot{x}T_s}{b}}\le\frac{\gamma}{B}T_s\Rightarrow H^T\dot{x}\ge-\frac{\gamma b}{\gamma T_s+B}. \tag{G.7}$$

■

Appendix H. Simulation setup in Section 3.6

The parameters for simulations are listed in Table H.1.

Table H.1 Simulation Parameters

Parameter	Value	Meaning
$v_{d\max}$	$1.2m/s$	Maximum obstacle speed
$r_{\max}$	$3.4rad/s$	Maximum yaw rate
$a_{\max}$	$4m/s^2$	Maximum acceleration
$v_{\max}$	$2m/s$	Maximum vehicle speed
R_v	0.5m	Radius of AV
R_p	0.3m	Radius of pedestrians

T_s	0.05s	Sampling time
μ	0.7	Friction coefficient
σ_{ax}, σ_{ay}	1	Standard deviation of pedestrian acceleration on X and Y direction

The simulations are conducted using Matlab. The toolboxes used are shown in Table H.2.

Table H.2 Toolboxes Used

Polytope Calculation	Multi-Parametric Toolbox 3
Mixed Integer Programming	Gurobi 6.0.4
Hamilton Jacobi Calculation	Level Set Toolbox 1.1.1

Appendix I. MPC design in Section 3.6

The greedy MPC navigates the vehicle to the destination without any knowledge of the pedestrians. The nonlinear unicycle model of the vehicle is linearized for MPC.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{v} \\ \dot{\bar{\psi}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cos(\psi_0) & -v\sin(\psi_0) \\ 0 & 0 & \sin(\psi_0) & v\cos(\psi_0) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ v \\ \bar{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ a \\ r \end{bmatrix}, \tag{I.1}$$

where

$$\psi_0 = \psi(t_0),\ \bar{\psi} = \psi - \psi_0. \tag{I.2}$$

This linearized model is then discretized and used for the MPC, with prediction horizon $N_{pred} = 5$ and control horizon $N_{con} = 1$.

Appendix J. Potential field controller design in Section 3.6

The potential field controller is adapted from the method proposed by Shimoda et al. in [66] and tune the parameter to enhance its performance. The potential field is built on the trajectory space, which is the Cartesian product of velocity and yaw rate (different from the original trajectory space in [66] because of the different inputs of the dynamic model). The potential field function used is

$$f(v,r) = K_g\left(r - r_{des}\right)^2 + K_{v1}\left(v - v_{des}\right)^{k_{v2}} + \sum_{i=1}^{N_{ped}} \frac{K_o\left(K_{ov}v + 1\right)}{\left(K_{oa}A_d^{\,i} + 1\right)\exp\left(\frac{-\left(r - r_o^i\right)^2}{2\sigma^i} - K_{od}d^i / v^2\right)}, \tag{J.1}$$

where A_d, r_o, d, and σ are defined in the following

$$\begin{aligned} A_d &= \left|\arctan\frac{Y_{des} - Y}{X_{des} - X} - \arctan\frac{Y_d - Y}{X_d - X}\right|, \\ r_o &= \frac{1}{2}\left(r_1 + r_2\right), \\ d &= \sqrt{\left(X_d - X\right)^2 + \left(Y_d - Y\right)^2} - R_v - R_p, \\ \sigma &= \left|r_2 - r_1\right|. \end{aligned} \tag{J.2}$$

r_1 and r_2 are the maximum and minimum yaw rate that will lead to a collision, as demonstrated in Figure J.1. See [66] for more details.

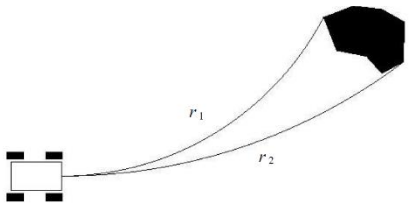

Figure J.1 Maximum and minimum yaw rates leading to collision (original figure in [66])

The parameters for the potential field method are listed in Table J.1.

Table J.1 Parameter of the potential field controller

K_g	5
K_{v1}	15
K_{v2}	2
K_o	100
K_{oa}	0.1
K_{ov}	0.1
K_{od}	2

Appendix K. Hamilton Jacobi controller design

The control problem is formed and solved as a pursue-evade game. First, a reachability set for a single obstacle is solved while restricting the vehicle's input to only braking, then apply this reachability set to multiple obstacles (the same "trick" used in the polar method). It should be noted

that the coordinate for relative dynamics in the Hamilton Jacobi method differs from that for the polar algorithm.

$$x=\begin{bmatrix}\Delta X_L\\ \Delta Y_L\\ v\end{bmatrix}=\begin{bmatrix}\Delta X\cos\psi+\Delta Y\sin\psi\\ -\Delta X\sin\psi+\Delta Y\cos\psi\\ v\end{bmatrix}, \tag{K.1}$$

where ΔX_L and ΔY_L are the relative position of the obstacle in the local frame attached to the vehicle. Theoretically, the three states in (K.1) are sufficient for describing the relative dynamics between the AV and the obstacle. The reason for using four states in the polar algorithm is that the model shown in (3.7) is more similar to double integrators, which makes the simplification in the polar algorithm easier. The dynamic equation for the states in (K.1) is

$$\dot{x}=\begin{bmatrix}-v+\Delta Y_L r+v_{Ldx}\\ -\Delta X_L r+v_{Ldy}\\ a\end{bmatrix}, \tag{K.2}$$

where v_{Ldx} and v_{Ldy} are the longitudinal and lateral projection of the obstacle's speed in the local coordinates. The dynamics are then reversed in time to calculate the backward reachable set.

The value function is defined as

$$\varphi(x,\mathrm{t})=\min_{a\in\mathcal{A}}\max_{b\in\mathcal{B}}\int_{t_0}^{t}0dt+\mathcal{G}\left(x(t)\right), \tag{K.3}$$

and $\mathcal{G}(x)$ satisfies:

$$\mathcal{G}(x)\begin{cases}<0,\ \Delta X_L{}^2+\Delta Y_L{}^2<\left(R_v+R_p\right)^2\\ =0,\ \Delta X_L{}^2+\Delta Y_L{}^2=\left(R_v+R_p\right)^2\\ >0,\ \Delta X_L{}^2+\Delta Y_L{}^2>\left(R_v+R_p\right)^2\end{cases}. \tag{K.4}$$

The Hamiltonian of the reversed dynamics allowing only braking is solved analytically:

$$\begin{aligned}H&=\min_{b\in\mathcal{B}}\max_{a\in\mathcal{A}}p^T f\left(x,t,a(x,t),b(x,t)\right)\\ &=p_1 v+k\left(p_3,v\right)-\sqrt{\frac{p_1{}^2+p_2{}^2}{2}}v_{d\max},\\ k\left(p_3,v\right)&=\begin{cases}\min\left\{0,p_3 a_{\max}\right\}, & v=0\\ -\left|p_3 a_{\max}\right|, & 0<v<v_{\max}\\ \min\left\{0,-p_3 a_{\max}\right\}, & v=v_{\max}\end{cases},\end{aligned} \tag{K.5}$$

where $a \in \mathcal{A}$ and $b \in \mathcal{B}$ are the strategy for the vehicle and obstacle, respectively. $p = \frac{\partial \phi}{\partial x}$ is the conjugate momenta; $k(p_3, v)$ ensures that the vehicle speed is within the limit. To enforce the responsibility rules presented in Section 3.2.2,

$$\left(\Delta X_L \le 0 \vee v = 0\right) \rightarrow H = 0\,. \tag{K.6}$$

The final time is chosen as $T = 10s$. The calculated zero level set is shown in Figure K.1.

