ROBUST CONTROL, PLANNING, AND INFERENCE FOR SAFE ROBOT AUTONOMY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Sumeet Singh
August 2019

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Rock

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mac Schwager

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Integrating autonomous robots into safety-critical settings requires reasoning about uncertainty at all levels of the autonomy stack. This thesis presents novel algorithmic tools for imbuing robustness within two hierarchically complementary areas, namely: motion planning and decision-making.

In Part I of the thesis, by harnessing the theories of contraction and semi-infinite convex optimization and the computational tool of sum-of-squares programming, we present a unified framework for robust real-time motion planning for complex underactuated nonlinear systems. Broadly, the approach entails pairing open-loop motion planning algorithms that neglect uncertainty and are optimized for generating trajectories for simple kinodynamic models in real-time, with robust nonlinear trajectory-tracking feedback controllers. We demonstrate how to systematically synthesize these controllers and integrate them within planning to generate and execute certifiably safe trajectories that are robust to the closed-loop effects of disturbances and planning with simplified models.

In Part II of the thesis, we demonstrate how to embed the control-theoretic advancements developed in Part I as constraints within a novel semi-supervised algorithm for learning dynamical systems from user demonstrations. The constraints act as a form of context-driven hypothesis pruning to yield learned models that jointly balance regression performance and stabilizability, ultimately resulting in generated trajectories for the robot that are conditioned for feedback control. Experimental results on a quadrotor testbed illustrate the efficacy of the proposed algorithms in Parts I and II of the thesis, and clear connections between theory and hardware.

Finally, in Part III of the thesis, we describe a framework for lifting notions of robustness from low-level motion planning to higher-level sequential decision-making using the theory of risk measures. Leveraging a class of risk measures with favorable axiomatic foundations, we demonstrate how to formulate decision-making algorithms with tunable robustness properties. In particular, we focus on a novel application of this framework to inverse reinforcement learning where we learn predictive motion models for humans in safety-critical scenarios, and illustrate their effectiveness within a commercial driving simulator featuring humans in-the-loop.

The contributions within this thesis constitute an important step towards endowing modern robotic systems with the ability to systematically and hierarchically reason about safety and efficiency in the face of uncertainty, which is crucial for safety-critical applications.

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Marco Pavone. Marco's support and guidance as a mentor and academic role model, throughout my time at Stanford, cannot be overstated. His dedication to scientific excellence, academic integrity, and concern for students are qualities that I hope will be a part of me throughout my professional career. I thank him for his patience while I explored several research directions during my initial years in the lab, and his consistent encouragement to engage and collaborate with the broader research community. I look forward to collaborating with Marco as a colleague.

I sincerely thank Prof. Jean-Jacques Slotine at MIT Mechanical Engineering. Jean-Jacques has essentially been an unofficial "co-advisor" and his tremendous insight and experience have been a significant factor for the majority of the work presented herein. Additionally, I thank Jean-Jacques for his support during the academic job search and look forward to continuing our collaboration.

A special thank you must be said to Prof. Anirudha Majumdar at Princeton University MAE. Ani's tenure as a postdoc with the Autonomous Systems Lab was a huge turning point for my research. Within the span of a year, he helped me establish myself as a robotics researcher and provided unwavering support as a mentor and collaborator. He was and continues to be a significant figure for inspiration and advice and I eagerly look forward to continuing our collaboration.

I thank Prof. Mac Schwager and Prof. Stephen Rock for serving on the reading committee for this thesis and providing invaluable feedback. I additionally thank Mac for his sincere advice and support during the academic job search, and Prof. Allison Okamura for serving as the chair for the defense committee and providing thoughtful feedback.

Research is a team effort. I am sincerely grateful for having the opportunity to work with so many talented co-authors. In particular, I thank Prof. Claire Tomlin at UC Berkeley EECS for her advice and mentorship during our collaboration and her invaluable support during the academic job search. I gratefully acknowledge Dr. Vikas Sindhwani at Google Brain Robotics for his sincere mentorship during our collaboration and supporting me during the job search. I eagerly look forward to joining his team in New York to continue to work on impactful projects within robotics. I sincerely thank Dr. Edward Schmerling, Prof. Simone D'Amico, Dr. Yinlam Chow, Ajay Mandlekar, Jonathan Lacotte, Dr. Zijian Wang, Sylvia Herbert, Prof. Mo Chen, Joseph Lorenzetti, Benoit Landry, and

Spencer M. Richards. It has been a privilege to work with and learn from all of you, and I hope to continue to do so in the future.

At the personal level, I thank each and every member, past and present, of the Autonomous Systems Lab. Undertaking a PhD is by no means, an easy endeavor. However, your companionship through all the stressful deadlines and constant source of inspiration and spirited discussion made it worthwhile. I am additionally grateful to all the friends I made playing with the Stanford orchestras, in particular, the outstanding percussion section. I will cherish our concerts together and impromptu mini-jam sessions during rehearsals.

To my parents, Dr. Raminder Pal Singh and Sunila Singh, and my grandparents, Tarlochan Singh Sachdev and Amarjit Kaur Sachdev, this thesis is the result of your unwavering support, and unconditional love and understanding. I promise to visit more often!

I acknowledge the support of the Stanford Graduate Fellowship – Office of Technology Licensing, Office of Naval Research (ONR) – Science of Autonomy Program grant number N00014-15-1-2673 and grant number N00014-17-1-2749, NASA Space Technology Research Grant NNX12AQ43G, King Abdulaziz City for Science and Technology (KACST), and Toyota Research Institute (TRI). This thesis solely reflects the opinions and conclusions of myself and not ONR, NASA, KACST, TRI, or any other Toyota entity.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The progress in robotics and machine learning research in the last decade is undeniably staggering. Complementarily, advancements in off- and on-board computational technology have unlocked the potential to equip robots with increasingly sophisticated, rich sensing modalities and the supporting algorithms to generate intelligent actions. However, the state-of-the-art in robotics excels at optimizing performance of pre-defined tasks in constrained isolated environments such as factory floors. This limits their usefulness within open unstructured environments, potentially shared by other agents, where optimizing efficiency is no longer the sole objective as performance must be balanced with safety. *The goal of my research is to devise algorithms for planning and decision-making that carefully balance efficiency and safety, for robots operating in* dynamic uncertain environments *co-inhabited by humans and other robots.* Target domains of this research include applications in agile motion planning, safety-critical human-robot interaction, and robust machine learning.

Within such a broad field, this thesis focuses on two hierarchically distinct areas, namely, (1) *robust trajectory optimization*, and (2) *risk-sensitive decision-making.* Robust trajectory optimization entails generating and tracking trajectories for agile robots with complex nonlinear dynamics in cluttered environments subject to external disturbances (e.g., a UAV buffeted by uncertain winds), and constitutes a form of *low-level decision-making.* Complementarily, risk-sensitive decision-making focuses on *higher-level decision-making*, in particular, in the use of more nuanced statistical measures of performance, (i.e., beyond expected value or variance) to better capture events of low probability yet potentially catastrophic outcomes. The contributions of this thesis are outlined in the following discussion.

1.1 Robust Trajectory Optimization

Robustness here entails three key challenges. First, the presence of uncertainty (e.g., from external disturbances) in the dynamics forces us to reason about the *range* of possible outcomes that

the disturbances may drive the system to, rather than a single planned trajectory. Second, reliably computing global trajectories that are dynamically feasible for complex nonlinear systems is a challenging problem that is generally intractable to solve in real-time. In the pursuit of fast online planning, a typical solution approach leverages simplified lower-dimensional kinematic models to generate motion plans online, which are then tracked by the full-order system. The resulting mismatch between the planning and actual dynamics of the robot leads to inevitable tracking error that must be factored within the planning stage. Third, in the absence of accurate (or any) dynamic models, one must use statistical learning techniques to first estimate a model from observed trajectories on the robot and then construct a control policy for the desired task using this learned model. However, the performance and stability of such policies can vary tremendously depending on the quality of the learned model and by extension, the learning algorithm itself.

1.1.1 Robust Motion Planning

To address the first two challenges, we pair existing motion planning algorithms that ignore uncertainty and are optimized for generating nominal trajectories for simple kinodynamic models *in real-time*, with robust nonlinear trajectory-tracking feedback controllers.

Specifically, to address the challenge of robustness with respect to exogenous disturbances, we develop original proofs for establishing trajectory-based stabilizability of a nonlinear system using Contraction Theory (Lohmiller and Slotine, 1998), quantify measures of robustness using the notion of *invariant tubes*, and develop an *offline* quasiconvex optimization algorithm using sum-of-squares (SOS) programming for *automatically synthesizing* controllers that optimize robust performance. The computed invariant tubes represent a *conservative* approximation to the region of state-space surrounding a trajectory within which the system is *guaranteed* to remain while using the robust tracking controller *in the presence of disturbances* (cf. Figure 1.1).

To address the dynamics mismatch between a simple planning model used by the real-time planner and the true dynamics of the robot, we formulate an *offline* optimization whereby one computes, for a pair of "planner" (i.e., low-dimensional) and "tracking" (i.e., high-dimensional) models, a feedback tracking controller and associated tracking bound. This bound is then used as a safety margin when generating motion plans *online* via the low-dimensional model. The approach leverages SOS programming and is shown to be competitive with the state-of-the-art exact Hamilton-Jacobi (HJ) differential game method for low-dimensional problems, while being scalable to higher-dimensional problems beyond the reach of HJ.

Combining the two solutions together yields an algorithm that features planning with fast but lower-fidelity models (e.g., using sampling-based planners), supported by real-time geometric collision checking algorithms using buffers (cf. Figure 1.1) that account for both (i) dynamic infeasibility, *and* (ii) external disturbances. The key insight is to combine nonlinear control and *systematic offline optimization-based synthesis* to endow the online motion generating algorithms with the ability to

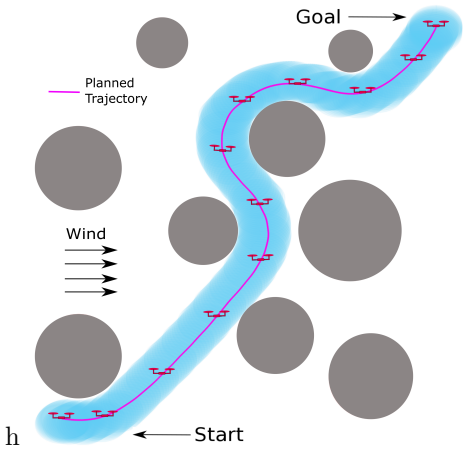
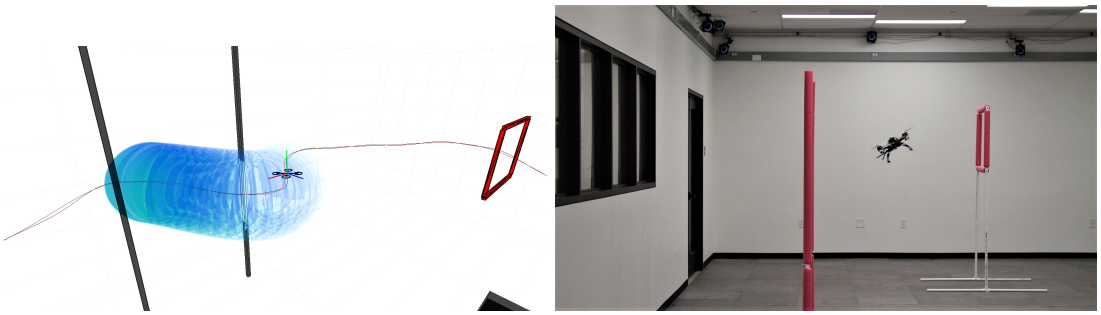


Figure 1.1: Planning with invariant tubes. The robot, in this case, a planar quadrotor, subject to bounded disturbances (e.g., from the cross-wind), plans a path (magenta) around the obstacles (shaded grey) using a collision margin (shaded blue) derived from the robustness properties of the automatically synthesized tracking controller.

adapt to challenging dynamic scenarios in real-time. The approach has been extensively validated through numerical experiments and recently implemented on a quadrotor (see Figure 1.2) to illustrate the feasibility of converting the rich theoretical analysis into a practical tool capable of running on modern hardware.



(a) (b)

Figure 1.2: Illustration of our robust planning with invariant tubes methodology on a full 3D quadrotor in simulation and experiment.

Relevant Publications:

Singh S, Majumdar A, Slotine JJE and Pavone M (2017) Robust online motion planning via contraction theory and convex optimization. In: *Proc. IEEE Conf. on Robotics and Automation*. Extended Version, Available at `http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf`.

Singh S, Chen M, Herbert SL, Tomlin CJ and Pavone M (2018a) Robust tracking with model

mismatch for fast and safe planning: an SOS optimization approach. In: *Workshop on Algorithmic Foundations of Robotics*.

Singh S, Landry B, Majumdar A, Slotine JJE and Pavone M (2019a) Robust online motion planning via contraction theory and convex optimization. *Int. Journal of Robotics Research* Submitted.

