Title
Robust Hierarchical Control with Connected Layers

Author
Schweidel, Katherine

Publication Date
2023

Peer reviewed|Thesis/dissertation

Robust Hierarchical Control with Connected Layers

By

Katherine Schweidel

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Murat Arcak, Chair
Professor Peter Seiler
Professor Francesco Borrelli
Professor Kameshwar Poolla

Summer 2023

Robust Hierarchical Control with Connected Layers

Abstract

Robust Hierarchical Control with Connected Layers

by

Katherine Schweidel

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Murat Arcak, Chair

Complex systems, such as autonomous vehicles and missile guidance systems, use hierarchical control schemes where each control layer employs a different system model. This approach enhances computational efficiency because using a simpler model in the higher-level control layer reduces computation times, enabling real-time control strategies. This dissertation presents a framework in which a lower-fidelity planning model is employed for online planning, and a tracking controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set. To ensure safety, the error that arises from the different models in each control layer is rigorously accounted for through augmentation of the planner safety constraints with the tracking error bound.

Accommodating more sources of real-world uncertainty enhances the safety and usefulness of the control scheme. We next describe a robust extension which utilizes integral quadratic constraints to accommodate input uncertainties such as unknown delays or unmodeled actuator dynamics in the tracking model. Finally, through a case study of shared vehicle control between a human driver and a supervisory autonomous system in longitudinal driving scenarios, we present a novel method called Driver-in-the-Loop Contingency MPC that leverages simplified dynamics to compute invariant sets that guarantee safety with respect to other vehicles. This contribution can be viewed as adding robustness to other agents in the planning layer.

To my parents David and Linda, my sister Laurel, and my furry brother Leonard.

Contents

List of Figures

List of Tables

Acknowledgments

There are so many people I would like to thank who have shaped my PhD experience over the past six years. I would like to thank Professor Mark Mueller, Professor Laurent El Ghaoui, and Professor Jonathan Shewchuk for serving on my qualifying exam committee, Professor Francesco Borrelli and Professor Peter Seiler for serving on my dissertation committee, and Professor Kameshwar Poolla for serving on both committees.

Thank you to Professor Murat Arcak for being a great co-advisor for the first two years of my PhD and a wonderful primary advisor for the last four. Thank you for all the invaluable research meetings and for humoring all of my little questions. Thank you for guiding my research and supporting me along the way.

I was fortunate to work closely with Professor Peter Seiler during my PhD. Pete, you share Andy's sharpness, enthusiasm, and generosity. Thank you for always being willing to meet whenever I needed help. Working with you has kept an Andy spark in my life after his passing, and I'm so grateful.

Professor Andy Packard was one of the main reasons I decided to come to Berkeley for graduate school. He was brilliant, his excitement for controls was contagious, and he was such a fun person to be around. In the two years I worked with Andy, I felt lucky to be his student and friend. He was so generous with his time and was always excited to talk to his students, never making us feel like we were interrupting. Even when his health was poor, he made a huge effort to connect with his students, in and out of the lab. I will remember lab dinners, baseball games, and a wonderful log-rolling trip about a month before Andy's passing.

On that log-rolling trip I got to spend time with Johanna Sedman, who has become a dear friend and the world's best landlord. Thank you to Johanna for welcoming me into your life and your home. Thank you for your incredible generosity and for throwing me a wonderful graduation party that I will never forget. I want to be like you when I grow up!

One cool thing about being a PhD student is that you get to have awesome experiences outside the university that still count toward your degree. I would like to thank the planning and controls teams at Toyota Research Institute (Sarah Koehler, Selina Pan, Manuel Ahumada, Matt Brown, Carrie Bobier-Tiu, Miroslav Barić, Vishnu Desaraju, and Timothée Cazenave) for a fun and engaging internship during the summer of 2021. I would also like to thank Professor Astrid Brodtkorb, Markus Fossdal, and Raphaël Mounet for welcoming me to Trondheim, Norway and collaborating on fun, challenging experiments at NTNU during the fall of 2022.

I am lucky to have had many amazing labmates during my time at Berkeley. Thank you to Emmanuel Sin, Galaxy Yin, Octavio Narváez-Aroche, Arun Hegde, Akhil Shetty, and Jared Porter for creating a fun, kind, and supportive BCCI community. I would also like to thank my honorary labmate Jyot Buch; collaborating with you on our paper in 2020 was one of my graduate school highlights. I would also like to thank Douglas Philbrick, who is not a labmate, but who showed up to our lab meetings as diligently as anyone. Thank you for your help and insight over the years. Finally, it was great to work with the students in

Murat's research group. Thanks especially to Emily Jensen, Alex Devonport, and Neelay Junnarkar for the recent work sessions at Yali's that have been so fun and helpful.

Emily has patiently listened to me explain our meeting story to many people, so why stop now? Andy introduced me to Emily in 2019 when he suggested we be roommates at ACC. At the time, I thought it was a funny anecdote: "there are so few women in controls that my advisor literally had to invite another woman to the conference to be my roommate," but I'm now happy to call Emily my dear friend and treasured labmate. Thank you Emily for fostering a sense of lab community that was missing after the pandemic. It has been such a joy to work together at Yali's, proofread each other's papers and emails, and walk Paisley together. Thank you for encouraging me to exercise and to work on campus with you, two things that I resist but that you know are good for me!

Many other friends, in and out of school, have made my life as a graduate student rich and rewarding. Thank you to my Etcheverry buddies Tony Zheng, Ed Zhu, and Monica Li. I would also like to thank Eric Thacher for his support and encouragement throughout my degree. Thank you for all the adventures, big and small, over the last four years.

Last but certainly not least, thank you to Charlott Vallon, my first friend at UC Berkeley and the person who made me conclude that I'd made the right choice by coming here for graduate school. Since our first friend date to Jupiter and Ben & Jerry's, we've made countless memories together through Cardio Dance, group projects, winter trips, and Brewed Awakening study sessions. Thank you for making my time at Berkeley so fun and full of laughter, and thank you for *finally* moving back to me.

I am lucky in a lot of ways, but I think I'm most lucky to have a family that I love so much and also *like* so much. Going to graduate school in my hometown has led to so much quality time, especially during the pandemic, that we wouldn't have gotten otherwise. Thank you to my parents David and Linda, my sister Laurel, and our dog Leonard for consistently being a huge source of joy in my life! I'd say everything I've ever done has been made possible by your unconditional love and support, this degree included.

Chapter 1

Introduction

1.1 Motivation

Complex systems, such as autonomous vehicles [1] and missile guidance systems [2], use hierarchical control schemes where each control layer uses a different system model. This approach can enhance computational efficiency because the higher-level control layer can use a simpler design model, which can reduce computation times and enable control strategies that may not have been possible with the more complex design model. Faster higher-level control can also allow the system to respond to changing environments in real-time.

However, the resulting hierarchical control scheme may be unsafe if the error between the models in different layers is not accounted for. A high-level motion plan generated using a simplified model may avoid obstacles that the lower-level controller is unable to avoid, resulting in a collision. Thus, it is important to rigorously handle the error that arises from the different models in each control layer.

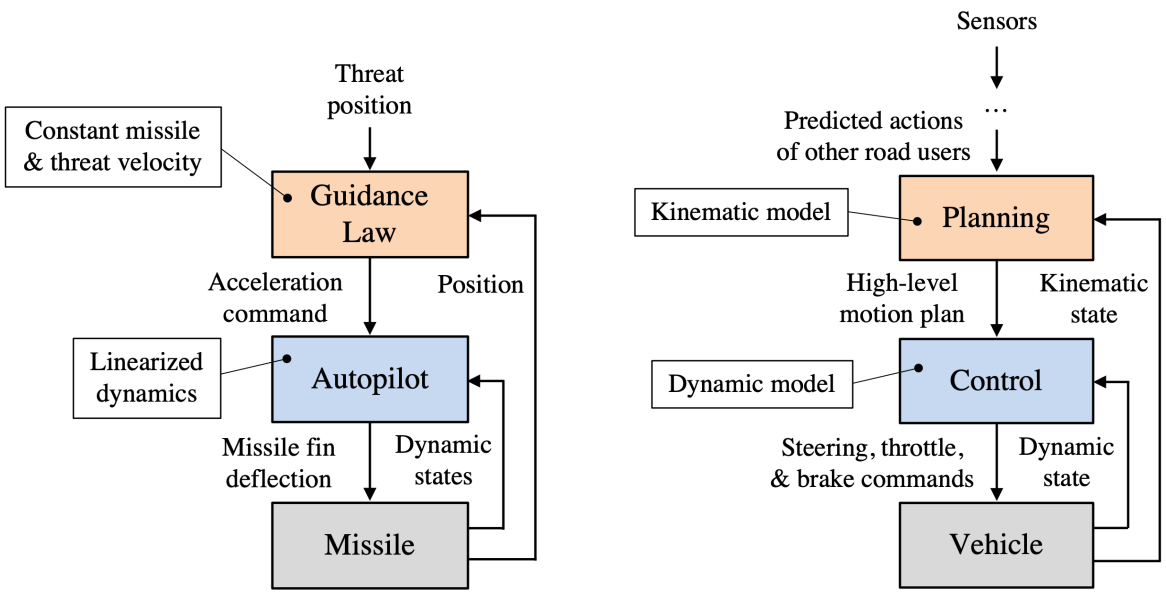
Accounting for this error can allow us to enjoy the benefits of the simpler higher-level control while still maintaining safety guarantees. This is the main idea behind the planner-tracker framework [3–10], which we will formally introduce in Chapter 2.

Furthermore, there are many sources of uncertainty in the real world, and the more sources of uncertainty we can accommodate, the safer and more useful the framework is in practice. The main contribution of this dissertation is adding robustness to the layers of planner-tracker framework in order to accommodate various sources of uncertainty.

We focus on the different sources of uncertainty that arise in different layers. In Chapter 2, we account for exogenous disturbances in the tracking layer. In Chapter 4, we account for uncertainty arising from other agents in the planning layer. In Chapters 3 and 6, we account for unmodeled dynamics in the tracking layer.

Motivating Examples

Missile guidance systems [2,11] use a hierarchical control structure, where different modeling assumptions are present in each layer (see Figure 1.1a). When a missile is trying to intercept



(a) Missile block diagram.

(b) Autonomous vehicle block diagram.

Figure 1.1: Examples of hierarchical control schemes with different modeling assumptions in each layer.

a threat, the guidance law gives an acceleration command. Proportional navigation is a common choice of guidance law, which is derived under the oversimplified modeling assumption that the missile and the threat have constant velocities [12]. This acceleration command is fed to the autopilot. The popular "three-loop autopilot" is designed using a linearization of the system about the equilibrium point at a fixed angle of attack and speed, with gains scheduled on these variables [11]. This autopilot computes a missile fin deflection that is used to actuate the missile control surfaces.

Autonomous vehicles [1, 13–17] are another example of a layered control scheme with different models used in each layer (see Figure 1.1b). The overall control scheme typically has many layers, such as localization, perception, prediction, planning, and control. We are interested in the planning and control layers. The planning layer receives predictions about other road users and generates a high-level motion plan, typically using a kinematic model of the vehicle, which describes the vehicle motion without taking forces into account. The control layer attempts to track this motion plan using a dynamic model of the vehicle, which includes tire forces. The control layer generates steering, throttle, and brake commands which are sent to the vehicle.

These are examples where the differences in the models are typically *not* accounted for and thus rigorous guarantees about safety and performance cannot be made. Next we show, in a simplified setting, how accounting for the errors between models can guarantee satisfaction of the true system constraints.

1.2 Problem Overview

We now present a hierarchical control framework for systems like the ones above, where different models are used in each control layer. We focus our attention on the case of a two-layer control architecture, and we explicitly account for the differences between models to guarantee performance and safety. Throughout this dissertation, we will consider a higher-fidelity model with state x and a lower-fidelity model with state $\hat{x}$. The lower-fidelity model has dynamics

$$\dot{\hat{x}} = \hat{f}(\hat{x}, \hat{u}), \tag{1.1}$$

where $\hat{u}$ is the input to the lower-fidelity model. We assume there is a given planning controller (e.g., MPC) that keeps the state and input within constraints $\hat{x} \in \hat{\mathcal{X}}$ and $\hat{u} \in \hat{\mathcal{U}}$. We denote this controller as $\hat{u} = \hat{\kappa}(\hat{x}, x, \hat{\mathcal{X}}, \hat{\mathcal{U}})$, which can depend on the state x of the higher-fidelity model. Note that $\hat{\kappa}$ can be the output of an online optimization problem (e.g., an MPC problem) and does not in general have a closed-form solution. The higher-fidelity model has dynamics

$$\dot{x} = f(x, u, d), \tag{1.2}$$

where u is the control input and $d \in \mathcal{D}$ represents some kind of disturbance or unmodeled dynamics. The high-fidelity model has state constraint $x \in \mathcal{X}$ encoding, for example, obstacles to avoid.

By analyzing the error between the low- and high-fidelity models, we can leverage the inputs $(\hat{x}, \hat{u})$ from the planning layer to generate a control input u in the tracking layer that will lead to satisfaction of the tracking constraint $x \in \mathcal{X}$. Consider, in the simplest case, an error between the two systems defined as

$$e := x - \hat{x}. \tag{1.3}$$

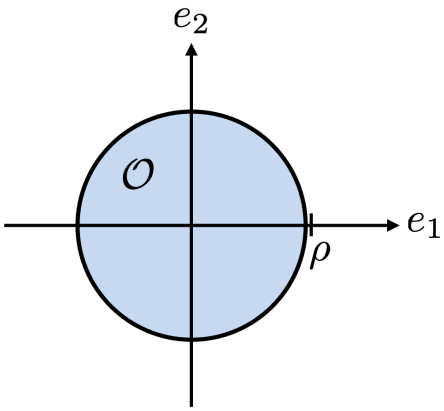
A more general error definition will be allowed in Chapter 2, allowing for different dimensions of x and $\hat{x}$. For now, we focus on this simple case to build intuition. This error variable has dynamics

$$\dot{e} = f(e + \hat{x}, u, d) - \hat{f}(\hat{x}, \hat{u}). \tag{1.4}$$

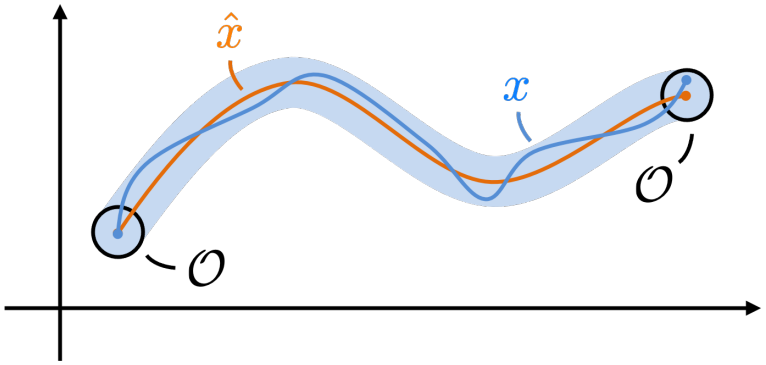
The goal is to find a tracking control law $u = \kappa(e, \hat{x}, \hat{u})$ and an associated error bound set $\mathcal{O}$, as small as possible, such that $e(t) \in \mathcal{O}$ for all $t \geq 0$. This set can be used to shrink constraints such that, when $e \in \mathcal{O}$, $\hat{x} \in \hat{\mathcal{X}}$ ensures that $x \in \mathcal{X}$.

For example, consider the set $\mathcal{O}$ shown in Figure 1.2a. If we know that $e(t)$ will remain in $\mathcal{O}$ for all $t \geq 0$, then $x = e + \hat{x}$ will live in a tube about $\hat{x}$ that is bloated by $\mathcal{O}$ as in Figure 1.2b. Thus, if $\hat{x} \in \hat{\mathcal{X}}$, $e \in \mathcal{O}$, and $\hat{\mathcal{X}} \subseteq \mathcal{X} \ominus \mathcal{O}$, then $x \in \mathcal{X}$. For example, if e is a position error $\mathcal{X}$ represents obstacle-avoidance constraints, then it suffices to select $\mathcal{X}$ as an obstacle-avoidance constraint with the same obstacles bloated by $\mathcal{O}$.

This hierarchical control scheme is summarized in Figure 1.3, which shows the interactions between the models, controllers, and constraints used in each layer.



(a) Example of error bound set $\mathcal{O}$.



(b) Trajectory for $\hat{x}$ in an inertial frame, bloated by $\mathcal{O}$. The trajectory for x remains within this bloated tube.

Figure 1.2: The role of an error bound set $\mathcal{O}$.

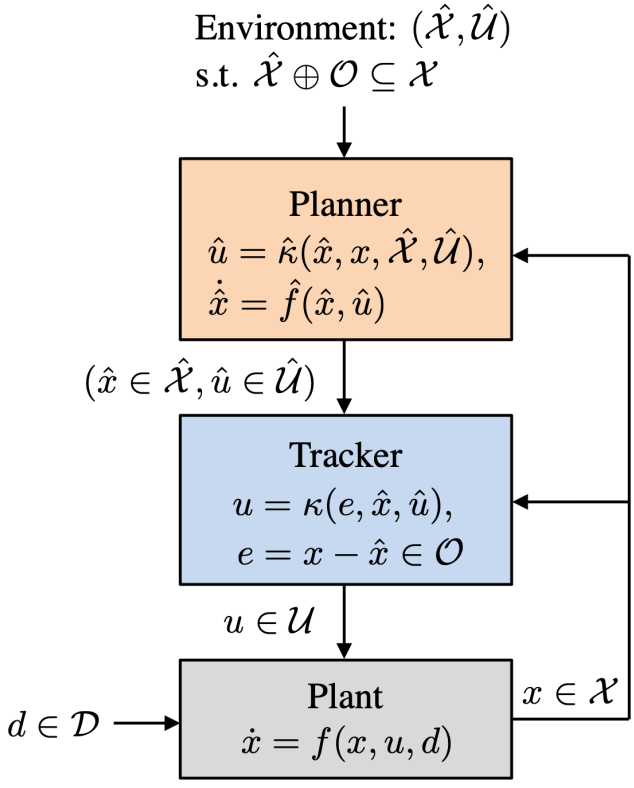


Figure 1.3: Planner-tracker block diagram.

1.3 Background

There are many existing approaches for combining the planning and control of complex, nonlinear and/or uncertain systems. Sequential convex programming (SCP) is a strategy for solving a nonconvex trajectory generation problem for a nonlinear system by solving multiple convex subproblems that approximate the original problem [18, 19]. SCP is a powerful approach that has been successfully applied in many domains, including rocket landing and robot motion planning. SCP can be combined with a feedback controller for tracking this reference trajectory.

Another class of algorithms for trajectory generation and feedback control involves piecing together multiple nominal trajectories, each with its own tracking controller and associated region of validity. In the funnel libraries method [20], a library of nominal trajectories is analyzed offline. For each nominal trajectory there is a tracking controller and a time-varying funnel centered at the nominal trajectory that describes how far the closed loop system can deviate from the nominal trajectory under uncertainty. Online, these trajectories and funnels can be pieced together to create a safe trajectory that avoids obstacles in the presence of uncertainty. The LQR Trees algorithm [3] works backwards from a goal set to compute a region of attraction that probabilistically covers the entire controllable space. Both of these works leverage SOS programming to certify that a given nominal trajectory/tracking controller pair has the desired property.

Model predictive control (MPC) is a popular trajectory planning and control strategy that solves a finite-time, receding horizon optimal control problem subject to state and input constraints [21]. Tube MPC [22–30] is a *robust* control strategy where the error between the true model and a nominal model, free of disturbances, is bounded within a tube, and MPC is performed on the nominal model with state constraints shrunk by the size of the tube. In rigid tube MPC, the tube has a fixed cross section, whereas in shrinking tube MPC, the cross section of the tube grows over the prediction horizon. Typically, the true and nominal models differ by only an additive disturbance. However, there are also works considering multiplicative uncertainty [27, 28], parametric uncertainty [30], and unmodeled dynamics [31, 32]. Furthermore, tube MPC methods typically apply to linear systems, but there is a growing body of literature on nonlinear tube MPC [33–37].

Reduced order MPC [38, 39] obtains a low-dimensional approximation of a high-dimensional model, often using a Galerkin or Petrov-Galerkin projection. An MPC controller is formulated for the reduced order model, and the input to the higher order model is computed using a feedback law involving the error between the reduced order state and the projection of the higher order state. This feedback control law is designed so that the state-error and input-error remain in bounded sets, which are used to shrink the state and input constraints for the reduced order model.

Abstraction-based methods, originating from the field of formal methods for software verification, approximate the "concrete" system of interest with a simpler "abstraction." The abstraction can have a discrete state-space (see [40] and references therein) or a continuous state-space (see [41]). The simpler abstraction is easier to control, and an abstract controller

is assumed available. The abstract controller is then "refined" so that it can be applied to the concrete system. In [41], refining the abstract control involves passing it through an interface control law that uses the concrete state and the abstract state/input to compute the concrete control input. This interface control law guarantees that the distance between the output trajectories of the concrete and abstract systems stays within a computable bound.

The planner-tracker framework [4–10, 42] is another hierarchical control framework where a lower-fidelity "planning" model is employed for online planning and a "tracking" controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set. System safety is then guaranteed if the planner constraints, when augmented by the tracking error bound, lie within the safety constraints. A simple illustration of this framework was described in Section 1.2, and a thorough presentation will be given in Chapter 2.

Comparing the planner-tracker framework to the methods summarized above, SCP with feedback control does not formally bound the error from the planned path. Unlike the methods which compute a bound for a particular nominal trajectory and associated tracking controller, the planner-tracker framework uses a single tracking controller with a single error bound for all trajectories. The planner-tracker framework can handle both the uncertainty present in tube MPC as well as the different system dimensions present in reduced order MPC. In the planner-tracker framework, the choice of the planning model is flexible: it can remove uncertainties, linearize nonlinearities, and reduce the system dimension. Furthermore, MPC is a suitable choice of planning controller, but it is not the only option. For example, symbolic control was used in [42]. The planner-tracker framework has a structure similar to abstraction-based control, but recent works [8] and [42] eliminate the restrictive geometric conditions in [41], also implicit in [7], which require that the set where the tracking error vanishes be invariant. Removing this requirement and allowing the tracking error to depend on planner inputs greatly expand the applicability of the planner-tracker framework.

1.4 Dissertation Outline

The dissertation is organized as follows. Chapter 2 gives a tutorial of the planner-tracker framework. We expand upon the brief overview given in Section 1.2, giving more detailed descriptions of the planner, tracker, and error systems. We formulate a sum-of-squares (SOS) program to search for a tracking controller and an error bound set. We extend the error definition to depend on the planner input and formulate a new SOS program for this setting. This additional flexibility in the error definition is useful when the inputs to a kinematic planner model correspond to states in a dynamic tracker model. The goal of this Chapter is to provide a tutorial review of this hierarchical framework and to illustrate it with examples, including a design for vehicle obstacle avoidance.

Chapter 3 extends the planner-tracker framework to handle unmodeled dynamics at the input to the tracker model. This allows for systems with unknown input delays, unmodeled actuator dynamics, etc. The unmodeled dynamics are described by integral quadratic con-

straints (IQCs). A sum-of-squares (SOS) program is formulated to search for the tracking controller and error bound, and the method is demonstrated on a vehicle obstacle avoidance example with an input delay.

Chapter 4 describes adding robustness to the planner in the case study of shared vehicle control between a human driver and an autonomous system. The proposed control strategy, termed Driver-in-the-Loop Contingency Model Predictive Control (MPC), is inspired by the concept of contingency planning and is designed to intervene from the driver under emergency conditions in a manner that is smooth and not overly conservative. Driver-in-the-Loop Contingency MPC relies on the computation of invariant sets, which are used as MPC terminal sets. The computation of these sets is made possible by the simple, kinematic model used to describe the longitudinal vehicle motion, making this method a suitable choice for a planning controller in the planner-tracker framework. The Driver-in-the-Loop Contingency MPC method is demonstrated on two longitudinal traffic scenarios: (1) vehicle-following, and (2) an intersection where the cross-traffic has the right of way. We use these examples to demonstrate the safety of the controller as well as the inherent trade-off between smooth intervention and minimal intervention.

Chapter 5 formulates a backstepping-based tracking controller for the planner-tracker framework as an alternative to the SOS-based controllers used in Chapters 2 and 3. This controller mitigates some shortcomings of the SOS approach: (1) the fact that SOS control laws are a generic polynomial which may not be interpretable, and (2) the numerical challenges associated with SOS programming. However, this chapter requires a particular structure in the planner and tracker models.

Chapter 6 provides further robustness analysis of a tracking controller in a hierarchical control scheme such as the planner-tracker framework. This Chapter examines the setting where a reference trajectory is provided for a nominal system with no disturbance or modeling uncertainty. However, the true system may differ from the nominal system by unmodeled dynamics (described by an IQC) and an additive disturbance at the input. Robustness to these two sources of uncertainty is handled by (1) sampling over the unmodeled dynamics, and (2) for each sample, solving an optimal control problem over the set of possible disturbances to compute an error bound for the linearization of the system about the reference trajectory. Furthermore, we directly compute the worst-case disturbance that achieves this error bound. This Chapter provides an alternative method for computing an approximate error bound (due to sampling and linearizations) compared to the planner-tracker framework. Finally, concluding remarks and directions for future work are provided in Chapter 7.

1.5 Contributions

This thesis makes four contributions relative to the current literature. First, we extend the planner-tracker framework to accommodate unmodeled dynamics at the tracker input, characterized by integral quadratic constraints. This is covered in Chapter 3 and manuscript [??]. Next, we propose a novel safe control framework for shared vehicle control between a human

driver and a supervisory autonomous system based on the principle of contingency planning. This is covered in Chapter 4 and publication [43]. Next, we develop a novel backstepping-based control strategy for planner and tracker systems with a particular cascaded structure, where an error between the two systems is defined and a control law is derived to keep that error in a bounded set. This is covered in Chapter 5. Finally, we develop a novel method for assessing the robustness of an uncertain nonlinear system on a finite time horizon where we linearize about a nominal trajectory, sample over the model uncertainty, and then compute the disturbance that maximizes an error variable. This is covered in Chapter 6 and publication [44].

1.6 List of Publications

The chapters of this dissertation are based on other publications by the author. In particular:

- Chapter 2 is based on [10]:
  - K. S. Schweidel, H. Yin, S. W. Smith, and M. Arcak, "Safe-by-design planner–tracker synthesis with a hierarchy of system models," *Annual Reviews in Control*, vol. 53, pp. 138–146, 2022.
- Chapter 3 is based on [45]:
  - K. S. Schweidel, P.J. Seiler, and M. Arcak, "Safe-by-design planner-tracker synthesis with unmodeled input dynamics," *To appear in IEEE Control Systems Letters*.
- Chapter 4 is based on [43]:
  - K. S. Schweidel, S. M. Koehler, V. R. Desaraju and M. Barić, "Driver-in-the-Loop Contingency MPC with Invariant Sets," *2022 European Control Conference (ECC)*, London, United Kingdom, 2022, pp. 808-813.
- Chapter 6 is based on [44]:
  - K. S. Schweidel, J. R. Buch, P. J. Seiler and M. Arcak, "Computing Worst-Case Disturbances for Finite-Horizon Linear Time-Varying Approximations of Uncertain Systems," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1753-1758.

Chapter 2

Planner-Tracker Framework

2.1 Abstract

We present a safe-by-design trajectory planning and tracking framework for nonlinear dynamical systems using a hierarchy of system models. The planning layer uses a low-fidelity model to plan a feasible trajectory satisfying the planning constraints, and the tracking layer utilizes the high-fidelity model to design a controller that restricts the error states between the low- and high-fidelity models to a bounded set. The simplicity of the low-fidelity model enables the planning to be performed online (e.g. using Model Predictive Control) and the tracking controller and error bound are derived offline (e.g. using sum-of-squares programming). This error bound is then used by the planner to ensure safety for the combined planner-tracker system. To provide freedom in the choice of the low-fidelity model, we allow the tracking error to depend on both the states and inputs of the planner. The goal of this Chapter is to provide a tutorial review of this hierarchical framework and to illustrate it with examples, including a design for vehicle obstacle avoidance.

2.2 Introduction

Modern engineering systems such as autonomous vehicles and unmanned aerial vehicles (UAVs) must operate subject to complex safety and performance requirements in changing environments. Designing controllers that meet such requirements in real-time may be computationally intractable, e.g., due to large system dimension or nonlinearities in a high-fidelity dynamical model of the system. The planner-tracker framework [3–9] addresses this challenge with a layered architecture where a lower-fidelity "planning" model is employed for online planning and a "tracking" controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set. System safety is then guaranteed if the planner constraints, when augmented by the tracking error bound, lie within the safety constraints.

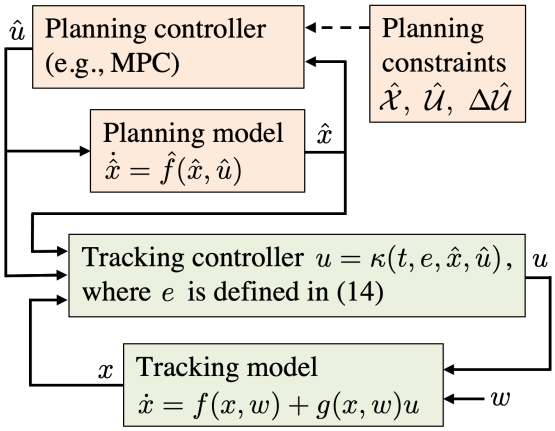
There is a choice to be made when defining the tracking error between the planner and tracker systems. In [7, 8, 46], the tracking error depends on only the planner/tracker *states*. In [42], which studies a ship control problem, the tracking error is generalized to also depend on the planner *input*. This is achieved by accounting for the jumps in the error variable that are induced by jumps in the zero-order hold input between time-steps. Including the planner input in the error definition allows for a lower-order planning model whose states mimic the tracker *position* states while the planner inputs correspond to the tracker *velocity* states. References [8] and [42] further make a connection between the layered planner-tracker architecture and the notion of abstractions introduced in [41]. In doing so, they also eliminate the restrictive geometric conditions in [41], also implicit in [7], which require that the set where the tracking error vanishes be invariant. Removing this requirement and allowing the tracking error to depend on planner inputs greatly expand the applicability of the planner-tracker framework.

In this chapter we introduce a broad framework which encompasses those earlier results while further generalizing the error definition compared to [42]. In addition, the framework described here is not restricted to a particular planner. Indeed, unlike the computationally heavy symbolic design method used for planning in [42], the numerical example presented here uses the popular choice of Model Predictive Control (MPC), which is appropriate for real-time implementation.

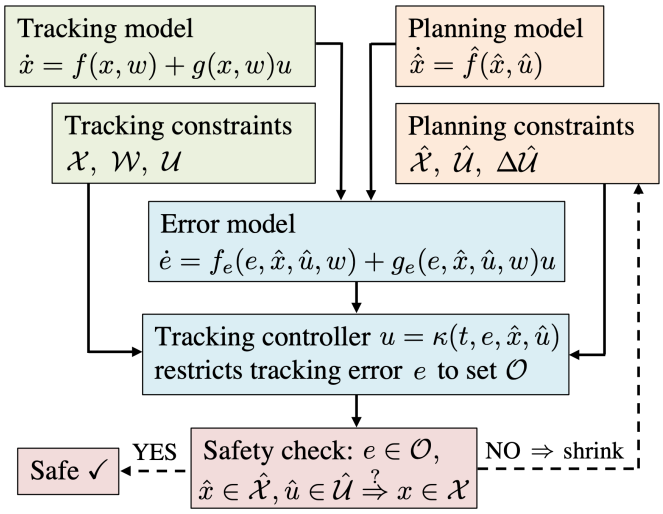
Although MPC is often used for both planning and control, under mismatch of planning model and the plant, the MPC optimization problem must be robustified. Feasibility and stability properties of robust MPC have been studied in [47, 48] and subsequent publications. For linear systems, Tube MPC [22–30] is a widely used approach that solves a computationally efficient convex optimization problem for robust control synthesis. Although Tube MPC design with feasibility and stability properties are proposed for nonlinear systems in [33, 35–37, 49, 50], control synthesis can become either too conservative, or computationally demanding.

Another related work is [20], in which multiple tracking controllers and error-bound funnels are computed for a library of nominal trajectories. By contrast, the planner-tracker framework presented here generates a single tracking controller and can accommodate any trajectory from the planner, not just one that belongs to a pre-specified library.

The remainder of this Chapter is organized as follows. Section 2.3 introduces the high-fidelity tracking model and the low-fidelity planning model and defines a simple tracking error that depends only on the planner/tracker *states*. We build intuition with this simple error model and present the method for constructing a tracking controller and an error bound using sum-of-squares (SOS) programming. Section 2.4 generalizes the tracking error definition to additionally depend on the planner *input* and extends the results in Section 2.3 to handle this generalized error. In Section 2.5, we demonstrate the method on a vehicle obstacle avoidance example, and we provide concluding remarks in Section 2.6.