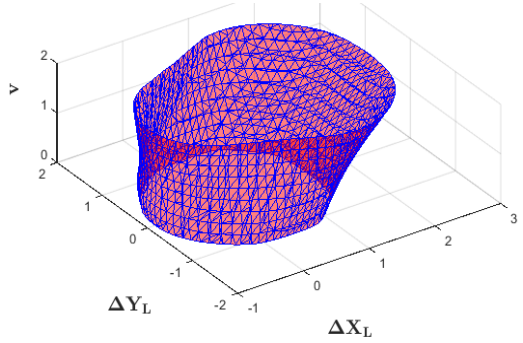

Figure K.1 Hamilton Jacobi reachability set

The result is calculated numerically and stored in a look-up table. In the implementation, local regression is performed to obtain the value and the gradient of φ. Then the following constraint is enforced:

$$\nabla_{\varphi}(x)^T \; f(x,u,d) \ge -\alpha\varphi(x), \tag{K.7}$$

where ∇_{φ} is the gradient at that point and α is a positive constant. For more details, please refer to [143] and [33].

Appendix L. Zero dynamics of the truck lateral dynamics in Section 4.2

To show the zero dynamics the following state transformation is used that renders a new choice of states:

$$\chi = Tx = \begin{bmatrix} z & \dot{z} & \sigma^T \end{bmatrix}^T, \tag{L.1}$$

where T is a full rank matrix:

$$T=\begin{bmatrix} 1 & 0 & v_x T_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & v_x & v_x T_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -B^4 & 0 & B^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -B^6 & 0 & 0 & 0 & B^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -B^8 & 0 & 0 & 0 & 0 & 0 & B^2 \end{bmatrix}, \tag{L.2}$$

and B^i is the *i-th* entry of B. The transformation is linear and full rank. The dynamics under χ is

$$\dot{\chi}=\overline{A}\chi+\overline{B}u+\overline{E}r_d, \tag{L.3}$$

Moreover,

$$\overline{B}=TB=\begin{bmatrix}0 & CAB & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T, \tag{L.4}$$

where $C=\begin{bmatrix}1 & 0 & T_0 v_x & \mathbf{0}_{1\times 5}\end{bmatrix}$. So the dynamics under χ can be written as

$$\dot{\chi}=\begin{bmatrix}\dot{z}\\ \ddot{z}\\ \dot{\sigma}\end{bmatrix}=\begin{bmatrix}\dot{z}\\ CA^2x\\ \Gamma\left(z,\dot{z},\sigma\right)\end{bmatrix}+\begin{bmatrix}0\\ CAB\\ 0\end{bmatrix}u+\begin{bmatrix}0\\ CAE_1\\ 0\end{bmatrix}r_d, \tag{L.5}$$

where $\Gamma\left(z,\dot{z},\sigma\right)$ has exponentially stable zero dynamics, meaning $\dot{\sigma}=\Gamma\left(0,0,\sigma\right)$ is exponentially stable.

Appendix M. Analysis of continuous hold controller in Chapter 4

M.1 Continuous hold controller for systems with disturbance

Systems of the following form are considered:

$$\dot{x}=f\left(x,u,d\right), \tag{M.1}$$

where x, u and d are the state, the input, and the measured disturbance respectively.

Remark M.1: *The result of the CH controller is applicable to general nonlinear dynamic model, the control-affine nonlinear model in* (4.1) *(assuming* $d_2=0$*) is a special case.*

The following assumptions about the system dynamics are made.

Assumption M.1: $f:\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^p\to\mathbb{R}^n$ *is locally Lipschitz continuous in* x, u, *and* d.

Assumption M.2: *$\exists \mathcal{D} \subseteq \mathbb{R}^p$ and mapping $\eta_x : \mathcal{D} \to \mathbb{R}^n, \eta_u : \mathcal{D} \to \mathbb{R}^p$, Lipschitz continuous, such that $f\left(\eta_x(d), \eta_u(d), d\right) = 0$, i.e., for every exogenous disturbance $d \in \mathcal{D}$, there exist two mappings η_x and η_u that map any $d \in \mathcal{D}$ to a unique equilibrium point and a unique input that maintains the equilibrium.*

Remark M.2: *There may be non-unique equilibrium points of the system due to the cyclic coordinates, i.e., states that do not affect the dynamics, see [144] for detail. Therefore, a function η_x is needed to select a single equilibrium point given d. Assumption 4.3 gives a possible definition of η_x.*

Assumption M.3: *$\forall d \in \mathcal{D}$, there is an open ball about the origin, $B_d \subset \mathbb{R}^n$, a positive-definite, locally Lipschitz-continuous function $V_d : B_d \to \mathbb{R}$, and constants $0 \le \alpha_1 \le \alpha_2$ such that $\forall x \in B_d + \eta_x(d)$,*

$$\begin{aligned} &\bar{x} = x - \eta_x(d), \\ &\alpha_1 \bar{x}^T \bar{x} \le V_d(\bar{x}) \le \alpha_2 \bar{x}^T \bar{x}. \end{aligned} \tag{M.2}$$

Assumption M.4: *$\exists \mathcal{S} \subseteq \mathbb{R}^n$, compact, such that $\forall d \in \mathcal{D}, \eta_x(d) \in \mathcal{S}$. There exists CBF $b(x)$, such that $\forall x \notin S, b(x) \le 0$, $\forall d \in \mathcal{D}$, $b\left(\eta_x(d)\right) > 0$. Moreover, $\forall \xi \in \mathcal{S}, \forall d \in \mathcal{D}$, $\exists u_\xi^d : \left[0, T_p\right] \to \mathbb{R}^m$ and a corresponding state trajectory $\varphi_\xi^d : \left[0, T_p\right] \to \mathbb{R}^n$ satisfying*

$$\begin{aligned} &V_d\left(\varphi_\xi^d(T_p) - \eta_x(d)\right) \le c_1 V_d\left(\varphi_\xi^d(0) - \eta_x(d)\right), \\ &\frac{db}{dx}\Big|_{\varphi_\xi^d(t)} \dot{\varphi}_\xi^d(t) + \kappa b\left(\varphi_\xi^d(t)\right) \ge 0, \\ &\lim_{\xi \to \eta_x(d)} \varphi_\xi^d(t) = \eta_x(d), \lim_{\xi \to \eta_x(d)} u_\xi^d(t) \equiv \eta_u(d), \end{aligned} \tag{M.3}$$

where $\kappa > 0, 1 > c_1 > 0$ are predefined constants.