1.1.2 Control-Theoretic Regularization for Model-Based RL

The work on trajectory-wise stabilizability gives rise to an interesting application in model-based reinforcement learning (RL), where we noted that existing algorithms do not encode any notion of controllability within the dynamics learning problem. Consequently, there is no reason to expect the learned model to satisfy properties such as controllability or stabilizability. To address this limitation, we formulate a *semi-supervised* dynamics learning algorithm, regularized using the conditions for stabilizability developed within the work on robust planning. The resulting algorithm generates significantly higher quality dynamics models (in that the resulting generated trajectories were always stabilizable) than traditional regression techniques, especially when given small supervised training datasets. The approach has been validated extensively in simulations and on the quadrotor depicted in Figure 1.2, where, most notably, we demonstrate that with just 150 sampled tuples of dynamics state-transition supervised examples, we are able to stably track a challenging test trajectory, which is generated with the learned model and substantially different from any of the training data. In contrast, a model learned using traditional regression techniques leads to consistently unstable behavior and eventual failure as the quadrotor repeatedly flips out of control and crashes.

Relevant Publications:

Singh S, Sindhwani V, Slotine JJE and Pavone M (2018d) Learning stabilizable dynamical systems via control contraction metrics. In: *Workshop on Algorithmic Foundations of Robotics*.

Singh S, Richards SM, Sindhwani V, Slotine JJE and Pavone M (2019b) Learning stabilizable nonlinear dynamics with contraction-based regularization. *Int. Journal of Robotics Research* Submitted.

1.2 Risk-Sensitive Decision-Making

While the fast time-scale planning work addresses uncertainty at the trajectory optimization level, the work on risk-sensitive optimization focuses on higher-level decision-making, where actions and uncertainty correspond to various *modes*. The notion of risk sensitivity originated within the operations research and human psychology fields, motivated by two independent observations. First, for non-trivial uncertainty representations (e.g., non-Gaussian), traditional measures such as expected

cost or variance can lead to highly illogical decisions when used as a basis for decision-making. Second, decades of research in human behavior suggests that humans behave in a manner that is inconsistent with expected utility model, especially in scenarios involving risk or ambiguity (when probabilities are uncertain). Accordingly, the field of risk-sensitive decision-making was born, which entails using alternative statistical measures of performance (e.g., Conditional Value-at-Risk) instead of expected value or Markowitz-inspired objectives. As these measures are "nonlinear in probabilities" (as opposed to expected reward models that are linear), one obtains highly non-trivial stochastic optimization problems. This thesis' contributions to the field focus on the complementary topics of *risk-sensitive inference* and *decision-making* for robots operating in safety-critical scenarios, such as those involving human agents.

In particular, we present the *first* set of risk-sensitive inverse RL algorithms in which human agents are modeled as acting optimally according to an unknown cost *and* risk measure, both of which are inferred from their demonstrations. We developed both non- and semi-parametric algorithms which were evaluated using human-in-the-loop testing in a realistic driving simulator, and shown to vastly outperform traditional inverse RL algorithms in prediction accuracy. The second contribution (for brevity, not presented within this thesis) is a tractable, unifying (in the sense that it captured a full range of risk preferences from risk-neutral to worst-case analysis) risk-sensitive Model Predictive Control (MPC) algorithm to allow robots to use risk-sensitive measures as a basis for decision-making.

Relevant Publications:

> Majumdar A, Singh S, Mandlekar A and Pavone M (2017) Risk-sensitive inverse reinforcement learning via coherent risk models. In: *Robotics: Science and Systems*.
>
> Singh S, Lacotte J, Majumdar A and Pavone M (2018c) Risk-sensitive inverse reinforcement learning via semi- and non-parametric methods. *Int. Journal of Robotics Research* 37(13): 1713–1740.
>
> Singh S, Chow YL, Majumdar A and Pavone M (2018b) A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms. *IEEE Transactions on Automatic Control* 64(7): 2905–2912. Extended version available at: `http://arxiv.org/abs/1703.01029`.

1.3 Organization

This thesis is organized into three parts. Part I: *Robust Motion Planning*, spanning Chapters 2 – 6, details the contributions made towards robust real-time motion planning. Part II: *Control-Theoretic Regularization for Model-Based RL*, spanning Chapters 7 – 9, demonstrates how to embed the nonlinear control tools developed within the first part of the thesis within a novel dynamics learning

algorithm. Part III: *Risk-Sensitive Decision-Making*, spanning Chapters 10 – 12, illustrates the use of risk-sensitive optimization theory to develop a new class of control and inference algorithms that go beyond traditional statistical measures of performance. The contributions of each chapter are summarized as follows.

Chapter 2 formalizes the robust real-time motion planning problem and provides an overview of the state-of-the-art within this field. Additionally, we provide a brief introduction to semi-infinite programming and sum-of-squares programming, a key computational tool featured heavily within Part I of the thesis.

Chapter 3 introduces the foundations of stability analysis and control synthesis using contraction theory, specifically, control contraction metrics. We present proofs for bounds on exponential trajectory stabilizability and disturbance rejection using various tools from differential geometry and calculus of variations. Finally, we outline an *offline* quasiconvex optimization routine for synthesizing controllers that are optimized for robust tracking performance.

Chapter 4 addresses the *online* aspects of our robust motion planning framework. Specifically, we demonstrate how to implement a contraction-derived feedback controller online, derive bounds on the control effort, and demonstrate how to embed the feedback controller into any existing motion planner to obtain desired robustness guarantees. We illustrate the approach in simulation to validate the fundamentals of all parts of the algorithm.

Chapter 5 presents an extended case-study of a more challenging dynamics example – a 3D quadrotor. We present several numerical studies of the planning algorithm and provide a thorough empirical investigation into the conservatism associated with the computed bounds for contraction-derived controllers. Furthermore, we implement the entire robust planning pipeline on a quadrotor subject to aerodynamic disturbances stemming from unmodeled drag forces, and validate the theoretical performance bounds.

Chapter 6 addresses the challenge of planning for complex nonlinear systems in real-time by formalizing the model-mismatch paradigm. We present a similar hybrid offline/online algorithmic framework where, in the offline stage, we optimize "tracking controllers" for the full robotic system to track motion plans generated using simplified lower-order models, and derive bounds on the tracking error. In the online stage, we illustrate how such bounds and tracking controllers may be leveraged to generate plans for the full robotic system in real-time, with guaranteed collision-free properties. In essence, the contributions of this chapter form the "motion-planner" core of the entire robust planning pipeline.

Chapter 7 formalizes the "planning with unknown dynamics" problem, introduces the model-based RL solution methodology, and provides an overview of the state-of-the-art in model-based RL.

Chapter 8 presents a motivating example for the need for stabilizability-constrained dynamics learning and introduces a formulation based upon the contraction-theoretic tools developed in Part

I of the thesis. Leveraging novel results pertaining to learning in Reproducing Kernel Hilbert spaces, we derive the form of the optimal solution to the stabilizable dynamics learning problem and present a tractable finite-dimensional parameterization.

Chapter 9 outlines a bi-convex iterative algorithm for solving the stabilizable learning dynamics problem and presents an extensive numerical study into its convergence properties. The entire learning pipeline is then validated on a quadrotor testbed where we illustrate the efficacy of the stabilizability-constrained model-based RL approach and quantify its performance improvements over a traditional dynamics learning method.

Chapter 10 formalizes the inverse reinforcement learning (IRL) problem and motivates the need for incorporating risk-sensitivity as a means of capturing more nuanced decision models in the face of stochastic uncertainty. We introduce the necessary theory pertaining to risk measures for both static and dynamic decision-making, focusing in particular on a specific class of risk measures known as coherent risk measures (CRMs), with favorable axiomatic and computational properties.

Chapter 11 presents a risk-sensitive IRL (RS-IRL) algorithm for static, i.e., single-step decision-making. We formulate the human agent's decision problem using CRMs and present linear programming-based algorithms for inferring both the cost function and risk preferences of the agent, given a dataset of demonstrated state-control pairs. A proof of convergence for the predictions is presented for the case where only the risk preferences of the agent are unknown, and synthetic simulations are presented to validate the efficacy of the algorithm.

Chapter 12 extends the RS-IRL methodology to the multi-step dynamic decision-making setting where we are given state-control *trajectories* executed by the human agent. Under a receding-horizon planning assumption for the human agent, we propose semi-parametric maximum likelihood-based algorithms for inferring the agent's cost function and risk preferences. A key feature of the algorithms within this chapter and the preceding chapter on static decision-making, is that there is no a priori assumption on a specific risk measure for the human agent. That is, we are not merely estimating the parameters of a fixed risk measure. Instead, by leveraging certain elegant representation theorems for CRMs, we are able to infer *any* risk measure within the class of all CRMs, thereby illustrating significant modeling flexibility. We validate the proposed algorithms on a high-fidelity commercial driving simulator with 10 human participants in the loop, and demonstrate the need for modeling both a cost function *and* risk measure within the agent's decision-making problem to be able to capture the wide range of driving behaviors observed.

Chapter 13 concludes this thesis, discusses various limitations, and proposes several avenues for future work.

1.4 Additional Contributions

Additional contributions made during the PhD that are not discussed within this thesis are listed below:

(Singh et al., 2015b) presents a class of decentralized formation control algorithms for multi-robot teams inspired by *cyclic control*, and leverages contraction theory in order to prove global exponential stability for any desired 2D or 3D shape.

(Singh et al., 2015a) details the development of a high-fidelity simulation environment and H_∞-based control synthesis for a drag-free microsatellite.

(Wang et al., 2018) proposes a distributed optimization-based control algorithm for cooperative object transport using a collection of quadrotors that are not allowed to communicate with each other.

(Lorenzetti et al., 2019) demonstrates how to leverage reduced-order dynamic models for real-time MPC of higher-dimensional systems, while still guaranteeing robust constraint satisfaction and stability (i.e., convergence) for the full dynamical system.

Part I

Robust Real-Time Motion Planning

Chapter 2

A Review of Robust Motion Planning

As robotic systems become more pervasive, real-time motion planning becomes increasingly important. In general, motion planning algorithms must find a trade-off among two key challenges: (i) achieving fast computational speed to enable online re-planning for complex system dynamics, and (ii) ensuring formal safety and robustness guarantees in the presence of exogenous disturbances and unmodeled effects. In this chapter, we introduce the robust motion planning problem, review the state-of-the-art for addressing these two challenges, and situate our contributions within the respective fields. Additionally, we provide a short introduction to semi-infinite optimization and *sum-of-squares programming*, a computational tool that is featured heavily within the first part of this thesis.

2.1 Problem Statement

We consider robotic systems whose dynamics are described by the nonlinear differential equation:

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

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the control input, and $w(t) \in \mathbb{R}^{n_w}$ is the disturbance. The function $f : \mathbb{R}^n \to \mathbb{R}^n$ captures the *drift*, $B : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the input matrix mapping, depicted in column-stacked form as $(b_1, \ldots, b_m)$, and $B_w : \mathbb{R}^n \to \mathbb{R}^{n \times n_w}$ is the disturbance matrix mapping with $\bar{\sigma}(B_w(x)) = 1$ for all x, where $\bar{\sigma}(\cdot)$ denotes the maximum singular value. In other words, B_w simply selects the channels where the disturbance is active. A state-input trajectory satisfying (2.1) is denoted as a pair (x, u).

Additionally, we enforce state constraints (e.g., arising from obstacles in the robot's environment

and physical constraints such as joint limits) and input constraints, that is: $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$ for all t, where $\mathcal{X}$ and $\mathcal{U}$ are defined to be the closures of bounded, open, and connected sets in Euclidean space.

The motion planning problem we wish to address is to find a (possibly non-stationary) policy $\pi : \mathcal{X} \times \mathbb{R} \to \mathcal{U}$ that (i) drives the state x to a compact region $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}$, (ii) satisfies the state and input constraints, and (iii) minimizes a quadratic cost:

$$J(x(t), \pi) := \int_0^{T_{\text{goal}}} 1 + \|\pi(x(t), t)\|_R^2 \, dt,$$

where $\|(\cdot)\|_R := \sqrt{(\cdot)^T R(\cdot)}$ denotes the weighted norm with respect to R, a strictly positive definite matrix, and T_{goal} is the first time $x(t)$ enters $\mathcal{X}_{\text{goal}}$.

2.2 Robust Planning: State-of-the-Art

A key difficulty within robust planning is that uncertainty and disturbances in the dynamics force us to reason about the "funnel" (or tube) of possible outcomes that the disturbances may drive the system to, rather than a single planned trajectory. Consequently, the core challenges for planning translate into the ability to guarantee both *safety* (with respect to constraint satisfaction and collision avoidance) and *performance* (with respect to cost function optimality) for our robotic system in cluttered and possibly dynamically changing environments.

The topic of planning under uncertainty has been approached from two general methodologies within the robotics community. In the first approach, one seeks probabilistic guarantees on safety (e.g., collision probabilities) for a stochastic model of uncertainty. This is elegantly described by the chance-constrained programming framework (Charnes and Cooper, 1959). Typical solution methods generally consider linear systems affected by Gaussian noise (Blackmore et al., 2006, 2011; Ono et al., 2013) or more generally, exploit linear-Gaussian dependencies (Luders et al., 2016). The extension to nonlinear dynamics and/or non-Gaussian noise is inherently difficult; typical methods employ sampling (Monte Carlo) techniques (Janson et al., 2015b; Sun et al., 2015), stochastic Lyapunov theory (Battilotti and De Santis, 2003; Buehler et al., 2016), or Gaussian processes with (probabilistically) robust feedback linearization (Helwa et al., 2019). In the more general case with partial observability and/or noisy sensing, the stochastic formulation is lifted to the *belief space* planning framework (Kaelbling et al., 1998; Kurniawati et al., 2008; Prentice and Roy, 2009; Platt et al., 2012; Agha-mohammadi et al., 2014). This approach has been typically constrained by limitations such as Gaussian belief state assumptions, linear dynamics, and/or small state and action spaces, for instance in the Partially Observable Markov Decision Process (POMDP) framework. Deriving formal guarantees is significantly more challenging and in many cases impossible.