(a) Online implementation. The planning controller uses the state $\hat{x}$ and the constraint sets $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\Delta\hat{\mathcal{X}}$ to generate a reference input $\hat{u}$. The tracking controller converts this into a control u which is guaranteed to keep the tracker state x within state constraints $\mathcal{X}$. This is accomplished by keeping the tracking error e within a set $\mathcal{O}$ as described in Figure 2.1(b).



(b) Offline synthesis. The tracking and planning models are combined to obtain a model for the error system. A controller κ is derived to keep the error in the set $\mathcal{O}$ (e.g., using SOS). If the safety condition is met, the tracking system is guaranteed to satisfy the state constraints $\mathcal{X}$. Otherwise, the sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are shrunk and the process is repeated.

Figure 2.1: Online implementation and offline synthesis of the planner-tracker control scheme.

Notation

$\mathbb{S}^n$ denotes the set of n-by-n symmetric matrices. $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$ denote the sets of n-by-n symmetric positive semi-definite and positive definite matrices, respectively. For $\xi \in \mathbb{R}^n$, $\mathbb{R}[\xi]$ represents the set of polynomials in ξ with real coefficients, and $\mathbb{R}^m[\xi]$ and $\mathbb{R}^{m\times p}[\xi]$ denote all vector and matrix valued polynomial functions. The subset $\Sigma[\xi] := \{p = p_1^2 + p_2^2 + ... + p_M^2 : p_1, ..., p_M \in \mathbb{R}[\xi]\}$ of $\mathbb{R}[\xi]$ is the set of sum-of-squares polynomials in ξ. Unless defined otherwise, notation x^j denotes a variable x used in the j'th iteration of an iterative algorithm. The symbol "$\leq$" represents component-wise inequality.

2.3 Problem setup

In this section we describe the hierarchical approach to safe-by-design trajectory planning and control that consists of two layers: a planning layer, which uses a low-fidelity "planning" model, and a tracking layer, with a high-fidelity "tracking" model. The planning model might be a model with a lower state dimension than the tracking model or a linearization of the tracking model to reduce the computational burden of planning. By analyzing the dynamics of the error between these two systems' states, we will show how we can bound this error by synthesizing an appropriate tracking controller. In this Chapter, the controller

and corresponding error bound are designed via SOS programming.

The online implementation and offline synthesis of the planner-tracker control scheme are summarized in Figure 2.1. We begin with a high-fidelity tracking model and a low-fidelity planning model, each with state and input constraints. Defining an appropriate error variable, e, between the two models, and using the error dynamics and the planner/tracker constraints, we design a tracking controller and derive a tracking error bound. This bound takes the form of a set $\mathcal{O}$ such that $e(t) \in \mathcal{O}$. If the planner constraints, when augmented by $\mathcal{O}$, still satisfy the tracking constraints, then the tracking system is safe: it will satisfy all constraints with the synthesized controller. Otherwise, the planner constraints are shrunk and the process is repeated.

High-Fidelity Tracking Model

The high-fidelity model is of the form:

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

with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, disturbance $w(t) \in \mathcal{W} \subseteq \mathbb{R}^{n_w}$, bounded control $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$. The sets $\mathcal{X}$ and $\mathcal{U}$ are the constraint sets imposed on the states and control inputs in the high-fidelity model, respectively.

Low-Fidelity Planning Model

The low-fidelity model, which is a simplified version of (2.1), is of the form:

$$\dot{\hat{x}}(t) = \hat{f}(\hat{x}(t), \hat{u}(t)), \tag{2.2}$$

where $\hat{x}(t) \in \hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{u}(t) \in \hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, and $\hat{f} : \mathbb{R}^{\hat{n}_x \times \hat{n}_u} \to \mathbb{R}^{\hat{n}_x}$. The sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are constraint sets enforced by the planning layer. The control input for the low-fidelity model, computed via the planning algorithm of choice, is assumed to be a zero-order hold signal with sampling time $T_s > 0$. This means:

$$\hat{u}(t) = \hat{u}(\tau_k), \ \forall t \in [\tau_k, \tau_{k+1}), \ \text{with } \tau_k = k \cdot T_s, \tag{2.3a}$$

$$\hat{u}(\tau_{k+1}) = \hat{u}(\tau_k) + \Delta\hat{u}(\tau_{k+1}), \tag{2.3b}$$

where $\Delta\hat{u}(t)$ is the change in the control input between sampling periods (also referred to as the "input jump"), restricted to a set $\Delta\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$.

Note that the planning model does not depend directly on the tracker state x, in contrast to reduced order methods where planning is done on a lower dimensional state that is a projection of the true higher dimensional state [38].

Remark 1. *The planner-tracker synthesis framework is applicable to any planning algorithm that is able to bound $\hat{x}(t), \hat{u}(t)$, and $\Delta\hat{u}(t)$. For example, this framework has been applied to different planning algorithms, using Nonlinear MPC in [8, 46], signal temporal logic (STL) in [51], and discrete abstraction in [42].*

Error Dynamics

The goal is to design a controller for the high-fidelity tracking model (2.1) to track a reference trajectory planned using the low-fidelity planning model (2.2). In order to do so, we proceed by deriving the evolution of the error between (2.1) and (2.2). Since $\hat{n}_x \leq n_x$ in general, we define a $\mathcal{C}^1$ map $\pi : \mathbb{R}^{\hat{n}_x} \to \mathbb{R}^{n_x}$, called the *comparison* map, and we define the tracking error as:

$$e(t) = x(t) - \pi(\hat{x}(t)). \tag{2.4}$$

If the planning model is simply a linearization, we may select π to be the identity map, but our primary interest is in the case where $\hat{x}$ is of lower dimension and π lifts it to the dimension of x. We will first describe the method with this simple error definition to build intuition before generalizing the error definition in Section 2.4, where π is allowed to also depend on $\hat{u}$.

Differentiating (2.4) with respect to time (dropping time arguments to improve readability), and eliminating the variable x, we obtain:

$$\begin{aligned}
\dot{e} &= \dot{x} - \frac{\partial \pi}{\partial \hat{x}} \cdot \dot{\hat{x}} \\
&= f(x,w) + g(x,w) \cdot u - \frac{\partial \pi}{\partial \hat{x}} \cdot \hat{f}(\hat{x},\hat{u})\Big|_{x=e+\pi(\hat{x})}, \\
&= f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},w) \cdot u,
\end{aligned} \tag{2.5}$$

where we have defined:

$$\begin{aligned}
f_e(e,\hat{x},\hat{u},w) &= f(\pi(\hat{x}) + e, w) - \frac{\partial \pi}{\partial \hat{x}} \cdot \hat{f}(\hat{x},\hat{u}), \\
g_e(e,\hat{x},w) &= g(\pi(\hat{x}) + e, w).
\end{aligned} \tag{2.6}$$

In this section, we consider controllers of the form

$$u(t) = \kappa(e(t), \hat{x}(t), \hat{u}(t)),\ \kappa \in \mathcal{K}_{\mathcal{U}} \tag{2.7}$$

where the set $\mathcal{K}_{\mathcal{U}} := \{\kappa : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathcal{U}\}$ defines a set of admissible error-state feedback control laws. Plugging in this controller, the closed-loop dynamics become

$$\dot{e} = f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},w) \cdot \kappa(e,\hat{x},\hat{u}). \tag{2.8}$$

Our goal is to design a control law κ and an associated error bound $\mathcal{O}$ that is ideally as small as possible.

Definition 1 (Tracking Error Bound)**.** *Given the closed loop error dynamics* (2.8)*, we say that a set $\mathcal{O}$ is a tracking error bound (TEB) from an initial set $\mathcal{I}$ if*

$$\begin{aligned}
e(0) \in \mathcal{I},\ \hat{x}(t) \in \hat{\mathcal{X}},\ \hat{u}(t) \in \hat{\mathcal{U}},\ w(t) \in \mathcal{W}\ \forall\ t \geq 0 \\
\Rightarrow\ e(t) \in \mathcal{O}\ \forall\ t \geq 0.
\end{aligned} \tag{2.9}$$

The initial set $\mathcal{I}$ will be constructed together with the TEB $\mathcal{O}$. In Theorem 1 in Section 2.3, $\mathcal{I}$ and $\mathcal{O}$ will be identical, while in Theorem 2 in Section 2.4, $\mathcal{I} \subseteq \mathcal{O}$.

As we will see in the next subsection, we aim to minimize the volume of the set $\mathcal{O}$ when designing the tracking controller κ; however, we do not emphasize asymptotic behavior of the error $e(t)$ since we do not need perfect tracking of the planning model. Indeed we allow the dynamics (2.8) to depend on $\hat{x}$ besides $\hat{u}$ and w, and we do not require the right-hand side to vanish when $e = 0$. The benefit of this relaxed approach, as alluded to in the Introduction, is to remove restrictive geometric constraints from the selection of the map π and controller κ that would render the set $e = x - \pi(\hat{x}) = 0$ invariant and attractive.

Computing the TEB and Tracking Controller

The TEB $\mathcal{O}$ and the tracking controller κ can be obtained with the help of the following theorem, which gives conditions under which the error can be constrained to lie within a certain sublevel set of a storage function $V(e)$.

Theorem 1. *Given the error dynamics (2.5) with $f_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, $g_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, and $\gamma \in \mathbb{R}$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\mathcal{W} \subseteq \mathbb{R}^{n_w}$, if there exists a $\mathcal{C}^1$ function $V : \mathbb{R}^{n_x} \to \mathbb{R}$ and $\kappa : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$ such that*

$$\begin{aligned} &\frac{\partial V(e)}{\partial e} \cdot (f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, w) \cdot \kappa(e, \hat{x}, \hat{u})) < 0, \\ &\forall e, \hat{x}, \hat{u}, w, \ \text{s.t. } V(e) = \gamma, \ \hat{x} \in \hat{\mathcal{X}}, \ \hat{u} \in \hat{\mathcal{U}}, \ w \in \mathcal{W}, \end{aligned} \tag{2.10}$$

then the sublevel set $\Omega(V, \gamma) := \{e \in \mathbb{R}^{n_x} : V(e) \leq \gamma\}$ is a TEB as in Definition 1 with $\mathcal{I} = \mathcal{O} = \Omega(V, \gamma)$.

Proof. The theorem is proved by contradiction. Assume there exist a time $t_2 > 0$ and a trajectory $e(\cdot)$ such that $e(0) \in \Omega(V, \gamma)$ but $e(t_2) \notin \Omega(V, \gamma)$, i.e., $V(e(t_2)) > \gamma$. Since $V(e(0)) \leq \gamma$, by continuity of V there exists t_1 such that $0 \leq t_1 < t_2$, $V(e(t_1)) = \gamma$, and $\frac{d}{dt} V(e(t))|_{t=t_1} \geq 0$. (If all crossings of $V(e(t)) = \gamma$ satisfied $\frac{d}{dt} V(e(t)) < 0$, then V would not be continuous.) This contradicts (2.10). □

It is straightforward to augment the statement above to ensure a bound on the input. Adding the following constraint ensures $u = \kappa(e, \hat{x}, \hat{u}) \in \mathcal{U}$:

$$\Omega(V, \gamma) \subseteq \{e \in \mathbb{R}^{n_x} : \kappa(e, \hat{x}, \hat{u}) \in \mathcal{U}\} \ \forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}. \tag{2.11}$$

Furthermore, if a user-specified initial error set $\mathcal{E}_0$ is known, then adding the set constraint

$$\Omega(V, \gamma) = \mathcal{I} \supseteq \mathcal{E}_0 \tag{2.12}$$

will ensure that $e(t) \in \mathcal{O}$ for all $t \geq 0$. As a practical consideration, we add such a constraint when searching for V and κ.

Finding generic functions V and κ that satisfy constraints (2.10), (2.11), and (2.12) is a difficult problem. Below we show how SOS programming can be used to search for these functions by restricting to polynomial candidates $V \in \mathbb{R}[e]$ and $\kappa \in \mathbb{R}^{n_u}[(e, \hat{x}, \hat{u})]$. Besides this restriction, we make the following assumption:

Assumption 1. *The mappings $f_e \in \mathbb{R}^{n_x}[(e, \hat{x}, \hat{u}, w)]$ and $g_e \in \mathbb{R}^{n_x \times n_u}[(e, \hat{x}, w)]$ in error dynamics (2.5) are polynomials. Sets $\mathcal{E}_0$, $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ are semi-algebraic sets, i.e., there exists $p_0 \in \mathbb{R}[e]$ such that $\mathcal{E}_0 = \{e \in \mathbb{R}^{n_x} \ : \ p_0(e) \leq 0\}$; with similar definitions for $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ with polynomials $p_{\hat{x}} \in \mathbb{R}[\hat{x}]$, $p_{\hat{u}} \in \mathbb{R}[\hat{u}]$, and $p_w \in \mathbb{R}[w]$. The control constraint set $\mathcal{U}$ is a hypercube $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : \underline{u} \leq u \leq \overline{u}\}$, where $\underline{u}, \overline{u} \in \mathbb{R}^{n_u}$.*

By applying the generalized S-procedure [52] to the set containment constraints (2.10), (2.11), and (2.12), and using the volume of $\Omega(V, \gamma)$ as the cost function to minimize, we obtain the following SOS optimization problem for finding V and κ:

$$
\begin{aligned}
\min_{V,\kappa,\gamma,s,l} \ & \text{volume}(\Omega(V,\gamma)) \\
\text{s.t. } & s_0 \in \Sigma[e], s_{1\to 3} \in \Sigma[(e,\hat{x},\hat{u},w)], l \in \mathbb{R}[(e,\hat{x},\hat{u},w)] \\
& s_{4\to 9,i} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,...,n_u\} && \text{(2.13a)} \\
& -(V(e)-\gamma) + s_0 \cdot p_0 \in \Sigma[e], && \text{(2.13b)} \\
& -\frac{\partial V}{\partial e} \cdot (f_e + g_e \cdot \kappa) - \epsilon e^\top e + l \cdot (V-\gamma) + s_1 \cdot p_{\hat{x}} \\
& \quad + s_2 \cdot p_{\hat{u}} + s_3 \cdot p_w \in \Sigma[(e,\hat{x},\hat{u},w)], && \text{(2.13c)} \\
& \overline{u}_i - \kappa_i + s_{4,i} \cdot (V-\gamma) + s_{5,i} \cdot p_{\hat{x}} \\
& \quad + s_{6,i} \cdot p_{\hat{u}} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,...,n_u\}, && \text{(2.13d)} \\
& \kappa_i - \underline{u}_i + s_{7,i} \cdot (V-\gamma) + s_{8,i} \cdot p_{\hat{x}} \\
& \quad + s_{9,i} \cdot p_{\hat{u}} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,...,n_u\}. && \text{(2.13e)}
\end{aligned}
$$

In the formulation above, SOS polynomials $s_{1\to 3}$ and $s_{4\to 9,i}$ are multipliers used in the generalized S-procedure, and $\epsilon > 0$ is on the order of 10^{-6}. Constraint (2.13a) ensures that all the polynomial multipliers are SOS. Constraint (2.13b) is a relaxation of (2.12) (for $\mathcal{I} = \mathcal{O} = \Omega(V, \gamma)$), (2.13c) is a relaxation of (2.10), and together (2.13d) and (2.13e) are a relaxation of (2.11) under the hypercube assumption for $\mathcal{U}$. The optimization (2.13) is non-convex as there are two groups of decision variables V and $(\kappa, l, s_{4,i}, s_{7,i})$ bilinear in each other. To tackle this problem, similarly to [53, Algorithm 1], we decompose it into two tractable subproblems to iteratively search between the two groups of decision variables, as shown in Algorithm 1 in the Appendix. We note that V and γ always appear in the optimization as $V - \gamma$, so from a theoretical perspective there is no need for two separate optimization variables. However, the variable γ is practical algorithmically because in the subproblem where V is fixed, we can minimize over γ via bisection to find the smallest level set of V that forms a viable TEB.

Remark 2. *For simplicity, we define V to be a function of e. However, in principle, we could have defined $V(x, \hat{x})$, as is done in the incremental stability literature, e.g., [54].*

Safety Check

After synthesizing V and κ, we check the following safety condition with $\mathcal{O} = \Omega(V, \gamma)$:

$$\pi(\hat{\mathcal{X}}) \oplus \mathcal{O} \subseteq \mathcal{X}. \tag{2.14}$$

If (2.14) is satisfied, then the tracker state x is guaranteed to satisfy state constraints $\mathcal{X}$ and the design is considered successful. If (2.14) is *not* satisfied, we shrink the planner sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ and repeat the process as indicated in Figure 2.1.

The details of how to shrink the sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are not the focus of this Chapter, but we refer the reader to [46] for an in-depth treatment. In [46], the constraint sets are parameterized by some parameter, and a bisection over this parameter is performed to find the most permissive sets that still guarantee safety.

2.4 Generalized Tracking Error Definition

So far, we have used a map π that only depends on the planner state $\hat{x}$. However, as illustrated in the example below, this map may fail to provide reference signals for all the tracker states. Therefore, in Section 2.4, we move to a more general error definition that also depends on the planner input $\hat{u}$.

Example 1. *As a simple illustration of why it is useful to include the planner input $\hat{u}$ in the error definition, consider the double integrator tracking model*

$$x = \begin{bmatrix} s \\ v \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} v \\ u \end{bmatrix}, \tag{2.15}$$

where s is the position, v is the velocity, and u is the acceleration input. Let the planning model be a single integrator, where the only state is the planner position ($\hat{x} = \hat{s}$) and the input is the planner velocity ($\dot{\hat{x}} = \hat{v} =: \hat{u}$). Then, letting $\pi(\hat{x}, \hat{u}) = [\hat{x}; \hat{u}]$, the error is

$$e = x - \pi(\hat{x}, \hat{u}) = \begin{bmatrix} s - \hat{s} \\ v - \hat{v} \end{bmatrix}, \tag{2.16}$$

which is the deviation of the planner and tracker positions and velocities. Thus, by keeping e small, we keep the planner and tracker positions and velocities close to one another, which is desirable. On the other hand, if we had used the naïve map $\pi(\hat{x}) = [\hat{x}; 0]$, the error would be $[(s - \hat{s}) \;\; v]^\top$, and bounding the error would mean keeping v close to zero, which is overly conservative and may not align with planning objectives.

Modified Error Dynamics

As motivated above, we will use a more general $\mathcal{C}^1$ map $\pi : \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x}$ to provide better reference trajectories for the tracking model, as was done in [42] for the first time. For further generality, in this Chapter we redefine the error state as

$$e = \phi(x, \hat{x}, \hat{u})(x - \pi(\hat{x}, \hat{u})), \tag{2.17}$$

where we add the $\mathcal{C}^1$ map $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x \times n_x}$ which provides additional flexibility, as will be demonstrated in Section 2.5.

Assume that for each e, $\hat{x}$, $\hat{u}$, there exists a unique x satisfying (2.17), and denote this inverse as

$$x = \nu(e, \hat{x}, \hat{u}). \tag{2.18}$$

The error dynamics resulting from (2.17) are

$$\dot{e} = f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)u - h_e(e, \hat{x}, \hat{u})\dot{\hat{u}}, \tag{2.19}$$

where

$$\begin{aligned} f_e(e, \hat{x}, \hat{u}, w) := &\left\{ \frac{\partial \phi}{\partial x} f(x, w) + \frac{\partial \phi}{\partial \hat{x}} \hat{f}(\hat{x}, \hat{u}) \right\} (x - \pi(\hat{x}, \hat{u})) \\ &+ \phi(x, \hat{x}, \hat{u}) \left\{ f(x, w) - \frac{\partial \pi}{\partial \hat{x}} \hat{f}(\hat{x}, \hat{u}) \right\} \Bigg|_{x = \nu(e, \hat{x}, \hat{u})}, \end{aligned} \tag{2.20}$$

$$\begin{aligned} g_e(e, \hat{x}, \hat{u}, w) := &\left\{ \frac{\partial \phi}{\partial x} (x - \pi(\hat{x}, \hat{u})) \right. \\ &\left. + \phi(x, \hat{x}, \hat{u}) \right\} g(x, w) \Big|_{x = \nu(e, \hat{x}, \hat{u})}, \end{aligned} \tag{2.21}$$

and h_e can be computed but is not written explicitly since it multiplies $\dot{\hat{u}}$, which is zero within sampling periods.

Note that the planner input is applied in a zero order hold fashion within each sampling period as described in (2.3). As the tracking error dynamics (2.19) have a term containing $\dot{\hat{u}}$ (unlike (2.5)), these dynamics change discontinuously at each sampling instant τ_k. Therefore, we break up the error analysis into two parts. In Section 2.4, we bound the error *within* a single sampling period $[\tau_{k-1}, \tau_k)$. In Section 2.4, we bound the error jump *across* sampling periods from τ_k^- to τ_k^+ induced by the input jump $\Delta\hat{u}_k$. This two-part approach can be visualized in Figure 2.2.

Analysis Within Sampling Periods

Since the signal $\hat{u}$ is piece-wise constant, within a single sampling period we have

$$\dot{\hat{u}}(t) = 0, \ \forall t \in [\tau_{k-1}, \tau_k). \tag{2.22}$$

Therefore, the error dynamics (2.19) during the time interval $[\tau_{k-1}, \tau_k)$ are:

$$\dot{e} = f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)u. \tag{2.23}$$

We want to enforce the boundedness of the error state during $[0, T_s)$ by introducing a tracking controller

$$u(t) = \kappa(t, e(t), \hat{x}(t), \hat{u}(t)), \tag{2.24}$$

which is now defined by a *time-varying*, error-state feedback control law $\kappa : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$. We use a time-varying controller κ and a time-varying storage function V in this section for additional flexibility. In particular, we may find a storage functions whose level sets shrink as time increases. The main idea is that if the level set at the end of the sampling period is sufficiently smaller than the level set at the beginning of the sampling period, then this size difference can accommodate the error jump across sampling periods as in Figure 2.2.

Below we provide conditions on κ and V for bounding the error in a level set of V within each sampling period, where now each level set of V is a *funnel* in (e, t) space.

Proposition 1. *Given the error dynamics* (2.23) *with mappings* $f_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, $g_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, *and* $\gamma \in \mathbb{R}$, $T_s > 0$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\mathcal{W} \subseteq \mathbb{R}^{n_w}$, *suppose there exists a* $\mathcal{C}^1$ *function* $V : \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}$, *and* $\kappa : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$, *such that*

$$\begin{aligned} &\frac{\partial V(t,e)}{\partial e} \cdot (f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w) \cdot \kappa(t, e, \hat{x}, \hat{u})) \\ &\qquad + \frac{\partial V(t,e)}{\partial t} < 0, \ \forall t, e, \hat{x}, \hat{u}, w, \ \text{s.t. } t \in [0, T_s), \\ &\qquad\qquad V(t,e) = \gamma, \ \hat{x} \in \hat{\mathcal{X}}, \ \hat{u} \in \hat{\mathcal{U}}, \ w \in \mathcal{W}. \end{aligned} \tag{2.25}$$

Define the funnel $\Omega(V, t, \gamma) := \{e \in \mathbb{R}^{n_x} : V(t, e) \leq \gamma\}$. *If* $e(0) \in \Omega(V, 0, \gamma)$, *then* $e(t) \in \Omega(V, t, \gamma)$ *for all* $t \in [0, T_s)$.

Proof. The proof is a simple modification of the proof of Theorem 1, where now $\dot{V}(t, e(t)) := \frac{\partial V(t,e)}{\partial e}\dot{e}(t) + \frac{\partial V(t,e)}{\partial t}$ contains the additional $\partial V/\partial t$ term from (2.25). □

Remark 3. *Although Proposition 1 is stated for the first sampling period* $[0, T_s)$, *it can be used for any other sampling period* $[\tau_k, \tau_{k+1})$ *with* $\tau_k = k \cdot T_s$. *Let* $e(\tau_k) \in \Omega(V, 0, \gamma)$. *Then we have* $e(\tau_k + t) \in \Omega(V, t, \gamma)$, *for all* $t \in [0, T_s)$, *under the control signal* $u(\tau_k + t) = \kappa(t, e(\tau_k + t), \hat{x}(\tau_k + t), \hat{u}(\tau_k + t)))$.

Analysis Across Sampling Periods

Next, we focus on the effect of the input jump $\Delta\hat{u}$ at each sampling instant τ_k as in (2.3b). From (2.17), $\Delta\hat{u}$ induces a jump on the error. Let τ_k^- and τ_k^+ denote sampling instant τ_k

before and after the discrete jump, respectively, and for simplicity use the notation $e_k^+ := e(\tau_k^+)$. Then we have

$$\begin{aligned} e_k^+ &= \phi(x_k^+, \hat{x}_k^+, \hat{u}_k^+)(x_k^+ - \pi(\hat{x}_k^+, \hat{u}_k^+)) \\ &= \phi(x_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)(x_k^- - \pi(\hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)) \\ &= \phi(\nu(e_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+), \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+) \\ &\qquad \cdot (\nu(e_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+) - \pi(\hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)) \\ &=: h(e_k^-, \hat{x}_k^-, \hat{u}_k^-, \Delta\hat{u}_k^+). \end{aligned} \tag{2.26}$$

We refer to h as the *jump function*, as it reflects how the error may jump from the end of one sampling period to the beginning of the next, due to the jump in the input.

We introduce the additional condition below to characterize the error jump induced by the control jump $\Delta\hat{u}$ in terms of the funnel $\Omega(V, t, \gamma)$.

Proposition 2. *Given $\gamma \in \mathbb{R}$, $T_s \in \mathbb{R}$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\Delta\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $h : \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x}$, if there exists a function $V : \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}$ satisfying*

$$\begin{aligned} &V(0, h(e, \hat{x}, \hat{u}, \Delta\hat{u})) \leq \gamma, \\ &\qquad \forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}, \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \text{ and } \forall e \text{ s.t. } V(T_s, e) \leq \gamma \end{aligned} \tag{2.27}$$

then for all $e_k^- \in \Omega(V, T_s, \gamma)$, $e_k^+ \in \Omega(V, 0, \gamma)$.

Proof. Suppose $e_k^- \in \Omega(V, T_s, \gamma)$, i.e., $V(T_s, e_k^-) \leq \gamma$. By Eq. 2.26, $e_k^+ = h(e_k^-, \hat{x}_k^-, \hat{u}_k^-, \Delta\hat{u}_k^+)$. Thus by Eq. 2.27, $V(0, e_k^+) \leq \gamma$, i.e., $e_k^+ \in \Omega(V, 0, \gamma)$. □

Remark 4. *For the special case $\phi(x, \hat{x}, \hat{u}) = 1$ and $\pi(\hat{x}, \hat{u}) = \theta(\hat{x}) + Q\hat{u}$ for some $Q \in \mathbb{R}^{n_x \times \hat{n}_u}$, the $\hat{x}$ and $\hat{u}$ terms cancel, and so (2.27) simplifies to*

$$V(0, e - Q\Delta\hat{u}) \leq \gamma, \ \forall \Delta\hat{u} \in \Delta\hat{\mathcal{U}}, \ \forall e \text{ s.t. } V(T_s, e) \leq \gamma.$$

Combining Within- and Across-Sample Analysis

We next combine the conditions for within- and across-sample error boundedness from Propositions 1 and 2, respectively, to obtain the main result on the boundedness of the error at all time, formulated below and illustrated in Figure 2.2.

Theorem 2. *If there exist V and κ satisfying (2.25) and (2.27), define $\mathcal{O} \subset \mathbb{R}^{n_x}$ such that*

$$\cup_{t \in [0, T_s)} \Omega(V, t, \gamma) \subseteq \mathcal{O}.$$

Then for all $\hat{x}(t) \in \hat{\mathcal{X}}$, $\hat{u}(t) \in \hat{\mathcal{U}}$, $\Delta\hat{u}(t) \in \Delta\hat{\mathcal{U}}$, and $w(t) \in \mathcal{W}$, the error system (2.19) under control law $u(t) = \kappa(\tilde{t}, e(t), \hat{x}(t), \hat{u}(t)))$ with $\tilde{t} = (t \mod T_s) \in [0, T_s)$ satisfies:

$$e(0) \in \Omega(V, 0, \gamma) = \mathcal{I} \ \Rightarrow \ e(t) \in \mathcal{O}, \ \forall t \geq 0,$$

that is to say, $\mathcal{O}$ is a TEB from $\mathcal{I}$ achieved by the tracking control law κ.

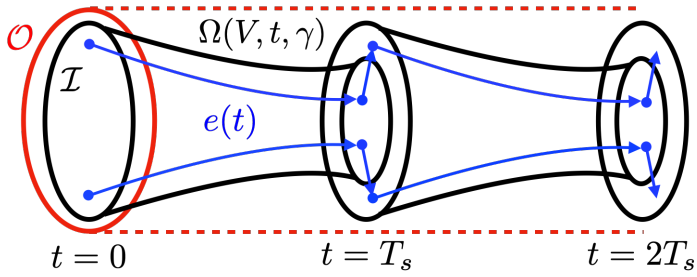


Figure 2.2: Illustration of Theorem 2, with initial error set $\mathcal{I} = \Omega(V, 0, \gamma)$, funnels $\Omega(V, t, \gamma)$ over two sampling periods, bounded error jumps at sampling times, and TEB $\mathcal{O}$.

Proof. From Remark 3 and for all $\tau_k = k \cdot T_s$, we have if $e(\tau_k) \in \Omega(V, 0, \gamma)$, then $e(\tau_k + \tilde{t}) \in \Omega(V, \tilde{t}, \gamma)$ and $e(\tau_{k+1}^-) \in \Omega(V, T_s, \gamma)$. Then it follows from Proposition 2 that $e(\tau_{k+1}^+) \in \Omega(V, 0, \gamma)$. As a result, for all $e(0) \in \Omega(V, 0, \gamma)$, we have $e(k \cdot T_s + \tilde{t}) \in \Omega(V, \tilde{t}, \gamma) \subseteq \mathcal{O}$, for all $k \geq 0$, and $\tilde{t} \in [0, T_s)$. □

Example 2. *We now revisit the planner/tracker dynamics from Example 1 and perform the analysis above to bound the error within and across sampling periods. Instead of using SOS to search for a storage function, we propose a fixed form of a controller, and we construct a storage function that satisfies the conditions of Theorem 2 above.*

Within sampling periods, $\dot{\hat{u}} = 0$, so the open loop error dynamics are

$$\dot{e} = \begin{bmatrix} \dot{s} - \dot{\hat{s}} \\ \dot{v} - \dot{\hat{v}} \end{bmatrix} = \begin{bmatrix} v - \hat{v} \\ u - 0 \end{bmatrix} = \begin{bmatrix} e_2 \\ u \end{bmatrix}. \tag{2.28}$$

Selecting a state-feedback controller $u(e) = -k_1 e_1 - k_2 e_2$, the closed loop error dynamics are

$$\dot{e} = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} e =: Ae. \tag{2.29}$$

Because this is a LTI system, constructing a Lyapunov function is straightforward. If there exists $P = P^\top > 0$ such that $PA + A^\top P < -\alpha P$ for some $\alpha > 0$, then it is simple to show that $V(t, e) = \exp(\alpha t) \cdot e^\top P e$ satisfies $\dot{V}(t, e) < 0$ for all $e \neq 0$. Hence, for any $\gamma > 0$, $\cup_{t \in [0, T_s)} \Omega(V, t, \gamma)$ forms a valid TEB from $\mathcal{I} = \Omega(V, 0, \gamma)$ by Theorem 2. We can examine the form of the level sets to see how they shrink with time

$$\Omega(V, t, \gamma) = \left\{ e \in \mathbb{R}^{n_x} : e^\top P e \leq \frac{\gamma}{\exp(\alpha t)} \right\}. \tag{2.30}$$

As t increases, e is forced to lie in smaller and smaller ellipsoids. Then the jump condition is

$$\left(e - \begin{bmatrix} 0 \\ \Delta\hat{u} \end{bmatrix} \right)^\top P \left(e - \begin{bmatrix} 0 \\ \Delta\hat{u} \end{bmatrix} \right) \leq \gamma \tag{2.31}$$
$$\text{for all } \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \text{ and } e \text{ s.t. } e^\top P e \leq \frac{\gamma}{\exp(\alpha T_s)},$$

meaning that if the error lies in the smallest ellipsoid at the end of the sampling period, then for all values of $\Delta\hat{u}$ the perturbed error will lie in the largest ellipsoid at the start of the next sampling period, as illustrated in Figure 2.2.

The variable e can be eliminated by maximizing the left hand side of (2.31) over e and observing that the optimizer e^* is aligned with $[0; \Delta\hat{u}]$. Then (2.31) can be simplified to

$$\Delta\hat{u}^\top P_{22}\Delta\hat{u} \leq \gamma\left(1 - \exp(-\tfrac{1}{2}\alpha T_s)\right)^2 \ \forall\ \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \tag{2.32}$$

where $P_{22} \in \mathbb{R}^{\hat{n}_u \times \hat{n}_u}$ is the lower right block of P. Furthermore, if $\Delta\hat{\mathcal{U}}$ is a polytope, condition (2.32) can simply be checked at the vertices of $\Delta\hat{\mathcal{U}}$ due to the convexity of the expression $\Delta\hat{u}^\top P_{22}\Delta\hat{u}$.