A CH controller keeps a timer $\hat{t}$ that is reset to 0 when the triggering event happens and the desired trajectory is updated, then keep flowing as the trajectory is being executed. The update can happen when the trajectory is executed to the end, i.e. $\hat{t} = T_p$, or when an interruption is detected. A possible interruption includes a change in d or an unexpected disturbance that makes the tracking error too large. The CH input is

$$u^{CH}(\hat{t},x,d)=u_{\xi}^{d}(\hat{t})+u^{fb}\left(x\left(\hat{t}\right),\varphi_{\xi}^{d}\left(\hat{t}\right)\right), \tag{M.4}$$

where ξ is the initial state when $\hat{t}=0$, and $u^{fb}:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}^m$ is a feedback controller that tracks φ_{ξ}^{d}.

The closed-loop system under CH feedback is then

$$\dot{x}=f^{CH}\left(\hat{t},x,d\right):=f\left(x,u_{\xi}^{d}(\hat{t})+u^{fb}\left(x\left(\hat{t}\right),\varphi_{\xi}^{d}\left(\hat{t}\right)\right),d\right). \tag{M.5}$$

Assumption M.5: *For any trajectory φ_{ξ}^{d} in Assumption M.4 that a CH controller tries to follow, there exists a feedback controller $u^{fb}:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}^m$ that makes φ_{ξ}^{d} uniformly locally exponentially stable, i.e., the closed loop system in (M.5) satisfies*

$$\begin{aligned}&\exists B\subseteq\mathbb{R}^n, s.t.\ \forall 0\le t_1\le t_2\le T_p, \left(x\left(t_1\right)-\varphi_{\xi}^{d}\left(t_1\right)\right)\in B,\\ &\left\|x\left(t_2\right)-\varphi_{\xi}^{d}\left(t_2\right)\right\|\le e^{-c_2\left(t_2-t_1\right)}\left\|x\left(t_1\right)-\varphi_{\xi}^{d}\left(t_1\right)\right\|\end{aligned} \tag{M.6}$$

for some $c_2>0$.

Next, the result on the stability of the CH controller is presented. First, consider the case when d is fixed.

Theorem M.1: *Under Assumption M.1, Assumption M.2, Assumption M.3, Assumption M.4 and Assumption M.5, for an initial condition $\xi\in\left\{x\,|\,b\left(x\right)\ge 0\right\}$ the closed-loop system in (M.5) will stay inside $\left\{x\,|\,b\left(x\right)\ge 0\right\}$, and if d stops changing after $T\ge 0$, the state will converge to $\eta_x\left(d\right)$ exponentially.*

Proof: From Assumption M.4, since $\xi\in\left\{x\,|\,b\left(x\right)\ge 0\right\}$, $\xi\in\mathcal{S}$, which means the feedback system in (M.5) is well defined. From (M.3), the CBF $b\left(x\right)$ remains nonnegative as discussed in Section 2.2, which proves that the state will stay inside $\left\{x\,|\,b\left(x\right)\ge 0\right\}$, and thus $x\left(t\right)\in\mathcal{S},\forall t\ge 0$.

When d stops changing, from the Lyapunov condition in Assumption M.4, $V_d\left(x\left(nT_p\right)-\eta_x\left(d\right)\right)\le c_1^{n-1}V_d\left(x\left(0\right)-\eta_x\left(d\right)\right)$, $n=1,2,3,\ldots$, which implies $\lim_{n\to\infty}V_d\left(x\left(nT_p\right)\right)=0$. So from Assumption M.3, the sequence $x\left(nT_p\right)$ converges to $\eta_x\left(d\right)$. Therefore from the last assumption in (M.3), the state stays at $\eta_x\left(d\right)$.

■

M.2 State decomposition and dimension reduction

As discussed in Section 4.3.2, under a grid fashion sampling of the initial condition, the computation power limits the dimension of the feature that describes the initial condition. To parameterize the initial condition with a subset of states, the states are decomposed into two parts: $x=\left[x_1;x_2\right]$, where in practice $x_1\in\mathbb{R}^{n_1}$ are states with slow dynamics and $x_2\in\mathbb{R}^{n_2}$ are states with fast and stable dynamics. The case where the trajectory and tracking feedback u^{fb} is parameterized by only x_1 is considered.

Definition M.1: *A locally Lipschitz continuous function* $\gamma:\mathbb{R}^{n_1}\to\mathbb{R}^{n_2}$ *such that* $\gamma(0)=0$ *and satisfies*

$$\begin{aligned}\forall d\in\mathcal{D},\eta_x(d)&=\left[\eta_x^1(d);\eta_x^2(d)\right],\\ \eta_x^2(d)&=\gamma\left(\eta_x^1(d)\right),\end{aligned}\tag{M.7}$$

is called an insertion map.

The condition in (M.7) states that for any $d\in\mathcal{D}$, the insertion map maps the steady state of x_1 to the steady state of x_2. To extend the previous conclusion to cases where trajectories are parameterized with only x_1, the following assumptions are made.

Assumption M.6: $\exists\mathcal{S}\subseteq\mathbb{R}^n$, *compact, such that* $\forall d\in\mathcal{D},\eta_x(d)\in\mathcal{S}$. *There exists CBF* $b(x)$, *s.t.* $\forall x\notin\mathcal{S},b(x)\le 0$; $\forall d\in\mathcal{D},b\left(\eta_x(d)\right)\ge 0$; $\forall d\in\mathcal{D},\xi=\left[\xi_1;\xi_2\right]\in\mathcal{S},\quad \xi_2=\gamma(\xi_1)$, *there exists* $u_{\xi_1}^d:\left[0,T_p\right]\to\mathbb{R}^m$ *and a corresponding state trajectory,* $\varphi_{\xi_1}^d:\left[0,T_p\right]\to\mathbb{R}^n$ *satisfying*

$$\begin{aligned}V_d\left(\varphi_{\xi_1}^d(T_p)-\eta_x(d)\right)&\le cV_d\left(\varphi_{\xi_1}^d(0)-\eta_x(d)\right),\\ \frac{db}{dx}|_{\varphi_{\xi_1}^d(t)}\dot{\varphi}_{\xi_1}^d(t)+\kappa\varphi_{\xi_1}^d(t)&\ge 0,\\ \lim_{\left[\xi_1,\gamma(\xi_1)\right]\to\eta_x(d)}\varphi_{\xi}^d(t)&=\eta_x(d),\\ \lim_{\left[\xi_1,\gamma(\xi_1)\right]\to\eta_x(d)}u_{\xi}^d(t)&\equiv\eta_u(d),\\ \varphi_{2\xi_1}^d(T_p)&=\gamma\left(\varphi_{1\xi_1}^d(T_p)\right)\end{aligned}\tag{M.8}$$