In the second approach, in contrast to the stochastic methodology, one considers *bounded* models

of uncertainty – typically described as the robust planning problem, and is the formulation adopted within this thesis. A fundamental component of solving this problem is based around computing and/or optimizing over reachable sets, or their *over*-approximations. Figure 2.1 depicts a rough classification of various methods for computing bounds on the reachable sets, ranked by the conservativeness of the approximation. The figure is by *no means* exhaustive and additional references are provided following the figure, but it is useful for understanding the overarching *categorization* of various methods, ranging from approximative linear analysis, through Lyapunov-based certificates computed using convex optimization, to formal verification using dynamic programming and differential games; see also the recent review (Bansal et al., 2017b).

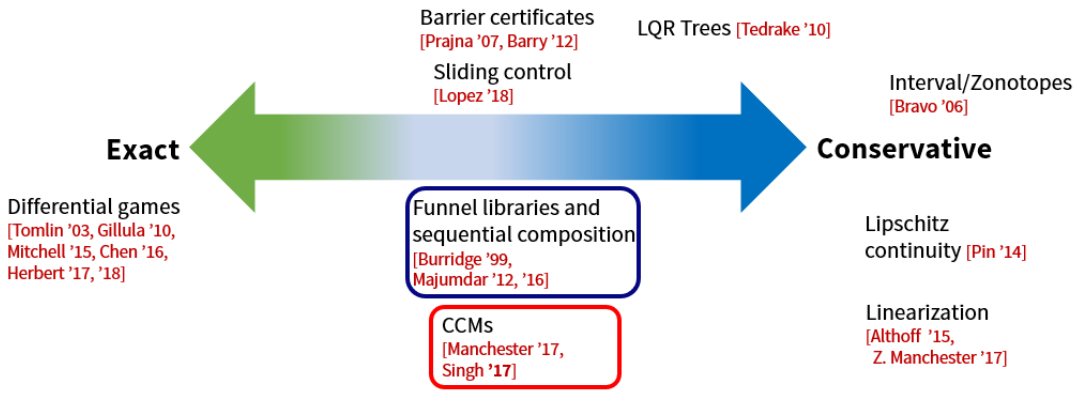


Figure 2.1: An overview of methods for computing exactly or outer-approximating the reachability set for nonlinear systems. The references in the blue-outline box form the fundamental basis for our planning framework, i.e., sequential composition of invariant tubes. The red-outline box indicates the *method* for computing these tubes proposed within this thesis.

On the exact end of the spectrum, one may leverage logic-based methods (e.g., quantifier elimination) to recursively compute the backward reachable set of a given goal set (Raković et al., 2006a; Kong et al., 2015). In a similar vein, the differential game formulation treats any admissable disturbance as an adversarial agent and one can compute the backward reachable set of unsafe sets (e.g., obstacle locations) as the solution to a Hamilton-Jacobi PDE using level-set methods (Tomlin et al., 2003; Gillula et al., 2010; Chen et al., 2016b; Herbert et al., 2017; Fridovich-Keil et al., 2018). While these methods are exact in that they *precisely* characterize a collision-free "roadmap" to the goal set, and in the differential game formulation, also yield the optimal closed-loop controller, implementation of these methods for dynamically changing or unknown environments and/or high-dimensional systems is computationally prohibitive. A computational relaxation in this spirit is the barrier certificates method (Prajna et al., 2007; Barry et al., 2012) in which one characterizes the unsafe regions in the state-space as the zero-superlevel set of a function whose time derivative on the zero-level set is negative for a given controller and all admissible disturbances. While this yields a more conservative sufficient condition, the method is again incompatible with generalizing

to unknown environments discovered in real-time.

As posed within Section 2.1, the motion planning problem entails an optimization over the class of state-feedback functions – a computationally intractable task in general with a solution that is incompatible with changing environments (e.g., obstacle locations). In an effort to reduce computational complexity, the prevailing solution approach, and indeed the strategy adopted within this thesis, is to parameterize general state-feedback policies as a sum of a nominal (open-loop) input u^* and a feedback term designed to track the nominal state trajectory x^* (induced by u^* assuming no disturbances):

$$\pi(x(t),\, t) = u^*(t) + k(x^*(t), x(t)), \tag{2.2}$$

where $k(\cdot,\cdot)$ is the feedback tracking controller. Commonly referred to as *feedback motion planning*, such a solution approach represents a compromise between the general class of state-feedback control laws and a purely open-loop formulation (i.e., no tracking). In order to ensure satisfaction of all constraints in the presence of disturbances, one needs to characterize the reachable set or "funnel" (Burridge et al., 1999), around the trajectory being tracked. Formally, this is defined through the notion of *robust control invariant* (RCI) tubes.

Suppose (x^*, u^*) is a state-input trajectory satisfying the nominal dynamics (i.e., (2.1) with $w \equiv 0$) and (x, u) is a state-input trajectory satisfying (2.1) under the action of a parameterized policy (2.2). Let T^*_{goal} be the first time $x^*(t)$ enters $\mathcal{X}_{\text{goal}}$. An RCI tube is defined as follows.

Definition 1 (RCI Tube)**.** *Let $\Omega : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be a mapping s.t. $x \in \Omega(x)$ and $\Omega(x)$ is a closed and bounded set for every x. Then, $\Omega(\cdot)$ is an RCI mapping (additionally $\Omega(x)$ is an RCI set centered on x) if there exists a tracking controller $k(x^*, x)$ s.t. if $x(t_0) \in \Omega(x^*(t_0))$, then for all allowable realizations of the disturbance $w(t)$, $x(t) \in \Omega(x^*(t))$ for all $t_0 \le t \le T^*_{\text{goal}}$. Given an RCI mapping $\Omega(\cdot)$, an RCI tube centered on the trajectory $x^*(t)$, $t_0 \le t \le T^*_{\text{goal}}$, is the swept region $\bigcup_{t_0 \le t \le T^*_{\text{goal}}} \Omega(x^*(t))$.*

Intuitively, a tracking controller with an associated RCI tube $\Omega(\cdot)$ guarantees that the state of the system is always "close" to its nominal value $x^*(t)$ (precisely, within set $\Omega(x^*(t))$. Thus, by planning a nominal state-input trajectory satisfying the *tightened* constraints:

$$x^*(\cdot) \in \bar{\mathcal{X}} := \mathcal{X} \ominus \Omega, \tag{2.3a}$$

$$u^*(\cdot) \in \bar{\mathcal{U}} := \{\bar{u} \in \mathcal{U} : \forall x^*(t) \in \bar{\mathcal{X}}, \forall x(t) \in \mathcal{X} \text{ s.t. } x(t) \in \Omega(x^*(t)),\, \bar{u} + k(x^*(t), x(t)) \in \mathcal{U}\}, \tag{2.3b}$$

where $\ominus$ denotes the Minkowski set difference, one can ensure that the robotic system will safely reach the goal region $\mathcal{X}_{\text{goal}}$ (modulo the size of the RCI set) in the presence of disturbances. Note that one can ensure that the system reaches $\mathcal{X}_{\text{goal}}$ (without the extra buffer from the RCI set) by constraining the RCI set to be contained within $\mathcal{X}_{\text{goal}}$ at T^*_{goal}. Constraint (2.3a) ensures that the RCI tube around the trajectory does not intersect any obstacles, while (2.3b) is defined for a given tracking controller and ensures that the net applied control satisfies the input limit.

For fully-actuated (i.e., feedback linearizable) systems, RCI tubes or funnels may be computed and optimized using sliding control (Slotine, 2007; Lopez et al., 2018). More conservative approximations of these sets may be found using *linear* reachability analysis, where one computes continuous linearizations of the dynamics about reference trajectories and treats nonlinearities as bounded disturbances (Althoff and Dolan, 2014; Althoff et al., 2015). Alternatively, by bounding the dynamics' Lipschitz constant or Jacobian, one obtains an exponentially growing outer approximation of the reachable set. For instance, in (Pin et al., 2009), a bound on the global Lipschitz constant is used to compute sequentially tightened constraint sets for a reference trajectory to ensure robust constraint satisfaction, while (Bravo et al., 2006) presents a general framework for computing outer approximations of reachable sets using zonotopes, which, while possessing favorable computational properties, again yield overly conservative approximations. The differential inequality approach in (Scott and Barton, 2013; Villanueva et al., 2017) attempts to alleviate the conservativeness resulting from such linear methods through online co-optimization of the reference trajectory and its associated reachable set. In similar spirit, (Manchester and Kuindersma, 2017, 2019) leverage a measure of the size of the approximate invariant funnels computed using linear analysis (i.e., propagation of ellipsoids under linearized dynamics) within the formulation of the cost function for the nominal trajectory itself. Ultimately, however, these methods cannot be applied for real-time re-computation for fast robotic systems. In general, treating nonlinearities as bounded disturbances naturally leads to overly conservative approximations.

Recently, convex programming-based verification methods such as sum-of-squares (SOS) programming have gained increasing popularity in feedback motion planning. For instance, the LQR-Trees algorithm (Tedrake et al., 2010) constructs a tree of locally LQR feedback controllers, however, it cannot handle scenarios in which the task and environment are unknown until runtime. Recently, the *funnel library* approach (Majumdar and Tedrake, 2012, 2017) has been proposed to handle online geometric constraints (e.g., obstacles) that force the robot to re-plan in real-time. The approach leverages SOS programming to compute, offline, a library of funnels around a set of nominal trajectories in which the state is *guaranteed* to remain despite bounded disturbances. These funnels are then sequentially composed online to avoid obstacles. However, this approach is restricted to employing a *fixed set* of trajectories computed offline. While the richness of the funnel library may be increased by exploiting invariances in the dynamics (Majumdar and Tedrake, 2017) or pre-computing a family of funnels parameterized by shifts to a nominal trajectory (Majumdar et al., 2012), one would ideally like to generate a funnel around *any* nominal trajectory generated online.

The concept of feedback motion planning is also fundamental within Tube Model Predictive Control (TMPC), whereby one computes a tracking feedback (also termed ancillary) controller that keeps the state within an invariant "tube" around the nominal MPC trajectory despite disturbances. TMPC has been studied extensively for linear systems with bounded disturbances or model uncertainties (Langson et al., 2004; Mayne et al., 2005; Limon et al., 2010; Farina and Scattolini, 2012;

Rakovic et al., 2012), and for linear systems with stochastic disturbances (Fleming et al., 2015) (see also the recent review (Mayne, 2014)). The application of TMPC to nonlinear systems is certainly not new, see for instance (Raković, 2009) where the properties of TMPC for nonlinear systems are explored via lifting the analysis to set dynamics and employing the Banach fixed-point theorem. However, the construction of invariant tubes and the design of the associated ancillary controller in the nonlinear setup is significantly more complicated than in the linear case. In (Raković, 2009) for instance, the existence of a stabilizing (nonlinear) ancillary controller (that results in contracting set iterates) is simply assumed, while in (Kögel and Findeisen, 2015), a static linear state feedback ancillary controller is used to stabilize the "linear" component of the nonlinear dynamics and Lipschitz continuity is used to bound the effect of disturbance propagation. Some techniques to construct the ancillary controller and accompanying invariant tube proposed in existing literature include systems with matched nonlinearities and linear ancillary feedback (Raković et al., 2006b); integral sliding mode ancillary feedback with Lipschitz-based reachability analysis (Rubagotti et al., 2011); ellipsoidal invariant tubes constructed using linear-matrix-inequalities (LMI) and bounds on the Lipschitz constant (Yu et al., 2013) or assuming a polytopic linear differential inclusion model for the dynamics (Yu et al., 2010); ancillary-MPC with Lipschitz-based reachability analysis (Mayne et al., 2011); systems with incrementally conic uncertainties stabilized using linear ancillary feedback (Carson III et al., 2013); and linearization with static or time-varying linear ancillary feedback with nonlinearities treated as bounded disturbances (Cannon et al., 2011). As with the robotic planning literature, methods leveraging linearization or Lipschitz constants are inherently overly conservative. The notion of incremental input-to-state stability (δ-ISS) for discrete-time systems was used in (Bayer et al., 2013) to derive the invariant tube as a sublevel set of the associated δ-ISS Lyapunov function, which was assumed to be given. More recently, (Köhler et al., 2019) leverages the assumed *existence* of such a function and implicitly incorporates constraint tightening via constraining the growth of this function (using a nonlinear dynamical representation of the scalar defining the Lyapunov sublevel sets) as part of the MPC optimization problem. Again, these bounding functions are either assumed given, or constructed assuming local linear feedback. In contrast, the work presented herein focuses on the *design and optimization* of the functions themselves (and the associated feedback controllers), subsequently permitting the use of algorithms such as the one presented in (Köhler et al., 2019).