Remark 5. *As we saw in Example 2, it's not always necessary to use a SOS tracking controller. SOS is a versatile option since it can handle general polynomial systems, but for a given system, a practitioner may wish to use a fixed controller or a fixed form of a controller with some parameters to be determined. This fixed or parameter-dependent κ can be plugged into the SOS optimization, rather than leaving κ as a totally free decision variable. Then the SOS optimization can search for V, γ, and any parameters in κ assuming the optimization remains convex in these parameters. Otherwise, an iterative search can be performed over the parameters of κ.*

SOS Optimization

Again, to use SOS optimization to search for V and κ, we restrict them to polynomials: $V \in \mathbb{R}[(t, e)]$, and $\kappa \in \mathbb{R}[(t, e, \hat{x}, \hat{u})]$. We further assume that f_e (2.20), g_e (2.21), and the jump function h (2.26) are polynomials. In addition to Assumption 1, we assume $\Delta\hat{\mathcal{U}} = \{\Delta\hat{u} \in \mathbb{R}^{\hat{n}_u} : p_\Delta(\Delta\hat{u}) \leq 0\}$, where $p_\Delta \in \mathbb{R}[\Delta\hat{u}]$. Similarly to (2.11), we enforce tracking input constraints $u \in \mathcal{U}$ via the constraint

$$\begin{aligned}&\{e \in \mathbb{R}^{n_x} : V(t, e) \leq \gamma\} \subseteq \{e \in \mathbb{R}^{n_x} : \kappa(t, e, \hat{x}, \hat{u}) \in \mathcal{U}\},\\ &\quad \forall (t, \hat{x}, \hat{u}) \in [0, T_s) \times \hat{\mathcal{X}} \times \hat{\mathcal{U}}.\end{aligned} \tag{2.33}$$

By choosing the integral of the volume of $\Omega(V, t, \gamma)$ over the time interval $[0, T_s]$ as the cost function, and applying the generalized S-procedure to (2.12), (2.25), (2.27), and (2.33), we obtain the following optimization problem:

$$\begin{aligned}\min_{V,\kappa,\gamma,s,l}\ &\int_0^{T_s} \text{volume}(\Omega(V, t, \gamma))dt\\ \text{s.t. }\ &s_{1\to4} \in \Sigma[(t, e, \hat{x}, \hat{u}, w)], s_{5\to6} \in \Sigma[(e, \Delta\hat{u})],\\ &l \in \mathbb{R}[(t, e, \hat{x}, \hat{u}, w)], s_0 \in \Sigma[e],\\ &s_{7\to14,i} \in \Sigma[(t, e, \hat{x}, \hat{u})], i \in \{1, ..., n_u\},\end{aligned} \tag{2.34a}$$

$$\gamma - V(0,e) + s_0 \cdot p_0 \in \Sigma[e], \tag{2.34b}$$

$$\begin{aligned} &-\left(\frac{\partial V}{\partial t} + \frac{\partial V}{\partial e}\cdot(f_e + g_e\kappa)\right) - \epsilon e^\top e + l\cdot(V-\gamma) \\ &\quad + s_1\cdot p_{\hat{x}} + s_2 \cdot p_{\hat{u}} + s_3 \cdot p_w - s_4 \cdot t(T_s - t) \\ &\quad \in \Sigma[(t,e,\hat{x},\hat{u},w)], \end{aligned} \tag{2.34c}$$

$$\begin{aligned} &-(V(0,h(e,\hat{x},\hat{u},\Delta\hat{u})) - \gamma) + s_5\cdot(V(T_s,e)-\gamma) \\ &\quad + s_6\cdot p_\Delta \in \Sigma[(e,\Delta\hat{u})], \end{aligned} \tag{2.34d}$$

$$\begin{aligned} &\overline{u}_i - \kappa_i + s_{7,i}\cdot(V-\gamma) - s_{8,i}\cdot t(T_s-t) + s_{9,i}\cdot p_{\hat{x}} \\ &\quad + s_{10,i}\cdot p_{\hat{u}} \in \Sigma[(t,e,\hat{x},\hat{u})], i \in \{1,...,n_u\}, \end{aligned} \tag{2.34e}$$

$$\begin{aligned} &\kappa_i - \underline{u}_i + s_{11,i}\cdot(V-\gamma) - s_{12,i}\cdot t(T_s-t) + s_{13,i}\cdot p_{\hat{x}} \\ &\quad + s_{14,i}\cdot p_{\hat{u}} \in \Sigma[(t,e,\hat{x},\hat{u})], i \in \{1,...,n_u\}. \end{aligned} \tag{2.34f}$$

Note that the condition $t \in [0, T_s]$ is reformulated in (2.34e)-(2.34f) via the inequality $-t(T_s - t) \leq 0$. The optimization is bilinear in two groups of decision variables V and $(\kappa, l, s_5, s_{7,i}, s_{11,i})$, and can also be solved using alternating direction method similar to Algorithm 1 in the Appendix.

After the funnel $\Omega(V,t,\gamma)$ is found, the next step is to compute a TEB $\mathcal{O}$ by solving a convex optimization:

$$\begin{aligned} \min \quad & \text{volume}(\mathcal{O}) \\ \text{s.t.} \quad & \Omega(V,t,\gamma) \subseteq \mathcal{O}, \ \forall t \in [0,T_s]. \end{aligned} \tag{2.35}$$

The set $\mathcal{O}$ is restricted to a semi-algebraic set in order to convert the set containment constraint into a SOS constraint. Depending on the parameterization of $\mathcal{O}$, different cost functions can be chosen. For example, if $\mathcal{O}$ is an ellipsoid, $\mathcal{O} = \{e \in \mathbb{R}^{n_x} : e^\top P_\mathcal{O} e \leq 1\}$, where $P_\mathcal{O} \in \mathbb{S}^{n_x}_{++}$ is a decision variable, then $-\log\det(P_\mathcal{O})$ can be used as a cost function. If $\mathcal{O}$ is a polytope, $\mathcal{O} = \{e \in \mathbb{R}^{n_x} : A_\mathcal{O} e \leq b_\mathcal{O}\}$, where $A_\mathcal{O} \in \mathbb{R}^{n_\mathcal{O} \times n_x}$ is fixed, and $b_\mathcal{O} \in \mathbb{R}^{n_\mathcal{O}}$ is a decision variable, then $\sum_{i=1}^{n_\mathcal{O}} b_{\mathcal{O},i}$ can be used as a cost function, where $b_{\mathcal{O},i}$ is the i-th element of $b_\mathcal{O}$.

Once a TEB $\mathcal{O}$ is computed from the SOS optimization (2.34)-(2.35), we can check the following safety condition, which is a generalized version of (2.14):

$$\nu(\mathcal{O}, \hat{\mathcal{X}}, \hat{\mathcal{U}}) \subseteq \mathcal{X}. \tag{2.36}$$

If (2.36) is satisfied, then the tracker state x is guaranteed to satisfy state constraints $\mathcal{X}$ and the design is considered successful. If (2.14) is *not* satisfied, we shrink the planner sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ and repeat the process.

2.5 Vehicle Obstacle Avoidance Example

We now apply the planner-tracker control scheme to a vehicle obstacle avoidance example.

For the high-fidelity tracking model, we use the dynamic bicycle model from [55]:

$$\begin{aligned}
\dot{x}_1(t) &= x_5(t)\cos(x_3(t)) - x_6(t)\sin(x_3(t)),\\
\dot{x}_2(t) &= x_5(t)\sin(x_3(t)) + x_6(t)\cos(x_3(t)),\\
\dot{x}_3(t) &= x_4(t),\\
\dot{x}_4(t) &= \frac{2}{I_z}(l_f F_{c,f}(t) - l_r F_{c,r}(t)),\\
\dot{x}_5(t) &= x_4(t)x_6(t) + u_2(t),\\
\dot{x}_6(t) &= -x_4(t)x_5(t) + \frac{2}{m}(F_{c,f}(t) + F_{c,r}(t))
\end{aligned} \tag{2.37}$$

with

$$F_{c,f} = -C_{\alpha,f}\alpha_f,\ F_{c,r} = -C_{\alpha,r}\alpha_r \tag{2.38}$$

$$\alpha_f = \frac{x_6 + l_f x_4}{x_5} - u_1,\ \alpha_r = \frac{x_6 - l_r x_4}{x_5} \tag{2.39}$$

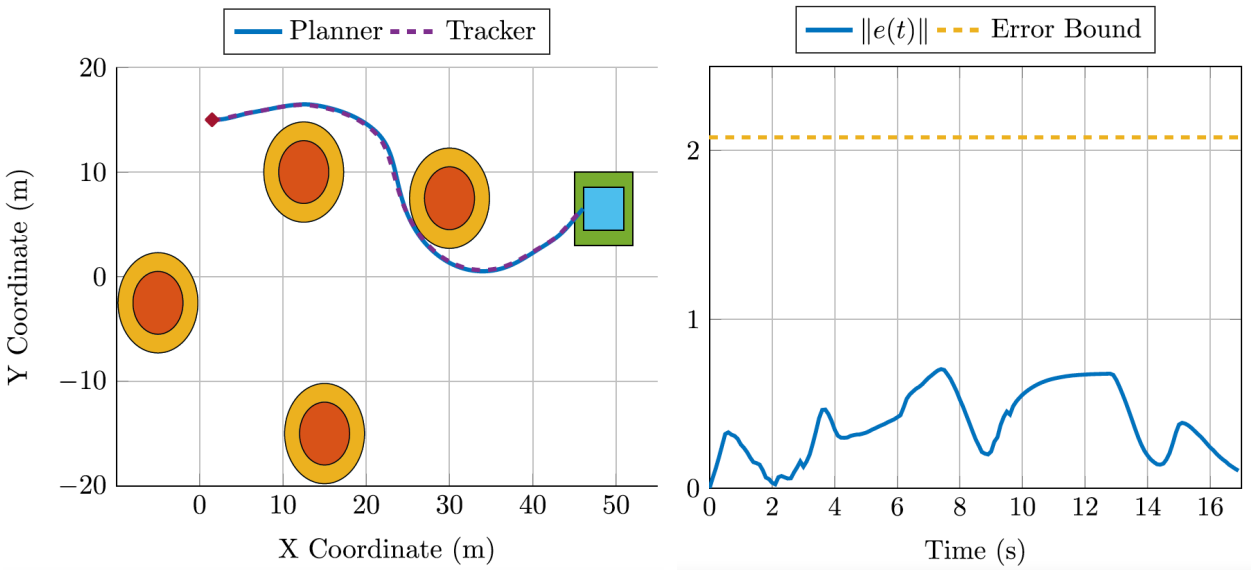
where x_1 to x_6 represent x, y positions in an inertial frame, inertial heading, yaw rate, and longitudinal and lateral speeds in the body frame. Variables u_1, u_2 represent front wheel steering angle and longitudinal acceleration, m and I_z denote the vehicle's mass and yaw inertia, and l_f and l_r represent the distance from the center of mass of the vehicle to the front and rear axles. $C_{\alpha,i}$ is the tire cornering stiffness, where $i \in \{f, r\}$. We use the parameter values $m = 1.67 \times 10^3$ kg, $I_z = 2.1 \times 10^3$ kg·m^2, $l_f = 0.99$ m, $l_r = 1.7$ m, $C_{\alpha,f} = 6.1595 \times 10^4$ N/rad, and $C_{\alpha,r} = 5.2095 \times 10^4$ N/rad.

The planning model is a Dubin's vehicle model:

$$\begin{aligned}
\dot{\hat{x}}_1(t) &= \hat{u}_2(t)\cos(\hat{x}_3(t)),\\
\dot{\hat{x}}_2(t) &= \hat{u}_2(t)\sin(\hat{x}_3(t)),\\
\dot{\hat{x}}_3(t) &= \hat{u}_1(t),
\end{aligned}$$

where $\hat{x}_1$ to $\hat{x}_3$ represent x, y positions and heading angle, and $\hat{u}_1$ and $\hat{u}_2$ represent angular velocity and velocity. If we use the map $\pi(\hat{x}) = [\hat{x}; 0_{3\times 1}]$, where $\hat{x} = [\hat{x}_1; \hat{x}_2; \hat{x}_3]$, then x_4 and x_5 will become part of the resulting error state. As a result, the magnitude of the absolute state x_4 and x_5 will be minimized in optimization (2.13), which is practically undesirable. To eliminate this issue, we use a map $\pi(\hat{x}, \hat{u}) = [\hat{x}; \hat{u}; 0]$, where $\hat{u} = [\hat{u}_1; \hat{u}_2]$, which also provides reference signals for x_4 and x_5.

The error is defined as in (2.17), with $\pi(\hat{x}, \hat{u}) = [\hat{x}; \hat{u}; 0]$ and $\phi(\hat{x}) = \text{diag}(R^{-1}(\hat{x}_3), I_4)$, where $R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}$. In this example, ϕ allows us to replace the trigonometric functions in $\hat{x}_3$ in the error dynamics by trigonometric functions in $e_3 = (x_3 - \hat{x}_3)$, which can easily be approximated by polynomials in a certain range of e_3. The sampling time used in this example is $T_s = 0.1$ s. The input and input jump spaces for the planning model are



(a) Planner and tracker system trajectories.

(b) Error trajectory.

Figure 2.3: Simulation results for the vehicle obstacle avoidance example. In Figure 2.3a we plot the trajectories of the planner and tracker systems through the environment, and in Figure 2.3b we plot $\|e(t)\|$ and its guaranteed upper bound. In Figure 2.3a, the initial position of the vehicle is marked with a red diamond, the goal set is represented with a green box, and the shrunken goal set is represented with a blue box. The four orange circles are the obstacles the vehicle must avoid. For each obstacle, the expanded unsafe region is shown in yellow.

$\hat{\mathcal{U}} = [-\pi/8, \pi/8] \times [2, 4]$, and $\Delta\hat{\mathcal{U}} = [-\pi/50, \pi/50] \times [-0.075, 0.075]$. The tracking control is unconstrained, i.e., $\mathcal{U} = \mathbb{R}^3$.

In this example, the SOS optimizations are formulated using SOSOPT [56], and solved by MOSEK. To compute the tracking controller, we parameterize the storage function V, control law κ, and multipliers s, l as degree-2 polynomials. We solve optimization (2.34) with these decision variables and then solve optimization (2.35) with the error bound $\mathcal{O}$ parameterized as a hypercube. The resulting error bound on (e_1, e_2, e_3) is $[-1.07, 1.07] \times [-1.44, 1.44] \times [-1.05, 1.05]$.

The resulting tracking controller is then tested in simulation with a corresponding planner; see Figure 2.3. The objective for the planner system is to generate a pathway through the environment that avoids all obstacles and eventually reaches a goal set, which is accom-

plished using a standard model-predictive controller (using the solver Ipopt [57]):

$$\min_{\hat{u}(\cdot)} \quad J = \ell_f(\hat{x}(t+N_p+1)) + \sum_{k=t}^{t+N_p} \ell(\hat{x}(k), \hat{u}(k)) \tag{2.40a}$$

$$\text{s.t.} \quad \hat{x}(k+1) = \hat{x}(k) + T_s \cdot \hat{f}_{\text{approx}}(\hat{x}(k), \hat{u}(k)), \tag{2.40b}$$

$$\hat{x}(k) \in \hat{\mathcal{X}}, \tag{2.40c}$$

$$\hat{u}(k) \in \hat{\mathcal{U}}, \tag{2.40d}$$

$$\hat{x}(t) = \hat{x}_0, \tag{2.40e}$$

$$\forall k = t, \ldots, t+N_p,$$

$$\hat{u}(k) - \hat{u}(k-1) \in \Delta\hat{\mathcal{U}}, \tag{2.40f}$$

$$\hat{u}(t) - \hat{u}_0 \in \Delta\hat{\mathcal{U}}, \tag{2.40g}$$

$$\forall k = t+1, \ldots, t+N_p,$$

where $\ell(\cdot,\cdot)$ in (2.40a) is the state/input cost at each time step, $\ell_f(\cdot)$ is the final state cost, (2.40b) is the polynomial approximation of the discretized Dubin's vehicle dynamics, (2.40c) and (2.40d) ensure state and input constraints are obeyed, and (2.40e) is the initial state constraint. Furthermore, $\hat{u}_0$ is the input that was applied at the previous time step, and therefore (2.40f) and (2.40g) ensure the input jump constraints are respected. The objective to reach the goal set is encoded using the functions ℓ and ℓ_f. The initial state of the vehicle is $\hat{x}_0 = [0;\ 15;\ 0]$ and the goal set is a square region centered at $(48.5, 6.5)$ with a height and width of 7m. This goal set is shrunk by the error bound to ensure that if the planner state reaches the shrunk goal set, the tracker state will reach the true goal set. There are four circular obstacles centered at $(-5, -2.5)$, $(12.5, 10)$, $(30, 7.5)$, and $(15, -15)$, each with a radius of 3m. Since the maximum position tracking error the vehicle will experience is $\sqrt{1.07^2 + 1.44^2} = 1.79$m, for each obstacle, we constrain the vehicle to avoid a circular region centered at the obstacle coordinates with an expanded radius of 4.79m. This ensures the vehicle will not collide with any of the obstacles. Indeed, in simulation the vehicle successfully navigates past each obstacle and eventually reaches the goal set, as shown in Figure 2.3.

2.6 Conclusion

In this chapter, we address robust trajectory planning and control design for nonlinear systems. A hierarchical trajectory planning and control framework is proposed, where a low-fidelity model is used to plan trajectories satisfying planning constraints, and a high-fidelity model is used for synthesizing tracking controllers guaranteeing the boundedness of the error state between the low- and high-fidelity models. We consider error states that are functions of both planner states and inputs, which offers more freedom in the choice of the low-fidelity model. SOS optimizations are formulated for computing the tracking controllers and their

associated tracking error bound simultaneously. Finally, we demonstrate the planner-tracker control scheme on a vehicle obstacle avoidance example.

When implementing the planner-tracker framework in real-time, there are still challenges for providing a full guarantee of safety. Two sources of error in the planner dynamics are present in the example above: (1) the discretization error from the forward Euler discretization, and (2) the polynomial approximation error from the trigonometric terms. If a bound on these errors were known, it would be possible to incorporate them into the design process, ensuring instead that the planner constraints, when augmented with the discretization error, polynomial approximation error, *and* the tracking error still satisfy the tracker constraints. We do not perform such an analysis in this chapter. One could also avoid discretization error by using a MPC solver that is designed to perform numerical integration for *continuous-time* dynamical systems, such as in the software package `acados` [58].

In this chapter, we also do not address the question of MPC terminal sets and costs for stability and persistent feasibility guarantees for the MPC problem. These sets/costs can be computed in simple cases but may increase the computational burden, both offline and online. Real-time reliability of solvers for MPC, especially for nonlinear models, should also be considered in practical applications. Finally, defining the error variable can require clever selection of the function ϕ to make terms in the dynamics cancel, which isn't always intuitive.

Appendix

The algorithm to solve the bilinear optimization (2.13) is summarized below, the (κ, γ)-step of which treats γ as a decision variable. By minimizing γ, the volume of $\Omega(V^{j-1}, \gamma)$ can be shrunk. In the V-step, (2.41) enforces $\Omega(V^j, \gamma^j) \subseteq \Omega(V^{j-1}, \gamma^j)$.

The input to Algorithm 1 is a feasible initial guess V^0. One candidate might be a quadratic Lyapunov function $\bar{V}$ obtained by solving Lyapunov equations using the linearized error dynamics with LQR controllers. However, $\bar{V}$ might be too coarse to be feasible for the constraints (2.13). Here, we introduced a slack variable $\lambda > 0$ to the constraint (2.13c) to relax the constraint, and quantify how far $\bar{V}$ is away from a feasible candidate:

$$\begin{aligned} -\frac{\partial V}{\partial e} \cdot (f_e + g_e \cdot \kappa) + \lambda - \epsilon e^\top e + l \cdot (V - \gamma) + s_1 \cdot p_{\hat{x}} \\ + s_2 \cdot p_{\hat{u}} + s_3 \cdot p_w \in \Sigma[(e, \hat{x}, \hat{u}, w)]. \end{aligned} \tag{2.42}$$

By iteratively searching over two bilinear groups of decision variables, we minimize λ until $\lambda \leq 0$. Based on this idea, an algorithm to compute V^0 from $\bar{V}$ is proposed as Algorithm 2.

Algorithm 1 Alternating direction method

Require: function V^0 such that constraints (2.13) are feasible by proper choice of s, l, κ, γ.
Ensure: κ, γ, V.

1: **for** $j = 1 : N_{\text{iter}}$ **do**
2: **$(\boldsymbol{\kappa}, \boldsymbol{\gamma})$-step**: decision variables (s, l, κ, γ).
Minimize γ subject to (2.13) using $V = V^{j-1}$.
This yields $(l^j, s^j_{4,i}, s^j_{7,i}, \kappa^j)$ and the cost γ^j.
3: **$\boldsymbol{V}$-step**: decision variables $(s_{1\to 3}, s_{5\to 6,i}$, $s_{8\to 9,i}, V)$; Maximize the feasibility subject to (2.13) as well as $s_{10} - \epsilon \in \Sigma[e]$, and

$$-s_{10} \cdot (V^{j-1} - \gamma^j) + (V - \gamma^j) \in \Sigma[e], \tag{2.41}$$

using $(\gamma = \gamma^j, s_{4,i} = s^j_{4,i}, s_{7,i} = s^j_{7,i}, \kappa = \kappa^j$, $l = l^j)$. This yields V^j.
4: **end for**

Algorithm 2 Computation of V^0

Require: function $\bar{V}$, and $\bar{\gamma} > 0$.
Ensure: V^0.

1: $V^{\text{pre}} \leftarrow \bar{V}$
2: **while** $\lambda > 0$ **do**
3: **$\boldsymbol{\kappa}$-step**: decision variables (s, l, κ).
Minimize λ subject to (2.13a–2.13b, 2.42, 2.13d–2.13e), using $V = V^{\text{pre}}$, $\gamma = \bar{\gamma}$.
$(l^{\text{pre}}, s^{\text{pre}}_{4,i}, s^{\text{pre}}_{7,i}, \kappa^{\text{pre}}) \leftarrow (l, s_{4,i}, s_{7,i}, \kappa)$
4: **$\boldsymbol{V}$-step**: decision variables $(s_{1\to 3}, s_{5\to 6,i}$, $s_{8\to 9,i}, V)$; Minimize λ subject to (2.13a–2.13b, 2.42, 2.13d–2.13e) using $(\gamma = \bar{\gamma}, s_{4,i} = s^{\text{pre}}_{4,i}, s_{7,i} = s^{\text{pre}}_{7,i}, \kappa = \kappa^{\text{pre}}$, $l = l^{\text{pre}})$.
$V^{\text{pre}} \leftarrow V$
5: **end while**
6: $V^0 \leftarrow V^{\text{pre}}$

Chapter 3

Extension to Unmodeled Input Dynamics

3.1 Abstract

The planner-tracker framework is a hierarchical control scheme wherein a motion plan is generated using a simplified model, and a tracking controller synthesized offline keeps the error between the true and simplified model trajectories small. This chapter extends the planner-tracker framework to accommodate unmodeled input dynamics, described by integral quadratic constraints (IQCs). A sum-of-squares (SOS) program is formulated to search for the tracking controller and error bound, and the method is demonstrated on a vehicle obstacle avoidance example with input delay.

3.2 Introduction

Complex systems, such as autonomous vehicles [1] and missile guidance systems [2], use hierarchical control schemes where each control layer uses a different system model. This approach can enhance computational efficiency, as the higher-level control layer can use a simpler model to reduce computation times and to enable control strategies that may not be possible with the more complex model. Faster higher-level control can also allow the system to respond to changing environments in real-time.

Such hierarchical schemes, however, may be unsafe if the error between the models in different layers is not accounted for. A high-level motion plan generated using a simplified model may avoid obstacles that the lower-level controller is unable to. Thus, for safety, each control layer must accommodate the error arising from different models.

In the planner-tracker framework [3–10], a lower-fidelity "planning" model is employed for online planning and a "tracking" controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set.

System safety is then guaranteed if the planner constraints, when augmented by the tracking error bound, lie within the safety constraints.

Tube Model Predictive Control (MPC) is another approach for handling the error between an uncertain model and a nominal model used for planning [22–30]. Tube MPC is a robust control strategy where the error between the true model and a nominal model, free of disturbances, is bounded within a tube, and MPC is performed on the nominal model with state constraints shrunk by the size of the tube. Typically, the true and nominal models differ by only an additive disturbance in the true model. However, this may not capture the total uncertainty in the true model. Besides disturbances, another source of uncertainty is unmodeled dynamics.

The more sources of uncertainty we can accommodate in the higher-fidelity model, the more the models will reflect reality. In this chapter, we extend the planner-tracker framework to handle unmodeled dynamics at the input of the tracker model. We characterize the uncertainty using an Integral Quadratic Constraint (IQC), which is an essential tool for ensuring robustness to unmodeled dynamics [59, 60]. In particular, we use α-IQCs, which contain an exponential weighting factor compared to standard IQCs [31, 32, 61].

These α-IQCs can be used to describe many uncertainties including unknown delays or unmodeled actuator dynamics. The Smith predictor [62] is a common compensator for systems with large delays, but the approach presented here provides safety guarantees for an *unknown* delay in a range.

The most closely related works are [31,32], which also use α-IQCs to incorporate unmodeled dynamics in shrinking tube MPC. However, attention is restricted to linear systems, whereas this chapter is applicable to nonlinear systems.

In Section 3.3, we derive the error dynamics between the planner and the tracker model with unmodeled input dynamics. In Section 3.4, we describe conditions that an error bound set must satisfy, and in Section 3.5 we convert these conditions into an SOS program. We solve this SOS program for a vehicle obstacle avoidance example in Section 3.6.

Notation: For $\xi \in \mathbb{R}^n$, $\mathbb{R}[\xi]$ represents the set of polynomials in ξ with real coefficients, and $\mathbb{R}^m[\xi]$ and $\mathbb{R}^{m\times p}[\xi]$ denote all vector and matrix valued polynomial functions. The subset $\Sigma[\xi] := \{p = p_1^2 + p_2^2 + ... + p_M^2 : p_1, ..., p_M \in \mathbb{R}[\xi]\}$ of $\mathbb{R}[\xi]$ is the set of sum-of-squares polynomials in ξ. We say a matrix-valued polynomial $P \in \mathbb{R}^{m\times m}[\xi]$ is in $\Sigma_m[\xi]$ if $y^\top P y \in \Sigma[\xi, y]$, which implies P is positive semidefinite for all ξ. For a function $V : \mathbb{R}^n \to \mathbb{R}$ and a scalar $\gamma \in \mathbb{R}$, we denote the γ-sublevel set of V as $\Omega(V, \gamma) := \{x \in \mathbb{R}^n : V(x) \leq \gamma\}$.

3.3 Problem Formulation

Planner-Tracker Framework

We now describe the planner-tracker framework, extended to include unmodeled input dynamics (Figure 3.1). The low-fidelity model, called the *planning* model P, has the form:

$$\dot{\hat{x}} = \hat{f}(\hat{x}, \hat{u}), \tag{3.1}$$

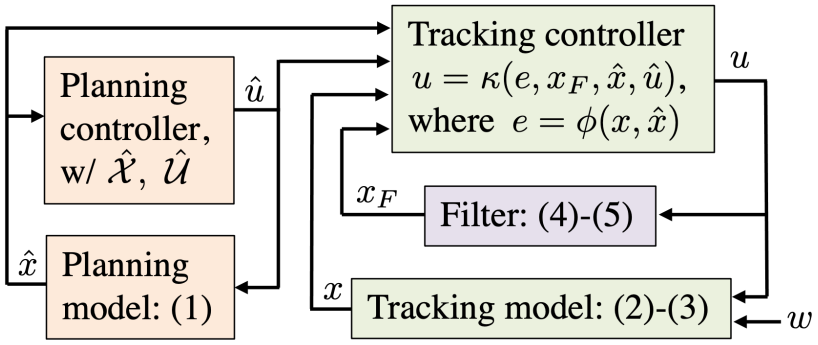


Figure 3.1: Planner-tracker scheme. The planning controller uses the state $\hat{x}$ and constraints $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ to generate a reference input $\hat{u}$. Using a filter state x_F as an additional input, the tracking controller converts this into a control u which is guaranteed to keep the tracker state x within state constraints $\mathcal{X}$. This is accomplished by keeping the tracking error e within a set $\mathcal{O}$ whose volume is minimized.

where $\hat{x} \in \hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$ and $\hat{u} \in \hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$ are the state and control input, respectively, of the planner system.

A higher-fidelity model, referred to as the *tracking* model, is used to track the trajectory generated for the planning model above. The tracking model is of the form:

$$\dot{x} = f(x, w) + g(x, w)(u + l), \tag{3.2}$$
$$l = \Delta(u), \tag{3.3}$$

where $x \in \mathbb{R}^{n_x}$ is the tracker state, $w \in \mathcal{W} \subseteq \mathbb{R}^{n_w}$ is an external disturbance, and $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the control input which will depend on $\hat{x}$ and $\hat{u}$ with the goal of keeping the error between the planner and tracker systems small.

An uncertain block Δ at the tracker input with output $l \in \mathbb{R}^{n_l}$ is added to the tracker input in (3.2). This block encompasses unmodeled dynamics, e.g., delays, where the resulting signal l is a function of u and/or x. This is distinct from the exogeneous disturbance w which does not depend on x or u and is restricted, a priori, to $\mathcal{W}$. The presence of Δ and l is what distinguishes this chapter from previous works in the planner-tracker framework literature.

Integral Quadratic Constraint

We assume that Δ satisfies an input-output relationship called an α Integral Quadratic Constraint (α-IQC) for some constant scalar $\alpha \in \mathbb{R}$. To define this relationship, the input u is passed through a linear time-invariant filter F with state $x_F \in \mathbb{R}^{n_F}$ and output $z \in \mathbb{R}^{n_z}$:

$$\dot{x}_F = A_F x_F + B_F u, \tag{3.4}$$
$$z = C_F x_F + D_F u. \tag{3.5}$$

Then Δ is said to satisfy the α-IQC defined by filter F if

$$\int_0^T e^{\alpha t}(z(t)^\top z(t) - l(t)^\top l(t))dt \geq 0 \tag{3.6}$$

for all $T \geq 0$ and for $x_F(0) = 0_{n_F}$. We write this compactly as $\Delta \in \text{IQC}(F, \alpha)$. The α-IQC can be used to describe uncertainties such as unknown delays or unmodeled actuators. It allows us to augment the system with F and analyze the augmented system with Δ removed, treating l as a free input.

There is a more general IQC framework where the filter has both u and l as inputs. In this chapter, we restrict the filter input to be the tracker input u so that there is no uncertainty affecting the filter state x_F. This allows us to compute x_F and use it as an input to the tracking controller.

Error System

We define a tracking error $e \in \mathbb{R}^{n_e}$ between the states of the planner and tracker models:

$$e = \phi(x, \hat{x}). \tag{3.7}$$

We assume ϕ is invertible in x for all $\hat{x} \in \hat{\mathcal{X}}$, i.e., it admits an inverse, ψ, such that $e = \phi(x, \hat{x}) \Rightarrow x = \psi(e, \hat{x})$. We further assume the error dynamics can be written as

$$\dot{e} = f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)(u + l), \tag{3.8}$$
$$l = \Delta(u), \tag{3.9}$$

for some functions $f_e(e, \hat{x}, \hat{u}, w)$ and $g_e(e, \hat{x}, \hat{u}, w)$. Both of these requirements hold for the simple definition $\phi(x, \hat{x}) = x - \hat{x}$. They also hold for the example in Section 3.6, where $\phi(x, \hat{x}) = r(\hat{x})(x - \hat{x})$, and $r(\hat{x}) \in \mathbb{R}^{n_x \times n_x}$ is invertible for all $\hat{x} \in \hat{\mathcal{X}}$. We denote this error system as E in Figure 3.2.

Objective

We wish to derive a tracking controller $u = \kappa(e, x_F, \hat{x}, \hat{u})$ and an associated error bound $\mathcal{O} \subseteq \mathbb{R}^{n_e}$, as small as possible, for the following closed loop error dynamics:

$$\dot{e} = f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)(\kappa(e, x_F, \hat{x}, \hat{u}) + l), \tag{3.10}$$
$$l = \Delta(\kappa(e, x_F, \hat{x}, \hat{u})). \tag{3.11}$$

$\mathcal{O}$ is used for shrinking the planner state constraint $\hat{\mathcal{X}}$ such that, when $e \in \mathcal{O}$, $\hat{x} \in \hat{\mathcal{X}}$ guarantees the tracker constraint $x \in \mathcal{X}$. Minimizing the volume of $\mathcal{O}$ ensures the constraint $\hat{\mathcal{X}}$ is minimally restrictive.

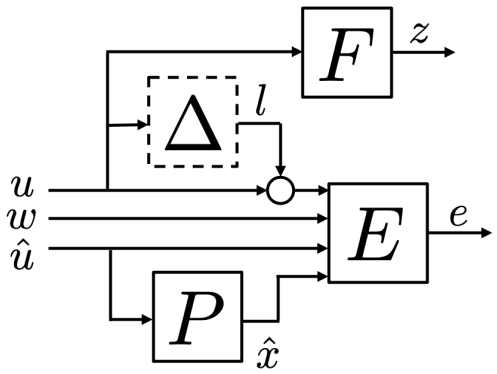


Figure 3.2: Block diagram for the error system E. By analyzing the system augmented with the filter F, we can ignore Δ and treat l as a free input, where z and l satisfy the α-IQC (3.6).