Assumption M.7: *There exists a feedback* $u_1^{fb}:\mathbb{R}^{n_1}\times\mathbb{R}^{n_1}\to\mathbb{R}^m$ *that* $u_1^{fb}\left(x_1(t),\varphi_{1\xi_1}^d\left(\hat{t}\right)\right)$ *makes* $\varphi_{\xi_1}^d$ *uniformly locally exponentially stable, i.e. (M.6) is satisfied with* $u\left(\hat{t}\right)=u_\xi^d\left(\hat{t}\right)+u_1^{fb}\left(x_1\left(\hat{t}\right),\varphi_{1\xi_1}^d\left(\hat{t}\right)\right)$.

Remark M.3: *The subscript* $\varphi_{1\xi_1}^d$ *means the desired trajectory of* x_1 *with initial condition* $x_1(0)=\xi_1$, *and* $\varphi_{2\xi_1}^d$ *means the desired trajectory of* x_2, $\varphi_{\xi_1}^d=\left[\varphi_{1\xi_1}^d;\varphi_{2\xi_1}^d\right]$. *Assumption M.7 is possible if the dynamic subsystem of* x_2 *is locally exponentially stable.*

Theorem M.2: *Under Assumption M.1, Assumption M.*2*, Assumption M.3, Assumption M.6 and Assumption M.7,* $\forall d\in\mathcal{D}$ *fixed,* $\forall\xi=\left[\xi_1;\gamma(\xi_1)\right]\in\left\{x\mid b(x)\geq 0\right\}$, *the closed-loop system under CH feedback will stay inside* $\left\{x\mid b(x)\geq 0\right\}$, *and if d stop changing after* $T\geq 0$, *the state will converge to* $\eta_x(d)$ *exponentially.*

Proof: By Assumption M.7, the closed loop system exponentially converges to the CH desired trajectory. From Assumption M.6, by CBF condition, $\left\{x\mid b(x)\geq 0\right\}$ is invariant under the CH controller. When d stops changing, the closed loop system exponentially converges to φ_ξ^d and $V\left(x\left(nT_p\right)-\eta_x(d)\right)\leq c^{n-1}V\left(\xi-\eta_x(d)\right)$, $n=1,2,3,\ldots$, and satisfies $x_2\left(nT_p\right)=\gamma\left(x_1\left(nT_p\right)\right)$. So every time the desired trajectory is executed to the end, there exists $\varphi_{x_1\left(nT_p\right)}^d$ that follows the previous trajectory, By the definition of insertion map in (M.7) make sure that when $x_1\to\eta_x^1(d)$, $\gamma(x_1)\to\eta_x^2(d)$. By Theorem M.2, $x(t)$ converges to $\eta_x(d)$ exponentially.

■

Remark M.4: *When the dynamics of* x_2 *is stable and fast,* $y:=x_2-\varphi_{2\xi_1}$ *converges to zero quickly, the influence of initial condition of* x_2 *is small enough to be neglected. Therefore, the CH can be parameterized only by* x_1.

Now consider a CH controller with trajectories generated with the procedure described in (4.20). Assumption M.1and Assumption M.2 are trivially satisfied by the linear dynamics, where η_x is defined such that it maps r_d to the equilibrium point that renders $z=h(x)=0$, which is

unique. It can be shown that η_x is Lipschitz continuous. The cost-to-go function of a Linear Quadratic Regulator (LQR) is used as the Lyapunov function V by solving the Riccati equation. Since V is quadratic, and the truck dynamic is linear, V satisfies Assumption M.3 for all r_d. The CBF condition and Lyapunov condition in Assumption M.6 are enforced in the trajectory optimization by the last two constraints in (4.20). Pick $x_1 = \begin{bmatrix} z & \dot{z} & \psi & -B_4 v_y + B_2 r & \psi_a \end{bmatrix}$, since z and $\dot{z}$ are part of x_1, the closed loop dynamics is indeed stable under the PD control that only depends on x_1, which is the direct result of a stable zero dynamics, therefore satisfies the exponential stability condition in Assumption M.7. Note that the initial conditions in the training set are parameterized by Φ, which is a full rank linear transformation of x_1 and r_d. By Theorem M.2, the closed-loop system with CH feedback stays within $\{x \mid b(x) \geq 0\}$, and converges to $\eta_x(d)$ exponentially once r_d stops changing.

Appendix N. Smoothing of the desired trajectory in Section 4.4.2

The smoothing of Bezier curve is very simple. For an *m-th* order Bezier curve, the value for 0th to 2nd derivative at $s = 0$ are

$$\begin{aligned} \mathbf{B}(0) &= \alpha_0, \\ \mathbf{B}'(0) &= m\alpha_1 - m\alpha_0, \\ \mathbf{B}''(0) &= m(m-1)(\alpha_2 + \alpha_0 - 2\alpha_1). \end{aligned} \tag{N.1}$$

Solving for α_0, α_1 and α_2:

$$\begin{aligned} \alpha_0 &= h_{des}^0, \\ \alpha_1 &= \frac{\dot{h}_{des}^0}{m} + \alpha_0, \\ \alpha_2 &= \frac{\ddot{h}_{des}^0}{m(m-1)} + 2\alpha_1 - \alpha_0, \end{aligned} \tag{N.2}$$

where h_{des}^0, $\dot{h}_{des}^0$ and $\ddot{h}_{des}^0$ are the current value and derivatives of the desired trajectory before the update. The smoothing process requires that the Bezier order should be high enough so that the influence of the smoothing is limited to only the beginning of the curve. The Bezier order is chosen to be 8.