Contribution: Adopting the feedback motion planning paradigm, we propose a framework for planning in the presence of *bounded, additive* disturbances using a hybrid offline/online approach. In the *offline* stage, one synthesizes the structure of a tracking controller which can be efficiently implemented online to guarantee exponential convergence to *any* feasible nominal trajectory in the absence of disturbances (Chapter 3). Additionally, the offline computation yields a fixed-size invariant "tube" (akin to a funnel) that can be centered around *any nominal trajectory* as a guaranteed collision-free envelope in the presence of bounded disturbances. In the *online* phase, when the robot

is faced with obstacles, one uses such a tube as a robustness margin during collision checking, thus yielding nominal trajectories that can be safely executed (Chapter 4). The key idea behind our approach is to leverage *contraction theory* (Lohmiller and Slotine, 1998), a method for analyzing nonlinear systems by studying convergence between *pairs of trajectories*. This makes it particularly well-suited to the problem we consider here since it does not require us to commit to a particular nominal trajectory in order to analyze the stability properties of a feedback controller designed to track it. In particular, we design tracking controllers by using *control contraction metrics* (Manchester and Slotine, 2017), a generalization of control Lyapunov functions that can be computed using convex optimization. Notably, in contrast to the class of techniques that employ *linear* reachability analysis to conservatively approximate funnels/tubes for nonlinear systems by treating nonlinearities as bounded disturbances, our analysis directly reasons about intrinsic nonlinearities in the dynamics and thus has the potential to be less conservative for highly nonlinear systems. We demonstrate our approach through numerical simulations of planar and 3D quadrotors, and hardware results on a quadrotor platform navigating a complex obstacle environment while subject to aerodynamic disturbances (Chapters 4 and 5). The results demonstrate the ability of our approach to jointly balance motion safety and efficiency for non-trivial robotic systems.

2.3 Fast Planning for Complex System Dynamics

While the discussion in the previous section addresses the challenge of handling disturbances, one still needs to be able to compute the nominal state-control trajectory (i.e., ignoring disturbances) at runtime for complex nonlinear systems. In the pursuit of real-time motion planning and high-frequency re-planning, a commonly adopted practice is to compute a trajectory by running a planning algorithm on a simplified, low-dimensional dynamical model, oftentimes just a single-integrator model. A feedback tracking controller that accounts for the full, high-dimensional dynamics of the system is then used to "track" such a trajectory, yielding a state-control trajectory that is dynamically feasible for the full system. We refer to this approach as *planning with model mismatch*. For instance, (Hernandéz et al., 2016) and (Lin and Saripalli, 2014) generate Dubins paths as guiding trajectories for autonomous underwater vehicles and UAVs respectively. This approach usually guarantees fast planning speeds, but guaranteeing safety becomes difficult, due to the mismatch between the model used for planning and the actual dynamics of the robotic system. This approximation can introduce tracking errors between the planned path and the actual trajectory of the system, potentially resulting in safety violations. Conversely, one could compute a collision-free motion plan by directly accounting for the full set of differential constraints of a robotic system (e.g., by using a general-purpose kinodynamic motion planning algorithm (LaValle, 2011)). However, despite significant progress in kinodynamic motion planning (LaValle and Kuffner, 2001; Karaman and Frazzoli, 2011; Janson et al., 2015a; Li et al., 2016; Bonalli et al., 2019), including recent contributions such

as learning efficient sampling distributions (Ichter et al., 2018), and "multi-layered" planning where fast geometric planners are used to bias the search for a full kinodynamic planner (Vidal et al., 2019), trajectory planning with full differential constraints is still computationally intensive. For robots characterized by "slow" or differentially flat dynamics, gradient-based trajectory optimization techniques, e.g., (Ratliff et al., 2009; Schulman et al., 2013; Richter et al., 2016), may provide quick convergence to a feasible solution, but extending such methods to systems with complex dynamics is challenging.

An alternative approach is to leverage a set of precomputed maneuvers and sequence them online to form a global path. Formally captured under the notion of a *maneuver automaton*, e.g., see (Frazzoli et al., 2005), trajectory libraries have been leveraged within various applications including humanoids (Liu and Atkeson, 2009), grasping (Berenson et al., 2007), and UAV navigation (Barry, 2016). Sampling-based planning on state-lattices – an unbounded graph generated using precomputed primitives, espouses a similar methodology (Likhachev and Ferguson, 2009; Heng et al., 2015). As discussed earlier, when augmented with feedback tracking controllers, the library of trajectories is transformed into a library of "funnels" (Burridge et al., 1999), where the key idea is to ensure that the trajectory of a system remains within precomputed regions of attraction that are sequenced together at runtime (Tedrake et al., 2010; Majumdar and Tedrake, 2013; Majumdar et al., 2013). However, while able to provide formal guarantees around nominal trajectories, these methods are less suitable for *a priori* unknown environments in which safe trajectories must be found online.

To tackle real-time motion planning problems, recent works such as (Chen et al., 2018a; Herbert et al., 2017; Kousik et al., 2017) combine low- and high-fidelity models to strike a balance between computational speed and model accuracy. In (Kousik et al., 2017), a forward reachable set for the high-fidelity model is computed offline and then used to prune trajectories generated online using the low-fidelity model. The approach relies on an *assumed* model mismatch bound, expressed as the maximum distance between the low- and high-fidelity models under a certain metric. FaSTrack (Chen et al., 2018a; Herbert et al., 2017) considers a slightly different definition of model mismatch, and casts a motion planning problem as a differential game wherein a low-fidelity "planning" system is being pursued (i.e., tracked) by a high-fidelity "tracking" system. Hamilton-Jacobi (HJ) reachability analysis is used to precompute a worst-case tracking error bound (TEB) as well as the optimal feedback tracking controller. Online, FaSTrack plans in real time using the low-fidelity planning system and the TEB as a collision margin, while simultaneously tracking the plan and remaining within *guaranteed* tracking error bounds – despite the model mismatch. FaSTrack's ability to ensure real-time planning of dynamic systems with safety guarantees makes it a desirable framework, but it requires a significant precomputation step in computing the TEB and controller. Executing this precomputation using HJ reachability is reliable and straightforward for nonlinear, highly-coupled dynamical systems with up to five states, or higher-dimensional systems of a specific form (Chen et al., 2018b). However, the computations required for HJ reachability analysis scale exponentially

with the number of states, thus making FaSTrack difficult to apply to systems with a large number of states and/or highly-coupled dynamics.

Contribution: Building upon FaSTrack, we present an approach to designing a feedback tracking controller along with guaranteed tracking error bounds via SOS programming (Chapter 6). SOS programs can be cast as semidefinite programs (SDPs), which may be solved using standard convex optimization toolboxes, and have been used extensively to provide convex relaxations to fundamentally non-convex or even NP-hard problems arising in robust control (Tedrake et al., 2010; Majumdar and Tedrake, 2013; Majumdar et al., 2013; Majumdar and Tedrake, 2017; Posa et al., 2017); see Section 2.4 for an overview of SOS programming. In context of this work, we leverage SOS programming to derive *sufficient* conditions for bounding tracking control error. The price paid for such a convex relaxation is in the tightness of the error bound guarantee (which corresponds to the size/computational tractability of the SOS SDP), but critically not in the existence of the guarantee itself provided the SOS program is feasible. Thus, SOS programming represents an attractive alternative to exact HJ reachability for those robotic systems where HJ is not directly applicable due to computational intractability. The algorithm is demonstrated via numerical experiments, with an emphasis on investigating the trade-off between the increased computational scalability afforded by SOS and its intrinsic conservativeness. Collectively, our results enable scaling the appealing strategy of planning with model mismatch to systems that are beyond the reach of HJ analysis, while maintaining safety guarantees. Additionally, these contributions, as well as FaSTrack (Chen et al., 2018a; Herbert et al., 2017), have potential to complement works such as (Kousik et al., 2017) that assume a model-mismatch error, by providing the TEB as well as a feedback tracking controller to achieve such a TEB.

2.4 Semi-Infinite Optimization and SOS Programming

A form of optimization that underpins the entire first part of this thesis is semi-infinite programming (SIP). In its most general form, a SIP problem can be written as the following optimization:

$$\begin{aligned} \min_{\theta \in \Theta} \quad & J(\theta) \\ \text{s.t.} \quad & g(s, \theta) \geq 0 \quad \forall s \in \mathcal{S}, \end{aligned} \tag{2.4}$$

where Θ is a compact subset of $\mathbb{R}^n$, $\mathcal{S}$ (often termed the *index set*) is a compact subset of $\mathbb{R}^d$, $J(\cdot)$ is the objective function, and $g : \Theta \times \mathcal{S} \to \mathbb{R}^m$ is a vector-valued function defined over the variable and index sets. Optimization problems in this form are termed *semi-infinite* since the variable to be optimized over is finite-dimensional (belonging to $\mathbb{R}^n$), while there are an infinite number of constraints. That is, the constraint in problem (2.4) represents a functional constraint. SIP problems

have been studied extensively within the literature and have appeared in numerous application domains such as motion planning (Hauser, 2018), feedback control design (Majumdar et al., 2014), and digital filter design (Zhu et al., 2000). The presence of the functional constraint makes solving such problems significantly more challenging than traditional finite-dimensional optimization. Solution techniques proposed include cutting plane methods (Kortanek and No, 1992; Mehrotra and Papp, 2014), adaptive convexification (Stein and Steuermann, 2012), sampling/exchange methods (Zhang et al., 2010), penalty and smoothing (Auslender et al., 2009), and homotopy methods (Liu, 2007). See also the reviews in (Hettich and Kortanek, 1993; López and Still, 2007; Goberna and López, 2018).

Of particular interest within this thesis are SIP problems where the function g is *affine* in θ and *polynomial* in s, and $\mathcal{S}$ is a *basic semialgebraic set*. Formally, a basic semialgebraic set $\mathcal{S}$ is a subset of the Euclidean space characterized by a finite set of polynomial inequalities and equalities, that is,

$$\mathcal{S} := \{s \in \mathbb{R}^d : \phi_i(s) \geq 0,\ \psi_j(s) = 0\}, \tag{2.5}$$

where $\{\phi_i\}, \{\psi_j\}$ are finite sets of multivariate polynomials in s. For simplicity, suppose g is scalar-valued and for any given θ, let g_θ denote the multivariate polynomial $g(\cdot, \theta) : \mathcal{S} \to \mathbb{R}$. Then, verifying nonnegativity of the polynomial g_θ over the set $\mathcal{S}$ is already an NP-hard task (Parrilo, 2000). SOS programming provides a convex relaxation approach to accomplish this, through the use of SDPs. For completeness, we briefly define the general form of SDPs before proceeding with our discussion on SOS programming.

Let $\mathbb{S}_d$ denote the set of symmetric matrices in $\mathbb{R}^{d\times d}$, and $\mathbb{S}_d^{\geq 0}$ the cone of positive semi-definite matrices in $\mathbb{R}^{d\times d}$. A matrix X in $\mathbb{S}_d$ is positive semi-definite (psd), if $s^T X s \geq 0$ for all $s \in \mathbb{R}^d, s \neq 0$, and is denoted as $X \succeq 0$. An SDP in primal standard form is written as:

$$\begin{aligned} \min_{X \in \mathbb{S}_d^{\geq 0}} \quad & \mathrm{Tr}(C^\top X) \\ \text{s.t.} \quad & A_i^\top X = b_i, \quad i = 1, 2, ..., m, \\ & X \succeq 0, \end{aligned} \tag{2.6}$$

where C and $\{A_i\}_i$ are elements of $\mathbb{S}_d$, and "Tr" denotes the trace operator. Various interior-point solvers are available for solving SDPs expressed in the form above, e.g., Mosek (ApS, 2017).

SOS programs provide a means of certifying nonnegativity of polynomials either globally or over basic semialgebraic sets. Specifically, a polynomial p is termed SOS if it can be written in the form $\sum_{k=1}^N z_k^2$ for some other polynomials $\{z_k\}_{k=1}^N$. While such a decomposition is not necessary, it is sufficient to guarantee (global) nonnegativity of p. Moreover, if one can find a set of SOS polynomials

$\{L_i\}$ and ordinary polynomials $\{q_j\}$ such that

$$p := g_\theta - \sum_i L_i \phi_i + \sum_j q_j \psi_j \quad \text{is SOS}, \tag{2.7}$$

then one obtains a *certificate* of nonnegativity of g_θ over $\mathcal{S}$. Indeed, as a consequence of p being SOS, for any s where $\phi_i(s) \geq 0$ and $\psi_j(s) = 0$, i.e., $s \in \mathcal{S}$, one has that $g_\theta(s) \geq \sum_i L_i(s)\phi_i(s) \geq 0$, as required. Such a certificate is the extension of the generalized S-procedure (Iwasaki and Hara, 2005) to the setting of real-valued polynomials (Parrilo, 2000), and additionally constitutes a necessary condition for a subclass of semialgebraic sets (Putinar, 1993). More complex necessary and sufficient conditions for verifying nonnegativity over any semialgebraic set using SOS decompositions also exist, and leverage the Stengle Positivstellensatz involving products of ϕ_i (Parrilo, 2000). For our purposes, however, the certificate of nonnegativity as per equation (2.7) will be sufficient.

The computational advantage of SOS programming stems from its intrinsic link to SDPs. Specifically, a polynomial p of degree $2d$ is SOS if and only if $p = z^T Q z$, where $Q \succeq 0$ and z is a vector of monomials up to order d. Thus, certifying that a polynomial is SOS reduces to the task of finding a psd matrix Q subject to a finite set of linear equalities, thus taking the form of the constraints in (2.6). Returning to our original SIP problem in (2.4), for the case where each component of g is affine in θ and polynomial in s, and $\mathcal{S}$ is semi-algebraic, the SOS relaxation of the problem is obtained by replacing each component of the functional constraint with an SOS constraint of the form in (2.7), and augmenting the set of optimization variables with the coefficients of the unknown polynomials $\{L_i\}$ and $\{q_j\}$ for each SOS constraint. For a more detailed review of SOS programming and its applications, please refer to (Parrilo, 2000; Ahmadi and Majumdar, 2016; Majumdar and Tedrake, 2017).