3.4 Error Bound

Finding the error bound $\mathcal{O}$ involves two conditions. First, in Section 3.4, we present a condition on the control law κ and an associated storage function V such that the γ-sublevel set of V for some scalar γ, denoted $\Omega(V,\gamma)$, is positively invariant under the closed loop error dynamics (3.10)-(3.11) for all $\Delta \in \text{IQC}(F,\alpha)$, $\hat{x} \in \hat{\mathcal{X}}$, $\hat{u} \in \hat{\mathcal{U}}$, and $w \in \mathcal{W}$. That is,

$$V(e(0),x_F(0)) \leq \gamma \;\Rightarrow V(e(t),x_F(t)) \leq \gamma \;\forall t \geq 0. \tag{3.12}$$

Next, in Section 3.4, we project $\Omega(V,\gamma)$ into the space of error variables. $\mathcal{O}$ is a bound on this projection, obtained using a "shape function" with an adjustable parameter that we use to minimize the volume. We accommodate an initial condition constraint and an input constraint in Section 3.4.

Invariance Condition

The following condition ensures that the γ-sublevel set of a storage function V is a positively invariant set for the closed loop error dynamics (3.10)-(3.11) with tracking control law κ. It is motivated by a related result on robust control barrier functions [61, Lemma 1].

Theorem 6. *Given the error dynamics (3.8)-(3.9), filter dynamics (3.4)-(3.5), and IQC $\Delta \in$ IQC(F,α), if there exist $\mathcal{C}^1$ functions $V : \mathbb{R}^{n_e} \times \mathbb{R}^{n_F} \to \mathbb{R}$ and $\kappa : \mathbb{R}^{n_e} \times \mathbb{R}^{n_F} \times \mathbb{R}^{\hat{n}} \times \mathbb{R}^{\hat{m}} \to \mathbb{R}$, and a scalar $\gamma > 0$ such that*

$$\begin{aligned} &\nabla_e V(e,x_F)^\top (f_e(e,\hat{x},\hat{u},w) \\ &\qquad + g_e(e,\hat{x},\hat{u},w)(\kappa(e,x_F,\hat{x},\hat{u})+l)) \\ &+ \nabla_{x_F} V(e,x_F)^\top (A_F x_F + B_F \cdot \kappa(e,x_F,\hat{x},\hat{u})) \\ &+ (C_F x_F + D_F \kappa(e,x_F,\hat{x},\hat{u}))^\top (C_F x_F + D_F \kappa(e,x_F,\hat{x},\hat{u})) \\ &\qquad - l^\top l < -\alpha(V(e,x_F) - \gamma) \end{aligned} \tag{3.13}$$

$$\forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}, w \in \mathcal{W}, l \in \mathbb{R}^{n_l}, (e, x_F) \text{ s.t. } V(e, x_F) \leq \gamma,$$

then $\Omega(V, \gamma)$ is a positively invariant set for the closed loop error dynamics with controller κ.

Proof. The dynamics of the augmented closed-loop system with state $\xi := [e; x_F]$, inputs $(l, \hat{x}, \hat{u}, w)$, and output z are:

$$\begin{aligned}\dot{\xi} &= \begin{bmatrix} f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)(\kappa(e, x_F, \hat{x}, \hat{u}) + l) \\ A_F x_F + B_F \cdot \kappa(e, x_F, \hat{x}, \hat{u}) \end{bmatrix} \\ &=: F(\xi, l, \hat{x}, \hat{u}, w), \qquad (3.14) \\ z &= C_F x_F + D_F \kappa(e, x_F, \hat{x}, \hat{u}) =: H(\xi, \hat{x}, \hat{u}). \qquad (3.15)\end{aligned}$$

Note that condition (3.13) is equivalent to

$$\nabla V(\xi)^\top F + H^\top H - l^\top l < -\alpha(V(\xi) - \gamma). \qquad (3.16)$$

For simplicity of notation, define $\bar{F}(t) := F(\xi(t), l(t), \hat{x}(t), \hat{u}(t), w(t))$, and similarly for $\bar{H}$. Define $\bar{V}(t) := V(\xi(t)) - \gamma$ so that $\bar{V}(t) = 0$ when $V(\xi(t)) = \gamma$.

Now the theorem is proved by contradiction. Assume there exist a time $t_3 > 0$ and signals $\xi(\cdot)$, $\hat{x}(\cdot)$, $\hat{u}(\cdot)$, $l(\cdot)$, and $w(\cdot)$ such that $\bar{V}(0) \leq 0$ but $\bar{V}(t_3) > 0$. Then, by continuity of $\bar{V}$, there exists $t_1 \in [0, t_3)$ such that $\bar{V}(t_1) = 0$, $\bar{V}(t) \leq 0$ for $t \leq t_1$, and $\bar{V}(t) > 0$ for $t \in (t_1, t_1 + \epsilon_1]$ for some $\epsilon_1 > 0$. Then the inequality (3.13) holds for $t \leq t_1$. Furthermore, since (3.13) is a *strict* inequality, by continuity of $\bar{V}$ there exists $\epsilon_2 > 0$ such that for $t \in [0, t_1 + \epsilon_2]$, the relaxed, non-strict version of (3.13) holds:

$$\nabla V(\xi(t))^\top \bar{F}(t) + \bar{H}(t)^\top \bar{H}(t) - l(t)^\top l(t) \leq -\alpha \bar{V}(t). \qquad (3.17)$$

Defining $t_2 = t_1 + \min\{\epsilon_1, \epsilon_2\}$, we see that (3.17) holds on $[0, t_2]$ and $\bar{V}(t_2) > 0$. Next, note that

$$\begin{aligned}\frac{d}{dt}\{e^{\alpha t}\bar{V}(t)\} &= e^{\alpha t}\{\alpha \bar{V}(t) + \nabla V(\xi(t))^\top \bar{F}(t)\} \\ &\leq -e^{\alpha t}(\bar{H}(t)^\top \bar{H}(t) - l(t)^\top l(t)). \qquad (3.18)\end{aligned}$$

Integrating (3.18) from $t = 0$ to $t = t_2$, we have

$$\begin{aligned}&e^{\alpha t_2}\bar{V}(t_2) - \bar{V}(0) \qquad (3.19) \\ &\leq -\int_0^{t_2} e^{\alpha t}(\bar{H}(t)^\top \bar{H}(t) - l(t)^\top l(t))dt \leq 0,\end{aligned}$$

where the last inequality used (3.6). Rearranging, we have

$$\bar{V}(t_2) \leq e^{-\alpha t_2}\bar{V}(0) \leq 0, \qquad (3.20)$$

which is a contradiction. □ □

Projection Condition

If we find V and κ satisfying (3.13), then $\Omega(V,\gamma) \subseteq \mathbb{R}^{n_e} \times \mathbb{R}^{n_F}$ is a positively invariant set. Next we obtain an error bound $\mathcal{O} \subseteq \mathbb{R}^{n_e}$ by bounding the projection of $\Omega(V,\gamma)$ from the (e, x_F)-space into the e-space $\mathbb{R}^{n_e}$. $\mathcal{O}$ will be used to shrink planning state constraints $\hat{\mathcal{X}}$ such that $\hat{x} \in \hat{\mathcal{X}}$ implies $x \in \mathcal{X}$. Then safety of the tracker system is guaranteed as long as the planner constraints are satisfied. We would like to obtain an error bound $\mathcal{O}$ that is as small as possible so that the constraints on the planner are minimally restrictive.

We introduce a "shape function" $P : \mathbb{R}^{n_e} \to \mathbb{R}$, whose sublevel sets are regular objects like balls or hyper-rectangles that can be conveniently used to shrink state constraints. P need not depend on all error variables (e.g., if the only constraint in $\mathcal{X}$ is for obstacle avoidance, P may just depend on the position errors). Then we enforce the constraint

$$\text{proj}_{\mathbb{R}^{n_e}}(\Omega(V,\gamma)) \subseteq \Omega(P,c) =: \mathcal{O}, \tag{3.21}$$

i.e., the projection onto $\mathbb{R}^{n_e}$ of the γ-sublevel set of V is contained within the c-sublevel set of P. By minimizing c, we can shrink the error bound $\mathcal{O}$ as much as possible.

Additional Conditions

There are two additional conditions that we may want the storage function V and the tracking controller κ to satisfy. First, the initial error $e(0)$ may be known to lie in a set $\mathcal{E}_0 \subseteq \mathbb{R}^{n_e}$. To ensure this set is included in the invariant set, we can enforce the constraint

$$\mathcal{E}_0 \times \{0_{n_F}\} \subseteq \Omega(V,\gamma), \tag{3.22}$$

where we used the fact that the filter initial condition is $x_F(0) = 0_{n_F}$. Secondly, we can enforce input constraints $\mathcal{U}$ with the following constraint:

$$\begin{aligned} &\kappa(e, x_F, \hat{x}, \hat{u}) \in \mathcal{U} \;\; \forall \hat{x} \in \hat{\mathcal{X}},\; \hat{u} \in \hat{\mathcal{U}}, \\ &\forall (e, x_F) \in \mathbb{R}^{n_e} \times \mathbb{R}^{n_F} \text{ s.t. } V(e, x_F) \leq \gamma. \end{aligned} \tag{3.23}$$

3.5 SOS Optimization

We now formulate an SOS optimization that searches for a tracking controller κ and a storage function V satisfying (3.13) and (3.21)-(3.23) while minimizing the volume of the error bound $\mathcal{O}$.

Finding generic functions V and κ that satisfy (3.13) is a difficult problem. Below we show how SOS programming can be used to search for these functions by restricting to polynomial candidates $V \in \mathbb{R}[e, x_F]$ and $\kappa \in \mathbb{R}^{n_u}[e, x_F, \hat{x}, \hat{u}]$. Besides this restriction, we make the following assumption:

Assumption 2. *The mappings $f_e \in \mathbb{R}^{n_x}[(e, \hat{x}, \hat{u}, w)]$ and $g_e \in \mathbb{R}^{n_x \times n_u}[(e, \hat{x}, w)]$ in error dynamics (3.8) are polynomials. Sets $\mathcal{E}_0$, $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ are semi-algebraic sets, i.e., there exists $p_0 \in \mathbb{R}[e]$ such that $\mathcal{E}_0 = \{e \in \mathbb{R}^{n_x} \ : \ p_0(e) \leq 0\}$; with similar definitions for $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ with polynomials $p_{\hat{x}} \in \mathbb{R}[\hat{x}]$, $p_{\hat{u}} \in \mathbb{R}[\hat{u}]$, and $p_w \in \mathbb{R}[w]$. The control constraint set $\mathcal{U}$ is a hypercube $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : \underline{u} \leq u \leq \overline{u}\}$, where $\underline{u}, \overline{u} \in \mathbb{R}^{n_u}$.*

The invariance condition (3.13) involves the unbounded variable l and contains a term that is quadratic in the decision variable κ. The following lemma gives a sufficient condition for (3.13) that is suitable for use in an SOS program.

Lemma 1. *A sufficient condition for (3.13) is*

$$-\mathcal{Q}(V, \kappa, \gamma, s_V, s_X, s_U, s_W) \in \Sigma_{2n_u+1}[e, x_F, \hat{x}, \hat{u}, w], \tag{3.24}$$

$$\text{where } \mathcal{Q} = \begin{bmatrix} \mathcal{Q}_{11} & \nabla_e V^\top g & (C_F x_F + D_F \kappa)^\top \\ g^\top \nabla_e V & -4I & 0 \\ C_F x_F + D_F \kappa & 0 & -I \end{bmatrix}, \tag{3.25}$$

$$\begin{aligned} \text{and } \mathcal{Q}_{11}(V, \kappa, \gamma, s_V, s_X, s_U, s_W) &= \nabla_e V^\top (f_e + g_e \cdot \kappa) \\ &+ \nabla_{x_F} V^\top (A_F x_F + B_F \kappa) + (\alpha - s_V) \cdot (V - \gamma) \\ &- s_X \cdot p_X - s_U \cdot p_U - s_W \cdot p_W + \epsilon. \end{aligned} \tag{3.26}$$

for some $\epsilon > 0$.

Proof. We manipulate (3.13), reproduced below, into an equivalent condition that has no quadratic terms in the decision variables:

$$\begin{aligned} &\nabla_e V^\top (f_e + g_e \cdot (k + l)) + \nabla_{x_F} V^\top (A_F x_F + B_F \kappa) \\ &\quad + (C_F x_F + D_F \kappa)^\top (C_F x_F + D_F \kappa) - l^\top l < -\alpha(V - \gamma) \\ &\forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}, w \in \mathcal{W}, l \in \mathbb{R}^{n_l}, (e, x_F) \text{ s.t. } V(e, x_F) \leq \gamma. \end{aligned} \tag{3.27}$$

Maximizing over the unconstrained variable l gives a worst-case value of $l^* = \frac{1}{2} g_e^\top \nabla_e V$. Plugging this in yields

$$\begin{aligned} &\nabla_e V^\top (f_e + g_e \cdot \kappa) + \nabla_{x_F} V^\top (A_F x_F + B_F \kappa) \\ &\qquad + (C_F x_F + D_F \kappa)^\top (C_F x_F + D_F \kappa) \\ &\qquad + \frac{1}{4} \nabla_e V^\top g_e g_e^\top \nabla_e V < -\alpha(V - \gamma) \\ &\forall \hat{x} \in \hat{\mathcal{X}}, \ \hat{u} \in \hat{\mathcal{U}}, \ w \in \mathcal{W}, \ (e, x_F) \text{ s.t. } V(e, x_F) \leq \gamma. \end{aligned} \tag{3.28}$$

While (3.28) only needs to hold for certain values of $(e, \hat{x}, \hat{u}, w)$, the S-procedure [52] is a method that gives a sufficient condition that holds for all $(e, \hat{x}, \hat{u}, w)$, where constraints such as $\hat{x} \in \hat{\mathcal{X}}$ are encoded via nonnegative multipliers. We now use the S-procedure [52] to

ensure that (3.28) holds whenever $V \leq \gamma$, $\hat{x} \in \hat{\mathcal{X}}$, $\hat{u} \in \hat{\mathcal{U}}$, and $w \in \mathcal{W}$, and add some $\epsilon > 0$ to the left hand side so the inequality is non-strict:

$$\begin{aligned} \nabla_e V^\top (f_e + g_e \cdot \kappa) + \nabla_{x_F} V^\top (A_F x_F + B_F \kappa) \\ + (C_F x_F + D_F \kappa)^\top (C_F x_F + D_F \kappa) \\ + \frac{1}{4} \nabla_e V^\top g_e g_e^\top \nabla_e V + (\alpha - s_V)(V - \gamma) \\ - s_X \cdot p_X - s_U \cdot p_U - s_W \cdot p_W + \epsilon \leq 0, \end{aligned} \tag{3.29}$$

where s_V, s_X, s_U, and s_W are nonnegative multipliers. Note that (3.29) has quadratic terms in $\nabla_e V$ and κ which are both decision variables, so (3.29) is not convex in either variable. Thus, start by applying Schur complements to expand the term $\frac{1}{4} \nabla_e V^\top g_e g_e^\top \nabla_e V$ that is quadratic in $\nabla_e V$. Then (3.29) is equivalent to

$$\begin{bmatrix} \mathcal{Q}_{11} + (C_F x_F + D_F \kappa)^\top (C_F x_F + D_F \kappa) & \nabla_e V^\top g \\ g^\top \nabla_e V & -4I \end{bmatrix} \leq 0, \tag{3.30}$$

where $\mathcal{Q}_{11}$ is defined in (3.26). Applying Schur complements a second time to expand the term $(C_F x_F + D_F \kappa)^\top (C_F x_F + D_F \kappa)$ that is quadratic in κ, (3.30) is equivalent to

$$\mathcal{Q}(V, \kappa, \gamma, s_V, s_X, s_U, s_W) \leq 0, \tag{3.31}$$

with $\mathcal{Q}$ as in (3.25)-(3.26). Finally, (3.31) can be relaxed as the SOS condition (3.24). □

Using Lemma 1 and applying the generalized S-procedure [52] to the set containment constraints (3.21)-(3.23), we obtain the following SOS optimization problem for finding V, κ, and $\mathcal{O}$:

$$\min_{\substack{V,\kappa,\gamma,s_0,s_F \\ s_V^i,s_X^i,s_U^i,s_W \\ \delta>0,c^{-1}>0,\gamma>0}} \quad \beta\delta - c^{-1} \tag{3.32a}$$

$$\text{s.t. } s_W, s_0 \in \Sigma[e, x_F, \hat{x}, \hat{u}] \tag{3.32b}$$

$$s_V^i, s_X^i, s_U^i \in \Sigma[e, x_F, \hat{x}, \hat{u}],\ i \in \{-n_u, ..., n_u\} \tag{3.32c}$$

$$\delta(Z^\top Z)I - \mathcal{Q}(V, \kappa, \gamma, s_V^0, s_X^0, s_U^0, s_W) \in \Sigma_{2n_u+1}[e, x_F, \hat{x}, \hat{u}] \tag{3.32d}$$

$$(V - \gamma) - (c^{-1}P - 1) \in \Sigma[e, x_F] \tag{3.32e}$$

$$s_0 \cdot p_0 + s_F \cdot x_F - (V - \gamma) \in \Sigma[e, x_F] \tag{3.32f}$$

$$\overline{u}_i - \kappa_i + s_V^i \cdot (V - \gamma) + s_X^i \cdot p_{\hat{x}} + s_U^i \cdot p_{\hat{u}} \in \Sigma[(e, \hat{x}, \hat{u})],\ i \in \{1, ..., n_u\} \tag{3.32g}$$

$$\kappa_i - \underline{u}_i + s_V^{-i} \cdot (V - \gamma) + s_X^{-i} \cdot p_{\hat{x}} + s_U^{-i} \cdot p_{\hat{u}} \in \Sigma[(e, \hat{x}, \hat{u})],\ i \in \{1, ..., n_u\} \tag{3.32h}$$

The variables in (3.32b) and (3.32c) are SOS multipliers. Condition (3.32d) is the result of applying the S-procedure to (3.13) (justified in Lemma 1), with an added slack variable δ multiplying $(Z^\top Z)I$, where Z is the vector of monomials in $\mathcal{Q}$. Condition (3.32e) is the result of applying the S-procedure to (3.21). Condition (3.32f) is the result of applying the S-procedure to (3.22), where the multiplier s_F need not be SOS. Conditions (3.32g) and (3.32h) are the result of applying the S-procedure to (3.23) at each vertex of $\mathcal{U}$. The cost function (3.32a) is a weighted difference of the slack variable δ from (3.32d) and the parameter c^{-1} from (3.32e) with a weight β that the user can tune. This cost helps to achieve the joint goals of having the invariance condition (3.13) hold up to numerical tolerances, and making the error bound $\mathcal{O}$ as small as possible by maximizing c^{-1} (i.e., minimizing c). We found that in practice, formulating the problem in terms of c^{-1} rather than c led to a smaller error bound.

Even after removing quadratic terms in $\nabla_e V$ and κ in Lemma 1, the SOS program (3.32) is nonconvex because it has terms that are bilinear in the decision variables, but it can be solved by alternately solving convex subproblems with the decision variables $(V, \gamma, \delta, c, s_0, s_F, s_X^i, s_U^i, s_W)$ and $(\kappa, \delta, c, s_0, s_F, s_V^i, s_X^i, s_U^i, s_W)$.

3.6 Numerical Example

Setup

We demonstrate the method on an obstacle avoidance example, where the planner uses a Dubin's vehicle model:

$$\dot{\hat{x}} = \begin{bmatrix} \hat{u}_2 \cos(\hat{x}_3) \\ \hat{u}_2 \sin(\hat{x}_3) \\ \hat{u}_1 \end{bmatrix}. \tag{3.33}$$

The states are positions $(\hat{x}_1, \hat{x}_2)$ and heading angle $\hat{x}_3$, and the inputs are angular rate $\hat{u}_1$ and longitudinal velocity $\hat{u}_2$.

In the tracker model, the input u_1 is delayed by τ seconds, denoted $D_\tau(u_1)$. Using an uncertain block Δ, the tracker model can be put into the form of (3.2)-(3.3):

$$\dot{x} = \begin{bmatrix} u_2 \cos(x_3) \\ u_2 \sin(x_3) \\ D_\tau(u_1) \end{bmatrix} = \begin{bmatrix} u_2 \cos(x_3) \\ u_2 \sin(x_3) \\ u_1 + l \end{bmatrix}, \quad l = \Delta(u_1). \tag{3.34}$$

Thus we have $\Delta(u) = D_\tau(u_1) - u_1$, so Δ is the *deviation* due to the delay. The value of the delay is unknown but is known to lie in the interval $\tau \in [0, \tau_{\max}]$. The α-IQC associated with Δ is computed as in Section V of [61], using the MATLAB function `fitmagfrd` to obtain a filter F whose Bode plot upper bounds the Bode plot of all possible values of Δ.

For simplicity, there is no exogenous disturbance w in this example. We define the error as

$$e = R(\hat{x}_3)^\top (x - \hat{x}), \text{ where } R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.35}$$

Rotating the error into the frame of the planner model gives:

$$\dot{e} = \begin{bmatrix} \hat{u}_1 e_2 + u_2 \cos(e_3) - \hat{u}_2 \\ -\hat{u}_1 e_1 + u_2 \sin(e_3) \\ u_1 + l - \hat{u}_1 \end{bmatrix}, \quad l = \Delta(u_1), \tag{3.36}$$

and the sin/cos terms can be approximated as polynomials using a second order Taylor series expansion about $e_3 = 0$.

SOS Tracking Controller

We solve the SOS program (3.32) with $\alpha = 5$, $\tau_{\max} = 0.05$s, $P(e) = e_1^2 + e_2^2$, $\hat{\mathcal{U}} = \{\hat{u} \in \mathbb{R}^2 : \|u\|_2^2 \leq 1\}$, and $\mathcal{U} = \mathbb{R}^2$. As can be seen in the SOS cost function (3.32a), larger β encourages a small value of δ and smaller β encourages a smaller value of c. After searching over several values, $\beta = 1e3$ was selected because it gave the smallest error bound while still yielding a δ value on the order of $1e{-}4$ which was approximate magnitude of the numerical tolerance from the SOS solver. Since the dynamics (3.36) don't depend on $\hat{\mathcal{X}}$, we don't need to select $\hat{\mathcal{X}}$ before solving the SOS program. We use the SOSTOOLS toolbox [63] in MATLAB with solver MOSEK [64], and after 15 iterations (28 minutes):

$$\begin{aligned} &\gamma = 1,\ \delta = 7 \times 10^{-4},\ c = 10.63 \\ &V = 0.30e_1^2 + 0.75e_1x_F + 0.13e_2^2 + 0.23e_3^2 + 0.70x_F^2 \\ &k_1 = 0.61e_1e_2 - 0.07e_2\hat{u}_2 - 0.28e_2x_F - 5.76e_3 + 0.82\hat{u}_1 \\ &k_2 = -0.001e_2\hat{u}_1 - 0.67e_1 + 0.11\hat{u}_2 + 2.01x_F. \end{aligned} \tag{3.37}$$

Terms with coefficient magnitudes less than 1e-4 have been omitted. The value $c = 10.63$ indicates an error bound radius of $\sqrt{c} = 3.26$. Hence, the planner state constraints will be shrunk by the set $\mathcal{O} = \{e \in \mathbb{R}^3 : e_1^2 + e_2^2 \leq 3.26^2\}$.

MPC Planning Controller

MPC is used as the planning controller to generate a motion plan for the vehicle that will avoid circular obstacles (bloated by the error bound) and reach a target region (shrunk by the error bound). At each time step, the following optimization is solved. We apply the

input $\hat{u}_t$ at the current time step and we resolve the optimization at the next step.

$$\min_{\substack{u_t,\dots,\\ u_{t+T-1}}} \sum_{k=t}^{t+T-1} (\hat{u}_k^\top R \hat{u}_k + (\hat{x}_k - \hat{x}_{\text{des}})^\top Q (\hat{x}_k - \hat{x}_{\text{des}})) \tag{3.38a}$$
$$+ (\hat{x}_{t+T} - \hat{x}_{\text{des}})^\top Q_T (\hat{x}_{t+T} - \hat{x}_{\text{des}})$$

$$\text{s.t. } \forall k \in \{t, \dots, t+T-1\}:$$
$$\hat{x}_{k+1} = \hat{x}_k + dt \cdot \hat{f}(x_k, u_k), \tag{3.38b}$$
$$\hat{x}_k \in \hat{\mathcal{X}} = \mathcal{X} \ominus \mathcal{O}, \tag{3.38c}$$
$$\hat{u}_k \in \hat{\mathcal{U}}, \tag{3.38d}$$
$$\hat{x}_{t+T} \in \hat{\mathcal{X}} = \mathcal{X} \ominus \mathcal{O}, \tag{3.38e}$$
$$\hat{x}_t = \hat{x}(dt \cdot t) \tag{3.38f}$$

The cost (3.38a) penalizes the control input and the distance from the desired final point $\hat{x}_{\text{des}}$ at the center of the target set. Constraint (3.38b) is the forward Euler discretized planner dynamics, and (3.38c) and (3.38e) are planner state constraints, which are tracker obstacle avoidance constraints shrunk by the error bound $\mathcal{O}$. Constraint (3.38d) restricts the input, and (3.38f) ensures that the optimization starts from the current planner state. We solve the optimization using the toolbox YALMIP [65] in MATLAB and the solver Ipopt [57], with parameters $T = 20$, $dt = 0.1$s, $R = \frac{1}{2} I_2$, $Q = 10 I_3$, $Q = 100 I_3$, and $\hat{x}_{\text{des}} = [46\text{m}; 14\text{m}]$. $\mathcal{X} = \{x \in \mathbb{R}^3 : (x_{1:2} - x_{\text{obs},j})^2 \geq r_{\text{obs},j}^2, j = 1, 2\}$, $x_{\text{obs},1} = [15\text{m}; 5\text{m}]$, $x_{\text{obs},2} = [30\text{m}; 15\text{m}]$, $r_{\text{obs},1} = r_{\text{obs},2} = 2.74$m.

Results

Combining the planning and tracking controllers, we simulate the combined system using the scenario shown in Figure 3.3. The planner trajectory in red avoids the bloated obstacles and reaches the shrunk target region. The tracker trajectory in blue tracks the planner trajectory closely; it does intersect with the bloated obstacles, but not with the true obstacles, preserving safety. It also reaches the target region.

Figure 3.4 shows the evolution of the error in (e_1, e_2) space and confirms that the error never leaves the bound $\mathcal{O}$. Note that while $\Omega(V, \gamma)$ is invariant and its projection lies in $\mathcal{O}$, $\mathcal{O}$ itself is not necessarily invariant.

3.7 Conclusion

This chapter used α-IQCs to extend the planner-tracker framework to accommodate unmodeled dynamics at the input of the tracker model. An SOS program was formulated to search for a tracking controller κ and an error bound $\mathcal{O}$. The method was demonstrated on a vehicle obstacle avoidance example with an MPC planner and an input delay in the tracker model.

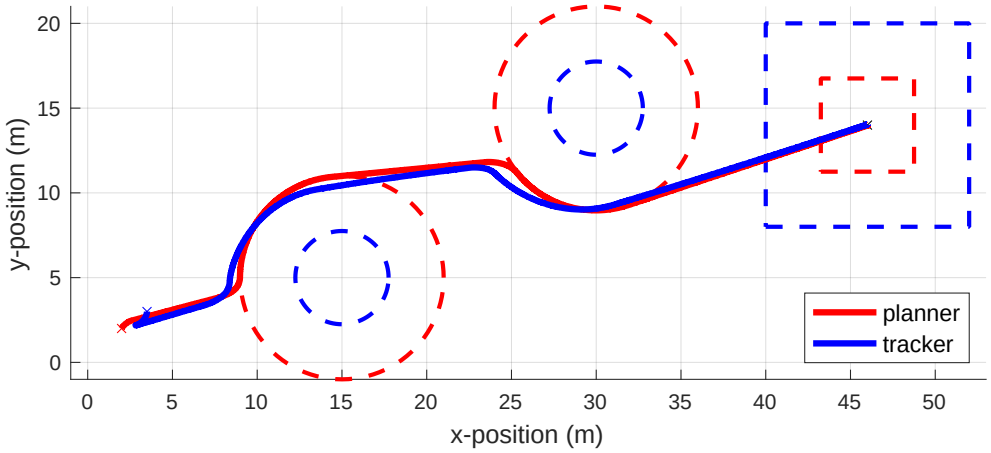


Figure 3.3: Planner-tracker simulation. The planner trajectory in red avoids the bloated obstacle (dashed red circles) and reaches the shrunk target region (dashed red square). The tracker trajectory in blue tracks the planner and avoids the true obstacles (dashed blue circles) and reaches the true target region (dashed blue square).

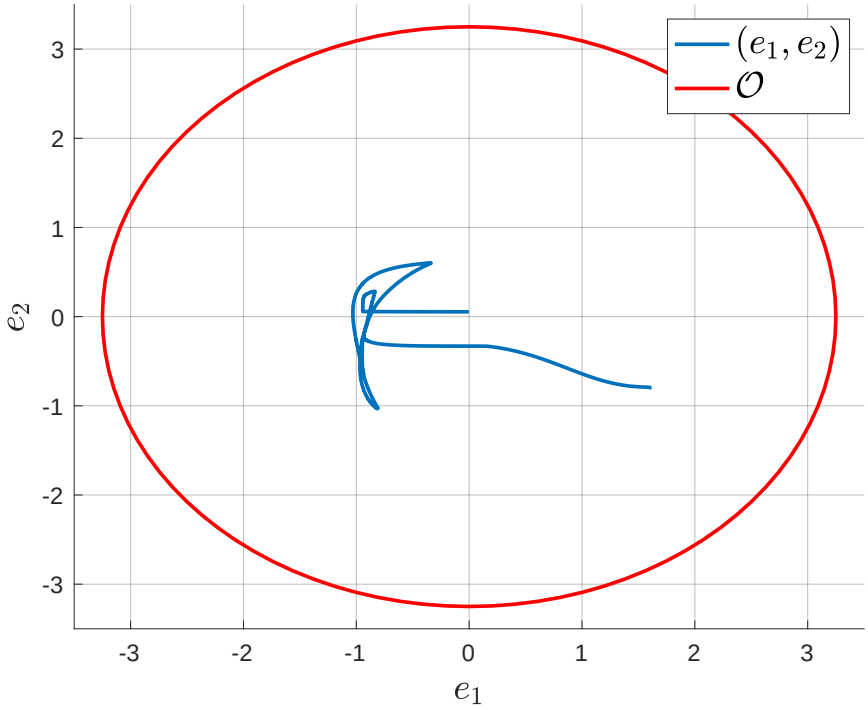


Figure 3.4: Position error (e_1, e_2) and the error bound $\mathcal{O}$.

Chapter 4

Case Study: Driver-in-the-Loop Contingency MPC

4.1 Abstract

We present a framework for shared vehicle control between a human driver and an autonomous system. The proposed control strategy, termed Driver-in-the-Loop Contingency Model Predictive Control (MPC), is inspired by the concept of contingency planning and is designed to intervene from the driver under emergency conditions in a manner that is smooth and not overly conservative. Driver-in-the-Loop Contingency MPC relies on the computation of invariant sets, which are used as MPC terminal sets. The proposed method is demonstrated on two longitudinal traffic scenarios: (1) vehicle-following, and (2) an intersection where the cross-traffic has the right of way. We use these examples to demonstrate the safety of the controller as well as the inherent trade-off between smooth intervention and minimal intervention.

4.2 Introduction

Shared vehicle control between an autonomous system and a human driver has the potential to improve traffic safety if widely implemented and adopted by drivers. Advanced Driver Assistance Systems (ADAS) are already prevalent, with features like blind spot monitoring, lane-keeping, and automatic braking, and these features have decreased the number of driver-caused accidents [66]. However, these systems are often disabled by drivers because even a low false positive rate is strongly undesirable [67].

In this chapter, we consider a shared control framework [68] where the autonomy has ultimate control of the vehicle, as in [69–71]. The driver applies a control input, i.e., a steering, braking, or throttle command, and the autonomy can decide whether to apply the driver's exact command, or to *intervene* from the driver and apply a different command. To improve the driver experience, whenever the autonomy must intervene, it should do so in

the least conservative manner possible in order to maintain the driver's sense of control over the vehicle motion.

To reduce conservatism, we employ the framework of contingency planning [72], a planning strategy which preserves the existence of a safe trajectory at all times. The aim in applying contingency planning to shared vehicle control is to delay any intervention from the driver as much as possible. To the best of our knowledge, the idea of contingency planning for driver assistance systems has not been previously explored.

We demonstrate the method on the two longitudinal scenarios shown in Figure 4.1: (a) vehicle-following, and (b) an intersection where the cross-traffic has the right of way. These longitudinal scenarios are highly relevant and provide a useful case study for implementing contingency planning in shared vehicle control. Other safety-critical longitudinal scenarios, where a similar approach could be implemented, include approaching an occluded area, driving in areas with human road users, etc.