Appendix O. Proof of Theorem 5.1

To simplify the notation, define

$$\begin{aligned}\mathcal{M}(x,d) &= \frac{dV}{dx}\left(f(x)+g_1(x)u_1(x,d)+g_d(x)d+g_2(x)u_2(x,d)\right),\\ \mathcal{M}_1(x,d) &= \frac{dV}{dx}\left(f(x)+g_1(x)u_1(x,d)+g_d(x)d\right),\\ \mathcal{M}_2(x,d) &= \frac{dV}{dx}g_2(x)u_2(x,d),\end{aligned} \tag{O.1}$$

and note that

$$\mathcal{M}(x,d)=\mathcal{M}_1(x,d)+\mathcal{M}_2(x,d). \tag{O.2}$$

Assume the hypothesis of part (a) of Theorem 5.1, then

$$\mathcal{M}(x,d)\le 0. \tag{O.3}$$

To show part (a), select

$$\begin{aligned}\alpha_1(x,d)&=\mathcal{M}_1(x,d),\\ \alpha_2(x,d)&=\mathcal{M}_2(x,d),\end{aligned} \tag{O.4}$$

then (5.12) is immediately satisfied with (O.3). Since

$$\begin{aligned}\mathcal{M}_1(x,d)&=\alpha_1(x,d)\le\alpha_1(x,d),\\ \mathcal{M}_2(x,d)&=\alpha_2(x,d)\le\alpha_2(x,d),\end{aligned} \tag{O.5}$$

(5.12) is also satisfied.

Next, let us consider part (b). Since $\mathcal{D}$ is a semialgebraic set defined with polynomials, and V is a polynomial of x, the following representation is used:

$$\left\{x,d \mid h_i(x,d)\ge 0\right\}=\left\{x \mid V(x)\ge 1\right\}\times\mathcal{D}. \tag{O.6}$$

Suppose (5.13) can be verified by SOS, that is, through Parrilo's Hierarchy, there exist SOS multipliers $\{c_i\}$ such that

$$SOS\left(-\mathcal{M}(x,d)-\sum_i c_i(x,d)h_i(x,d)\right). \tag{O.7}$$

Let

$$\begin{aligned}\alpha_1(x,d)&=\mathcal{M}_1(x,d)+\sum_i c_i(x,d)h_i(x,d),\\ \alpha_2(x,d)&=\mathcal{M}_2(x,d),\end{aligned} \tag{O.8}$$

then such α_1 and α_2 satisfy (5.11) and (5.12). Moreover, since

$$\begin{aligned}\mathcal{M}_1(x,d)-\alpha_1(x,d) &= 0-\sum_i c_i(x,d)h_i(x,d),\\ \mathcal{M}_2(x,d)-\alpha_2(x,d) &= 0,\\ \alpha_1(x,d)+\alpha_2(x,d) &= \mathcal{M}(x,d)+\sum_i c_i(x,d)h_i(x,d),\end{aligned} \tag{O.9}$$

which means that (5.11) and (5.12) can be verified with SOS. This proves part (b).

Next, it is shown that when one of $\mathcal{M}_1$ and $\mathcal{M}_2$ depends only on x, α_1 and α_2 can be functions of x only. Suppose

$$\mathcal{M}_2(x,d)=\frac{dV}{dx}g_2(x)u_2(x)=\mathcal{M}_2(x), \tag{O.10}$$

and $\mathcal{M}(x,d)=\mathcal{M}_1(x,d)+\mathcal{M}_2(x)$ satisfies (5.13), then let

$$\alpha_1(x)=-\mathcal{M}_2(x),\alpha_2(x)=\mathcal{M}_2(x), \tag{O.11}$$

and the above α_1 and α_2 satisfy (5.11) and (5.12). If (5.13) can be verified with SOS, that is, (O.7) is satisfied with some c_i, then (5.11) and (5.12) can be verified by SOS with the above α_1 and α_2.

■

Appendix P. Proof of Lemma 6.1

Proof: Lemma 6.1 is a specific instantiation of the general theory developed in [145]. The robust optimization in (6.15) is equivalent to the following:

$$\begin{aligned}&\min c^T\alpha \ s.t.\\ &\left\{\max_{\beta\in\mathcal{P}(F,f)} H_1^i\beta+\alpha^T H_2^i\beta+H_3^i\alpha\right\}\le h^i, i=1,...M,\end{aligned} \tag{P.1}$$

where the maximum is taken over the uncertainty set. Then the maximum is written as its dual:

$$\begin{aligned}\mathbf{p}^i &= \max_{\beta\in\mathcal{P}(F,f)} H_1^i\beta+\alpha^T H_2^i\beta+H_3^i\alpha,\\ \mathbf{d}^i &= \min_{\lambda^i\ge 0}\max_{\beta} H_1^i\beta+\alpha^T H_2^i\beta+H_3^i\alpha+\left(\lambda^i\right)^T(f-F\beta).\end{aligned} \tag{P.2}$$

According to the duality theory, $\mathbf{d}^i\ge\mathbf{p}^i$, and with some mild assumptions, by strong duality, $\mathbf{d}^i=\mathbf{p}^i$. Note that

$$\max_{\beta} H_1^i\beta+\alpha^T H_2^i\beta+H_3^i\alpha+\left(\lambda^i\right)^T(f-F\beta)=\begin{cases}H_3^i\alpha+\left(\lambda^i\right)^T f, if\ H_1^i+\alpha^T H_2^i=\left(\lambda^i\right)^T F\\ \infty,\quad otherwise\end{cases}. \tag{P.3}$$

So the dual is written as

$$\mathbf{d}^i = \min_{\lambda^i \geq 0, H_1^i + \alpha^T H_2^i = \left(\lambda^i\right)^T F} H_3^i \alpha + \left(\lambda^i\right)^T f. \tag{P.4}$$

Combined with the original optimization in (6.15), the robust optimization is transformed to the following:

$$\begin{aligned} &\min c^T \alpha \ s.t. \\ &H_3^i \alpha + \left(\lambda^i\right)^T f \leq h^i, \\ &H_1^i + \alpha^T H_2^i = \left(\lambda^i\right)^T F, \\ &\lambda^i \geq 0, i = 1, \ldots M. \end{aligned} \tag{P.5}$$

Note that due to the duality, $\mathbf{d}^i$ is always an upper bound of the right-hand side of the constraint in (6.15), therefore a solution to (P.5) is always a feasible solution to (6.15), and when strong duality holds, the two formulations are equivalent.

■

Bibliography

[1] E. COMPUTING, "Cyber-physical systems," 2009.

[2] K. J. Åström and T. Hägglund, *PID controllers: theory, design, and tuning* vol. 2: Instrument society of America Research Triangle Park, NC, 1995.

[3] S. Skogestad and I. Postlethwaite, *Multivariable feedback control: analysis and design* vol. 2: Wiley New York, 2007.

[4] M. Morari, C. Garcia, J. Lee, and D. Prett, *Model predictive control*: Prentice Hall Englewood Cliffs, NJ, 1993.

[5] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *ACM Computing Surveys (CSUR)*, vol. 28, pp. 626-643, 1996.

[6] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, pp. 183-235, 1994.

[7] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *International Conference on Computer Aided Verification*, Berlin, Heidelberg, 2001, pp. 53-65.

[8] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *Protocol Specification, Testing and Verification XV*, ed: Springer, 1996, pp. 3-18.

[9] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*: MIT press, 1999.

[10] K. L. McMillan, "Symbolic model checking," in *Symbolic Model Checking*, ed: Springer, 1993, pp. 25-60.