2.5 Summary

Common to the solution strategies proposed for fast planning with model-mismatch and robust planning with contraction theory, is the use of a *nominal planner* and a *feedback controller* with *invariance guarantees*. Within the model mismatch setting, the nominal planner is a simple, yet fast, real-time compatible planning technique leveraging a lower-order approximation of the full nonlinear dynamics. The feedback controller is used to "track" the trajectory generated by this planner, in the process, generating a state and control trajectory that is both (i) dynamically feasible with respect to the full nonlinear dynamics of the robot, and (ii) collision-free, since the nominal plan was generated using the invariance guarantee (provided by the feedback controller) as a safety margin.

Within the robust planning context, the nominal planner represents *any motion planner* capable of generating dynamically feasible trajectories for the full robot dynamics. For instance, these

trajectories may be generated using any of the techniques discussed within Section 2.3. The feedback controller is used on the actual robot to track these trajectories in the presence of bounded disturbances. This execution is also guaranteed to be collision-free, courtesy of the safety margin (derived from the invariance guarantee provided by the feedback controller) used when generating the dynamically feasible trajectory.

Additionally, there are three key advantages that are common to both contributions. First, by explicitly enforcing safety in the online planning process, our approach is particularly suited to planning in previously unseen and tightly-constrained environments, where it might be difficult to find feasible solutions by sequencing a pre-computed set of maneuvers. Second, we allow for both the *design and optimization* of invariant tubes and tracking feedback controllers. Third, the use of convex optimization, specifically, SOS programming, carries a smaller computational burden than differential game formulations for computing reachable tubes, which require numerical solutions to PDEs.

Chapter 3

Trajectory Tracking with Contraction Theory

In this chapter, we demonstrate how to derive trajectory tracking controllers with (i) exponential convergence properties in the absence of disturbances, and (ii) strong boundedness properties in the presence of *bounded* disturbances, by leveraging contraction theory. In particular, this chapter is concerned with the task of *robust nonlinear feedback control design*. The integration of the results of this chapter into a feedback planning algorithm will be the focus of Chapters 4 and 5. In particular, we leverage recent advances in contraction theory for control design through the use of *control contraction metrics* (CCM) (Manchester and Slotine, 2017), and make the following two contributions. First, on the theoretical side, while our CCM approach is directly inspired by (Manchester and Slotine, 2017), we present an alternative proof of incremental exponential stability using a suitable CCM-derived controller. This proof employs techniques from calculus of variations and differential geometry in order to (i) derive a tighter characterization of the controller's disturbance rejection properties and the size of the corresponding invariant tube, and (ii) simplify the online implementation of the controller, under significantly weaker conditions for the CCM as compared to (Manchester and Slotine, 2017) and (Zamani et al., 2013). Second, on the computational side, we formulate an *offline* quasiconvex optimization using SOS programming that searches for an *optimal* CCM with minimal cross-section of the RCI tube, and present two representative synthesis examples.

Notation: Let $\mathbb{S}_j$ be the set of symmetric matrices in $\mathbb{R}^{j\times j}$ and denote $\mathbb{S}_j^{\geq 0}$, respectively $\mathbb{S}_j^{>0}$, to be the set of symmetric positive semi-definite, respectively positive definite matrices in $\mathbb{R}^{j\times j}$. Given a matrix X, let $\widehat{X} := X + X^T$. The set of $\mathcal{C}^2$ functions from $\mathcal{D}$ to $\mathcal{R}$ is denoted by $\mathcal{C}^2(\mathcal{D},\mathcal{R})$. We denote the components of a vector $y \in \mathbb{R}^n$ as y^j, $j = 1,\ldots,n$, and its Euclidean norm as $\|y\|$. Let $\|y\|_A = \sqrt{y^T A y}$ denote a weighted norm with respect to some $A \in \mathbb{S}_n^{>0}$. Let $(\overline{\sigma}(A), \underline{\sigma}(A))$ denote the maximum and minimum singular values of a matrix A, and $(\bar{\lambda}(A), \underline{\lambda}(A))$ the maximum

and minimum eigenvalues of A. Finally, let $\partial_y F(x)$ denote the Lie derivative of the matrix-valued function F at x along the vector y.

3.1 Contraction Theory

Contraction theory (Lohmiller and Slotine, 1998) is a method of analyzing nonlinear systems in a differential framework, i.e., via the associated variational system (Crouch and van der Schaft, 1987, Chp 3), and is focused on the study of convergence between pairs of state trajectories towards each other. Thus, at its core, contraction explores a stronger notion of stability – that of incremental stability between solution trajectories, instead of the stability of an equilibrium point or invariant set. While the analysis in (Lohmiller and Slotine, 1998) and most other works on contraction theory focus on analyzing the stability of closed-loop vector fields – (Jouffroy, 2003; Sontag, 2010; Sontag et al., 2014; Simpson-Porco and Bullo, 2014; Forni and Sepulchre, 2014); see also the recent review in (Aminzarey and Sontag, 2014) and references therein), recent results demonstrate the applicability of contraction theory for *constructive control design*, e.g., control via backstepping (Sharma and Kar, 2009; Zamani et al., 2013), control for singularly perturbed systems using multiple timescales (Rayguru and Kar, 2015), and control via CCMs (Manchester et al., 2015; Manchester and Slotine, 2017; Singh et al., 2017). The concept of δ-ISS has also been studied within a contraction theory framework, e.g., in (Zamani et al., 2013), where contraction metrics are derived for a class of nonlinear systems stabilized using backstepping. Compared to works on establishing incremental stability through suitable δ-ISS Lyapunov functions (Angeli, 2002, 2009), contraction metrics are an *intrinsic* characterization of incremental stability (i.e., coordinate invariant) and the search for a suitable metric and associated stabilizing controller via convex optimization (see Chapter 4) boasts obvious practical benefits.

The core principle behind contraction theory (Lohmiller and Slotine, 1998) is to study the evolution of distance between any two *infinitesimally close* neighboring trajectories and draw conclusions on the distance between any *finitely apart* pair of trajectories. We begin by first introducing the fundamentals of contraction theory, in the absence of control.

3.1.1 Introduction to Contraction

Given an autonomous system of the form: $\dot{x}(t) = f(x(t))$, consider two neighboring trajectories separated by an infinitesimal (virtual) displacement δ_x; formally, δ_x is a vector in the tangent space $\mathcal{T}_x\mathcal{X}$ at x. The dynamics of this virtual displacement are given by:

$$\dot{\delta}_x = \frac{\partial f}{\partial x}\delta_x,$$

where $\partial f/\partial x$ is the Jacobian of f. The dynamics of the infinitesimal squared distance $\delta_x^T \delta_x$ between these two trajectories is then given by:

$$\frac{d}{dt}\left(\delta_x^T \delta_x\right) = 2\delta_x^T \frac{\partial f}{\partial x}\delta_x.$$

Then, if the (symmetric part) of the Jacobian matrix $\partial f/\partial x$ is *uniformly* negative definite, i.e.,

$$\sup_x \bar{\lambda}\left(\frac{1}{2}\widehat{\frac{\partial f(x)}{\partial x}}\right) \leq -\lambda < 0,$$

for some $\lambda \in \mathbb{R}_{>0}$, one has that the squared infinitesimal length $\delta_x^T \delta_x$ is exponentially convergent to zero at rate 2λ. By path integration of δ_x between *any* pair of trajectories, one has that the distance between any two trajectories shrinks exponentially to zero. The vector field f is thereby referred to be *contracting at rate* λ, and λ is referred to as the *contraction rate*.

Contraction metrics generalize this observation by considering as infinitesimal squared length distance, a symmetric positive definite function $V(x, \delta_x) = \delta_x^T M(x)\delta_x$, where $M : \mathcal{X} \to \mathbb{S}_n^{>0}$ is a mapping from $\mathcal{X}$ to the set of uniformly positive definite $n \times n$ symmetric matrices. Formally, $M(x)$ may be interpreted as a Riemannian metric tensor, endowing the space $\mathcal{X}$ with the Riemannian squared length element $V(x, \delta_x)$. A fundamental result in contraction theory (Lohmiller and Slotine, 1998) is that *any* contracting system admits a contraction metric $M(x)$ such that the associated function $V(x, \delta_x)$ satisfies:

$$\dot{V}(x, \delta_x) \leq -2\lambda V(x, \delta_x), \quad \forall (x, \delta_x) \in \mathcal{T}\mathcal{X},$$

for some positive contraction rate λ. Thus, the function $V(x, \delta_x)$ may be interpreted as a *differential Lyapunov function*; see also (Forni and Sepulchre, 2014) for such an analogy, albeit using Finsler metrics.

3.1.2 Control Contraction Metrics

Control contraction metrics (CCMs) generalize contraction analysis to the controlled dynamical setting, in the sense that the analysis searches *jointly* for a controller design and the metric that describes the contraction properties of the resulting closed-loop system. To introduce this concept, we first define the notion of incremental exponential stabilizability.

Definition 2 (Incremental Exponential Stability)**.** *Consider a nominal state-input trajectory pair* $(x^*(t), u^*(t))$ *for the unperturbed controlled dynamics (i.e.,* (2.1) *with* $w \equiv 0$*). Suppose there exist* $\lambda, C > 0$ *and a feedback controller* $k(x^*(t), x(t))$ *such that the resulting state trajectory* $x(t)$ *for the*

unperturbed dynamics with control $u(x(t)) = u^*(t) + k(x^*(t), x(t))$ *satisfies*

$$\|x^*(t) - x(t)\| \leq Ce^{-\lambda t}\|x^*(0) - x(0)\|. \tag{3.1}$$

Then, the trajectory $x^*(t)$ *is said to be incrementally exponentially stabilizable (IES) with rate* λ *and overshoot constant* C*.*

We now illustrate how to leverage contraction theory to derive trajectory tracking controllers that guarantee IES for any nominal state trajectory $x^*(t)$. Denote the tangent space of $\mathcal{X}$ at $x \in \mathcal{X}$ by $T_x\mathcal{X}$[1] and the tangent bundle of $\mathcal{X}$ by $T\mathcal{X} = \bigcup_{x\in\mathcal{X}}\{x\} \times T_x\mathcal{X}$. The variational dynamics (i.e., dynamics of the virtual displacement δ_x along any nominal state-input trajectory $(x(t), u(t))$) for the unperturbed controlled system are given by (Crouch and van der Schaft, 1987, Chp 3):

$$\dot{\delta}_x = \overbrace{\left(\frac{\partial f(x)}{\partial x} + \sum_{j=1}^{m} u^j \frac{\partial b_j(x)}{\partial x}\right)}^{:=A(x,u)} \delta_x + B(x)\delta_u, \tag{3.2}$$

where $\delta_x \in T_x\mathcal{X}$ is a tangent vector to a smooth path of states at $x \in \mathcal{X}$, and $\delta_u \in T_u\mathcal{U}$ is a tangent vector to a smooth path of controls at $u \in \mathcal{U}$. Let $M : \mathcal{X} \to \mathbb{S}_n^{>0}$ be a smooth matrix function that is uniformly bounded (i.e., there exist constants $0 < \underline{\alpha} < \overline{\alpha}$ such that $\underline{\alpha} I_n \preceq M(x) \preceq \overline{\alpha} I_n$). Continuing our interpretation of $V(x, \delta_x) = \delta_x^T M(x)\delta_x$ as a Riemannian squared differential length element, for a given smooth curve $c : [0, 1] \to \mathcal{X}$, we define its length $l(c)$ and energy $\mathcal{E}(c)$ as $l(c) := \int_0^1 \sqrt{V(c(s), c_s(s))}ds$, $\mathcal{E}(c) := \int_0^1 V(c(s), c_s(s))ds$, where $c_s(s) = \partial c(s)/\partial s$.

Let $\Gamma(p, q)$ be the set of smooth curves on $\mathcal{X}$ that connect points p and q. The Riemann distance between points p and q is defined as the quantity $d(p, q) := \inf_{c\in\Gamma(p,q)} l(c)$, and denote $\mathcal{E}(p, q) := d^2(p, q)$ to be the Riemann energy. Let the curve $\gamma \in \Gamma(p, q)$ be the (possibly non-unique) *minimizing geodesic* which achieves this infimum. Notice that $\mathcal{E}(\gamma) = d^2(p, q)$.

A smooth, uniformly positive definite matrix-valued function $M(x)$ is a CCM for the system $\dot{x} = f(x) + B(x)u$, if there *exists* a *differential* controller $\delta_u : T\mathcal{X} \to T_u\mathcal{U}$, such that $\dot{V}(x, \delta_x) < 0$, $\forall(x, \delta_x) \in T\mathcal{X}$. Thus, $V(x, \delta_x)$ may be interpreted as a *differential control Lyapunov function* (CLF) on the tangent bundle $T\mathcal{X}$. In the next section, we study the necessary and sufficient conditions for the *existence* of CCMs.