In particular, we extend a contingency planning strategy called Contingency MPC (CMPC) [73] to the setting of shared vehicle control, and we refer to this extension as Driver-in-the-Loop CMPC. We use a new MPC cost function to reflect the priority of maximizing driver control, and we use an MPC terminal set constraint to guarantee persistent feasibility with respect to all constraints, including safety constraints that describe the interaction between the ego and ado[1] vehicles. We explore a trade-off that arises between *minimal* intervention and *smooth* intervention from the driver.

In Section 4.3, we introduce set-theoretic concepts utilized in defining and computing the CMPC terminal set. In Section 4.4, we formulate the MPC optimization problem, discuss extensions from CMPC, and highlight resulting properties of the controller. We apply Driver-in-the-Loop CMPC to the vehicle-following scenario in Section 4.5 and the intersection scenario in Section 4.6, and we demonstrate the trade-off between smooth intervention and minimal intervention. Finally, we provide concluding remarks and directions for future work in Section 4.7.

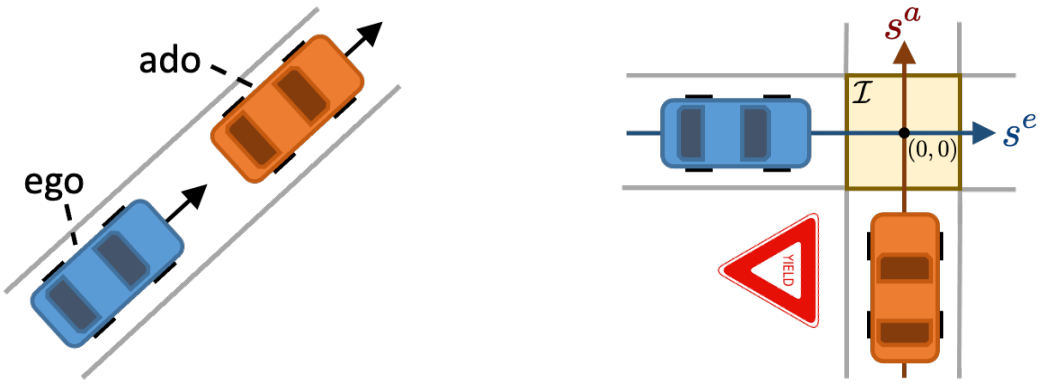
4.3 Preliminaries

Contingency MPC

Contingency planning, introduced in [72], is a trajectory planning strategy wherein the feasibility of a safe maneuver is preserved at all times. Safety is defined with respect to a particular contingency event (e.g., a vehicle ahead of the ego vehicle suddenly applying maximum braking), and if that event occurs, the safety maneuver can be executed.

CMPC is a particular form of contingency planning that uses an MPC problem to preserve the existence of the safe trajectory [73]. In a standard MPC problem, a sequence of optimal inputs is calculated over a prediction horizon, the first input is applied, and the procedure is repeated at the next time step in a receding horizon fashion [74]. In CMPC, however, *two*

[1]Vehicles other than the ego vehicle are referred to as *ado* vehicles.



(a) Vehicle-following scenario.

(b) Intersection scenario.

Figure 4.1: Two longitudinal traffic scenarios where the ego vehicle (blue) must maintain safety with respect to the ado vehicle (red).

sequences of optimal inputs and states are computed: a *nominal* branch, and a *contingency* branch. The contingency branch includes constraints assuming the contingency event takes place. The two branches are constrained to have a common first input, which is applied in a receding horizon fashion as in standard MPC.

Invariant Set Computations

Driver-in-the-Loop CMPC includes a terminal set constraint in order to guarantee persistent feasibility, meaning that if the MPC problem is feasible at one time step, it is guaranteed to be feasible at the next time step [74]. Constructing terminal sets for the scenarios in this Chapter involves two operations: backward reachable sets and maximum control invariant sets. In each definition, consider a system $x_{k+1} = f(x_k, u_k)$ with state constraints $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ and input constraints $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$.

(1) **Backward Reachable Set** (BRS). The K-step BRS from $\mathcal{S}$, denoted $\mathcal{B}(\mathcal{S}, K)$, is the set of states that can be driven to $\mathcal{S}$ in K steps while obeying state and input constraints. It can be computed by recursively applying the one-step backward reachable set operator, defined as:

$$\text{Pre}(\mathcal{S}) = \{x \in \mathcal{X} : \exists u \in \mathcal{U} \text{ s.t. } f(x, u) \in \mathcal{S}\}. \tag{4.1}$$

Initializing the recursion with $\mathcal{B}(\mathcal{S}, 0) = \mathcal{S}$, we have

$$\mathcal{B}(\mathcal{S}, k) = \text{Pre}(\mathcal{B}(\mathcal{S}, k-1)), \quad k = 1, \ldots, K. \tag{4.2}$$

Note that this set depends implicitly on f, $\mathcal{X}$, and $\mathcal{U}$.

(2) **Maximum Control Invariant Set** (MCIS). A nonempty set $\mathcal{C} \subseteq \mathcal{X}$ is a control invariant set if it satisfies the relation $\mathcal{C} \subseteq \text{Pre}(\mathcal{C})$. Then the MCIS, denoted $\mathcal{C}^{\max}(\mathcal{X})$, is the largest control invariant set within $\mathcal{X}$ (in the sense of set inclusion) [75]. In other words,

$\mathcal{C}^{\max}(\mathcal{X})$ is the largest subset of $\mathcal{X}$ that the system can be controlled to remain within for all time.

4.4 Problem Formulation

Considered Scenarios

In this Chapter we consider longitudinal scenarios, where the behavior of each vehicle is sufficiently captured by a double integrator model with speed and acceleration constraints. Let the superscript $i \in \{e, a\}$ denote the ego and ado vehicle, respectively. Let $x^i = [s^i; v^i]$ denote the vehicle state, where s^i is the longitudinal displacement along the vehicle path and v^i is the velocity, and let u^i be the acceleration input. Then the vehicle dynamics and constraints are

$$\dot{s}^i = v^i, \;\; \dot{v}^i = u^i, \;\; 0 \leq v^i \leq v^i_{\max}, \;\; a^i_{\min} \leq u^i \leq a^i_{\max}. \tag{4.3}$$

The model is discretized using a zero order hold scheme, and the state and input constraints are represented as polytopes $\mathcal{X}^i$ and $\mathcal{U}^i$, respectively. These linear time-invariant (LTI) dynamics and polytope constraints will be useful in computing the terminal set in the following MPC problem.

Note that this method is applicable to scenarios including systems other than double integrators, but the offline and/or online computations may prove difficult in some cases; future research is needed to investigate other safety critical scenarios and carefully consider the design in those cases.

Driver-in-the-Loop CMPC Optimization

For simplicity, we drop the superscript $i \in \{e, a\}$ for the ego/ado vehicles since all states and inputs in this optimization refer to the *ego* vehicle. Instead, we use the superscript $j \in \{n, c\}$ to refer to the nominal and contingency branches. Subscript $k \in \{0, \ldots, N\}$ denotes the time step. At each time step, the following optimization is solved *online*, i.e., during controller

operation:

$$\min_{\substack{u_0^n,\dots,u_{N-1}^n \\ u_0^c,\dots,u_{N-1}^c}} (1-P_c)\sum_{k=0}^{N-1}\left(u_k^n - u_k^{d,n}\right)^2 + P_c\sum_{k=0}^{N-1}\left(u_k^c - u_k^{d,c}\right)^2 \tag{4.4a}$$

$$\text{s.t. } x_{k+1}^n = Ax_k^n + Bu_k^n, \quad \forall\ k \tag{4.4b}$$

$$x_{k+1}^c = Ax_k^c + Bu_k^c, \quad \forall\ k \tag{4.4c}$$

$$h_k^n(x_k^n, u_k^n) \le 0, \quad \forall\ k \tag{4.4d}$$

$$h_k^c(x_k^c, u_k^c) \le 0, \quad \forall\ k \tag{4.4e}$$

$$x_N^c \in \mathcal{Z} \tag{4.4f}$$

$$u_0^n = u_0^c \tag{4.4g}$$

$$x_0^n = x_0^c = x(t) \tag{4.4h}$$

Here, $u_k^{d,j}$ is the driver's current ($k = 0$) or predicted ($k > 0$) input. We assume a model is given for calculating the driver's predicted inputs, which may differ along the nominal and contingency branches. This model may come from an inference module higher in the autonomy stack. Constraints (4.4b)-(4.4c) are the discretized dynamics for the nominal and contingency branches. N is the prediction horizon, which can be selected based on the amount of time into the future for which a driver input prediction is available. The functions h_k^j represents joint state and input constraints. The state constraints $\mathcal{X}^e$ and input constraints $\mathcal{U}^e$ are embedded into these functions, along with constraints regarding the interaction between the ego and ado vehicles. P_c reflects the contingency likelihood, which is assumed to be available a priori, e.g., by an inference module higher in the autonomy stack. $\mathcal{Z}$ is the terminal set, and details on its computation for each scenario are included in Sections 4.5 and 4.6. The constraints (4.4d)-(4.4f) are linear for the scenarios in this Chapter, which means the optimization is a quadratic program and can be efficiently solved.

Remark 7. *In this formulation the constraint $u_0^n = u_0^c$ requires the nominal and contingency branches to share only one common input, as in [73]. In general, however, there could be a longer shared segment between the two branches, i.e., $u_i^n = u_i^c,\ i = 1, \dots, N_{branch}$.*

Extensions from CMPC

We propose two key extensions from CMPC [73]:

(1) CMPC, as originally formulated, is amenable to a fully autonomous vehicle rather than a vehicle with a driver. In general, the CMPC cost function can include terms for comfort, fuel-efficiency, path-following, time-optimality, etc. In the presence of a driver, however, the priority is matching the driver's input whenever it is safe to do so. For this reason, we modify the cost function (4.4a) to be a weighted sum of deviations from the driver's current and predicted inputs along each branch.

(2) Persistent feasibility of an MPC problem can be guaranteed by using a suitably designed terminal set constraint [74]. Persistent feasibility is not addressed in [73], and in

general terminal sets are difficult to compute. However, for the longitudinal scenarios considered in this Chapter, the dynamics and constraints are linear, which means the terminal sets are polyhedra for which efficient algorithms and software exist, such as the Multi-Parametric Toolbox (MPT) [76] used in this work. Thus, we extend the standard CMPC formulation to include a terminal set constraint (4.4f).

Formulation Properties

The proposed MPC (4.4) has two key properties, which will be demonstrated numerically in Sections 4.5 and 4.6.

Property 1. The controller only intervenes if a current or anticipated driver input is unsafe. Otherwise, it is possible to match the driver's current measured input ($u_0^n = u_0^c = u_0^d$) and predicted future inputs ($u_k^n = u_k^{d,n}$ and $u_k^c = u_k^{d,c}$, $k = 1, \ldots, N-1$) without violating safety constraints. This is optimal since the cost achieves its minimum value of zero.

Property 2. A natural trade-off between delayed intervention and more prompt but gradual intervention can be achieved by varying the parameters P_c and N. For larger P_c, a higher weight is placed on the contingency branch cost. Qualitatively, this means the controller has a higher incentive to take preventative action sooner (e.g., via u_0^c) to prevent abrupt interventions further along the contingency branch.

Similarly, if the driver's predicted inputs are unsafe, increasing N further incentivizes the controller to intervene sooner in order to avoid late-horizon high-cost deviations from the driver's input. This may result in smaller variations between subsequent controller actions, i.e., smoother intervention, as will be demonstrated in Section 4.6. This smoother intervention for high N can be contrasted to the case where $N = 1$ and the controller intervenes if and only if the *current* input is unsafe.

4.5 Vehicle-Following Scenario

Consider the scenario shown in Figure 4.1(a), where the ego vehicle is driving on a straight road, following an ado vehicle. The objective of the ego vehicle is to follow the driver's commands while remaining safely behind the ado vehicle at all times, even if the ado vehicle applies maximum braking. That is, the *contingency event* in this scenario is that the ado vehicle applies maximum braking.

For this scenario, we enforce state constraints at each step of the prediction horizon to ensure that the ego vehicle remains at least some buffer distance d behind the ado vehicle, i.e., $s_k^e \leq s_k^a - d$. For the nominal constraints (4.4d), the ado position s_k^a is propagated assuming the ado vehicle maintains its current speed. For the contingency constraints (4.4e), s_k^a is propagated assuming the ado vehicle applies maximum braking.

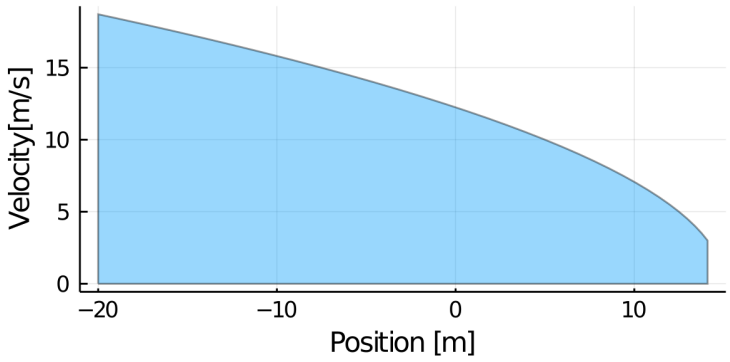


Figure 4.2: Terminal set for the vehicle-following scenario. Here, $d = 5\text{m}$, $N = 14$, $T = 0.1\text{s}$, and $x^a = [10\text{m}; 10\text{m/s}]$. From this we can calculate $\bar{x}^a = [19.1\text{m}; 3\text{m/s}]$ and $F = 20$.

Terminal Set

We follow the procedure in [77], summarized below, to construct the terminal set for this scenario. Note that these set computations depend implicitly on the system dynamics, which in this case are the *joint* dynamics of the ego and ado vehicles, where the ado input is fixed.

(1) Compute the MCIS for the stopped ado vehicle, i.e., the set of states such that the ego vehicle can stop behind the stopped ado vehicle's position:

$$\mathcal{S} = \mathcal{C}^{\max}\left(\{[x^e; x^a] : s^e \leq s^a - d,\ v^a = 0\}\right)|_{u^a=0}. \tag{4.5}$$

(2) Compute the $(F - N)$-step BRS from this set (assuming the ado vehicle applies maximum braking), where F is the minimum number of time steps in which the ado vehicle can come to a stop:

$$\mathcal{Q} = \begin{cases} \mathcal{B}(\mathcal{S}, F - N)|_{u^a = u^a_{\min}}, & F > N \\ \mathcal{S}, & \text{else} \end{cases}. \tag{4.6}$$

(3) Slice $\mathcal{Q}$ at $\bar{x}^a$, where $\bar{x}^a$ is the ado state at the end of the horizon under maximum braking: $\mathcal{Z} = \{x^e : [x^e; \bar{x}^a] \in \mathcal{Q}\}$.

Note that steps (1) and (2) are done offline, while step (3) is done online. The set computations are done using MPT [76] in MATLAB. Because the system is LTI with polytopic constraints, this primarily involves computing projections and intersections of polytopes. An example terminal set is displayed in Figure 4.2. The persistent feasibility of the set $\mathcal{Z}$ constructed in this manner is proved in [77].

Results

We implement the proposed controller (4.4) for this vehicle-following scenario. We generate $\{u_k^{d,n}\}_{k=0}^{N-1}$ and $\{u_k^{d,c}\}_{k=0}^{N-1}$ using a single-lane vehicle-following model called the Intelligent

Var.	Value	Var.	Value	Var.	Value
N	14	$a^e_{\min} = a^a_{\min}$	-5 m/s^2	d	5 m
M	35	$a^e_{\max} = a^a_{\max}$	1 m/s^2	x^e_0	[0 m; 10 m/s]
T	0.1 s	$v^e_{\max} = v^a_{\max}$	30 m/s	x^a_0	[10 m; 10 m/s]

Table 4.1: Problem parameters for the vehicle-following scenario.

Driver Model [78]. We simulate the controller using the parameters shown in Table 4.1. The MPC was run for M time steps. These computations were done using JuMP [79] (an optimization modeling package implemented in the Julia language) with the optimization solver Ipopt [57].

We simulate this controller for a range of P_c values for two different ado trajectories, and we plot the resulting controller intervention (controller input minus driver input). In Figure 4.3 (top), the ado vehicle initiates a braking maneuver with acceleration $a^a_{\min}$ at $t = 1$s (i.e., the contingency event *happens*), and in Figure 4.3 (bottom) the ado vehicle maintains a constant speed (i.e., the contingency event *does not happen*).

Figure 4.3 illustrates Property 2 from Section 4.4: the trade-off between *smooth* controller intervention when the contingency event occurs and *minimal* controller intervention when the contingency event does not occur. In Figure 4.3 (top), when $P_c = 0$, there are no terms in the cost function to make the controller take preventative action and smooth the intervention. Hence, for this value of P_c, the controller waits the longest amount of time before intervening, but when it does intervene, it does so the most abruptly. Increasing P_c causes the controller to intervene sooner and more smoothly.

Smooth intervention (high P_c) is desirable, but it comes at the cost of interventions when the contingency event does not occur, as in Figure 4.3 (bottom). (Here, the interventions are fairly minimal and may even be undetectable by the driver.) Hence, there exists a trade-off between intervening smoothly and delaying the intervention as much as possible. This trade-off can be achieved by tuning the parameter P_c based on the likelihood of the contingency event and the preferences of an individual driver.

4.6 Intersection Scenario

Now we consider the scenario shown in Figure 4.1(b), in which the ego and ado vehicles both approach an intersection, and the ado vehicle has the right of way. The two vehicles should never simultaneously occupy the intersection. Thus, the ego vehicle has to decide if it should stop before the intersection and wait for the ado vehicle to pass, or if it can safely move through the intersection before the ado vehicle enters the intersection. Again, the *contingency event* is the ado vehicle's worst-case behavior, which in this case is applying maximum acceleration.

Each vehicle is modeled as a double integrator where s^e and s^a represent the positions of

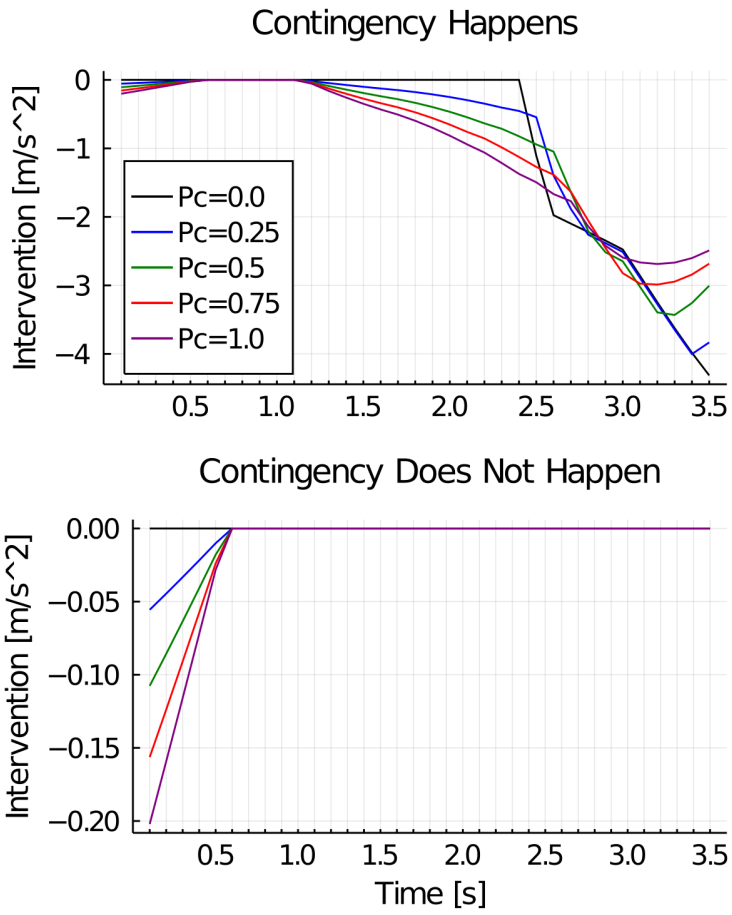


Figure 4.3: Controller intervention for a range of P_c values when the contingency event does happen (top) or does not happen (bottom).

the ego and ado vehicles along their straight-line paths and $s^e = s^a = 0$ is the point where the paths cross in the center of the intersection as shown in Figure 4.1(b). Let l be half the width of the intersection. The interaction set to be avoided becomes

$$\mathcal{I} = \{x^e, x^a : |s^e| \leq l \text{ and } |s^a| \leq l\}. \tag{4.7}$$

Terminal Sets

In order to avoid the interaction set $\mathcal{I}$, it is sufficient to guarantee that the ego vehicle can either *wait* for the ado vehicle to pass through the intersection first, or can successfully *go* through the intersection and exit it before the ado vehicle enters. We construct two invariant sets, $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}$, to achieve each of these goals and impose a joint terminal set constraint

$$x_N^e \in \mathcal{Z}_{\text{wait}} \cup \mathcal{Z}_{\text{go}}. \tag{4.8}$$

First we construct $\mathcal{Z}_{\text{wait}}$, the invariant set that guarantees the ego vehicle will be able to *wait* and let the ado vehicle pass. We note that this is simply the set of ego vehicle states

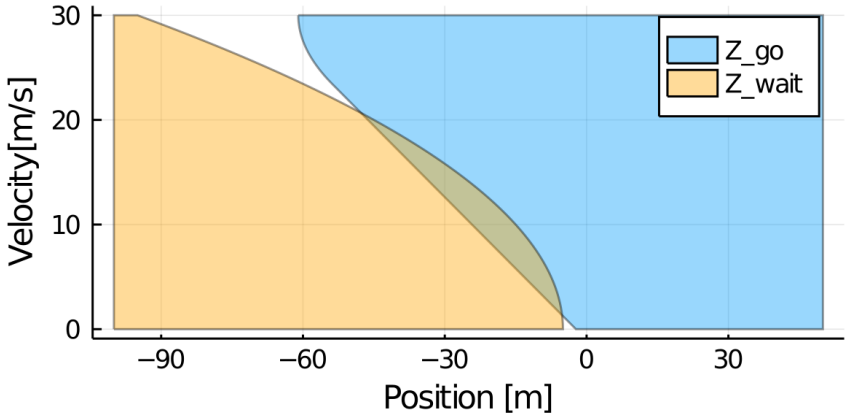


Figure 4.4: Terminal sets $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}(F)$, where $F = 27$ and $N = 5$.

such that the ego vehicle can stop by the beginning of the intersection ($s^e = -l$). This set can be computed (offline) as

$$\mathcal{Z}_{\text{wait}} = \mathcal{C}^{\max}(\{x^e : s^e \leq -l\}). \tag{4.9}$$

$\mathcal{Z}_{\text{go}}$, on the other hand, is the set of ego states such that the ego vehicle can exit the intersection before the ado vehicle enters it. Computation of this set is again straightforward. First, we compute F, the minimum number of time steps in which the ado vehicle can possibly enter the intersection (subject to speed and acceleration constraints). If $F > N$, i.e. the ado vehicle cannot enter the intersection during the current prediction horizon, then $\mathcal{Z}_{\text{go}}$ is the $(F - N)$-step backward reachable set from $\{x^e : s^e \geq l\}$, the set of ego states that are past the intersection.

If $F \leq N$, however, the ado vehicle is able to enter the intersection during the current prediction horizon, and hence $\mathcal{Z}_{\text{go}} = \varnothing$. In this case, we simply encode the pointwise constraint $s^e_F \geq l$ in (4.4e). We can express $\mathcal{Z}_{\text{go}}(F)$ as

$$\mathcal{Z}_{\text{go}}(F) = \begin{cases} \mathcal{B}\left(\{x^e : s^e \geq l\}, F - N\right), & F > N \\ \varnothing, \quad \text{otherwise} \end{cases}. \tag{4.10}$$

$\mathcal{Z}_{\text{go}}(F)$ is computed offline for a range of F-values. Online, F is computed for the current ado vehicle state and $\mathcal{Z}_{\text{go}}(F)$ is retrieved. Once the ado vehicle has exited the intersection, we remove the terminal constraint altogether, since the risk of contingency has passed. For a sample value of F, we plot $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}(F)$ in Figure 4.4.

Remark 8. *The terminal constraint* (4.8) *is in general non-convex. Practically, we can address this problem by solving two optimizations, one wait optimization with the constraint* $x^e_N \in \mathcal{Z}_{wait}$ *and one go optimization with the constraint* $x^e_N \in \mathcal{Z}_{go}$*, and then taking the solution to the optimization problem that yields the lower cost. It would also be possible to bias toward either waiting or going by weighting the cost functions of the wait and go optimizations differently.*

Var.	Value	Var.	Value	Var.	Value
P_c	0.5	$a^e_{\min} = a^a_{\min}$	-5 m/s^2	l	5 m
M	40	$a^e_{\max} = a^a_{\max}$	3 m/s^2	x^e_0	[-40 m; 15 m/s]
T	0.1 s	$v^e_{\max} = v^a_{\max}$	30 m/s	x^a_0	[-40 m; 20 m/s]

Table 4.2: Problem parameters for the intersection scenario.

Results

For sample ego and ado initial conditions, we simulate the controller to verify that safety is preserved. First, we use initial conditions such that only the terminal constraint $x^e_N \in \mathcal{Z}_{\text{go}}$ can be satisfied. Figure 4.5 shows the ego states (both the closed loop states and the open loop N-step MPC prediction) as well as the safe sets $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}$ for a collection of snapshot times.

As the ado vehicle moves, the set $\mathcal{Z}_{\text{go}}$ shrinks. Once the ado vehicle is able to reach the intersection within the current prediction horizon ($k = 17, 23$), $\mathcal{Z}_{\text{go}}$ becomes empty. By that time, however, the ego vehicle is guaranteed to be able to exit the intersection within the same number of time steps. As seen in Figure 4.5 (bottom), the ado vehicle enters the intersection at time step $k = 17$, by which point the ego vehicle has already left the intersection.

We show a similar set of results in Figure 4.6, this time for ego and ado initial conditions such that only waiting is feasible. The terminal set constraint $\mathcal{Z}_{\text{wait}}$ is satisfied for the first three snapshots, while the ado vehicle is still approaching or in the intersection. In the fourth snapshot, however, the ado vehicle has exited the intersection, so the set constraints no longer hold and the ego vehicle is free to enter the intersection.

Lastly, we illustrate Property 1 from Section 4.4: that the controller will intervene only when the driver's measured or predicted inputs are unsafe. For fixed ego/ado initial conditions (where only waiting is feasible) and for a fixed set of ado inputs, we consider two different driver input policies:

(1) Driver applies maximum braking (*safe*), and

(2) Driver tries to maintain a constant speed (*unsafe*).

In Figure 4.7, we plot the driver's input and the controller's input for several values of the parameter N, the MPC horizon. The controller only intervenes when the predicted driver input is unsafe, and when the controller does intervene, it does so more smoothly for larger values of N. The parameters used for this simulation are displayed in Table 4.2.

4.7 Conclusion

In this Chapter, we applied contingency planning to shared vehicle control, a setting where it is important to make the driver feel in control of the vehicle whenever possible. We proposed a method called Driver-in-the-Loop CMPC, which we applied to two longitudinal

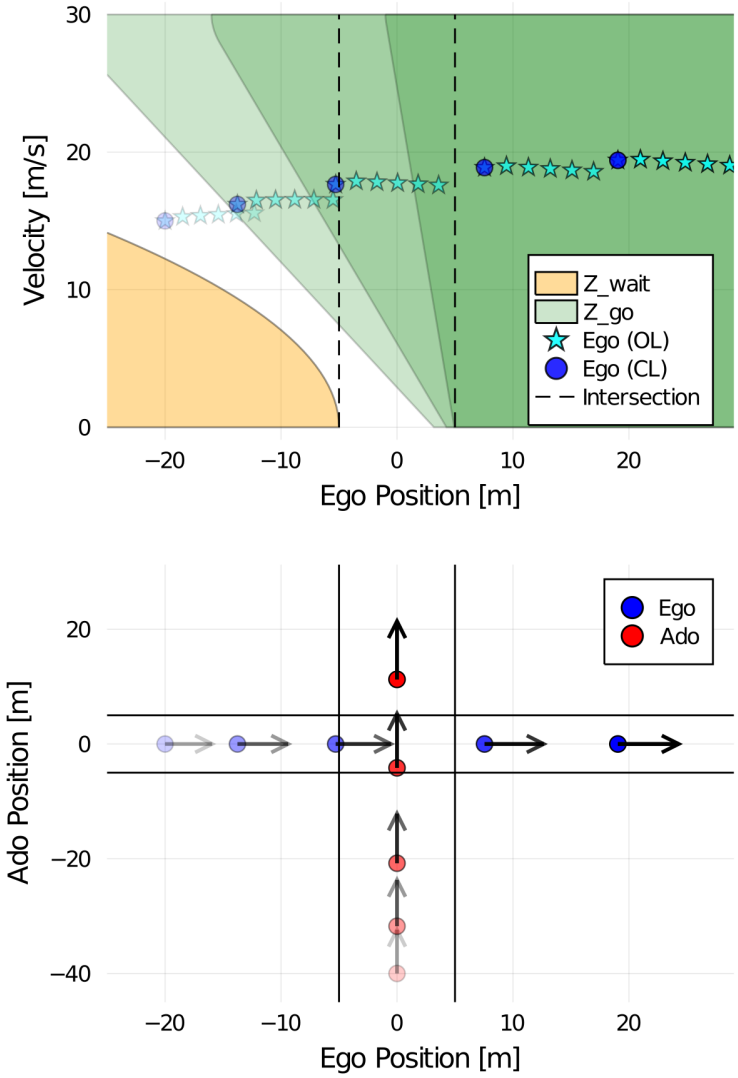


Figure 4.5: Ego vehicle crosses the intersection before the ado vehicle, showing a collection of snapshot times: $k \in \{1, 5, 10, 17, 23\}$. The earlier time steps are indicated via more transparent markers. **Top**: ego states (both the closed loop states and the open loop N-step MPC prediction) and safe sets $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}$. **Bottom**: ego and ado positions with lane boundaries and intersection. The length of the arrows are proportional to the magnitude of each vehicle's velocity.

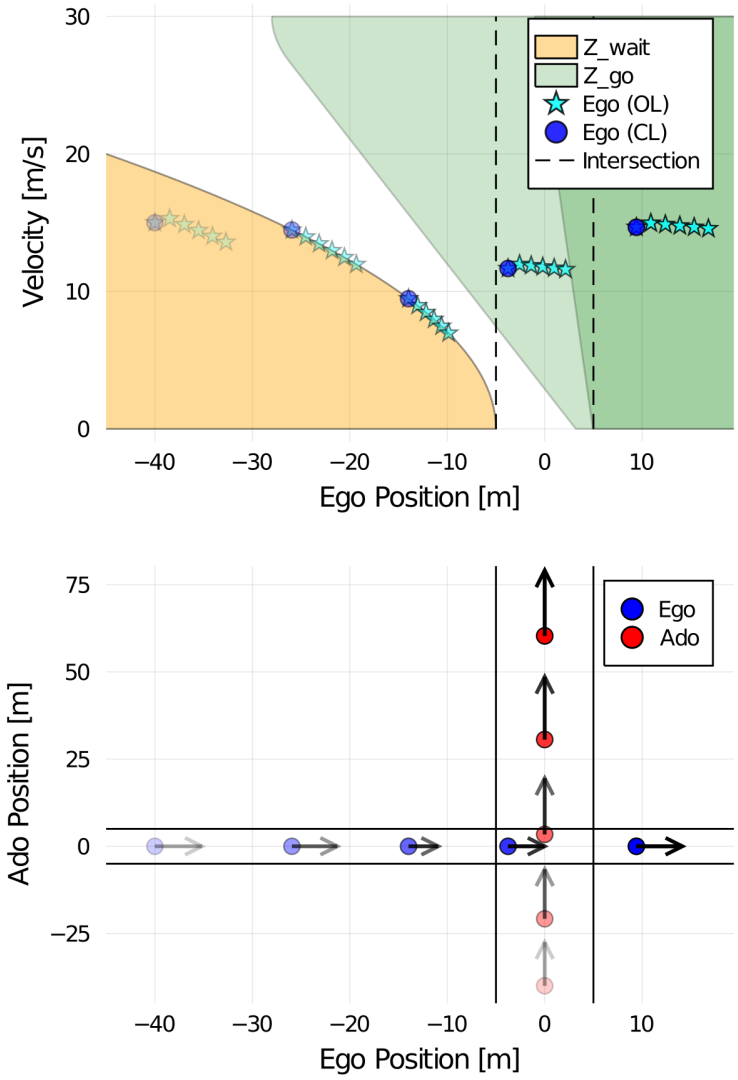


Figure 4.6: Ego vehicle waits for the ado vehicle to cross the intersection, showing snapshot times: $k \in \{1, 10, 20, 30, 40\}$. The earlier time steps are indicated via more transparent markers. **Top**: ego states (both the closed loop states and the open loop N-step MPC prediction) and safe sets $\mathcal{Z}_{\text{wait}}$ and $\mathcal{Z}_{\text{go}}$. **Bottom**: ego and ado positions with lane boundaries and intersection. The length of the arrows are proportional to the magnitude of each vehicle's velocity.