[11] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: Formal models, validation, and synthesis," *Proceedings of the IEEE*, vol. 85, pp. 366-390, 1997.

[12] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, Charleston, USA, 2006, pp. 364-380.

[13] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, Chicago, IL, USA, 2011, pp. 313-314.

[14] M. Risler and O. von Stryk, "Formal behavior specification of multi-robot systems using hierarchical state machines in XABSL," in *AAMAS08-workshop on formal models and methods for multi-robot systems, Estoril, Portugal*, 2008.

[15] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, pp. 61-70, 2007.

[16] R. Alur and E. Altug, "Linear Hybrid Automata," 2000.

[17] R. Alur, "Formal verification of hybrid systems," in *Proceedings of the International Conference on Embedded Software (EMSOFT), 2011*, Taipei, Taiwan, 2011, pp. 273-278.

[18] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, *et al.*, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, pp. 3-34, 1995.

[19] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*, Zurich, Switzerland, 2005, pp. 291-305.

[20] C. Le Guernic and A. Girard, "Reachability analysis of hybrid systems using support functions," in *International Conference on Computer Aided Verification*, Grenoble, France, 2009, pp. 540-554.

[21] E. Haghverdi, P. Tabuada, and G. J. Pappas, "Bisimulation relations for dynamical, control, and hybrid systems," *Theoretical Computer Science*, vol. 342, pp. 229-261, 2005.

[22] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 971-984, 2000.

[23] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Transactions on Automatic Control*, vol. 55, pp. 116-126, 2010.

[24] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *International Workshop on Hybrid Systems: Computation and Control*, Prague, Czech Republic, 2003, pp. 498-513.

[25] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," *Hybrid Systems: Computation and Control, Proceedings*, vol. 2993, pp. 477-492, 2004.

[26] S. Gulwani and A. Tiwari, "Constraint-based approach for analysis of hybrid systems," in *International Conference on Computer Aided Verification*, Princeton, USA, 2008, pp. 190-203.

[27] A. Platzer and E. M. Clarke, "Computing differential invariants of hybrid systems as fixedpoints," in *International Conference on Computer Aided Verification*, Pricneton, USA, 2008, pp. 176-189.

[28] A. Platzer, *Logical analysis of hybrid systems: proving theorems for complex dynamics*: Springer Science & Business Media, 2010.

[29] S. Prajna, "Barrier certificates for nonlinear model validation," *Automatica*, vol. 42, pp. 117-126, 2006.

[30] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, Los Angeles, USA, 2014, pp. 6271-6278.

[31] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. Ames, *et al.*, "Preliminary results on correct-by-construction control software synthesis for adaptive cruise control," in *IEEE 53rd Annual Conference on Decision and Control (CDC), 2014* Los Angeles, USA, 2014, pp. 816-823.

[32] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, pp. 1747-1767, 1999.

[33] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on automatic control,* vol. 50, pp. 947-957, 2005.

[34] D. Henrion and M. Korda, "Convex Computation of the Region of Attraction of Polynomial Control Systems," *IEEE Transactions on Automatic Control,* vol. 59, pp. 297-312, 2014.

[35] R. Wisniewski and C. Sloth, "Converse Barrier Certificate Theorems," *IEEE Transactions on Automatic Control,* vol. 61, pp. 1356-1361, May 2016.

[36] K. Chatterjee and T. A. Henzinger, "Assume-guarantee synthesis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Braga, Portugal, 2007, pp. 261-275.

[37] M. Arcak, C. Meissen, and A. Packard, *Networks of dissipative systems: compositional certification of stability, performance, and safety*: Springer, 2016.

[38] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, USA, 2002, pp. 3482-3487.

[39] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness Guarantees for the Composition of Lane Keeping and Adaptive Cruise Control," *arXiv preprint arXiv:1609.06807,* 2016.

[40] S. Prajna, A. Papachristodoulou, and F. Wu, "Nonlinear control synthesis by sum of squares optimization: A Lyapunov-based approach," in *5th Asian Control Conference, 2004*, Melbourne, Australia, 2004, pp. 157-165.

[41] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification," in *The International Journal of Robotics Research* vol. 29, ed, 2010, pp. 1038-1052.

[42] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: A general purpose sum of squares programming solver," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, Las Vegas, USA, 2002, pp. 741-746.

[43] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design, 2004* New Orleans, USA, 2004, pp. 284-289.

[44] F. P. Mark M. Tobenkin, Alexandre Megretski. *SPOTless: Conic and Polynomial Programming Toolbox*. Available: https://github.com/mmt/spotless

[45] P. A. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming,* vol. 96, pp. 293-320, 2003.

[46] J. B. Lasserre, "A sum of squares approximation of nonnegative polynomials," *Siam Journal on Optimization,* vol. 16, pp. 751-765, 2006.

[47] J. Bochnak, M. Coste, and M.-F. Roy, *Real algebraic geometry* vol. 36: Springer Science & Business Media, 2013.

[48] J.-B. Lasserre, *Moments, positive polynomials and their applications* vol. 1: World Scientific, 2010.

[49] S. Prajna and A. Jadbabaie, "Safety Verification of Hybrid Systems Using Barrier Certificates," in *International Workshop on Hybrid Systems: Computation and Control*, Berlin Heidelberg, 2004, pp. 477-492.

[50] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control,* vol. 62, pp. 3861-3876, 2017.

[51] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *American Control Conference (ACC), 2015*, Chicago, USA, 2015, pp. 4542-4548.

[52] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine,* vol. 48, pp. 68-73, 2015.

[53] L. Wang, A. D. Ames, and M. Egerstedt, "Safety Barrier Certificates for Collisions-Free Multirobot Systems," *IEEE Transactions on Robotics,* 2017.

[54] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to automotive safety systems," *arXiv preprint arXiv:1609.06408,* 2016.

[55] G. Stengle, "A Nullstellensatz and a Positivstellensatz in semialgebraic geometry," *Mathematische Annalen,* vol. 207, pp. 87-97, 1974.

[56] T. IIMURA and K. YAMAMOTO, "2A1-H03 Development of Single-passenger Mobility-Support Robot "ROPITS" : Verification of Self-Localization Function with Camera Images(Demonstration experiments of personal mobility robots in Mobility Robot Special Zone of Tsukuba-city)," presented at the ロボティクス・メカトロニクス講演会講演概要集, 2014.

[57] Transport-Systems-Catapult. (2016). *Self-Driving Pods*. Available: https://ts.catapult.org.uk/current-projects/self-driving-pods/

[58] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, *et al.*, "MINERVA: a second-generation museum tour-guide robot," in *Proceedings of IEEE International Conference on Robotics and Automation,1999*, Detroit, USA, 1999.