3.1.3 Conditions for CCMs

Consider the time-derivative of V along any given state-control trajectory (x, u):

$$\dot{V}(x, \delta_x, u) = \delta_x^T \left(\partial_{f+Bu} M(x) + \widehat{M(x)A(x, u)}\right) \delta_x + 2\delta_x^T M(x)B(x)\delta_u. \tag{3.3}$$

[1]Since $\mathcal{X}$ is the closure of an open set in $\mathbb{R}^n$, the tangent space $T_x\mathcal{X}$ for all x in the interior of $\mathcal{X}$ is simply $\mathbb{R}^n$, while $T_x\mathcal{X}$ on the boundary of $\mathcal{X}$ is a half-space in $\mathbb{R}^n$.

Suppose that the following equality holds for all $x \in \mathcal{X}$:

$$\partial_{b_j} M(x) + \widehat{M(x)\frac{\partial b_j(x)}{\partial x}} = 0, \quad j = 1, \dots, m. \tag{3.4}$$

Condition (3.4) implies that the vectors b_j form a Killing vector field for the metric tensor $M(x)$. Under this condition, $\dot{V}$ reduces to

$$\dot{V}(x, \delta_x, \delta_u) = \delta_x^T \left(\partial_f M(x) + \widehat{M(x)\frac{\partial f(x)}{\partial x}} \right) \delta_x + 2\delta_x^T M(x) B(x) \delta_u. \tag{3.5}$$

Then, if the following property holds for some constant $\lambda > 0$ and all $(x, \delta_x) \in T\mathcal{X}$:

$$\delta_x^T \left(\partial_f M(x) + \widehat{M(x)\frac{\partial f(x)}{\partial x}} \right) \delta_x \leq -2\lambda \delta_x^T M(x) \delta_x \tag{3.6}$$
$$\text{for all } \delta_x \text{ such that } \delta_x^T M(x) B(x) = 0,$$

then (Manchester and Slotine, 2017) shows that there always exists an integrable *differential* feedback controller of the form $\delta_u(x, \delta_x) = K(x)\delta_x$ such that the following inequality holds for all[2] $(x, \delta_x) \in \mathcal{X} \times \mathbb{R}^n$:

$$\begin{aligned} \dot{V}(x, \delta_x) &= \delta_x^T \left(\partial_f M(x) + \widehat{M(x)\frac{\partial f(x)}{\partial x}} + \widehat{M(x)B(x)K(x)} \right) \delta_x \\ &\leq -2\lambda \delta_x^T M(x) \delta_x = -2\lambda V(x, \delta_x). \end{aligned} \tag{3.7}$$

Notice that condition (3.6) simply indicates that for all directions where the *variational* system lacks controllability (given by the nullspace of $B^T(x)M(x)$), the system is naturally contracting with rate λ. In the scenario where (3.4) fails to be true, one may leverage the following weaker alternative to conditions (3.4) and (3.6):

$$\delta_x^T \left(\partial_{f+Bu} M(x) + \widehat{M(x)A(x,u)} \right) \delta_x \leq -2\lambda \delta_x^T M(x) \delta_x \tag{3.8}$$
$$\text{for all } \delta_x \text{ such that } \delta_x^T M(x) B(x) = 0.$$

In (Manchester and Slotine, 2017), it is proven, by construction, that the weaker condition above is still sufficient to guarantee the existence of a differential feedback controller. However, in this case, the differential controller becomes a function of u as well, i.e., $\delta_u = \delta_u(x, \delta_x, u)$.

We now illustrate how to leverage such a differential controller – which guarantees stability on an infinitesimal scale by the property $\dot{V}(x, \delta_x) \leq -2\lambda V(x, \delta_x)$ along any nominal trajectory, to derive

[2]Note that we drop the distinction between $T_x\mathcal{X}$ at the boundary versus the interior and simply assume that $T_x\mathcal{X} = \mathbb{R}^n$.

a stabilizing controller for any pair of finitely-apart trajectories.

3.1.4 Incrementally Stabilizing Controllers

Given a desired nominal state-input trajectory pair $(x^*(t), u^*(t))$, let $\gamma(\cdot, t) : [0, 1] \to \mathcal{X}$ denote a minimizing geodesic connecting $x^*(t)$ and $x(t)$. Consider the following control law:

$$\begin{aligned}\pi(x(t), t) &= u^*(t) + \int_{\gamma(\cdot,t)} \delta_u(\gamma(s,t), \delta_\gamma(s,t))\, ds \\ &= u^*(t) + \underbrace{\int_0^1 K(\gamma(s,t))\delta_\gamma(s,t)\, ds}_{= k(x^*(t),x(t))},\end{aligned} \tag{3.9}$$

where $\delta_\gamma(s,t) := \partial\gamma(s,t)/\partial s$. The geometric interpretation of (3.9) and inequality (3.7) is illustrated in Fig. 3.1.

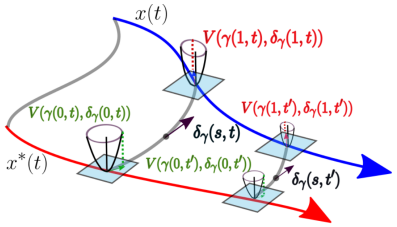


Figure 3.1: Schematic of the differential CLF $V(x, \delta x)$ at the endpoints of the geodesics $\gamma(\cdot, t)$ and $\gamma(\cdot, t')$, at times t and $t' > t$ respectively, and the geodesic velocity vector at the position $s \in (0, 1)$ along the geodesic. The contours of V are shaped according to the metric tensor $M(x)$. The differential controller ensures that at all points along the geodesic, $V(x, \delta_x)$ is shrinking in the direction tangent to the geodesic.

In case the weaker condition (3.8) is used, $\delta_u = \delta_u(x, \delta_x, u)$, and thus $\pi(x(t), t)$ will be given by the solution to the differential equation

$$\pi(x(t), t) = u^*(t) + \int_{\gamma(\cdot,t)} \delta_u(\gamma(s,t), \delta_\gamma(s,t), \pi(\gamma(s,t),t))\, ds, \tag{3.10}$$

where $\delta_u(\cdot)$ is designed such that $\dot{V}(x, \delta_x, u) \leq -2\lambda V(x, \delta_x)$ for all (x, δ_x) along the minimizing geodesic. Theorem 1, which is central to our approach, proves that control law (3.9) ensures the trajectory $x(t)$ is IES with respect to $x^*(t)$ in the sense of Definition 2. The proof differs significantly from the one presented in (Manchester and Slotine, 2017) as it is later adapted in the next section for deriving the RCI mapping.

Theorem 1 (Incrementally stabilizing controller)**.** *Let (x^*, u^*) and (x, u) be state-input trajectory pairs for the nominal dynamics where $\pi(x(t), t)$ is given by* (3.9) *using a CCM which satisfies equations*[3] (3.4) *and* (3.6)*. Then, $x(t)$ is IES with respect to $x^*(t)$ in the sense of Definition 2.*

[3] Alternatively, $\pi(x(t), t)$ is given by (3.10) using a CCM satisfying the weaker condition (3.8).

Proof. Notice that at each time instant t, (3.9) implicitly defines a smooth *virtual parameterized surface* of solutions $c_t : (s, t') \in [0, 1] \times (-\varepsilon, \varepsilon) \mapsto \mathcal{X}$ (where ε is an arbitrarily small positive value), to the nominal dynamics as follows:

$$c_t'(s, t') := \frac{\partial c_t}{\partial t'}(s, t') = f(c_t(s, t')) + B(c_t(s, t'))\pi(c_t(s, t'), t), \quad c_t(\cdot, 0) = \gamma(\cdot, t), \tag{3.11}$$

where

$$\pi(c_t(s, t'), t) = u^*(t + t') + \int_0^s \delta_u(c_t(\mathfrak{s}, t'), \delta_{c_t}(\mathfrak{s}, t'))\, d\mathfrak{s}, \tag{3.12}$$

where $\delta_{c_t}(\mathfrak{s}, t') = \partial c_t(\mathfrak{s}, t')/\partial \mathfrak{s}$. Consider the time derivative of the Riemann energy $\mathcal{E}(x^*(t), x(t))$ between $x^*(t)$ and $x(t)$. If $x(t) \notin \mathrm{Cut}(x^*(t))$ (where $\mathrm{Cut}(x)$ denotes the cut-locus at x), then the derivative of $\mathcal{E}(x^*(t), x(t))$ is well-defined at t (i.e., the exponential map is a local diffeomorphism) and given by (Spivak, 1999, Chp 9):

$$\dot{\mathcal{E}}(x^*(t), x(t)) = \left[\left(\frac{\partial V}{\partial \delta_x}(\gamma(s, t), \delta_\gamma(s, t)) \right)^T \frac{\partial \gamma(s, t)}{\partial t} \right] \Bigg|_{s=0}^{s=1}. \tag{3.13}$$

This result is true for all $t \geq 0$ such that $x(t) \notin \mathrm{Cut}(x^*(t))$. For completeness, suppose now that $x(t) \in \mathrm{Cut}(x^*(t))$, a Lebesgue measure zero set. In this scenario, the minimizing geodesic $\gamma(\cdot, t)$ may not be unique and thus the Riemannian energy $\mathcal{E}(x^*(t), x(t))$ is *not* smooth. However, it can be shown that the upper Dini derivative of $\mathcal{E}(x^*(t), x(t))$, defined as:

$$D^+\mathcal{E}(x^*(t), x(t)) := \limsup_{t' \searrow t} \frac{\mathcal{E}(x^*(t'), x(t')) - \mathcal{E}(x^*(t), x(t))}{t' - t}$$

exists, and satisfies the inequality[4] (Adelstein and Epstein, 2017):

$$D^+\mathcal{E}(x^*(t), x(t)) \leq \left[\left(\frac{\partial V}{\partial \delta_x}(\gamma(s, t), \delta_\gamma(s, t)) \right)^T \frac{\partial \gamma(s, t)}{\partial t} \right] \Bigg|_{s=0}^{s=1}. \tag{3.14}$$

Notice that the parameterized set of solutions in (3.11) satisfies: (i) $c_t(\cdot, 0) = \gamma(\cdot, t)$, (ii) $\delta_{c_t}(0, 0) = \delta_\gamma(0, t)$ and $\delta_{c_t}(1, 0) = \delta_\gamma(1, t)$, and (iii) $\partial\gamma(0, t)/\partial t = c_t'(0, 0) = \dot{x}^*(t)$ and $\partial\gamma(1, t)/\partial t = c_t'(1, 0) = \dot{x}(t)$. Thus, we have the following equality

$$\begin{aligned} \left[\left(\frac{\partial V}{\partial \delta_x}(\gamma(s, t), \delta_\gamma(s, t)) \right)^T \frac{\partial \gamma(s, t)}{\partial t} \right] \Bigg|_{s=0}^{s=1} &= \left[\left(\frac{\partial V}{\partial \delta_x}(c_t(s, 0), \delta_{c_t}(s, 0)) \right)^T c_t'(s, 0) \right] \Bigg|_{s=0}^{s=1} \\ &= \int_0^1 \frac{\partial}{\partial s} \left(\left(\frac{\partial V}{\partial \delta_x}(c_t(s, 0), \delta_{c_t}(s, 0)) \right)^T c_t'(s, 0) \right) ds. \end{aligned} \tag{3.15}$$

[4] In actual fact, it is shown in (Adelstein and Epstein, 2017) that the one-sided derivative of $\mathcal{E}(x^*(t), x(t))$ along any tangent vectors at $x^*(t)$ and $x(t)$ (defined using $\lim_{t' \searrow t}(\cdot)$) exists, from which it follows that the limit $\limsup_{t' \searrow t}(\cdot)$ exists.