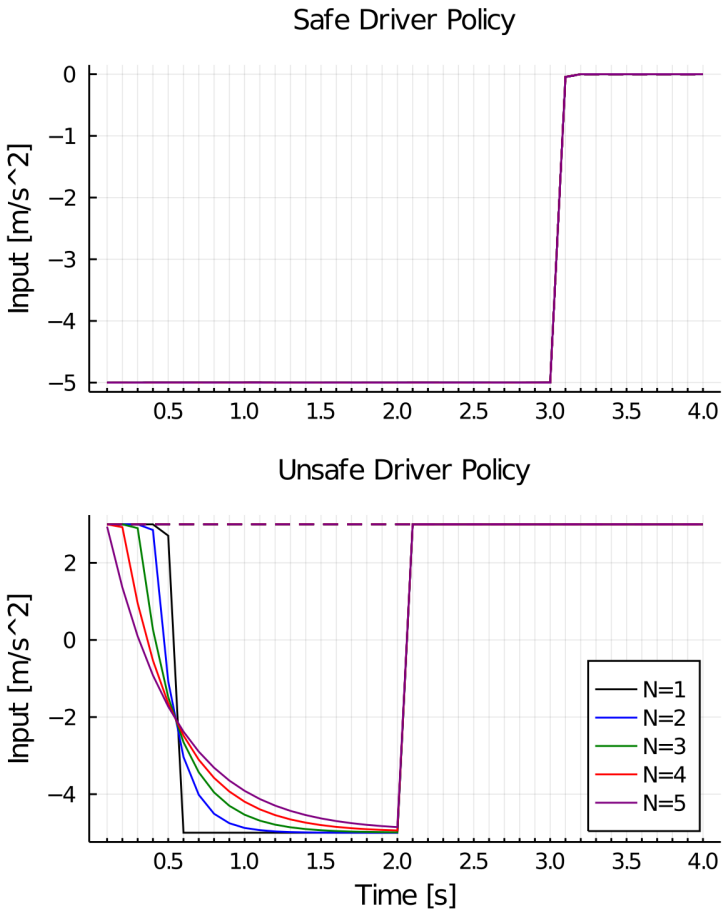


Figure 4.7: Driver's input and controller's input for safe and unsafe predicted driver polices, over a range of N values.

traffic scenarios. This involved the construction of terminal sets to guarantee persistent feasibility. We demonstrated that the proposed control scheme preserves safety, and we showed the trade-off between minimal intervention and smooth intervention that can be achieved by tuning parameters in the formulation. Future work will involve comparing the method with other shared control strategies and testing Driver-in-the-Loop CMPC on real vehicles to evaluate the method in practice.

Chapter 5

Backstepping Approach for Tracking

5.1 Motivation

In Chapters 2 and 3, we formulated SOS programs to solve for the storage function V and the tracking controller κ. These SOS programs allow us to encode initial condition constraints, input constraints, etc., and we can alternately solve two convex subproblems over different sets of decision variables. However, drawbacks of the SOS approach for tracking control include:

1. The controller κ will not, in general, have a clear interpretation. It will be a generic polynomial function of a user-specified order whose coefficients may not have a clear meaning.

2. SOS programming is prone to numerical sensitivities. The LMIs present in an SOS program often contain implicit equality constraints which can allow a solver to conclude that a feasible problem is infeasible. Adding slack variables, like δ in (3.32d), help to mitigate this challenge.

It is desirable to develop a tracking control strategy that does not have the numerical challenges of SOS programming and that has an interpretable form. In this Chapter, we derive a tracking controller using a backstepping-based approach [80].

5.2 Problem Formulation

In this Chapter, we impose more structure on the planning and tracking models, where the state variable is made of of multiple stages, and the each stage is the input to the next stage. In particular, the tracking model is of the form

$$\dot{x}_k = f_k(x_{1:k}) + g_k(x_{1:k})x_{k+1},\ k = 1, ..., n-1 \tag{5.1}$$

$$\dot{x}_n = f_n(x) + g_n(x)u + w. \tag{5.2}$$

The tracking state is $x \in \mathbb{R}^{nm}$ where $x = [x_1, \ldots, x_n]$ and each $x_i \in \mathbb{R}^m$. For $k < n$, let $x_{1:k}$ denote $[x_1, \ldots, x_k] \in \mathbb{R}^{mk}$. The tracking input is $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ and the disturbance is $w \in \mathcal{W} \subseteq \mathbb{R}^m$. The planning model is of the form

$$\dot{\hat{x}}_k = \hat{f}_k(\hat{x}_{1:k}) + \hat{g}_k(\hat{x}_{1:k})\hat{x}_{k+1},\ k = 1, ..., n-1 \tag{5.3}$$

$$\dot{\hat{x}}_n = \hat{f}_n(\hat{x}) + \hat{g}_n(\hat{x})\hat{u} \tag{5.4}$$

The planning state is $\hat{x} \in \mathbb{R}^{nm}$ where $\hat{x} = [\hat{x}_1, \ldots, \hat{x}_n]$ and each $\hat{x}_i \in \mathbb{R}^m$. For $k < n$, let $\hat{x}_{1:k}$ denote $[\hat{x}_1, \ldots, \hat{x}_k] \in \mathbb{R}^{mk}$. The planning input is $\hat{u} \in \hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$.

Our goal is to define an error variable and a control law that keeps that error variable in a bounded set $\mathcal{O}$. Unlike in Chapters 2 and 3 where the error was user-defined, here the error is defined in stages as part of the backstepping procedure.

5.3 Backstepping Procedure

Step 1: Define $e_1 = x_1 - \hat{x}_1$. Then

$$\dot{e}_1 = f_1(x_1) + g_1(x_1)x_2 - \hat{f}_1(\hat{x}_1) - \hat{g}_1(\hat{x}_1)\hat{x}_2. \tag{5.5}$$

Select $\kappa_1(x_1, \hat{x}_1)$ such that

$$e_1^\top \kappa_1(x_1, \hat{x}_1) \geq c_1 \|e_1\|^2 \tag{5.6}$$

for some $c_1 > 0$. Let α_1 be such that substituting $x_2 = \alpha_1$ gives $\dot{e}_1 = -\kappa_1(x_1, \hat{x}_1)$, i.e.,

$$\alpha_1(x_1, \hat{x}_{1:2}) = g_1(x_1)^{-1}(-\kappa_1(x_1, \hat{x}_1) - f_1(x_1) + \hat{f}_1(\hat{x}_1) - \hat{g}_1(\hat{x}_1)\hat{x}_2). \tag{5.7}$$

Accounting for the difference between x_2 and α_1, we have

$$\dot{e}_1 = -\kappa_1(x_1, \hat{x}_1) + g_1(x_1)(x_2 - \alpha_1(x_1, \hat{x}_{1:2})). \tag{5.8}$$

Step 2: Define $e_2 = x_2 - \alpha_1(x_1, \hat{x}_{1:2})$, yielding

$$\dot{e}_1 = -\kappa_1(x_1, \hat{x}_1) + g_1(x_1)e_2, \tag{5.9}$$

$$\dot{e}_2 = f_2(x_1, x_2) + g_2(x_1, x_2)x_3 - \dot{\alpha}_1(x_{1:2}, \hat{x}_{1:3}), \tag{5.10}$$

where $\dot{\alpha}_1$ is the algebraic expression for the time derivative of α_1. Select $\kappa_2(x_{1:2}, \hat{x}_{1:2})$ such that

$$\kappa_2(x_{1:2}, \hat{x}_{1:2}) \geq c_2 \|e_2\|^2 \tag{5.11}$$

for some $c_2 > 0$. Let α_2 be such that substituting $x_3 = \alpha_2$ gives $\dot{e}_2 = -g_1(x_1)^\top e_1 - \kappa_2(x_{1:2}, \hat{x}_{1:2})$, i.e.,

$$\alpha_2(x_{1:2}, \hat{x}_{1:3}) = g_2(x_{1:2})^{-1}(-g_1(x_1)^\top e_1 - \kappa_2(x_{1:2}, \hat{x}_{1:2}) - f_2(x_{1:2}) + \dot{\alpha}_1(x_{1:2}, \hat{x}_{1:3})) \tag{5.12}$$

Then we have

$$\dot{e}_2 = -g_1(x_1)^\top e_1 - \kappa_2(x_{1:2}, \hat{x}_{1:2}) + g_2(x_{1:2})(x_3 - \alpha_2(x_{1:2}, \hat{x}_{1:3})). \tag{5.13}$$

Step k: Suppose

$$\dot{e}_{k-1} = -g_{k-2}(x_{1:k-2})^\top e_{k-2} - \kappa_{k-1}(x_{1:k-1}, \hat{x}_{1:k-1}) + g_{k-1}(x_{1:k-1})(x_k - \alpha_{k-1}(x_{1:k-1}, \hat{x}_{1:k})). \tag{5.14}$$

Define $e_k = x_k - \alpha_{k-1}(x_{1:k-1}, \hat{x}_{1:k})$ so that

$$\dot{e}_{k-1} = -g_{k-2}(x_{1:k-2})^\top e_{k-2} - \kappa_{k-1}(x_{1:k-1}, \hat{x}_{1:k-1}) + g_{k-1}(x_{1:k-1})e_k, \tag{5.15}$$

$$\dot{e}_k = f_k(x_{1:k}) + g_k(x_{1:k})x_{k+1} - \dot{\alpha}_{k-1}(x_{1:k}, \hat{x}_{1:k+1}). \tag{5.16}$$

Select $\kappa_k(x_{1:k}, \hat{x}_{1:k})$ such that

$$e_k^\top \kappa_k(x_{1:k}, \hat{x}_{1:k}) \geq c_k \|e_k\|^2 \tag{5.17}$$

for some $c_k > 0$. Let α_k be such that substituting $x_k = \alpha_{k-1}$ gives $\dot{e}_k = -g_{k-1}(x_{1:k-1})^\top e_{k-1} - \kappa_k(x_{1:k}, \hat{x}_{1:k})$, i.e.,

$$\alpha_k(x_{1:k}, \hat{x}_{1:k+1}) = g_k(x_{1:k})^{-1}(-g_{k-1}(x_{1:k-1})^\top e_{k-1} - \kappa_k(x_{1:k}, \hat{x}_{1:k}) - f_k(x_{1:k}) + \dot{\alpha}_{k-1}(x_{1:k}, \hat{x}_{1:k+1})) \tag{5.18}$$

$$\Rightarrow \dot{e}_k = -g_{k-1}(x_{1:k-1})^\top e_{k-1} - \kappa_k(x_{1:k}, \hat{x}_{1:k}) + g_k(x_{1:k})(x_{k+1} - \alpha_k(x_{1:k}, \hat{x}_{1:k+1})). \tag{5.19}$$

Step n: By induction, we have

$$\dot{e}_{n-1} = -g_{n-2}(x_{1:n-2})^\top e_{n-2} - \kappa_{n-1}(x_{1:n-1}, \hat{x}_{1:n-1}) + g_{n-1}(x_{1:n-1})(x_n - \alpha_{n-1}(x_{1:n-1}, \hat{x})). \tag{5.20}$$

Defining $e_n = x_n - \alpha_{n-1}(x_{1:n-1}, \hat{x})$, we have

$$\dot{e}_{n-1} = -g_{n-2}(x_{1:n-2})^\top e_{n-2} - \kappa_{n-1}(x_{1:n-1}, \hat{x}_{1:n-1}) + g_{n-1}(x_{1:n-1})e_n \tag{5.21}$$

$$\dot{e}_n = f_n(x) + g_n(x)u + w - \dot{\alpha}_{n-1}(x, \hat{x}, \hat{u}). \tag{5.22}$$

Select $\kappa_n(x, \hat{x})$ such that

$$e_n^\top \kappa_n(x, \hat{x}) \geq c_n \|e_n\|^2 \tag{5.23}$$

for some $c_n > 0$. Select $u = \alpha_n(x, \hat{x}, \hat{u})$ such that

$$\dot{e}_n = -g_{n-1}(x_{1:n-1})^\top e_{n-1} - \kappa_n(x, \hat{x}) + w \tag{5.24}$$

$$\Rightarrow u = \alpha_n(x, \hat{x}, \hat{u}) = g_n(x)^{-1}(-g_{n-1}(x_{1:n-1})^\top e_{n-1} - \kappa_n(x, \hat{x}) - f_n(x) + \dot{\alpha}_{n-1}(x, \hat{x}, \hat{u})) \tag{5.25}$$

Finally, the closed loop dynamics are

$$\dot{e} = \begin{bmatrix} -c_1 I & g_1 & 0 & \cdots & & \\ -g_1^\top & -c_2 I & g_2 & 0 & \cdots & \\ & & & & & \\ & \cdots & 0 & -g_{k-1}^\top & -c_{k-1} I & g_k \\ & & \cdots & 0 & -g_{k-1}^\top & -c_k I \end{bmatrix} e + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix} w. \tag{5.26}$$

Lyapunov function: Define the Lyapunov function

$$V(e) = \frac{1}{2} e_1^\top e_1 + \frac{1}{2} e_2^\top e_2 + \cdots + \frac{1}{2} e_n^\top e_n. \tag{5.27}$$

Then cross-terms cancel and we have

$$\begin{aligned} \dot{V}(e, w) &= e_1^\top \dot{e}_1 + e_2^\top \dot{e}_2 + \cdots + e_n^\top \dot{e}_n \\ &= e_1^\top (-\kappa_1(x_1, \hat{x}_1) + g_1(x_1) e_2) \\ &\quad + e_2^\top (-g_1(x_1)^\top e_1 - \kappa_2(x_{1:2}, \hat{x}_{1:2}) g_2(x_{1:2}) e_3) \\ &\quad + \cdots + e_n^\top (-g_{n-1}(x_{1:n-1})^\top e_{n-1} - \kappa_n(x, \hat{x}) + w) \\ &= e_n^\top w - \sum_{k=1}^{n} e_k^\top \kappa_k(x_{1:k}, \hat{x}_{1:k}) \qquad (5.28) \\ &\leq e_n^\top w - \sum_{k=1}^{n} c_k \|e_k\|^2 =: \dot{V}_{\max}(e, w) \qquad (5.29) \end{aligned}$$

Level set: We want to find the smallest level set of V that is invariant.

$$\begin{aligned} \gamma^* &:= \min \gamma \qquad (5.30) \\ &\quad \text{s.t. } (V(e) = \gamma,\ w \in \mathcal{W}) \ \Rightarrow\ \dot{V}(e, w) < 0 \\ &\leq \min \gamma \qquad (5.31) \\ &\quad \text{s.t. } (V(e) = \gamma,\ w \in \mathcal{W}) \ \Rightarrow\ \dot{V}_{\max}(e, w) < 0 \\ &= \min \gamma \qquad (5.32) \\ &\quad \text{s.t. } (V(e) \geq \gamma,\ w \in \mathcal{W}) \ \Rightarrow\ \dot{V}_{\max}(e, w) < 0 \end{aligned}$$

We can prove the final equality by contrapositive, and we can solve optimization (5.32) using the S-lemma [52]. Observe that we can write $\dot{V}_{\max}(e, w) = -e^\top C e + e_n^\top w$.

Example 3. *Suppose $\mathcal{W}$ is an ellipsoid, i.e., $\mathcal{W} = \{w \in \mathcal{R}^m : w^\top R w \leq 1\}$ for some positive semidefinite R. Then adding a small constant $\epsilon > 0$ to make the inequalities non-strict and*

applying the S-lemma to (5.32) yields

$$\begin{aligned}
&\min\ \gamma && (5.33)\\
&\quad s.t.\ \left(\tfrac{1}{2}e^\top e \geq \gamma,\ w^\top R w \leq 1\right)\ \Rightarrow\ -e^\top C e + e_n^\top w + \epsilon\ \leq\ 0\\
&\leq \min_{\alpha,\beta,\gamma}\ \gamma && (5.34)\\
&\quad s.t.\ \alpha \geq 0, \beta \geq 0\\
&\qquad \alpha\left(\tfrac{1}{2}e^\top e - \gamma\right) + \beta(1 - w^\top R w) - e^\top C e + e_n^\top w + \epsilon \leq 0\\
&= \min_{\alpha,\beta,\gamma}\ \gamma && (5.35)\\
&\quad s.t.\ \alpha \geq 0, \beta \geq 0\\
&\qquad \begin{bmatrix} \frac{\alpha}{2}I - C & \begin{bmatrix} 0 \\ \frac{1}{2}I \end{bmatrix} & 0 \\ \begin{bmatrix} 0 & \frac{1}{2}I \end{bmatrix} & -\beta R & 0 \\ 0 & 0 & \beta - \alpha\gamma + \epsilon \end{bmatrix} \leq 0
\end{aligned}$$

For a fixed γ, this optimization is a semidefinite program (SDP) in α and β. Hence, we can bisect over γ to find the smallest value such that the SDP is feasible.

5.4 LgV Backstepping Procedure

Observe that in the closed loop dynamics (5.39), $\dot{e}_k$ depends on e_{k-1}. It can be useful to eliminate such cross-terms to obtain cascaded error dynamics: the evolution of e_k depends only on e_j for $j > k$. In practice, this can make the controller easier to tune. We now present an "LgV" backstepping procedure that eliminates such cross-terms. This concept comes from [81], where the name "LgV" refers to Lie derivative terms in $\dot{V}$ that are strategically dominated in this approach.

Step 1 proceeds identically as in Section 5.3, where $\bar{e}_1 = e_1$ and $\bar{\alpha}_1 = \alpha_1$. In Step 2, we again let $\bar{e}_2 = \bar{e}_1 - \bar{\alpha}_1$, and we now define

$$\bar{\alpha}_2(x_{1:2}, \hat{x}_{1:3}) = g_2(x_{1:2})^{-1}(-\kappa_2(x_{1:2}, \hat{x}_{1:2}) - f_2(x_{1:2}) + \dot{\alpha}_1(x_{1:2}, \hat{x}_{1:3}))$$

which yields

$$\dot{\bar{e}}_2 = -\kappa_2(x_{1:2}, \hat{x}_{1:2}) + g_2(x_{1:2})(x_3 - \bar{\alpha}_2). \tag{5.36}$$

Note that this is missing the term $-g_1(x_1)^\top e_1$ compared to (5.13). In this section, we will obtain cascaded error dynamics where $\dot{\bar{e}}_k$ does not depend on $\bar{e}_j$ for $j < k$. We continue in this manner so that

$$\dot{\bar{e}}_k = -\kappa_k(x_{1:k}, \hat{x}_{1:k}) + g_k(x_{1:k})\bar{e}_{k+1}, \tag{5.37}$$

without the $-g_{k-1}(x_{1:k-1})^\top e_{k-1}$ cross-term present in (5.19). Furthermore, we now make the assumption that $e_k^\top \kappa_k(x_{1:k}, \hat{x}_{1:k}) \geq c_k(x_{1:k})\|e_k\|^2$, where c_k can depend on $x_{1:k}$ unlike in Section 5.3. Finally, we have

$$\bar{u} = g_n(x)^{-1}(-\kappa_n(x, \hat{x}) - f_n(x) + \dot{\hat{\alpha}}_{n-1}(x, \hat{x}, \hat{u})), \tag{5.38}$$

which yields $\dot{e}_n = -\kappa_n(x, \hat{x}) + w$. Finally, the closed loop error dynamics become

$$\dot{e} = \begin{bmatrix} -c_1 I & g_1 & 0 & \cdots & & \\ 0 & -c_2 I & g_2 & 0 & \cdots & \\ & & & & & \\ & & \cdots & 0 & -c_{k-1}I & g_k \\ & & & \cdots & 0 & -c_k I \end{bmatrix} e + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix} w. \tag{5.39}$$

In Section 5.3, we included the $-g_k(x_{1:k})^\top e_k$ in each equation of the error dynamics so that with a Lyapunov function of $V = \frac{1}{2}e^\top e$, these terms would directly cancel the opposite cross-terms that arose from the $g_k e_{k+1}$ terms in the dynamics of e_k. In the LgV backstepping method, instead of directly cancelling these cross-terms, they are dominated by other terms using Young's inequality, $xy \leq lx^2 + \frac{1}{4l}y^2$ for $l > 0$. Let $V(\bar{e}) = \frac{1}{2}\bar{e}^\top \bar{e}$. Then

$$\begin{aligned}
\dot{V}(\bar{e}, w) &= \sum_{k=1}^{n} \bar{e}_k^\top \dot{\bar{e}}_k \\
&= \bar{e}_n^\top(-\kappa_n(x, \hat{x}) + w) + \sum_{k=1}^{n-1} \bar{e}_k^\top(-\kappa_k(x_{1:k}, \hat{x}_{1:k}) + g_k(x_{1:k})\bar{e}_{k+1}) \\
&\leq -c_n \bar{e}_n^\top \bar{e}_n + \bar{e}_n^\top w + \sum_{k=1}^{n-1}(-c_k \bar{e}_k^\top \bar{e}_k + l_k \bar{e}_k^\top \bar{e}_k + \frac{1}{4l_k}\bar{e}_{k+1}^\top g_k(x_{1:k})^\top g_k(x_{1:k})\bar{e}_{k+1}) \\
&= -c_1 \bar{e}_1^\top \bar{e}_1 + \bar{e}_n^\top w + \sum_{k=2}^{n} \bar{e}_k^\top(-(c_k - l_k)I + \tfrac{1}{4l_{k-1}} g_{k-1}(x_{1:k-1})^\top g_{k-1}(x_{1:k-1}))\bar{e}_k,
\end{aligned}$$

where $l_n = 0$. In order to ensure that $\dot{V}(\bar{e}, w) < 0$ for all $\bar{e} \neq 0$, let

$$-(c_k(x_{1:k}) - l_k)I + \tfrac{1}{4l_{k-1}} g_{k-1}(x_{1:k-1})^\top g_{k-1}(x_{1:k-1}) \prec -\bar{c}_k\|\bar{e}\|^2 \tag{5.40}$$

for some $\bar{c}_k > 0$, which is possible for sufficiently large $c_k(x_{1:k})$. Then we have

$$\dot{V}(\bar{e}, w) \leq \bar{e}_n^\top w - \sum_{k=1}^{n} \bar{c}_k\|\bar{e}_k\|^2 \tag{5.41}$$

which is of the same form as (5.29), and so we can find an invariant sublevel set of V in the same manner as in Section 5.3.

Using these backstepping approaches, with or without the $-g_k(x_{1:k})^\top e_k$ term in the error dynamics, we can avoid using SOS programming to search for a storage function. One drawback of the backstepping method, however, is that it becomes difficult to bound the tracking input $u \in \mathcal{U}$ without simply bounding each term and using the triangle inequality, which may lead to a quite conservative bound.

5.5 Example

We now demonstrate the backstepping method from Section 5.3 on an autonomous ship example, adapted from [42]. For the tracker model, we use the following ship model from [82]:

$$\dot{\eta} = R(\psi)\nu, \tag{5.42}$$

$$M\dot{\nu} = -C(\nu)\nu - D\nu + \tau + w, \tag{5.43}$$

where $\eta := [N; E; \psi]$ is the North position, East position, and heading angle, respectively. The velocity vector $\nu = [u; v; r]$ consists of the surge and sway velocities as well as the yaw rate, and τ is the control input. There are state constraints $[\eta; \nu] \in \mathcal{X} \subseteq \mathbb{R}^6$ that will be described for the particular scenario of interest. $R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is a rotation matrix, and $w \in \mathcal{W} = [-\bar{w}, \bar{w}]$ is a bounded disturbance due to wind, where $\bar{w} = [0.014\ 0.014\ 0.05]^\top$. The inertia matrix is $M = \begin{bmatrix} 87.4 & 0 & 0 \\ 0 & 98.3 & 2.48 \\ 0 & 2.48 & 22.2 \end{bmatrix}$, the damping matrix is $D = \begin{bmatrix} 6.58 & 0 & 0 \\ 0 & 37.7 & 2.66 \\ 0 & 2.66 & 19.3 \end{bmatrix}$, and the Coriolis matrix is $C(\nu) = u \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 98.3 \\ 0 & 0 & 2.48 \end{bmatrix}$. The planner model neglects the wind disturbance:

$$\dot{\hat{\eta}} = R(\hat{\psi})\hat{\nu}, \tag{5.44}$$

$$M\dot{\hat{\nu}} = -C(\hat{\nu})\hat{\nu} - D\hat{\nu} + \hat{\tau}. \tag{5.45}$$

There are planner input constraints $\hat{\tau} \in \hat{\mathcal{U}}$ and state constraints $[\hat{\eta}; \hat{\nu}] \in \hat{\mathcal{X}}$. $\hat{\mathcal{X}}$ will be restricted as in Chapters 2 and 3 to ensure that the tracker constraints $[\eta; \nu] \in \mathcal{X}$ are satisfied. Now we can follow the backstepping procedure from Section 5.3 to design a controller for this planner-tracker system.

Step 1: Define $\eta - \hat{\eta}$. Then we have

$$\dot{e}_1 = R(\psi)\nu - R(\hat{\psi})\hat{\nu}. \tag{5.46}$$

Let α_1 be such that substituting $\nu = \alpha_1$ gives $\dot{e}_1 = -K_1 e_1$, i.e.,

$$\alpha_1 = R(\psi)^\top (R(\hat{\psi})\hat{\nu} - K_1 e_1). \tag{5.47}$$

Accounting for the difference between x_2 and α_1, we have

$$\dot{e}_1 = -K_1 e_1 + R(\psi)(\nu - \alpha_1). \tag{5.48}$$

Step 2: Defining $e_2 := \nu - \alpha_1$, we have

$$\dot{e}_1 = -K_1 e_1 + R(\psi) e_2. \tag{5.49}$$

Differentiating to obtain the dynamics for e_2 yields

$$\begin{aligned} M\dot{e}_2 &= M\dot{\nu} - M\dot{\alpha}_1 \\ &= -C(\nu)\nu - D\nu + \tau + w - M\dot{\alpha}_1. \end{aligned} \tag{5.50}$$

We can select $\tau = -R(\psi)^\top e_1 - K_2 e_2 + C(\nu)\nu + D\nu + M\dot{\alpha}_1$ so that

$$M\dot{e}_2 = -R(\psi)^\top e_1 - K_2 e_2 + w. \tag{5.51}$$

To compute $\dot{\alpha}_1$, observe that $\frac{d}{dt}R(\psi(t)) = r(t)R(\psi(t))S$, where $S = \left[\begin{smallmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right]$ is skew-symmetric. Then

$$\begin{aligned} \dot{\alpha}_1 = -rSR(\psi)^\top(R(\hat{\psi})\hat{\nu} - K_1 e_1) + R(\psi)^\top(\hat{r}R(\hat{\psi})S\hat{\nu} + R(\hat{\psi})(M^{-1}(-C(\hat{\nu})\hat{\nu} - D\hat{\nu} + \hat{\tau}))) \\ + R(\psi)^\top K_1(-K_1 e_1 + R(\psi)e_2). \end{aligned} \tag{5.52}$$

Define the Lyapunov function $V(e) = \frac{1}{2}e_1^\top e_1 + \frac{1}{2}e_2^\top M e_2$. Differentiating, we have

$$\begin{aligned} \dot{V}(e,w) &= e_1^\top \dot{e}_1 + e_2^\top M\dot{e}_2 = e_1^\top(-K_1 e_1 + R(\psi)e_2) + e_2^\top(-R(\psi)^\top e_1 - K_2 e_2 + R(\psi)^\top w) \\ &= -e_1^\top K_1 e_1 - e_2^\top K_2 e_2 + e_2^\top w \end{aligned} \tag{5.53}$$

Error Bound

Selecting $K_1 = K_2 = 2I_3$, we wish to find the smallest sublevel set of V, denoted $\Omega(V,\gamma)$ for some γ, that is invariant for the closed loop error dynamics. We formulate an optimization problem to find the smallest value of γ that satisfies the invariance condition Theorem 1 from Chapter 2: $\dot{V}(e,w) < 0$ whenever $V(e) = \gamma$, and $w \in \mathcal{W}$:

$$\begin{aligned} &\min_{\gamma} \ \gamma \\ &\text{s.t. } V(e) = \gamma \ \Rightarrow \ \dot{V}(e,w) < 0 \ \forall \ w \in \mathcal{W} \end{aligned} \tag{5.54}$$

The disturbance set $\mathcal{W} = [-\bar{w}, \bar{w}]$ is a hyperrectangle in $\mathbb{R}^3$ with 6 vertices. Let these vertices be denoted w_j, $j = 1, \ldots, 6$. Because $\dot{V}(e,w)$ is linear in w, $\dot{V}(e,w) \leq 0$ will hold for all $w \in \mathcal{W}$ if and only if it holds at the vertices, i.e., $\dot{V}(e,w_j) \leq 0$, $j = 1, \ldots, 6$. Hence, the following optimization is equivalent:

$$\begin{aligned} &\min_{\gamma} \ \gamma \\ &\text{s.t. } V(e) = \gamma \ \Rightarrow \ \dot{V}(e,w_j) < 0, \ j = 1, \ldots, 6 \end{aligned} \tag{5.55}$$

We can apply the S-procedure to relax each of the 6 constraints above by introducing multipliers $s_j(e)$. Note that because these multipliers correspond to the *equality* constraint $V(e) = \gamma$, they need not be SOS. Furthermore, we add a small postive constant $\epsilon > 0$ to account for the strict inequality in (5.55). Then the following SOS program is a relaxation of the original optimization.

$$\begin{aligned} &\min_{\gamma, s_j} \ \gamma \\ &\text{s.t. } -\dot{V}(e, w_j) + s_j(e)(V(e) - \gamma) + \epsilon \in \Sigma[e], \ j = 1, \ldots, 6 \end{aligned} \tag{5.56}$$

Optimization (5.56) is bilinear in decision variables s_j and γ and can be solved by bisecting over γ to find the smallest value such that the optimization is feasible. We solve the optimization in this manner using SOSOPT [56] in MATLAB, which yields $\gamma = 0.0188$. Then we can use invariant set $\mathcal{O} = \Omega(V, \gamma)$ as the error bound.

Scenario

In this example, the goal of the ship is to steer around two obstacles and reach a target region which represents docking of the ship, as shown in Figure 5.1. The ship 2.578m long and 0.22m wide and is covered by 6 intersecting circles, and these bounding circles should never intersect with the obstacles (solid blue circles).

The state constraint $\mathcal{X}$ is simply a rectangular constraint (dashed blue rectangle) in the position space that excludes the obstacles (solid blue circles). The error bound set $\mathcal{O}$ is used to make the obstacles and target region more conservative, so that $\hat{x} \in \hat{\mathcal{X}}$ and $e \in \mathcal{O}$ imply $x \in \mathcal{X}$ as in Chapters 2 and 3.

MPC Planner

We use an MPC planner with a prediction horizon of M steps. The cost function is

$$J = \sum_{k=0}^{M-1} \left(\hat{\tau}_k^\top R \hat{\tau}_k - W_{\hat{N}} \exp\{\hat{N}_k\} \right) + W_{\text{term}} \max\{\hat{N}_{\text{target}} - \hat{N}_M, |\psi_M - \tfrac{\pi}{2}|\}. \tag{5.57}$$

The stage cost penalizes a large control input and a small North position, since the goal is for the ship to reach the target region at the maximum North position. The terminal cost penalizes whichever term is larger between (1) the distance from the terminal North position to the target North position, and (2) the difference between the terminal heading angle and the desired value of $\frac{\pi}{2}$. During the simulation, the first terminal cost term starts larger at the beginning, encouraging the ship to travel North. Towards the end of the simulation, the second terminal cost term becomes larger, encouraging the ship to turn and orient itself with the desired heading of $\frac{\pi}{2}$.

The MPC constraints include an initial condition constraint, discretized planner dynamics (using the midpoint method), scenario boundary constraints (dashed black rectangle in

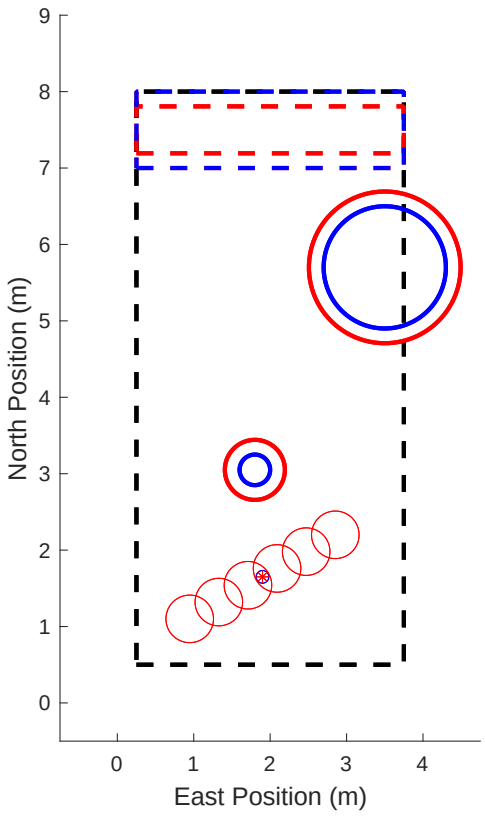


Figure 5.1: Simulation setup. The ship, which is covered by 6 intersecting circles, aims to avoid two obstacles and reach a target region. The true obstacles and target region are shown in blue dashed and solid lines, respectively. The planner aims to avoid the bloated obstacles (solid red circles) and reach the restricted target region (dashed red rectangle). The black dashed rectangle represents the scenario boundary.