[59] K. Yamazaki, A. Yamazaki, M. Okada, Y. Kuno, Y. Kobayashi, Y. Hoshi, *et al.*, "Revealing Gauguin : Engaging Visitors in Robot Guide ' s Explanation in an Art Museum," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, Boston, USA, 2009, pp. 1437-1446.

[60] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *American Control Conference (ACC), 2012*, Montreal, Canada, 2012, pp. 4239-4244.

[61] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice,* vol. 17, pp. 741-750, 2009.

[62] D. Fernandez Llorca, V. Milanes, I. Parra Alonso, M. Gavilan, I. Garcia Daza, J. Perez, *et al.*, "Autonomous Pedestrian Collision Avoidance Using a Fuzzy Steering Controller," *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, pp. 390-401, 2011.

[63] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles," in *Journal of Guidance, Control, and Dynamics* vol. 25, ed, 2002, pp. 116-129.

[64] R. Bis, H. Peng, and G. Ulsoy, "Velocity Occupancy Space: Robot Navigation and Moving Obstacle Avoidance With Sensor Uncertainty," in *ASME 2009 Dynamic Systems and Control Conference, Volume 1*, Hollywood, California, USA, 2009, pp. 363-370.

[65] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics and Research,* vol. 5, pp. 90-98, 1986.

[66] S. Shimoda, Y. Kuroda, and K. Iagnemma, "Potential field navigation of high speed unmanned ground vehicles on uneven terrain," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 2828-2833.

[67] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," *Robotics Research,* pp. 3-19, 2011.

[68] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine,* vol. 4, pp. 23-33, 1997.

[69] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics,* vol. 21, pp. 188-195, 2005.

[70] S. Mitsch, K. Ghorbal, and A. Platzer, "On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles," in *Proceedings of Robotics: Science and Systems*, 2013.

[71] A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake, "Convex optimization of nonlinear feedback controllers via occupation measures," *The International Journal of Robotics Research,* vol. 33, pp. 1209-1230, 2014.

[72] K. Macek, D. Alejandro Vasquez Govea, T. Fraichard, and R. Siegwart, "Towards Safe Vehicle Navigation in Dynamic Urban Scenarios," *Automatika,* vol. 50, pp. 184-194, 2009.

[73] G. M. Ziegler, *Lectures on Polytopes*: Springer Science & Business Media, 2012.

[74] Y. Chen, H. Peng, and J. Grizzle, "Obstacle Avoidance for Low-Speed Autonomous Vehicles With Barrier Function," *IEEE Transactions on Control Systems Technology,* 2017.

[75] H. Gomi and M. Kawato, "Neural network control for a closed-loop system using feedback-error-learning," *Neural Networks,* vol. 6, pp. 933-946, 1993.

[76] X. Da, R. Hartley, and J. W. Grizzle, "Supervised Learning for Stabilizing Underactuated Bipedal Robot Locomotion, with Outdoor Experiments on the Wave Field."

[77] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316,* 2016.

[78] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-End Deep Reinforcement Learning for Lane Keeping Assist," *arXiv preprint arXiv:1612.04340,* 2016.

[79] S.-Y. Oh, J.-H. Lee, and D.-H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Transactions on Vehicular Technology,* vol. 49, pp. 997-1005, 2000.

[80] D. Bertsekas, *Dynamic programming and optimal control* vol. 1: Athena Scientific Belmont, MA, 1995.

[81] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, USA, 2012, pp. 2723-2730.

[82] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *IEEE 53rd Annual Conference on Decision and Control (CDC)*, Los Angeles, USA, 2014, pp. 1424-1431.

[83] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *2015 European Control Conference (ECC)*, Linz, Austria, 2015, pp. 2496-2501.

[84] A. Isidori, *Nonlinear control systems*: Springer Science & Business Media, 2013.

[85] H. K. Khalil, "Noninear systems," *Prentice-Hall, New Jersey,* vol. 2, pp. 5-1, 1996.

[86] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, "Feedback control of dynamic bipedal robot locomotion," ed: CRC press, 2007.

[87] A. Shiriaev, J. W. Perram, and C. Canudas-de-Wit, "Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach," *IEEE Transactions on Automatic Control,* vol. 50, pp. 1164-1176, 2005.

[88] M. Maggiore and L. Consolini, "Virtual holonomic constraints for Euler–Lagrange systems," *IEEE Transactions on Automatic Control,* vol. 58, pp. 1001-1008, 2013.

[89] J. W. Grizzle, M. D. Di Benedetto, and F. Lamnabhi-Lagarrigue, "Necessary conditions for asymptotic tracking in nonlinear systems," *IEEE Transactions on Automatic Control,* vol. 39, pp. 1782-1794, 1994.

[90] A. J. Miege and D. Cebon, "Optimal roll control of an articulated vehicle: theory and model validation," *Vehicle system dynamics,* vol. 43, pp. 867-884, 2005.

[91] J. Y. Wong, *Theory of ground vehicles*: John Wiley & Sons, 2008.

[92] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*: SIAM, 2010.

[93] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *IEEE International Conference on Robotics and Automation (ICRA), 2016*, Stockholm, Sweden, 2016, pp. 1447-1454.

[94] J. Stoer and R. Bulirsch, *Introduction to numerical analysis* vol. 12: Springer Science & Business Media, 2013.

[95] A. Hereid and A. D. Ames, "FROST*: Fast Robot Optimization and Simulation Toolkit," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

[96] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467,* 2016.

[97] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control,* vol. 52, pp. 1680-1685, 2007.

[98] X. Da and J. Grizzle, "Combining Trajectory Optimization, Supervised Machine Learning, and Model Structure for Mitigating the Curse of Dimensionality in the Control of Bipedal Robots," *arXiv preprint arXiv:1711.02223,* 2017.

[99] Y. A. Ghoneim, W. C. Lin, D. M. Sidlosky, H. H. Chen, Y. K. Chin, and M. J. Tedrake, "Integrated chassis control system to enhance vehicle stability," *International Journal of Vehicle Design,* vol. 23, pp. 124-144, 2000.

[100] T. H. Hwang, K. Park, S. J. Heo, S. H. Lee, and J. C. Lee, "Design of integrated chassis control logics for AFS and ESP," *International Journal of Automotive Technology,* vol. 9, pp. 17-27, Feb 2008.

[101] M. Nagai, M. Shino, and F. Gao, "Study on integrated control of active front steer angle and direct yaw moment," *Jsae Review,* vol. 23, pp. 309-315, Jul 2002.

[102] N. A. Duffie, R. Chitturi, and J. I. Mou, "Fault-Tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities," *Journal of Manufacturing Systems*, vol. 7, pp. 315-328, 1988.

[103] Nico A. Kelling and W. Heck, "Centralized Versus Distributed Redundancy for Brake-by-Wire Systems," SAE technical paper series 0148-7191, 2002.