Now since $\gamma(\cdot, t)$ (and thus $c_t(\cdot, 0)$) is a geodesic, it satisfies the Euler-Lagrange equation, which in coordinates reads as:

$$\frac{\partial}{\partial s}\left(\frac{\partial V}{\partial \delta_x^i}(c_t(s,0), \delta_{c_t}(s,0))\right) = 2\frac{\partial}{\partial s}\left(\sum_{j=1}^{n} M_{ij}(c_t(s,0))\, \delta_{c_t}^j(s,0)\right) = \delta_{c_t}(s,0)^T \frac{\partial M}{\partial x^i}(c_t(s,0))\delta_{c_t}(s,0),$$

for all $i \in \{1, \dots, n\}$ and $s \in (0,1)$. By definition

$$\frac{\partial c_t'(s,t')}{\partial s} = A(c_t(s,t'), \pi(c_t(s,t'),t))\delta_{c_t}(s,t') + B(c_t(s,t'))\delta_u(s,t'),$$

i.e., the variational dynamics. Leveraging this equality with the Euler-Lagrange equation, one obtains

$$\begin{aligned}
&\frac{\partial}{\partial s}\left(\left(\frac{\partial V}{\partial \delta_x}(c_t(s,0), \delta_{c_t}(s,0))\right)^T c_t'(s,0)\right) \\
&\quad = \delta_{c_t}(s,0)^T \left(\partial_{c_t'(s,0)} M(s,0) + \widehat{M(s,0)A(s,0)}\right)\delta_{c_t}(s,0) + 2\delta_{c_t}(s,0)^T M(s,0)B(s,0)\delta_u(s,0) \\
&\quad = V'(c_t(s,0), \delta_{c_t}(s,0), \delta_u(s,0)) =: \frac{\partial V}{\partial t'}(c_t(s,0), \delta_{c_t}(s,0), \delta_u(s,0)),
\end{aligned}$$

where the last line follows by definition, and we use the abbreviation $(s,0)$ for sake of clarity. Now by deliberate design of δ_u, we have:

$$V'(c_t(s,0), \delta_{c_t}(s,0), \delta_u(s,0)) \leq -2\lambda V(c_t(s,0), \delta_{c_t}(s,0)),$$

for all $s \in [0,1]$. Combining the inequality above and equations (3.14) and (3.15), one obtains the following chain of inequalities:

$$\begin{aligned}
D^+\mathcal{E}(x^*(t), x(t)) &\leq \int_0^1 V'(c_t(s,0), \delta_{c_t}(s,0), \delta_u(s,0))\, ds \\
&\leq -2\lambda \int_0^1 V(c_t(s,0), \delta_{c_t}(s,0))\, ds \\
&= -2\lambda \int_0^1 V(\gamma(s,t), \delta_\gamma(s,t))\, ds = -2\lambda \mathcal{E}(x^*(t), x(t)).
\end{aligned} \tag{3.16}$$

It follows that $\mathcal{E}(x^*(t), x(t)) \leq \mathcal{E}(x^*(0), x(0))e^{-2\lambda t}$. Since $d^2(x^*(t), x(t)) = \mathcal{E}(x^*(t), x(t))$ and the metric tensor $M(x)$ is uniformly bounded, one concludes

$$\|x(t) - x^*(t)\| \leq \sqrt{\frac{\overline{\alpha}}{\underline{\alpha}}}\|x(0) - x^*(0)\|e^{-\lambda t}.$$

Thus, $x^*(t)$ is IES with rate λ and overshoot $\sqrt{\overline{\alpha}/\underline{\alpha}}$. □

The construction of the *virtual* parameterized set of solutions induced by (3.11) is needed as one *cannot* directly reason about the evolution of the minimizing geodesic. Instead, we leverage the Euler-Lagrange equation to relate the time derivative of the Riemann energy between the trajectories $x^*(\cdot)$ and $x(\cdot)$ and the Lie derivative of the metric tensor $M(x)$ with respect to the nominal dynamics. Thus, the Riemannian energy between $x^*(t)$ and $x(t)$ may be viewed as an *incremental* CLF, thereby defining the structure of the feedback controller as *any piecewise continuous (in time) function that satisfies inequality* (3.16). In Chapter 4, we will give the explicit form of such a controller.

Remark 1. *While the interpretation of the energy as an incremental CLF is an observation also made in (Manchester and Slotine, 2017), by showing equivalence between the right hand sides of* (3.14)*,* (3.15)*, and* (3.16) *and using this to establish IES for $x^*(t)$, we are able to: (i) derive the robustness guarantees for controller* (3.9) *under significantly weaker conditions than those necessary in (Manchester and Slotine, 2017); we explore this distinction in greater detail in Section 3.2, and (ii) derive a significantly simpler form of the controller than the integral construction in eq.* (3.9)*; see Chapter 4.*

Remark 2. *It can be shown that conditions* (3.4) *and* (3.6) *are* invariant *under state diffeomorphism and metric pushforward (Manchester and Slotine, 2017). This invariance allows one to relax the topological assumptions on the state space $\mathcal{X}$. In particular, one may take $\mathcal{X}$ to be any embedded smooth n-manifold in $\mathbb{R}^k$ where $k \geq n$, as long as the contraction conditions hold on one (and thus, any) choice of local coordinates in $\mathbb{R}^n$. This highlights the coordinate-free (i.e., intrinsic) nature of CCM-based stabilization.*

3.2 Contraction-Based Tubes

In this section we deduce the disturbance rejection properties of the feedback controller given by (3.9) and derive the resulting RCI mapping for the closed-loop system stabilized with this controller, under the assumption of bounded disturbances. Henceforth, $(x^*(t), u^*(t))$ is assumed to satisfy the unperturbed dynamics, while $x(t)$ denotes the actual state trajectory (i.e., with disturbances) using the control law $u^*(t) + k(x^*, x)$, where $k(x^*, x)$ is a CCM-derived tracking controller.

Theorem 2 (Disturbance Rejection)**.** *Assume there exists a CCM $M(x)$ satisfying conditions* (3.4) *and* (3.6) *that is uniformly bounded, i.e., $\underline{\alpha} I_n \preceq M(x) \preceq \overline{\alpha} I_n$, for all $x \in \mathcal{X}$ where $\underline{\alpha} > 0$. Factorize $M(x)$ as $\Theta(x)^T \Theta(x)$ and define*

$$\overline{\alpha}_w := \sup_{x \in \mathcal{X}} \overline{\sigma}(\Theta(x) B_w(x)).$$

Then, the geodesic energy between trajectories $x(t)$ and $x^*(t)$, i.e., $\mathcal{E}(x^*(t), x(t))$, satisfies the differential inequality:

$$D^+ \mathcal{E}(x^*(t), x(t)) \leq -2\lambda \mathcal{E}(x^*(t), x(t)) + 2\, d\,(x^*(t), x(t))\ \overline{\alpha}_w \left\| w(t) \right\|. \tag{3.17}$$

Proof. Inequality (3.14) implies:

$$\begin{aligned} D^+\mathcal{E}(x^*(t),x(t)) \leq\ & 2\delta_\gamma^T(1,t)M(x(t))\Big[f(x(t))+B(x(t))u(x(t))\Big] \\ & - 2\delta_\gamma^T(0,t)M(x^*(t))\Big[f(x^*(t))+B(x^*(t))u^*(t)\Big] \\ & + 2\delta_\gamma^T(1,t)M(x(t))B_w(x(t))w(t), \end{aligned} \tag{3.18}$$

where $u(x(t))$ is given by (3.9). By the IES property of the nominal system, i.e., inequality (3.16), the expression above can be bounded as:

$$D^+\mathcal{E}(x^*(t),x(t)) \leq -2\lambda\mathcal{E}(x^*(t),x(t)) + 2\delta_\gamma^T(1,t)M(x(t))B_w(x(t))w(t).$$

Defining $\delta_z(s,t) := \Theta(\gamma(s,t))\delta_\gamma(s,t)$, we obtain

$$D^+\mathcal{E}(x^*(t),x(t)) \leq -2\lambda\mathcal{E}(x^*(t),x(t)) + 2\delta_z^T(1,t)\Theta(x(t))B_w(x(t))w(t).$$

Recall that the velocity field of a geodesic is *parallel* along the geodesic (Spivak, 1999) and thus $V(\gamma(s,t),\delta_\gamma(s,t)) = \mathcal{E}(x^*(t),x(t))$ for all $s \in [0,1]$. This implies that $\sqrt{V(\gamma(s,t),\delta_\gamma(s,t))} = \|\delta_z(s,t)\| = d(x^*(t),x(t))$. Using the Cauchy-Schwarz inequality one obtains the stated differential inequality. □

Remark 3. *Notice that the differential inequality derived in* (3.17) *may be integrated in time to yield a δ-ISS statement similar to Definition 2.3 in (Zamani et al., 2013). Specifically, in (Zamani et al., 2013), the authors prove δ-ISS by leveraging a decrescent condition similar to inequality* (3.7) *with respect to tangent vectors δ_w defined on the tangent space of the disturbance manifold $\mathcal{W}$, yielding a differential ISS condition. The key difference here then is by leveraging the separability of the disturbance term (i.e., inequality* (3.18)*), and by using a tracking controller that enforces contraction along the minimizing geodesic at each time (see Fig. 3.1), we do not need to impose similar differential ISS like conditions on $V(x,\delta_x)$. We further note that a similar bound is also proved in (Manchester and Slotine, 2017) but under stronger conditions, which resemble the differential ISS conditions in (Zamani et al., 2013). Indeed, the relaxation of such conditions for Theorem 2 is the primary motivation for our alternative proof strategy for Theorem 1.*

Notice that the equality in (3.17) yields the well known Bernoulli differential equation. Suppose now that the disturbance $w(t)$ is piecewise $\mathcal{C}^1$ and norm-bounded, i.e., there exists $\bar{w}$ such that $\|w(t)\| \leq \bar{w}$, for all $t \geq 0$. Then, if $d(x^*(0),x(0)) \in (0, \overline{\alpha}_w\bar{w}/\lambda]$, it follows by the Comparison Lemma (Khalil, 2002), that the geodesic distance is upper bounded by $\overline{\alpha}_w\bar{w}/\lambda$, for all time $t \geq 0$.

Thus, the RCI mapping may be expressed as:

$$\Omega(x^*) = \{x \in \mathcal{X} : d(x^*, x) \leq \overline{\alpha}_w \bar{w}/\lambda := \bar{d}\}. \tag{3.19}$$

Notice that the mapping above is given using the Riemann distance which may depend upon a spatially varying metric. In order to efficiently plan a nominal trajectory whose associated RCI tube does not collide with obstacles, we would prefer a mapping that is *independent* of x^*. To this end, consider the following technical lemma.

Lemma 3. *(Geodesic Boundedness) Consider points $x^* \in \bar{\mathcal{X}}, x \in \mathcal{X}$ s.t. $x \in \Omega(x^*)$ where $\Omega(x^*)$ is given in* (3.19)*. Suppose the CCM $M(x)$ satisfies $M(x) \succeq \underline{M}$ for all $x \in \mathcal{X}$, where $\underline{M} \succeq \underline{\alpha} I_n$. Define the ellipsoid*

$$\widetilde{\Omega}(x^*) := \{x \in \mathcal{X} : \|x - x^*\|^2_{\underline{M}} \leq \bar{d}^2\}, \tag{3.20}$$

and suppose $\widetilde{\Omega}(x^*) \subset \mathcal{X}$. Then, the minimizing geodesic γ is contained entirely within $\widetilde{\Omega}(x^*)$, i.e., $\gamma(s) \in \widetilde{\Omega}(x^*)$ for all $s \in [0, 1]$, and thus $\Omega(x^*) \subseteq \widetilde{\Omega}(x^*)$.

Proof. Consider the following chain of inequalities:

$$\begin{aligned}
\mathcal{E}(x^*, x) &= \int_0^1 \delta_\gamma(s)^T M(\gamma(s)) \delta_\gamma(s) ds \\
&\geq \int_0^1 \delta_\gamma(s)^T \underline{M} \delta_\gamma(s) ds \\
&\geq \|x - x^*\|^2_{\underline{M}},
\end{aligned}$$

where the first inequality follows from the Lemma assumptions, and the second inequality follows from the fact that $\underline{M} \in \mathbb{S}^{>0}_n$ defines a flat Riemannian metric under which geodesics are straight lines. Thus, we obtain the implication:

$$\mathcal{E}(x^*, x) \leq \bar{d}^2 \Rightarrow \|x - x^*\|^2_{\underline{M}} \leq \bar{d}^2.$$

To complete the proof we note that since γ is a minimizing geodesic, it follows that $d(x^*, \gamma(s)) = sd(x^*, x)$ for all $s \in [0, 1]$. □

Equation (3.20) gives an ellipsoidal outer approximation of the RCI tube as defined in (3.19), and is thus independent of x^* (see Figure 3.2 for an illustration). This is essential for two reasons: (i) it drastically simplifies collision checking with respect to obstacles by avoiding geodesic computations, and (ii) the tightened state constraint set $\bar{\mathcal{X}}$ in (2.3a) must be taken to be $\mathcal{X} \ominus \widetilde{\Omega}$ to ensure that the minimizing geodesic lies within $\mathcal{X}$, i.e., where the CCM conditions hold.

So far we have shown how the *existence* of a feasible CCM $M(x)$ allows us to construct exponentially stabilizing controllers with bounded-input-bounded-output disturbance rejection guarantees.

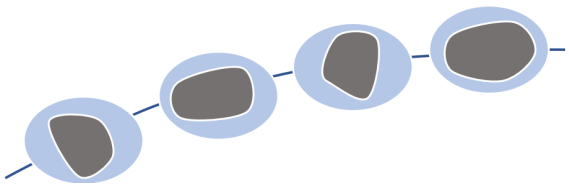


Figure 3.2: Illustration of the sets $\Omega(x^*)$ and $\tilde{\Omega}(x^*)$ along a trajectory $x^*(t)$. Due to the spatially varying $M(x)$, the set $\Omega(x^*)$ (shaded grey) continuously changes shape along the trajectory, making rapid collision checking difficult. The outer ellipsoidal approximation (shaded blue) $\tilde{\Omega}(x^*)$ is a fixed-size, easier to use collision margin.

In the next section, we demonstrate how one can leverage convex optimization techniques, specifically SOS programming, to compute offline, an *optimized* CCM that minimizes the size of the RCI tube and the outer ellipsoidal approximation.

3.3 Offline Synthesis of Optimized Contraction-Based Tubes

In this section we show how to compute CCMs *offline* that *minimize* a certain measure of the size of the RCI set (Section 3.3) which in turn minimizes the deviation of the perturbed trajectory from the nominal, thereby reducing the amount by which we must tighten the set $\mathcal{X}$. Specifically, we demonstrate how to transform the CCM conditions (3.4), (3.6), and (3.8) into convex constraints, and formulate the synthesis problem as a quasiconvex optimization problem with appropriate objective functions. Additionally, we illustrate the resulting methodology with a representative example.