Figure 5.1), and obstacle avoidance constraints. There is a circular obstacle avoidance constraint for each pair between the 2 obstacles and 6 bounding circles that cover the ship, i.e., 12 obstacle avoidance constraints in total.

Results

Using this MPC planner and the tracking controller described above, we simulate the ship performing this maneuver in MATLAB. Snapshots from the simulation are shown in Figure 5.2. The planner ship is covered by 6 intersecting red circles; the tracker ship is covered by 6 intersecting blue circles. The planner and tracker ship centers are shown by a small red star and a small blue circle, respectively. The planner ship successfully avoids the bloated obstacles and reaches the shrunk target region, and the tracker ship successfully avoids the true obstacle and reaches the true goal region.

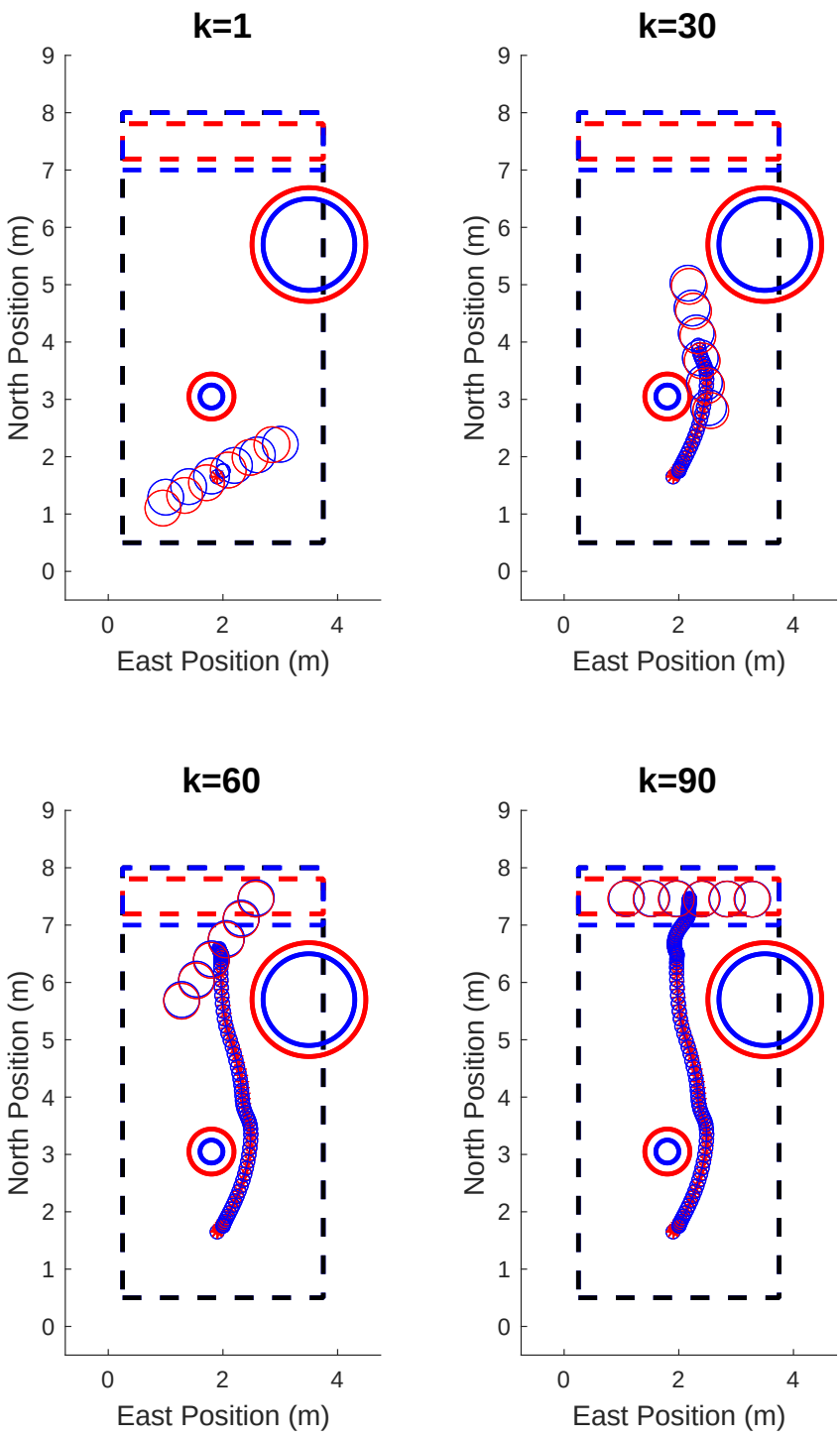


Figure 5.2: Simulation snapshots of ship avoiding obstacles and reaching the target region. The planner ship is covered by 6 intersecting red circles; the tracker ship is covered by 6 intersecting blue circles. The planner and tracker ship centers are shown by a small red star and a small blue circle, respectively.

Chapter 6

Further Robustness Analysis in Tracking

6.1 Abstract

A method is presented for computing the worst-case disturbance of a given norm for a finite-horizon linear time-varying system with a nonzero initial condition. This method is motivated by the linearized robustness analysis of a nonlinear system about a finite-horizon nominal trajectory. The system is linearized about this trajectory and interconnected with sampled model uncertainties. An additional input term to improve the fidelity of the linear interconnection is introduced and is absorbed into an augmented system with a nonzero initial condition. The method, which analyzes the robustness of the resulting interconnection to disturbance, is demonstrated on the numerical example of a two-link robot arm.

6.2 Introduction

Many engineering problems involve nonlinear systems following finite-horizon trajectories, such as space launch systems, robotic manipulators, etc. It is critical to understand the robustness of these systems to disturbances and model uncertainties. The dynamics are often linearized about a nominal trajectory to obtain a finite-horizon linear time-varying (LTV) system. Robustness can then be studied by sampling from a family of model uncertainties; however, each sampled model uncertainty changes the nominal trajectory.

To account for the shift in the nominal trajectory, we include an additional input term in the interconnection of the linearized system with the sampled model uncertainty. We then absorb the additional input term into an augmented system with a *nonzero* initial condition. The analysis tasks are to bound the final output along the trajectory in the presence of external disturbances and to compute the worst-case disturbance that achieves this bound.

The main technical contribution of this Chapter is a numerically efficient method for computing the worst-case disturbance of a given norm for a finite-horizon LTV system with

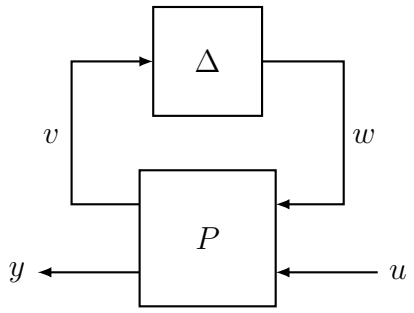


Figure 6.1: Interconnection of P and Δ.

a nonzero initial condition (Section 6.4). The performance measure of interest is a bound on the output at the final time. This bound serves as a measure of robustness and allows us to overapproximate reachable sets for the LTV system. The disturbance can be used to simulate the original nonlinear system to assess the accuracy of the linearized approximation; it can also be compared to available disturbance profiles to gauge the likelihood of a worst-case scenario.

This method is related to the paper [83] that constructs worst-case disturbances for a more general cost function but subject to zero initial condition. Another related work, [84], analyzes the robustness of uncertain nonlinear systems using a power iteration but does not provide any convergence guarantees. An algorithm is presented in [85] to compute an upper bound on the worst-case gain of an uncertain LTV system using integral quadratic constraints. Computation of a lower bound involves randomly sampling uncertainties and constructing corresponding worst-case disturbances using tools from [86]. Both [85] and [86] assume zero initial conditions. Other studies on bounding trajectories of LTV systems include [87], which studies disturbances but not model uncertainty.

In Section 6.5, we study a two-link robot arm example from [85]. The model is linearized about a nominal trajectory, and we analyze the robustness of the resulting LTV system by determining the effect of disturbance on interconnections of the system with sampled model uncertainties. We demonstrate that the modified interconnection with a sampled model uncertainty matches the nonlinear system far more closely than a naive linearized interconnection.

Notation: $\mathbb{R}^{n\times m}$ denotes the set of n-by-m real matrices, and $\mathbb{S}^n$ denotes the set of n-by-n real, symmetric matrices. For $u, y \in \mathbb{R}^n$, the Euclidean inner product $\langle u, y\rangle_2 := u^T y$ defines the norm $\|u\|_2 := \sqrt{\langle u, u\rangle_2}$. $\mathcal{L}_2^n[0,T]$ denotes the Hilbert space of Lebesgue integrable signals $f : [0,T] \to \mathbb{R}^n$ with inner product $\langle f, g\rangle_{2,[0,T]} := \int_0^T f(t)^T g(t)\, dt$. The inner product defines a norm $\|f\|_{2,[0,T]} := \sqrt{\langle f, f\rangle_{2,[0,T]}}$. If $\|f\|_{2,[0,T]} < \infty$ then $f \in \mathcal{L}_2^n[0,T]$.

6.3 Problem Formulation

Consider an uncertain time-varying system defined on the horizon $[0, T]$ as in Figure 6.1. The nominal part P is nonlinear and/or time-varying; the uncertainty Δ is an LTV system that is unknown but assumed to be in a known set $\mathbf{\Delta}$. We denote the mapping from u to y in Figure 6.1 as $F_U(P, \Delta)$. It is assumed that this interconnection is well-posed [88]. The response due to a nominal input $\bar{u}$ and $\Delta = 0$ corresponds to $(\bar{w} = 0, \bar{v}, \bar{y})$. The output y will deviate from the nominal trajectory $\bar{y}$ due to the uncertainty Δ and/or deviations of the input u from its nominal value $\bar{u}$. The goal is to estimate the maximum deviation (in the Euclidean norm) of the output at the final time, i.e.

$$\max_{\substack{\Delta \in \mathbf{\Delta} \\ \|u-\bar{u}\|_{2,[0,T]} \leq \alpha}} \|y(T) - \bar{y}(T)\|_2 \tag{6.1}$$
$$\text{s.t. } y = F_U(P, \Delta)\, u,\ \bar{y} = F_U(P, 0)\, \bar{u}$$

Let $\bar{P}$ denote the LTV system obtained by linearizing P along the nominal trajectory $(\bar{w} = 0, \bar{u}, \bar{v}, \bar{y})$. Define the deviations as $\delta w := w - \bar{w}, \delta u := u - \bar{u}, \delta v := v - \bar{v}, \delta y := y - \bar{y}$, to approximate the uncertain system in Figure 6.1 as:

$$\begin{bmatrix} \delta v \\ \delta y \end{bmatrix} = \bar{P} \begin{bmatrix} \delta w \\ \delta u \end{bmatrix} \tag{6.2}$$
$$\delta w = \Delta(\bar{v} + \delta v)$$

The linearization step is accurate if both δu and δw are "small", i.e., higher order terms are small compared to linear terms in the Taylor series expansion. The assumption that δw is small implies that the uncertainty has small deviation from the nominal value $\Delta = 0$. The relation $w = \Delta(v)$ has been replaced by $\delta w = \Delta(\bar{v} + \delta v)$, which is exact because Δ is LTV and $\bar{w} = 0$. It is important to note the additional forcing term $\bar{v}$, which accounts for a change in the nominal trajectory due to the uncertainty. The linearizations in typical robustness analysis approaches (such as [85]) do not include this additional term and thus are less accurate in general. We will show numerically in Section 6.5 that this additional forcing term yields a better approximation for the original uncertain nonlinear dynamics (as compared to simply setting $\bar{v} = 0$). This linearization step yields the (approximate) uncertain LTV system shown in Figure 6.2. We denote the mapping from δu to δy as $F_U(\bar{P}, \Delta, \bar{v})$.

The uncertainty poses a difficulty in solving Eq. 6.1, which we address through sampling. Specifically, we replace the maximization over all $\Delta \in \mathbf{\Delta}$ with a maximization over a finite collection of samples $\{\Delta_i\}_{i=1}^N$. (If $\mathbf{\Delta}$ is parameterizable, e.g., has a bounded number of states, and if $\{\Delta_i\}_{i=1}^N$ are i.i.d. samples from a probability distribution over $\mathbf{\Delta}$, then Theorem 8.1 of [89] can be used to compute N to make probabilistic guarantees on the robustness level.)

This leads to the following (approximate) worst-case analysis problem:

$$\max_{\substack{\Delta_i \in \{\Delta_i\}_{i=1}^N \\ \|\delta u\|_{2,[0,T]} \leq \alpha}} \|\delta y_i(T)\|_2 \tag{6.3}$$
$$\text{s.t. } \delta y_i = F_U(\bar{P}, \Delta_i, \bar{v})\, \delta u$$

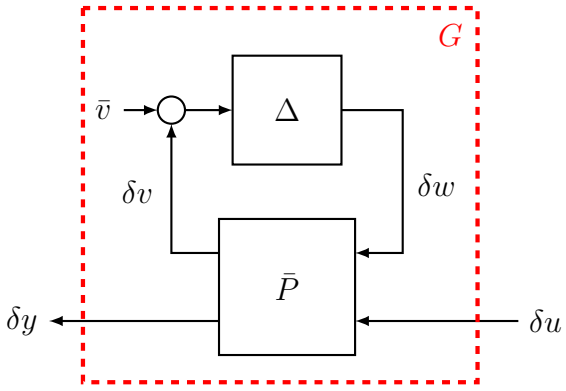


Figure 6.2: Linearized interconnection with affine term $\bar{v}$.

Eq. 6.3 involves a maximization over the disturbance and N uncertainty samples. A key subproblem is to select a single uncertainty $\Delta \in \{\Delta_i\}_{i=1}^N$ and maximize over the disturbance δu. For this sample Δ, the interconnection in Figure 6.2 can be represented as an LTV system with inputs $(\delta u, \bar{v})$ and output δy, where $\bar{v}$ is known and fixed. This can be rewritten as the LTV system G in Figure 6.2, where δu is the only input, by absorbing $\bar{v}$ into the state matrices and introducing a nonzero initial condition (see Appendix 6.6).

For simplicity, we denote the perturbation to the input as $d := \delta u$ and the resulting change in output as $e := \delta y$. This yields the following LTV system G defined on the finite horizon $[0, T]$:

$$\begin{aligned} \dot{x}(t) &= A(t)\, x(t) + B(t)\, d(t) \\ e(t) &= C(t)\, x(t), \end{aligned} \tag{6.4}$$

where $x(0) = x_0$, $x(t) \in \mathbb{R}^{n_x}$, $d(t) \in \mathbb{R}^{n_d}$, and $e(t) \in \mathbb{R}^{n_e}$. It is assumed that the interconnection in Figure 6.2 has zero feedthrough from δu to δy. This occurs if and only if $\bar{P}$ has zero feedthrough from δu to δy and one of the following: zero feedthrough from δu to δv or from δw to δy. The main technical task is to solve:

$$\begin{aligned} R_\alpha = \max_{\|d\|_{2,[0,T]} \le \alpha} & \|e(T)\|_2^2 \\ \text{s.t. Eq. (6.4) with } & x(0) = x_0. \end{aligned} \tag{6.5}$$

In addition, we would like to compute a worst-case disturbance d^* that achieves the maximal cost. Note that in the case where $x(0) = 0$, the map from d to $e(T)$ is linear, and so without loss of generality, we can solve Eq. 6.5 with $\alpha = 1$. However, when $x(0) \neq 0$, the shape of the worst-case disturbance depends on the value of α.

6.4 Worst-Case Disturbance

This section presents a series of technical lemmas that lead to a solution to Eq. 6.5, which is the main technical result of this Chapter. Let $\Phi(t, \tau)$ denote the state transition matrix

of the unforced system $\dot{x}(t) = A(t)\,x(t)$. For all $t, \tau \in [0, T]$, $\Phi(t, \tau)$ satisfies the following differential equation:

$$\frac{d}{dt}\Phi(t,\tau) = A(t)\,\Phi(t,\tau), \quad \Phi(\tau,\tau) = I$$

The output of G at the final time can be expressed as:

$$e(T) = L(d) + c, \tag{6.6}$$

where $c = C(T)\Phi(T, 0)x_0 \in \mathbb{R}^{n_e}$ and the linear map $L : \mathcal{L}_2^{n_d}[0, T] \to \mathbb{R}^{n_e}$ is defined by

$$L(d) = C(T) \int_0^T \Phi(T,\tau)B(\tau)d(\tau)d\tau$$

Thus, Eq. 6.5 can be rewritten as:

$$\begin{aligned} R_\alpha = \max_{d \in \mathcal{L}_2^{n_d}[0,T]} \; & \|L(d) + c\|_2^2 \\ \text{s.t. } & \|d\|_{2,[0,T]} \leq \alpha \end{aligned} \tag{6.7}$$

The following lemma, proven in Appendix 6.6, shows that the inequality constraint in Eq. 6.7 can be replaced with an equality constraint without loss of generality.

Lemma 2. *Assume there exists d such $L(d) \neq 0$. If d^* is optimal for Eq. 6.7, then $\|d^*\|_{2,[0,T]} = \alpha$.*

Thus, Eq. 6.7 can be rewritten as:

$$\begin{aligned} R_\alpha = \max_{d \in \mathcal{L}_2^{n_d}[0,T]} \; & \|L(d) + c\|_2^2 \\ \text{s.t. } & \|d\|_{2,[0,T]} = \alpha \end{aligned} \tag{6.8}$$

The next lemma, proven in Appendix 6.6, provides a sufficient condition for solving Eq. 6.8 using a Lagrange multiplier μ. This transforms the constrained optimization problem into an unconstrained one.

Lemma 3. *Consider the following optimization with $\mu \in \mathbb{R}$:*

$$\max_{d \in \mathcal{L}_2^{n_d}[0,T]} \; \|L(d) + c\|_2^2 - \mu\|d\|_{2,[0,T]}^2 \tag{6.9}$$

If d_μ^* is optimal for Eq. 6.9 with $\|d_\mu^*\|_{2,[0,T]} = \alpha$, then d_μ^* is also optimal for Eq. 6.8.

Now, the problem is to find $\mu \in \mathbb{R}$ and a maximizer d_μ^* of Eq. 6.9 such that $\|d_\mu^*\|_{2,[0,T]} = \alpha$. The remainder of this section builds towards proving that such a μ always exists. This requires the assumption that G is output-controllable at the final time T. To make this

precise, let $X : [0, T] \to \mathbb{S}^{n_x}$ denote the state controllability Gramian [90, 91] satisfying the following matrix Lyapunov differential equation:

$$\dot{X}(t) = A(t)X(t) + X(t)A(t)^T + B(t)B(t)^T, \quad X(0) = 0$$

The output controllability Gramian is defined as:

$$W(\tau) := C(\tau)X(\tau)C(\tau)^T, \quad \forall \tau \in [0, T] \tag{6.10}$$

Output controllability of G at time T means $W(T) \succ 0$. Since $W(T) \in \mathbb{S}^{n_e}$, we can perform an eigenvalue decomposition as $W(T) = Q\Lambda Q^T$, with Q unitary. Assume the eigenvalues of $W(T)$ are sorted in descending order, and let r be the multiplicity of the maximum eigenvalue. That is, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_{n_e})$ with $\lambda_1 = \cdots = \lambda_r > \lambda_{r+1} \geq \cdots \geq \lambda_{n_e} > 0$.

The Gramian $W(T)$ is related to the adjoint $L^\sim : \mathbb{R}^{n_e} \to \mathcal{L}_2^{n_d}[0, T]$, which has the following state-space realization:

$$\begin{aligned} &\dot{p}(t) = -A(t)^T p(t), \quad p(T) = C(T)^T \xi \\ &L^\sim(\xi)(t) = B(t)^T p(t), \end{aligned} \tag{6.11}$$

where $p(t) \in \mathbb{R}^{n_x}$ is the adjoint state. The operator $L \circ L^\sim : \mathbb{R}^{n_e} \to \mathbb{R}^{n_e}$ is equivalent to matrix multiplication by $W(T)$ (Theorem 3 of [83]).

Note that the optimization problem in Eq. 6.9 is over the infinite-dimensional space $\mathcal{L}_2^{n_d}[0, T]$. In the case where $\mu > 0$, the following lemma makes use of the eigenvalue decomposition of $W(T)$, as well as the adjoint map $L^\sim$, to reformulate Eq. 6.9 as a finite-dimensional maximization over $\mathbb{R}^{n_e}$. It is also shown that the optimal disturbance d_μ^* is in the range of $L^\sim$. The proof is provided in Appendix 6.6.

Lemma 4. *Consider the following finite-dimensional optimization problem for $\mu > 0$:*

$$\max_{z \in \mathbb{R}^{n_e}} -z^T \left(\mu\Lambda - \Lambda^2\right) z + 2(\Lambda Q^T c)^T z \tag{6.12}$$

If z_μ^* is optimal for Eq. 6.12, then $d_\mu^* = L^\sim(Qz_\mu^*)$ is optimal for Eq. 6.9.

Now it suffices to find $\mu > 0$ and z_μ^* optimal for Eq. 6.12 such that $d_\mu^* = L^\sim(Qz_\mu^*)$ has norm α. The following lemma makes use of the simple structure of Eq. 6.12 to find such a value of $\mu > 0$ for every $\alpha > 0$.

Lemma 5. *Let G be output-controllable. For all $\alpha > 0$, there exists $\mu \geq \lambda_1$ and z_μ^* optimal for Eq. 6.12 such that $d_\mu^* = L^\sim(Qz_\mu^*)$ satisfies $\|d_\mu^*\|_{2,[0,T]} = \alpha$.*

Proof. First, suppose $\mu > \lambda_1$. Then $(\mu\Lambda - \Lambda^2) \succ 0$, so the maximization in Eq. 6.12 has a unique maximizer

$$z_\mu^* = (\mu I - \Lambda)^{-1}\tilde{c}, \tag{6.13}$$

where $\tilde{c} := Q^T c$. Then $d_\mu^* = L^\sim(Qz_\mu^*)$ satisfies

$$\|d_\mu^*\|_{2,[0,T]}^2 = \langle L(L^\sim(Qz_\mu^*)), Qz_\mu^* \rangle_2$$

Using the relationship between $L \circ L^\sim$ and $W(T)$, as well as the eigenvalue decomposition for $W(T)$, we have:

$$\|d_\mu^*\|_{2,[0,T]}^2 = {z_\mu^*}^T Q^T W(T) Q z_\mu^* = {z_\mu^*}^T \Lambda z_\mu^* \tag{6.14}$$

Next, substitute for z_μ^*. This yields:

$$\begin{aligned} \|d_\mu^*\|_{2,[0,T]}^2 &= \tilde{c}^T \operatorname{diag}\left(\frac{\lambda_1}{(\mu-\lambda_1)^2}, \ldots, \frac{\lambda_{n_e}}{(\mu-\lambda_{n_e})^2} \right) \tilde{c} \\ &= \sum_{i=1}^{n_e} \tilde{c}_i^2 \frac{\lambda_i}{(\mu-\lambda_i)^2} =: f(\mu) \end{aligned} \tag{6.15}$$

We would like to solve the expression $f(\mu) = \alpha^2$ for μ. We consider two cases: (A) $\tilde{c}_i \neq 0$ for some $i = 1, \ldots, r$, and (B) $\tilde{c}_1 = \cdots = \tilde{c}_r = 0$.

Define $\alpha_{\max}^2 := \lim_{\mu \downarrow \lambda_1} f(\mu)$. In Case A, one of the first r terms of f grows unbounded, so $\alpha_{\max}^2 = \infty$. In Case B, the first r terms of f are zero and so $\alpha_{\max}^2 = \sum_{i=r+1}^{n_e} \tilde{c}_i^2 \frac{\lambda_i}{(\lambda_1-\lambda_i)^2} < \infty$.

Furthermore, for both cases, $f : (\lambda_1, \infty) \to \mathbb{R}$ is a continuous, decreasing function with $\lim_{\mu\to\infty} f(\mu) = 0$. Thus, f is onto the open interval $(0, \alpha_{\max}^2)$.

Suppose $\alpha < \alpha_{\max}$. Then there exists $\mu > \lambda_1$ satisfying $f(\mu) = \alpha^2$. Then, by construction, the disturbance $d_\mu^* = L^\sim(Qz_\mu^*)$ satisfies $\|d_\mu^*\|_{2,[0,T]} = \alpha$, where z_μ^* is given by Eq. 6.13. This completes the proof for the case $\alpha < \alpha_{\max}$.

Now suppose $\alpha \geq \alpha_{\max}$. Then there is no $\mu > \lambda_1$ such that $f(\mu) = \alpha^2$. However, for $\mu = \lambda_1$, Eq. 6.12 has an infinite set of maximizers of the form $\{z^* = \begin{bmatrix} z_N^T & {z_R^*}^T \end{bmatrix}^T \mid z_N \in \mathbb{R}^r\}$ (see Appendix 6.6). By varying z_N, we can choose

$$z^* \in \arg\max_{z \in \mathbb{R}^{n_e}} -z^T \left(\lambda_1 \Lambda - \Lambda^2 \right) z + 2(\Lambda Q^T c)^T z$$

such that $d^* = L(Qz^*)$ has any norm in $[\alpha_{\max}, \infty)$, including α. The remaining details for the case $\alpha \geq \alpha_{\max}$, as well as the costs resulting from these disturbances, are included in Appendix 6.6. □

Now we are ready to state the main result, which follows from the lemmas presented in this section.

Theorem 9. *Assume G is output-controllable. Then there exists $\mu \geq \lambda_1$ and z_μ^* optimal for Eq. 6.12 such that $d_\mu^* = L^\sim(Qz_\mu^*)$ is a worst-case disturbance that achieves the maximal cost R_α in Eq. 6.7.*

Proof. Since G is output-controllable, by Lemma 5 there exists $\mu \geq \lambda_1$ and z_μ^* optimal for Eq. 6.12 such that $d_\mu^* = L^\sim(Qz_\mu^*)$ satisfies $\|d_\mu^*\|_{2,[0,T]} = \alpha$. Then, by Lemma 4, since $\mu \geq \lambda_1 > 0$ (by output controllability), d_μ^* is optimal for Eq. 6.9. Since d_μ^* is optimal for Eq. 6.9 and $\|d_\mu^*\|_{2,[0,T]} = \alpha$, we have that d_μ^* is optimal for Eq. 6.8 by Lemma 3. Finally, Equations 6.7 and 6.8 have the same maximizers by Lemma 2 (there exists d such that $L(d) \neq 0$ by output controllability), so d_μ^* is optimal for Eq. 6.7. Thus, the cost achieved by d_μ^* (called $\hat{R}_\alpha$ in Appendix 6.6) is the optimal cost R_α. □

We summarize the method for the case $\alpha < \alpha_{\max}$. Otherwise, slight modifications are required (see Appendix 6.6). First, solve the matrix Lyapunov differential equation and compute $W(T)$ as in Eq. 6.10. Compute $c = C(T)x(T)$ after simulating $\dot{x}(t) = A(t)x(t)$ on $[0, T]$ from $x(0) = 0$. Perform the eigenvalue decomposition $W(T) = Q\Lambda Q^T$ and compute $\tilde{c} = Q^T c$. Solve $f(\mu) = \alpha^2$ for μ (which is possible since $\alpha < \alpha_{\max}$). Then, by simulating the adjoint system in Eq. 6.11, we can compute the worst-case disturbance as $d_\mu^* = L^\sim(Qz_\mu^*)$, where $z_\mu^* = (\mu I - \Lambda)^{-1}\tilde{c}$. The resulting performance level is $R_\alpha = \mu^2 \sum_{i=1}^{n_e} \tilde{c}_i^2/(\mu - \lambda_i)^2$.

6.5 Example: Two-Link Robot Arm

We demonstrate the proposed method for linearized robustness analysis on an uncertain, nonlinear system. The model is composed of a known nonlinear part P and an unknown part $\Delta \in \mathbf{\Delta}$, interconnected as in Figure 6.1. Here, P comes from a two-link robot arm from Ch. 4, Sec. 2.3 of [92] following a nominal trajectory from [93]. P is a 4-state system with 3 inputs and 3 outputs. $\mathbf{\Delta}$ is the set of SISO LTI systems with an $\mathcal{H}_\infty$ norm bound of 0.8.

We generate $\{\Delta_i\}_{i=1}^{100} \subseteq \mathbf{\Delta}$ by randomly sampling 100 LTI systems with 1-6 states, scaled to have an $\mathcal{H}_\infty$ norm of 0.8. Then our goal is to solve Eq. 6.3 for $\alpha = 0.5$, i.e., to bound the output of the LTV approximation in Figure 6.2 for all uncertainties $\Delta \in \{\Delta_i\}_{i=1}^{100}$ and all disturbances $\|d\|_{2,[0,T]} \leq 0.5$.

First, we linearize P about the nominal trajectory to obtain $\bar{P}$. Then, for each $\Delta_i \in \{\Delta_i\}_{i=1}^{100}$, we form the interconnection in Figure 6.2, and we follow the procedure in Appendix 6.6 to obtain an augmented system G_i, which has a nonzero initial condition by construction. Then, for $i = 1, \ldots, 100$, we use the construction in Lemma 5 to compute the performance level R_α^i and a worst-case disturbance for G_i (denoted d_i^*). Maximizing the performance over the sample space as in Eq. 6.5 leads to the worst-case performance of 1.50×10^{-3}. Next, we analyze the system in Figure 6.2, but with $\bar{v}$ set to zero. Let $G_{0,i}$ denote the system obtained when $\bar{v} = 0$ in system G_i. Again, we use the construction in Lemma 5 to compute the performance level $R_\alpha^{0,i}$ and a worst-case disturbance $d_{0,i}^*$ for $G_{0,i}$. The worst-case performance for the system with $\bar{v} = 0$ is 8.31×10^{-4}. Thus, setting $\bar{v} = 0$ causes a significant discrepancy in the worst-case performance.

In addition to comparing the worst-case performance for the two systems, we also compare R_α^i and $R_\alpha^{0,i}$ across the uncertainty samples. Figure 6.3 shows these values, sorted in

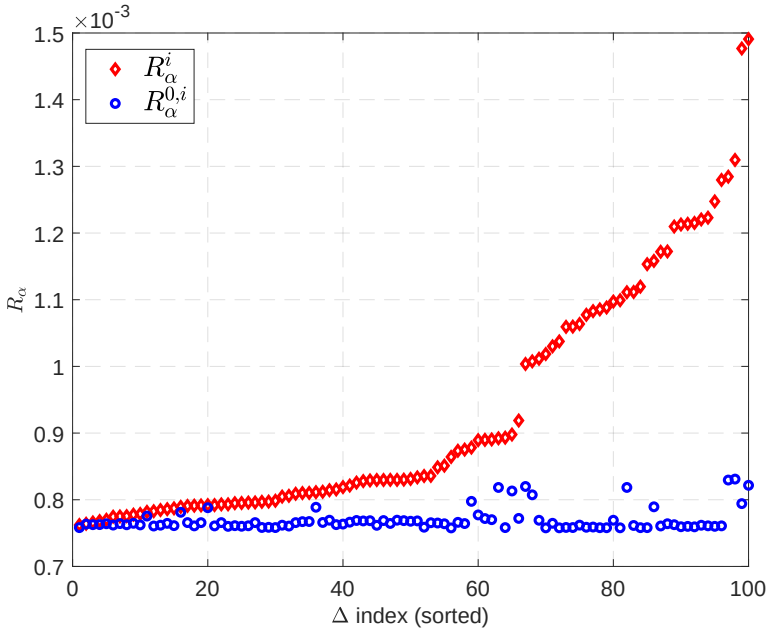


Figure 6.3: Performance R_α for each uncertainty, with and without $\bar{v}$.

	$F_U(P, \Delta_k)$	G_k	$G_{0,k}$
d_k^*	1.30×10^{-3}	1.30×10^{-3}	8.03×10^{-4}
$d_{0,k}^*$	5.51×10^{-4}	5.50×10^{-4}	8.31×10^{-4}

Table 6.1: $\|e(T)\|_2^2$ in response to d_k^* and $d_{0,k}^*$

increasing order of R_α^i. We see that $R_\alpha^i \geq R_\alpha^{0,i}$ $\forall i$, and for most i there is a significant discrepancy between the two.

Now we show numerically that including the term $\bar{v}$ leads to a linearized system that agrees more closely with the nonlinear system. We pick a particular sampled model uncertainty, Δ_k, to study the effect of $\bar{v}$. We generate a random disturbance $\|d_{\text{rand}}\|_{2,[0,T]} = \alpha$ and simulate it in (i) the nonlinear system $F_U(P, \Delta_k)$, (ii) the linearized system G_k, and (iii) the linearized system $G_{0,k}$, with $\bar{v} = 0$. The output e of all three systems to d_{rand} is plotted in Figure 6.4. The response of $F_U(P, \Delta_k)$ agrees much more closely with G_k than with $G_{0,k}$.