[104] Y. Kou, "Development and Evaluation of Integrated Chassis Control Systems," 2010.

[105] T. Gordon, M. Howell, and F. Brandao, "Integrated control methodologies for road vehicles," *Vehicle System Dynamics*, vol. 40, pp. 157-190, Sep 2003.

[106] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "New developments in sum of squares optimization and SOSTOOLS," in *Proceedings of the American Control Conference*, Boston, USA, 2004, pp. 5606-5611.

[107] W. Tan, A. Packard, and others, "Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming," *IEEE Transactions on Automatic Control*, vol. 53, pp. 565-570, 2008.

[108] L. Xiao, M. Johansson, and S. P. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE Transactions on Communications*, vol. 52, pp. 1136-1144, 2004.

[109] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, pp. 1-122, 2011.

[110] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, pp. 49-95, 1996.

[111] L. E. Ghaoui, "Lecture 13: SDP Duality," University of California, Berkeley, Ed, 2008.

[112] K. C. Goh, L. Turan, M. G. Safonov, G. P. Papavassilopoulos, and J. H. Ly, "Biaffine matrix inequality properties and computational methods," in *American Control Conference*, Baltimore, USA, 1994, pp. 850-855.

[113] M. Fukuda and M. Kojima, "Branch-and-cut algorithms for the bilinear matrix inequality eigenvalue problem," *Computational Optimization and Applications*, vol. 19, pp. 79-105, 2001.

[114] H. a. k. Terelius, U. Topcu, and R. M. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC Proceedings Volumes*, vol. 44, pp. 11245-11251, 2011.

[115] S. W. Smith, P. Nilsson, and N. Ozay, "Interdependence quantification for compositional control synthesis with an application in vehicle safety systems," in *IEEE 55th Conference on Decision and Control (CDC), 2016*, Las Vegas, USA, 2016, pp. 5700-5707.

[116] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory* vol. 15: Siam, 1994.

[117] J. B. Lasserre, D. Henrion, C. Prieur, and E. Trélat, "Nonlinear optimal control via occupation measures and LMI-relaxations," *SIAM journal on control and optimization*, vol. 47, pp. 1643-1666, 2008.

[118] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical problems in engineering*, vol. 4, pp. 317-367, 1998.

[119] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, pp. 219-224, 2005.

[120] S. V. Rakovic, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne, "Invariant approximations of the minimal robust positively invariant set," *IEEE Transactions on Automatic Control,* vol. 50, pp. 406-410, 2005.

[121] S. V. Rakovic and M. Baric, "Parameterized robust control invariant sets for linear systems: Theoretical advances and computational remarks," *IEEE Transactions on Automatic Control,* vol. 55, pp. 1599-1614, 2010.

[122] P. Trodden, "A one-step approach to computing a polytopic robust positively invariant set," *IEEE Transactions on Automatic Control,* vol. 61, pp. 4100-4105, 2016.

[123] T. B. Blanco, M. Cannon, and B. De Moor, "On efficient computation of low-complexity controlled invariant sets for uncertain linear systems," *International journal of Control,* vol. 83, pp. 1339-1346, 2010.

[124] F. Tahir and I. M. Jaimoukha, "Low-complexity polytopic invariant sets for linear systems subject to norm-bounded uncertainty," *IEEE Transactions on Automatic Control,* vol. 60, pp. 1416-1421, 2015.

[125] D. Bertsekas, "Infinite time reachability of state-space regions by using feedback control," *IEEE Transactions on Automatic Control,* vol. 17, pp. 604-613, 1972.

[126] M. Rungger and P. Tabuada, "Computing robust controlled invariant sets of linear systems," *IEEE Transactions on Automatic Control,* vol. 62, pp. 3665-3670, 2017.

[127] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control,* vol. 58, pp. 1771-1785, 2013.

[128] L. Ljung, "System identification," in *Signal analysis and prediction*, ed: Springer, 1998, pp. 163-173.

[129] R. L. Kosut, M. K. Lau, and S. P. Boyd, "Set-membership identification of systems with parametric and nonparametric uncertainty," *IEEE Transactions on Automatic Control,* vol. 37, pp. 929-941, 1992.

[130] J. Chen and G. Gu, *Control-oriented system identification: an H [infinity] approach* vol. 19: Wiley-Interscience, 2000.

[131] D. S. Shook, C. Mohtadi, and S. L. Shah, "A control-relevant identification strategy for GPC," *IEEE Transactions on Automatic Control,* vol. 37, pp. 975-980, 1992.

[132] G. C. Goodwin and M. E. Salgado, "A stochastic embedding approach for quantifying uncertainty in the estimation of restricted complexity models," *International Journal of Adaptive Control and Signal Processing,* vol. 3, pp. 333-356, 1989.

[133] S. Sadraddini and C. Belta, "Formal Guarantees in Data-Driven Model Identification and Control Synthesis," presented at the 21st international conference on Hybrid systems: computation and control, Porto, Portugal, 2018.

[134] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM review,* vol. 53, pp. 464-501, 2011.

[135] S. T. Wong and M. S. Roos, "A strategy for sampling on a sphere applied to 3D selective RF pulse design," *Magnetic Resonance in Medicine,* vol. 32, pp. 778-784, 1994.

[136] J. J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *Proceedings of IEEE International Conference on Robotics and Automation, 2004.*, New Orleans, USA, 2004, pp. 3993-3998.

[137] J. J. Hartung, P. Brink, J. Lamb, and D. P. Miller, "Autonomous Vehicle Platform and Safety Architecture," ed: Google Patents, 2017.

[138] G. Optimization. (2015). *Gurobi optimizer reference manual*. Available: http://www.gurobi.com

[139] H. Peng and M. Tomizuka, "Lateral control of front-wheel-steering rubber-tire vehicles," California Partners for Advanced Transportation Technology1990.

[140] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*: Cambridge university press, 1990.

[141] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain Markov decision processes with temporal logic specifications," in *IEEE 51st Annual Conference on Decision and Control (CDC), 2012* Maui, Hawaii, USA, 2012, pp. 3372-3379.

[142] T. M. Moldovan and P. Abbeel, "Safe exploration in Markov decision processes," *arXiv preprint arXiv:1205.4810*, 2012.

[143] L. C. Evans and P. E. Souganidis, "Differential Games and Representation Formulas for Solutions of Hamilton-Jacobi-Isaacs Equations.," Mathematics Research Center, University of Wisconsin-Madison 2492, 1983.

[144] V. I. Arnol'd, *Mathematical methods of classical mechanics* vol. 60: Springer Science & Business Media, 2013.

[145] A. Ben-Tal, D. Den Hertog, and J.-P. Vial, "Deriving robust counterparts of nonlinear uncertain inequalities," *Mathematical programming*, vol. 149, pp. 265-299, 2015.