Finally, we compare the performance of all three systems when simulated with both worst-case disturbances d_k^* and $d_{0,k}^*$. The resulting values of $\|e(T)\|_2^2$ for all three systems are shown in Table 6.1. Again, the response of the nonlinear system $F_U(P, \Delta_k)$ agrees well with the linear system G_k, but deviates significantly from $G_{0,k}$. Furthermore, we see that the worst-case disturbance d_k^* makes $\|e(T)\|_2^2$ much larger for the nonlinear system than does $d_{0,k}^*$.

For this example, we showed that $\bar{v}$ significantly affects the behavior of the linearized system, and we showed that the nonlinear system matches much more closely with the linear

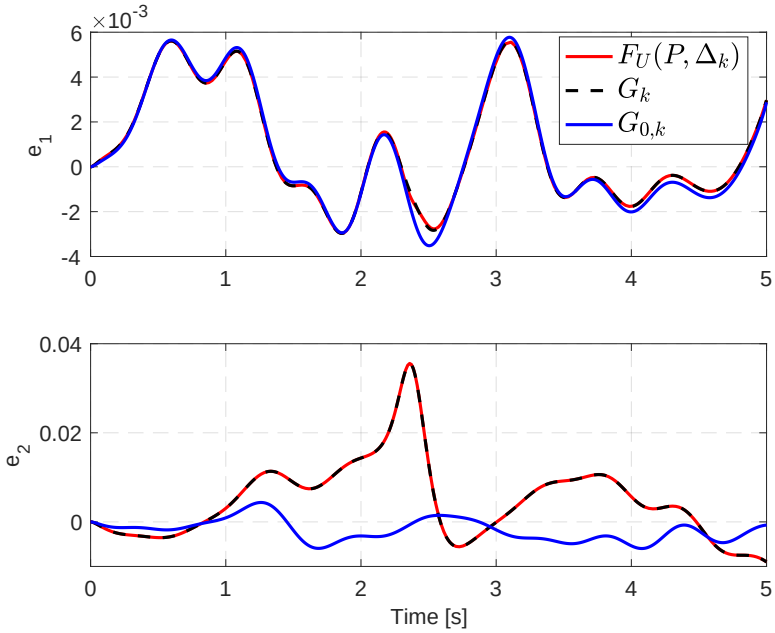


Figure 6.4: Response to random disturbance.

system that includes $\bar{v}$. Note that the effect of $\bar{v}$ is more pronounced for smaller values of α, where the effect of the uncertainty is large compared to the effect of the disturbance.

6.6 Conclusion

This Chapter presented a method for approximately analyzing the robustness of a nonlinear system to disturbances and model uncertainties. We linearized about a nominal trajectory to obtain a finite-horizon LTV system, which we interconnected with sampled model uncertainties. Each interconnection contained an affine input term to account for the shift in nominal trajectory, which we handled by creating an augmented system with a nonzero initial condition. We analytically computed a worst-case disturbance (and resulting performance level), and we demonstrated the method on a two-link robot arm example.

Augmented System

The interconnection in Figure 6.2 is an LTV system with inputs δu and $\bar{v}$ and output δy, with zero feedthrough δu to δy. Again, we refer to $(\delta u, \delta y)$ as (d, e). Let $\bar{x}$ denote the state of the interconnection, i.e., $\bar{x}$ contains the states of both $\bar{P}$ and Δ. Because of the zero feedthrough assumption, we can write the dynamics of the interconnection on $[0, T]$ in

general as:

$$\begin{aligned}\dot{\bar{x}}(t) &= \bar{A}(t)\bar{x}(t) + B_1(t)d(t) + B_2(t)\bar{v}(t) \\ e(t) &= \bar{C}(t)\bar{x}(t) + D_2(t)\bar{v}(t) \\ \bar{x}(0) &= \bar{x}_0\end{aligned} \tag{6.16}$$

Unlike the input d, the affine term $\bar{v}$ is fixed and known. We can rewrite the dynamics with d as the only input by augmenting the system with one additional state.

$$\begin{aligned}\underbrace{\begin{bmatrix}\dot{\bar{x}}(t)\\ 0\end{bmatrix}}_{\dot{x}(t)} &= \underbrace{\begin{bmatrix}\bar{A}(t) & B_2(t)\bar{v}(t)\\ 0 & 0\end{bmatrix}}_{A(t)}\underbrace{\begin{bmatrix}\bar{x}(t)\\ 1\end{bmatrix}}_{x(t)} + \underbrace{\begin{bmatrix}B_1(t)\\ 0\end{bmatrix}}_{B(t)} d(t) \\ e(t) &= \underbrace{\begin{bmatrix}\bar{C}(t) & D_2(t)\bar{v}(t)\end{bmatrix}}_{C(t)}\begin{bmatrix}\bar{x}(t)\\ 1\end{bmatrix}\end{aligned} \tag{6.17}$$

These are precisely the dynamics in Eq. 6.4. Because the augmented state is $x(t) = \begin{bmatrix}\bar{x}(t)^T & 1\end{bmatrix}^T$, the initial condition is $x(0) = \begin{bmatrix}\bar{x}(0)^T & 1\end{bmatrix}^T$, which is necessarily nonzero.

Proof of Lemma 1

Suppose d^* is optimal for Eq. 6.7.
Claim 1: $\langle L(d^*), c\rangle_2 \geq 0$. *Proof*: Suppose $\langle L(d^*), c\rangle_2 < 0$ and aim for contradiction. Let $J(d) = \|L(d) + c\|_2^2$ denote the cost function, and let $\tilde{d} = -d^*$.

$$\begin{aligned}J(\tilde{d}) &= \|L(\tilde{d}) + c\|_2^2 = \| - L(d^*) + c\|_2^2 \\ &= \|L(d^*)\|_2^2 + \|c\|_2^2 - 2\langle L(d^*), c\rangle_2 \\ &> \|L(d^*)\|_2^2 + \|c\|_2^2 + 2\langle L(d^*), c\rangle_2 = J(d^*)\end{aligned}$$

Thus, d^* does not maximize $J(d)$, which is a contradiction.
Claim 2: $\|L(d^*)\|_2 \neq 0$. *Proof*: Suppose $\|L(d^*)\|_2 = 0$ and aim for contradiction. There exists $\tilde{d}$ such that $L(\tilde{d}) \neq 0$ and $\|\tilde{d}\|_{2,[0,T]} \leq \alpha$. Then, either $\tilde{d}$ or $-\tilde{d}$ will outperform d^*:

$$\begin{aligned}\max\{J(\tilde{d}), J(-\tilde{d})\} &= \max\left\{\|L(\tilde{d}) + c\|_2^2, \| - L(\tilde{d}) + c\|_2^2\right\} \\ &= \|L(\tilde{d})\|_2^2 + \|c\|_2^2 + 2|\langle L(\tilde{d}), c\rangle_2| > \|c\|_2^2 = J(d^*)\end{aligned}$$

Thus, d^* does not maximize $J(d)$, which is a contradiction.

Proof of Lemma 1: Suppose $\|d^*\|_{2,[0,T]} < \alpha$ and aim for contradiction. Since $\|d^*\|_{2,[0,T]} < \alpha$, there exists $\epsilon > 0$ such that $\tilde{d} = (1+\epsilon)d^*$ satisfies $\|\tilde{d}\|_{2,[0,T]} \leq \alpha$. Then

$$\begin{aligned}J(\tilde{d}) &= \|L(\tilde{d}) + c\|_2^2 = \|(1+\epsilon)L(d^*) + c\|_2^2 \\ &= (1+\epsilon)^2\|L(d^*)\|_2^2 + \|c\|_2^2 + 2(1+\epsilon)\langle L(d^*), c\rangle_2 \\ &> \|L(d^*)\|_2^2 + \|c\|_2^2 + 2\langle L(d^*), c\rangle_2 = J(d^*)\end{aligned}$$

Claims 1 and 2 were used in the strict inequality step. Thus, d^* does not maximize $J(d)$, which is a contradiction. ■

Proof of Lemma 2

Let $\mu \in \mathbb{R}$. Suppose d_μ^* is optimal for Eq. 6.9 and $\|d_\mu^*\|_{2,[0,T]} = \alpha$. Then d_μ^* is also optimal for the following optimization problem:

$$\begin{aligned} \max_{d \in \mathcal{L}_2^{n_d}[0,T]} \ & \|L(d) + c\|_2^2 - \mu \|d\|_{2,[0,T]}^2 \\ \text{s.t. } & \|d\|_{2,[0,T]} = \alpha \end{aligned}$$

In the presence of the equality constraint, the term $-\mu\|d\|_2^2$ does not affect the set of maximizers of the optimization, so it can be removed from the cost function and d_μ^* will remain optimal. Hence, d_μ^* is optimal for Eq. 6.8. ■

Proof of Lemma 3

Let $\mu > 0$, and suppose z_μ^* is optimal for Eq. 6.12. We want to show that $d_\mu^* = L^\sim(Qz_\mu^*)$ is optimal for Eq. 6.9.

First, we need to establish some facts about the range and null space (denoted $\mathcal{R}$ and $\mathcal{N}$, respectively) of L and $L^\sim$.

1. $\mathcal{N}(L)$ is closed because L is bounded and linear (Theorem 1.18 of [94]). Therefore, $\mathcal{L}_2^{n_d}[0,T] = \mathcal{N}(L)^\perp \oplus \mathcal{N}(L)$ (Section 5.1 of [95]).
2. By output-controllability, $\mathcal{R}(L) = \mathbb{R}^{n_e}$, which is closed. Thus, $\mathcal{N}(L)^\perp = \mathcal{R}(L^\sim)$ (Ch. 4, Theorem 5.13 of [95]).

Thus, any $d \in \mathcal{L}_2^{n_d}[0,T]$ can be uniquely decomposed as $d = d_1 + d_2$, where $d_1 \in \mathcal{R}(L^\sim)$, $d_2 \in \mathcal{N}(L)$. Since $\mu > 0$, if $d_2 \neq 0$, we have

$$\|L(d) + c\|_2^2 - \mu\|d\|_{2,[0,T]}^2 < \|L(d_1) + c\|_2^2 - \mu\|d_1\|_{2,[0,T]}^2,$$

Thus, $d^* \in \mathcal{R}(L^\sim)$, i.e., $d^* = L^\sim(w^*)$ for some $w^* \in \mathbb{R}^{n_e}$. Since Q is invertible, we can also write $d^* = L^\sim(Qz^*)$ for some $z^* \in \mathbb{R}^{n_e}$. Then can rewrite Eq. 6.9 as

$$\begin{aligned} & \max_{z \in \mathbb{R}^{n_e}} \|L(L^\sim(Qz)) + c\|_2^2 - \mu\|L^\sim(Qz)\|_{2,[0,T]}^2 \\ = & \max_{z \in \mathbb{R}^{n_e}} \|Q\Lambda z + Q\tilde{c}\|_2^2 - \mu z^T \Lambda z \\ = & \max_{z \in \mathbb{R}^{n_e}} \|\Lambda z + \tilde{c}\|_2^2 - \mu z^T \Lambda z \\ = & \max_{z \in \mathbb{R}^{n_e}} -z^T(\mu\Lambda - \Lambda^2)z + 2(\Lambda\tilde{c})^T z + \tilde{c}^T\tilde{c} \end{aligned} \tag{6.18}$$

Because z_μ^* is optimal for Eq. 6.12, it is also optimal for Eq. 6.18 since the two optimizations differ by a constant. Thus $d^* = L^\sim(Qz_\mu^*)$ is optimal for Eq. 6.9. ■

Proof of Lemma 4

The case $\alpha < \alpha_{\max}$ is handled in the body of the Chapter. Now suppose $\alpha > \alpha_{\max}$ (which can only occur if $\tilde{c}_1 = \cdots = \tilde{c}_r = 0$). In this case, we need an alternative method for constructing the worst-case disturbance of norm α. It is useful to partition the variables as $z = [z_N^T, z_R^T]^T$, $\tilde{c} = [0, \tilde{c}_R^T]^T$, $\Lambda = \text{diag}(\lambda_1 I_r, \Lambda_R)$, $Q = [Q_N, Q_R]$, where $z_N \in \mathbb{R}^r$. Note that $\lambda_1\Lambda - \Lambda^2 = \text{blkdiag}\,(0, \Lambda_R(\lambda_1 I - \Lambda_R))$ and $\Lambda\tilde{c} = \Lambda_R\tilde{c}_R$. Then, at $\mu = \lambda_1$, we can rewrite Eq. 6.12:

$$\begin{aligned}&\max_{z\in\mathbb{R}^{n_e}} -z^T(\lambda_1\Lambda - \Lambda^2)z + 2(\Lambda\tilde{c})^T z \\ =&\max_{z_R\in\mathbb{R}^{n_e-r}} -z_R^T\Lambda_R(\lambda_1 I - \Lambda_R) + 2(\Lambda_R\tilde{c}_R)^T z_R\end{aligned}$$

Thus, the value of z_N does not affect the cost function, while the unique maximizing value for z_R is

$$z_R^* = (\lambda_1 I - \Lambda_R)^{-1}\,\tilde{c}_R$$

Therefore, for all $z_N \in \mathbb{R}^r$, $z_{\lambda_1}^* = [z_N^T, {z_R^*}^T]^T$ is optimal for Eq. 6.12 with $\mu = \lambda_1$. Now it suffices to select z_N such that $d_{\lambda_1}^* = L^{\sim}(Qz_{\lambda_1}^*)$ has norm α. Use Eq. 6.14 to write

$$\begin{aligned}\alpha^2 = \|d_{\lambda_1}^*\|_{2,[0,T]}^2 = {z_{\lambda_1}^*}^T\Lambda z_{\lambda_1}^* &= \begin{bmatrix} z_N \\ z_R^* \end{bmatrix}^T \begin{bmatrix} \lambda_1 I_r & 0 \\ 0 & \Lambda_R \end{bmatrix} \begin{bmatrix} z_N \\ z_R^* \end{bmatrix} \\ &= \lambda_1\|z_N\|_2^2 + \tilde{c}_R^T\,(\lambda_1 I - \Lambda_R)^{-1}\,\Lambda_R\,(\lambda_1 I - \Lambda_R)^{-1}\,\tilde{c}_R \\ &= \lambda_1\|z_N\|_2^2 + \sum_{i=r+1}^{n_e} \tilde{c}_i^2 \frac{\lambda_i}{(\lambda_1 - \lambda_i)^2} = \lambda_1\|z_N\|_2^2 + \alpha_{\max}^2\end{aligned}$$

Therefore, if we set $\|z_N\|_2^2 = (\alpha^2 - \alpha_{\max}^2)\,/\lambda_1$, then $d_{\lambda_1}^* = L^{\sim}(Qz_{\lambda_1}^*)$ has norm α.

For all $\alpha > 0$, we have now derived the form of d_μ^* (which has norm α). Let $\hat{R}_\alpha$ denote the cost function of Eq. 6.7 evaluated on this disturbance. To calculate $\hat{R}_\alpha$, first observe:

$$\begin{aligned}\hat{R}_\alpha &= \|L(d_\mu^*) + c\|_2^2 = \|L(L^{\sim}(Qz_\mu^*)) + c\|_2^2 \\ &= \|Q\Lambda z_\mu^* + c\|_2^2 = \|\Lambda z_\mu^* + \tilde{c}\|_2^2\end{aligned}$$

If $\alpha < \alpha_{\max}$, then plugging in z_μ^* yields:

$$\begin{aligned}\hat{R}_\alpha &= \|\Lambda(\mu I - \Lambda)^{-1}\tilde{c} + \tilde{c}\|_2^2 \\ &= \|\mu(\mu I - \Lambda)^{-1}\tilde{c}\|_2^2 = \mu^2 \sum_{i=1}^{n_e} \frac{\tilde{c}_i^2}{(\mu - \lambda_i)^2}\end{aligned}$$

If $\alpha \geq \alpha_{\max}$, then plugging in z_μ^* yields:

$$\begin{aligned}\hat{R}_\alpha &= \left\| \begin{bmatrix} \lambda_1 I & 0 \\ 0 & \Lambda_R \end{bmatrix} \begin{bmatrix} z_N \\ z_R^* \end{bmatrix} + \begin{bmatrix} \lambda_1 0 \\ \tilde{c}_R \Lambda_R \end{bmatrix} \right\|_2^2 \\ &= \lambda_1^2 \|z_N\|_2^2 + \|\Lambda_R z_R^* + \tilde{c}_R\|_2^2 \\ &= \lambda_1(\alpha^2 - \alpha_{\max}^2) + \lambda_1^2 \sum_{i=r+1}^{n_e} \frac{\tilde{c}_i^2}{(\lambda_1 - \lambda_i)^2}\end{aligned}$$

■

Chapter 7

Conclusion

A hierarchical control architecture that uses different models in each layer can be a powerful control strategy when the error between the layers is accounted for. In this dissertation, we described the planner-tracker framework, a hierarchical control framework where a lower-fidelity "planning" model is employed for online planning and a "tracking" controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set. The planner-tracker framework unites ideas from tube MPC, formal methods, and other areas to create a modular control architecture that can accommodate many modifications. Robustness can be added throughout the hierarchy to reflect the many sources of uncertainty in real systems, enhancing the safety and usefulness of the framework. In Chapter 2, we highlighted the practical advantages and technical approach for allowing an error definition that depends on the planner input $\hat{u}$. In Chapter 3, we added robustness to unmodeled dynamics, an extension that was application-agnostic. In Chapter 4, we focused on the application of shared vehicle control between a human driver and a supervisory autonomous system designed to intervene in an emergency. In this setting, we added robustness to other vehicles in longitudinal scenarios by designing scenario-specific MPC terminal sets that guarantee persistent feasibility with respect to a particular contingency event, such as the vehicle ahead of the ego vehicle braking abruptly. In Chapter 5, we formulated a backstepping-based tracking controller as an alternative to the SOS-based controllers used in Chapters 2 and 3. In Chapter 6, we described an additional method for assessing the robustness of a system around a given nominal trajectory.

Further modifications and extensions to the planner-tracker framework can be made. The following are areas for future work:

- One straightforward extension of the framework in Chapter 2 would be to allow the MPC planner state at each sampling time $\hat{x}(kT)$ to be a decision variable. Instead of setting $\hat{x}(kT)$ to be the planner state from the end of the previous time-step, the planner state can be any value such that the error lies in the invariant set, i.e, $e(kT) = \psi(x(kT), \hat{x}(kT), \hat{u}(kT)) \in \Omega(V, \gamma)$. This additional flexibility allows the planner to

lower its cost and more efficiently achieve its goals. With this modification, the planner-tracker framework with an MPC planner can be viewed as a tube MPC method.

- In Chapter 3, the error definition only depends on the tracker state x and the planner state $\hat{x}$, not the planner input $\hat{u}$. Extension of the error to be dependent on $\hat{u}$ would introduce discontinuous error dynamics whenever $\hat{u}$ is discontinuous. In Chapter 2, we handled this discontinuity by imposing an invariance condition for each sampling period $[kT, (k+1)T)$ (and a separate invariance condition between sample times), treating each $e(kT)$ as a new initial condition. However, with this interval-by-interval approach, the IQC used in Chapter 3 would require the filter initial condition to be $x_F(kT) = 0$, which will not generally be true at each sampling time. A separate generalization of the error could be to allow its definition to depend on the filter state x_F. This would not induce the error discontinuity described above.

- In the presence of unmodeled dynamics as in Chapter 3, additional flexibility may be incorporated by including the planner initial condition $\hat{x}(kT)$ as another decision variable subject to the constraint $e(kT) = \psi(x(kT), \hat{x}(kT)) \in \Omega(V, \gamma)$. This would decrease the planner cost and allow the system to more efficiently achieve its goals. However, this would create discontinuities in the planner and error dynamics between sampling periods, again leading to the possibility of nonzero $x_F(kT)$ for each sample period, so the IQC would not necessarily hold. It would be interesting to treat the error system as a hybrid system to try to obtain an invariant set for the error even subject to these discontinuities.

- The the error bound $\mathcal{O}$ discussed in Chapters 2 and 3 could be augmented to additionally account for discretization error (e.g., in an MPC planner) and polynomial approximation error (e.g., if dynamics have trigonometric terms that are approximated as polynomials).

- In Chapter 3, we allowed unmodeled dynamics at the *input* of the tracker model, characterized using IQCs. The only input to the filter F was the (known) tracking input u, so the filter state x_F was computed and used for control. Incorporating l (the unknown output of the uncertain block Δ) as an additional filter input would allow for the IQCs to capture more general unmodeled dynamics, rather than just those at the input. This additional (unknown) filter input would prevent measurement of the filter state, x_F, motivating the development of a control strategy that applies to all x_F within some bound and does not require exact knowledge of x_F.

Bibliography

[1] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[2] N. Palumbo, R. Blauwkamp, and J. Lloyd, "Modern homing missile guidance theory and techniques," *Johns Hopkins APL Technical Digest*, vol. 29, no. 1, pp. 42–59, 2010.

[3] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[4] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE Conference on Decision and Control (CDC)*, Dec. 2017, pp. 1517–1522.

[5] S. Singh, A. Majumdar, J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5883–5890.

[6] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020. [Online]. Available: https://doi.org/10.1177/0278364920943266

[7] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: An SOS optimization approach," in *Algorithmic Foundations of Robotics XIII*, M. Morales, L. Tapia, G. Sánchez-Ante, and S. Hutchinson, Eds. Cham: Springer International Publishing, 2020, pp. 545–564.

[8] S. W. Smith, H. Yin, and M. Arcak, "Continuous abstraction of nonlinear systems using sum-of-squares programming," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 8093–8098.

[9] U. Rosolia and A. D. Ames, "Multi-rate control design leveraging control barrier functions and model predictive control policies," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1007–1012, 2021.

[10] K. S. Schweidel, H. Yin, S. W. Smith, and M. Arcak, "Safe-by-design planner–tracker synthesis with a hierarchy of system models," *Annual Reviews in Control*, vol. 53, pp. 138–146, 2022.

[11] K. S. Schweidel, A. K. Packard, M. Arcak, P. J. Seiler, and D. O. Philbrick, "Classifier-based supervisory control with application to threat engagement," in *2020 American Control Conference (ACC)*, 2020, pp. 3757–3762.

[12] N. Palumbo, R. Blauwkamp, and J. Lloyd, "Basic principles of homing guidance," *Johns Hopkins APL Techical Digest*, vol. 29, no. 1, pp. 25–41, 2010.

[13] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017. [Online]. Available: https://www.mdpi.com/2075-1702/5/1/6

[14] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[15] H. Laghmara, M.-T. Boudali, T. Laurain, J. Ledy, R. Orjuela, J.-P. Lauffenburger, and M. Basset, "Obstacle avoidance, path planning and control for autonomous vehicles," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 529–534.

[16] I. Mir, F. Gul, S. Mir, M. A. Khan, N. Saeed, L. Abualigah, B. Abuhaija, and A. H. Gandomi, "A survey of trajectory planning techniques for autonomous systems," *Electronics*, vol. 11, no. 18, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/18/2801

[17] C. G. Bobier-Tiu, S. M. Koehler, M. Brown, and M. Ahumada, "A unified mpc envelope control formulation for toyota guardian and chauffeur," *IFAC-PapersOnLine*, vol. 55, no. 16, pp. 19–24, 2022, 18th IFAC Workshop on Control Applications of Optimization CAO 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589632201165X

[18] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022.

[19] E. Sin, "Trajectory optimization and control of small spacecraft constellations," *PhD thesis, University of California, Berkeley*, 2021.

[20] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017. [Online]. Available: https://doi.org/10.1177/0278364917712421

[21] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[22] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[23] W. Langson, I. Chryssochoos, S. V. Rakovic, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, pp. 125–133, 2004.

[24] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, "Optimization over state feedback policies for robust control with constraints," *Automatica*, vol. 42, no. 4, pp. 523–533, 2006.

[25] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen, "Parameterized tube model predictive control," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2746–2761, 2012.

[26] S. V. Raković and Q. Cheng, "Homothetic tube MPC for constrained linear difference inclusions," in *Chinese Control and Decision Conference (CCDC)*. IEEE, 2013, pp. 754–761.

[27] D. Muñoz-Carpintero, M. Cannon, and B. Kouvaritakis, "Recursively feasible robust MPC for linear systems with additive and multiplicative uncertainty using optimized polytopic dynamics," in *Conference on Decision and Control (CDC)*. IEEE, 2013, pp. 1101–1106.

[28] J. Fleming, B. Kouvaritakis, and M. Cannon, "Robust tube MPC for linear systems with multiplicative uncertainty," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 1087–1092, 2014.

[29] S. V. Raković, W. S. Levine, and B. Açikmese, "Elastic tube model predictive control," in *American Control Conference (ACC)*. IEEE, 2016, pp. 3594–3599.

[30] M. Bujarbaruah, U. Rosolia, Y. R. Stürz, X. Zhang, and F. Borrelli, "Robust MPC for linear systems with parametric and additive uncertainty: A novel constraint tightening approach," *arXiv preprint arXiv:2007.00930*, 2020.

[31] L. Schwenkel, J. Köhler, M. A. Müller, and F. Allgöwer, "Dynamic uncertainties in model predictive control: guaranteed stability for constrained linear systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1235–1241.

[32] ——, "Model predictive control for linear uncertain systems using integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 355–368, 2023.

[33] S. Yu, C. Maier, H. Chen, and F. Allgöwer, "Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems," *Systems & Control Letters*, vol. 62, no. 2, pp. 194–200, 2013.

[34] T. Faulwasser, L. Grüne, M. A. Müller *et al.*, "Economic nonlinear model predictive control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 1, pp. 1–98, 2018.

[35] J. Köhler, M. A. Müller, and F. Allgöwer, "A novel constraint tightening approach for nonlinear robust model predictive control," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 728–734.

[36] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.

[37] M. Cannon, J. Buerger, B. Kouvaritakis, and S. Rakovic, "Robust tubes in nonlinear model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1942–1947, 2011.

[38] M. Löhning, M. Reble, J. Hasenauer, S. Yu, and F. Allgöwer, "Model predictive control using reduced order models: Guaranteed stability for constrained linear systems," *Journal of Process Control*, vol. 24, no. 11, pp. 1647–1659, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095915241400211X

[39] J. Lorenzetti, A. McClellan, C. Farhat, and M. Pavone, "Linear reduced-order model predictive control," *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 5980–5995, 2022.

[40] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1781–1796, 2017.

[41] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, no. 2, p. 566–571, Feb. 2009. [Online]. Available: https://doi.org/10.1016/j.automatica.2008.09.016

[42] P.-J. Meyer, H. Yin, A. H. Brodtkorb, M. Arcak, and A. J. Sørensen, "Continuous and discrete abstractions for planning, applied to ship docking," in *21st IFAC World Congress*, 2020.

[43] K. S. Schweidel, S. M. Koehler, V. R. Desaraju, and M. Barić, "Driver-in-the-loop contingency mpc with invariant sets," in *2022 European Control Conference (ECC)*, 2022, pp. 808–813.

[44] K. S. Schweidel, J. R. Buch, P. J. Seiler, and M. Arcak, "Computing worst-case disturbances for finite-horizon linear time-varying approximations of uncertain systems," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1753–1758, 2021.

[45] K. S. Schweidel, P. J. Seiler, and M. Arcak, "Safe-by-design planner-tracker synthesis with unmodeled input dynamics," *To appear in IEEE Control Systems Letters*, 2023.

[46] H. Yin, M. Bujarbaruah, M. Arcak, and A. Packard, "Optimization based planner–tracker design for safety guarantees," in *2020 American Control Conference (ACC)*, 2020, pp. 5194–5200.

[47] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[48] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[49] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, Dec. 2018, pp. 6059–6066.

[50] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer, "A computationally efficient robust model predictive control framework for uncertain nonlinear systems," *submitted to IEEE Transactions on Automatic Control*, 2019.

[51] Y. Pant, H. Yin, M. Arcak, and S. Seshia, "Co-design of control and planning for multi-rotor UAVs with signal temporal logic specifications," in *Proceedings of the 2021 American Control Conference*, New Orleans, Louisiana, May 2021, pp. 4199–4206.

[52] P. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," *PhD thesis, California Institute of Technology*, 2000.

[53] H. Yin, M. Arcak, A. K. Packard, and P. Seiler, "Backward reachability for polynomial systems on a finite horizon," *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 6025–6032, 2021.

[54] D. Angeli, "A lyapunov approach to incremental stability properties," *IEEE Transactions on Automatic Control*, vol. 47, no. 3, pp. 410–421, 2002.

[55] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099.

[56] P. Seiler, "SOSOPT: A toolbox for polynomial optimization," *arXiv preprint arXiv:1308.1889*, 2013.

[57] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

[58] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, pp. 1–37, 2021.

[59] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[60] P. Seiler, R. M. Moore, C. Meissen, M. Arcak, and A. Packard, "Finite horizon robustness analysis of LTV systems using integral quadratic constraints," *Automatica*, vol. 100, pp. 135–143, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109818305417

[61] P. Seiler, M. Jankovic, and E. Hellstrom, "Control barrier functions with unmodeled input dynamics using integral quadratic constraints," *IEEE Control Systems Letters*, vol. 6, pp. 1664–1669, 2022.

[62] K. Astrom, C. Hang, and B. Lim, "A new Smith predictor for controlling a process with an integrator and long dead-time," *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 343–345, 1994.

[63] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P. A. Parrilo, M. M. Peet, and D. Jagt, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, `http://arxiv.org/abs/1310.4716`, 2021, available from `https://github.com/oxfordcontrol/SOSTOOLS`.

[64] *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html

[65] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[66] M. Kyriakidis, C. van de Weijer, B. van Arem, and R. Happee, "The deployment of advanced driver assistance systems in Europe," in *Proc. 22nd ITS World Cong.*, 2015.

[67] I. J. Reagan, J. B. Cicchino, L. B. Kerfoot, and R. A. Weast, "Crash avoidance and driver assistance technologies – are they used?" *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 52, pp. 176–190, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1369847817303674

[68] M. Marcano, S. Díaz, J. Pérez, and E. Irigoyen, "A review of shared control for automated vehicles: Theory and applications," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 6, pp. 475–491, 2020.

[69] C. G. Bobier and J. C. Gerdes, "Staying within the nullcline boundary for vehicle envelope control using a sliding surface," *Vehicle System Dynamics*, vol. 51, no. 2, pp. 199–217, 2013. [Online]. Available: https://doi.org/10.1080/00423114.2012.720377

[70] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2013.

[71] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, 2016.

[72] J. Hardy and M. Campbell, "Contingency planning over probabilistic hybrid obstacle predictions for autonomous road vehicles," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2237–2242.

[73] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *2019 American Control Conference (ACC)*, 2019, pp. 717–722.

[74] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[75] E. Kerrigan and J. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Proc. 39th IEEE Conference on Decision and Control*, vol. 5, 2000, pp. 4951–4956.

[76] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *2013 European Control Conference (ECC)*, 2013, pp. 502–510.

[77] S. Lefèvre, A. Carvalho, and F. Borrelli, "A learning-based framework for velocity control in autonomous driving," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 32–42, 2016.

[78] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, pp. 1805–1824, Aug 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.62.1805

[79] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[80] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Pearson, 2001.

[81] M. Arcak and P. Kokotović, "Redesign of backstepping for robustness against unmodeled dynamics," *International Journal of Robust and Nonlinear Control*, vol. 11, pp. 633–643, 2001.

[82] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley Sons, Ltd, 2011.

[83] J. Buch, M. Arcak, and P. Seiler, "An efficient algorithm to compute norms for finite horizon, linear time-varying systems," *IEEE Control Systems Letters*, pp. 1–1, 2020.

[84] J. E. Tierno, R. M. Murray, J. C. Doyle, and I. M. Gregory, "Numerically efficient robustness analysis of trajectory tracking for nonlinear systems," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 4, pp. 640–647, 1997.

[85] P. Seiler, R. M. Moore, C. Meissen, M. Arcak, and A. Packard, "Finite horizon robustness analysis of LTV systems using integral quadratic constraints," *Automatica*, vol. 100, pp. 135 – 143, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109818305417

[86] A. Iannelli, P. Seiler, and A. Marcos, "Worst-case disturbances for time-varying systems with application to flexible aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 6, pp. 1261–1271, 2019.

[87] A. B. Kurzhanski and P. Varaiya, "On ellipsoidal techniques for reachability analysis. part i: external approximations," *Optimization methods and software*, vol. 17, no. 2, pp. 177–206, 2002.

[88] K. Zhou, J. C. Doyle, and K. Glover, *Robust and optimal control*. Prentice hall New Jersey, 1996, vol. 40.

[89] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, 2nd ed. Springer-Verlag London, 2013.

[90] M. Green and D. J. Limebeer, *Linear Robust Control*. Courier Corporation, 2012.

[91] R. W. Brockett, *Finite dimensional linear systems*. SIAM, 2015.

[92] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robot Manipulation*. CRC Press, 1994.

[93] R. Moore, "Finite horizon robustness analysis using integral quadratic constraints," Master's thesis, University of California, Berkeley, 2015.

[94] W. Rudin, *Functional analysis*, 2nd ed. McGraw-Hill, 1973.

[95] T. Kato, *Perturbation theory for linear operators*. Springer Science & Business Media, 2013.