3.3.1 Optimized CCMs

As shown in (Manchester and Slotine, 2017), condition (3.6) can be written as a pointwise Linear-Matrix-Inequality (LMI) by introducing the *dual* metric $W(x) := M(x)^{-1}$ and the change of variables $\eta_x := M(x)\delta_x$. Specifically, define a matrix $B_\perp(x)$ whose columns form a basis for the null space of $B(x)^T$ (i.e., $B(x)^T B_\perp(x) = 0$). Then, conditions (3.4) and (3.6) are equivalent to:

$$\partial_{b_j} W(x) - \widehat{\frac{\partial b_j(x)}{\partial x} W(x)} = 0, \quad j = 1, \dots, m \tag{3.21}$$

$$\underbrace{B_\perp^T \left(-\partial_f W(x) + \widehat{\frac{\partial f(x)}{\partial x} W(x)} + 2\lambda W(x) \right) B_\perp}_{:=\mathcal{F}_\lambda(x)} \preceq 0. \tag{3.22}$$

The equivalent reformulation of the weaker condition (3.8) in terms of the dual metric is given by LMI (3.22) and:

$$B_\perp^T \left(\partial_{b_j} W(x) - \widehat{\frac{\partial b_j(x)}{\partial x} W(x)} \right) B_\perp = 0, \ j = 1, \dots, m. \tag{3.23}$$

In this section we replace the CCM *feasibility* problem, i.e., conditions (3.21) and (3.22), with a quasiconvex optimization problem to minimize the size of the outer approximation for the RCI mapping. A trade-off between the overshoot constant $\sqrt{\overline{\alpha}/\underline{\alpha}}$ and contraction rate λ was briefly discussed in (Manchester and Slotine, 2014). Here, we further explore this aspect by formulating a global *optimization* program to characterize such a trade-off and minimize conservatism. Ideally, one would directly like to minimize the bound in (3.20) subject to conditions (3.21) and (3.22). However, this problem is non-convex and infinite-dimensional.

First, to address the infinite-dimensionality of the problem, we consider a finite-dimensional approximation whereby the dual metric $W(x)$ is parameterized as a polynomial matrix and the LMIs are written as SOS constraints, enforced over the semi-algebraic set $\mathcal{X}$ using the relaxations introduced in Section 2.4. Second, we propose an objective function that can be solved using line-search and quasiconvex optimization. Formally, we define the offline synthesis problem $\mathcal{OPT}_{\widehat{CCM}}$:

Optimization Problem $\mathcal{OPT}_{\widehat{CCM}}$ — Solve

$$\min_{\lambda\in\mathbb{R}_{>0}} J_{CCM}(\lambda) := \min_{\substack{W\in\mathcal{C}^\infty(\mathcal{X},\mathbb{S}_n^{>0}) \\ \overline{W}\in\mathbb{S}_n^{>0},\ \underline{\beta},\overline{\beta}\in\mathbb{R}_{>0}}} \quad \frac{1}{\lambda^2}\left(\frac{\overline{\beta}}{\underline{\beta}}\right)$$

$$\text{s.t.} \qquad \text{eqs. (3.21), (3.22)} \tag{3.24}$$

$$\underline{\beta} I_n \preceq W(x) \preceq \overline{W} \preceq \overline{\beta} I_n \ , \tag{3.25}$$

where the conditions hold uniformly for all $x \in \mathcal{X}$.

The cost function above is an upper-bound on the *worst-case* (normalized) Euclidean distance within the ellipsoid defined in (3.20) since from (8.6c), we have: $\overline{\alpha} = 1/\underline{\beta}$, $\underline{\alpha} = 1/\overline{\beta}$, and $\underline{M} = \overline{W}^{-1}$. Thus,

$$\sup_{x\in\widetilde{\Omega}(x^*)} \frac{\|x-x^*\|^2}{\bar{w}^2} = \frac{\overline{\alpha}_w^2}{\lambda^2\underline{\alpha}} \leq \frac{1}{\lambda^2}\left(\frac{\overline{\alpha}}{\underline{\alpha}}\right) = \frac{1}{\lambda^2}\left(\frac{\overline{\beta}}{\underline{\beta}}\right).$$

Recognizing that for a fixed contraction rate λ, the CCM conditions define a convex feasibility region for $W(x)$, and that minimization of the condition number of a positive definite matrix over a closed convex set (i.e., the inner minimization in problem $\mathcal{OPT}_{\widehat{CCM}}$) is quasiconvex (Lu and Pong, 2011), problem $\mathcal{OPT}_{\widehat{CCM}}$ can then be solved using line search on λ and bisection search over the condition number of $W(x)$ (i.e., a sequence of feasibility problems). In our implementation of the bisection search we further minimized the expression $\mathrm{Tr}(W_s\overline{W})$ where $W_s \in \mathbb{S}_n^{>0}$ is a diagonal scaling matrix. This is motivated by the following observation:

Remark 4. *One may alternatively choose the following objective for problem* $\mathcal{OPT}_{\widehat{CCM}}$*:*

$$\frac{1}{\lambda^2}\left(\frac{\mathrm{Tr}(W_s\overline{W})}{\underline{\beta}}\right),$$

where $W_s \in \mathbb{S}_n^{>0}$ is a diagonal scaling matrix. This cost function is motivated by the observation:

$$\left(\frac{\overline{d}}{\overline{w}}\right)^2 = \frac{\overline{\alpha}_w^2}{\lambda^2} \leq \frac{\overline{\alpha}}{\lambda^2} = \frac{1}{\underline{\beta}\lambda^2},$$

and that the projection of the ellipsoid (3.20) *along the i^{th} state is the interval: $\left[-\overline{d}\sqrt{\overline{W}_{ii}}, \overline{d}\sqrt{\overline{W}_{ii}}\right]$. Thus, this cost function attempts to minimize the size of this interval along each state dimension (relatively weighted by the matrix W_s). Since $\mathrm{Tr}(\cdot)$ is convex and $\underline{\beta} = \lambda_{\min}(W)$ is concave, for a fixed λ, we obtain another quasiconvex objective for the inner minimization in problem $\mathcal{OPT}_{\widehat{CCM}}$, which can be solved using bisection search.*

3.3.2 Illustrative Example: Planar Quadrotor

Consider the 6-state planar quadrotor system depicted in Figure 3.3.

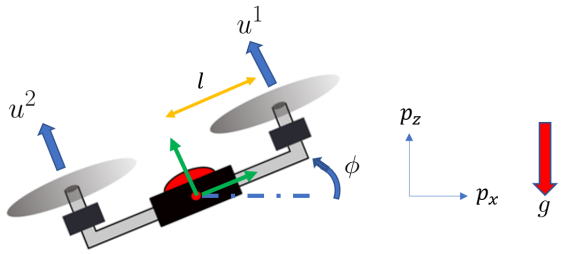


Figure 3.3: Definition of planar quadrotor state variables: l denotes the thrust moment arm (symmetric), and u^1 and u^2 denote the right and left motor thrust forces respectively.

The state vector is defined as $(p_x, p_z, \phi, \dot{p}_x, \dot{p}_z, \dot{\phi})^T$. For synthesizing the dual CCM however, it will be helpful to consider the alternative state-space representation: $(p_x, p_z, \phi, v_x, v_z, \dot{\phi})^T$ where (v_x, v_z) describe the velocity in the body frame of the vehicle with v_x representing the slip velocity (lateral) and v_z representing the velocity along the thrust axis. The control input $u \in \mathbb{R}^2_{\geq 0}$ corresponds to the individual motor thrusts. The dynamics in control affine form for this state representation are given as:

$$\dot{x} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_z \\ \dot{\phi} \\ \dot{v}_x \\ \dot{v}_z \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} v_x\cos(\phi) - v_z\sin(\phi) \\ v_x\sin(\phi) + v_z\cos(\phi) \\ \dot{\phi} \\ v_z\dot{\phi} - g\sin(\phi) \\ -v_x\dot{\phi} - g\cos(\phi) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 1/m \\ l/J & -l/J \end{bmatrix} \begin{bmatrix} u^1 \\ u^2 \end{bmatrix}, \tag{3.26}$$

where m and J denote the mass and moment of inertia about the out-of-plane axis. The inertial properties were taken from (Steinhardt and Tedrake, 2012), and were $m = 0.486$ kg, $J = 0.00383$

Kg m^2, and $l = 0.25$ m. By translation invariance of the dynamics, we expect that W will not be a function of (p_x, p_z). Furthermore, by leveraging this state representation, condition (3.21) requires that W is not a function of v_z or $\dot{\phi}$. Thus, for a 6-state system, the CCM may only be a function of (v_x, ϕ).

Let us impose the state-space bounds: $(v_x, v_z) \in [-2, 2] \times [-1, 1]$ m/s and $(\phi, \dot{\phi}) \in [-45^o, 45^o] \times [-60, 60]^o$/s, which may be concatenated together as the vector constraint $h(x) \geq 0$. A simple choice for $B_\perp$ is the matrix

$$\begin{bmatrix} I_4 \\ 0_{2\times 2} \end{bmatrix}.$$

Parameterize $W(x)$ as a polynomial matrix in (v_x, ϕ) with up to degree 4 monomials. With condition (3.21) dealt with, the remaining constraints in problem $\mathcal{OPT}_{\widehat{CCM}}$ may be written as:

$$\begin{aligned}
\underline{\beta} &\geq 1 && (3.27)\\
\overline{\beta} I_n - \overline{W} &\succeq 0 && (3.28)\\
h(x) \geq 0 \Rightarrow \quad \overline{W} - W(x) &\succeq 0 && (3.29)\\
h(x) \geq 0 \Rightarrow \quad W(x) - \underline{\beta} I_n &\succeq 0 && (3.30)\\
h(x) \geq 0 \Rightarrow \quad -\mathcal{F}_\lambda(x) &\succeq \epsilon I. && (3.31)
\end{aligned}$$

where ϵ is a small positive number and the constraint $\underline{\beta} \geq 1$ is imposed to ensure uniform definiteness. To enforce the semi-definite constraints of the type $h(x) \geq 0 \Rightarrow F(x) \succeq 0$ where $F : \mathcal{X} \to \mathbb{R}^{n_F \times n_F}$ is a placeholder for the matrix-valued functions in the constraints above, we leverage the quadratic form of the matrices by introducing auxiliary indeterminates y of dimension n_F so that the constraint may be equivalently written as $h(x) \geq 0 \Rightarrow y^T F(x) y \geq 0$, for all y. Leveraging certificates of the form introduced in Section 2.4, we pose the constraints:

$$\begin{aligned}
y^T F y - \sum_i L_i h^i \text{ is SOS}\\
\{L_i\} \text{ is SOS},
\end{aligned}$$

where $\{L_i\}$ is a set of SOS functions in (x, y). To keep the scale of the problem under control, one may wish to enforce simplifying structural properties for the multipliers $L(x, y)$ such as only retaining monomials quadratic in y (to reflect that $y^T F(x) y$ only contains quadratic terms in the auxiliary indeterminate y). The trigonometric terms in $\mathcal{F}_\lambda$ from the dynamics function were approximated using Chebyshev polynomial expansions up to third order.

Having cast the optimization as a SOS program, we may now proceed with solving problem $\mathcal{OPT}_{\widehat{CCM}}$. We swept through a range of values for λ, using the Spotless polynomial optimization toolbox (Tobenkin et al., 2013) and MOSEK SDP solver (ApS, 2017) to solve the SOS programs, each of which took about 40 seconds. Figure 3.4 plots the optimal curve for J_{CCM} as a function

of λ. The optimal contraction rate was determined to be $\lambda = 0.83$ and the corresponding dual

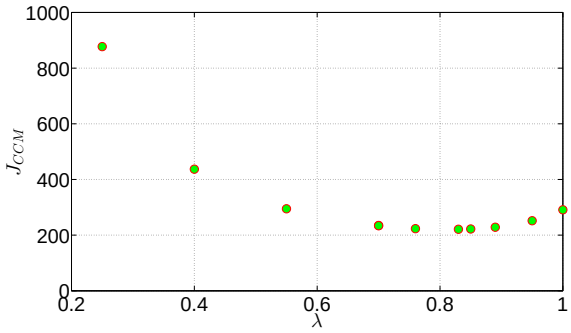
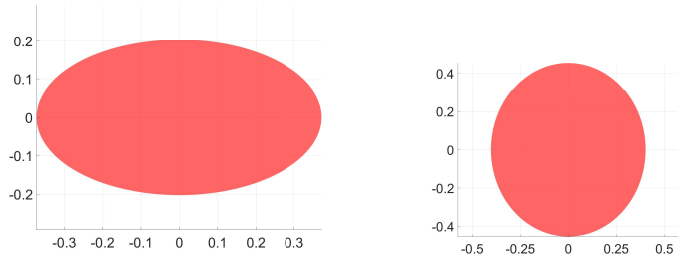


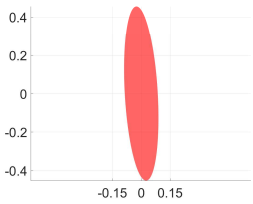
Figure 3.4: Problem $\mathcal{OPT}_{\widehat{CCM}}$ objective as a function of λ .

metric $W(x)$ contained 15 unique monomials in (ϕ, v_x). Assuming a cross-wind acting along either direction of the inertial p_x axis with effective acceleration up to 0.1 m/s^2, we used gridding to determine $\overline{d} = 0.038$. The resulting projections of the ellipsoid (3.20) are shown below in Figure 3.5.



(a) Projection onto the position states p_x-p_z.

(b) Projection onto the velocity states v_x-v_z.



(c) Projection onto the rotational states ϕ-$\dot{\phi}$.

Figure 3.5: Projections of the ellipsoidal tube upon various state-dimensions.

Notice that the projection of the outer ellipsoidal approximation of the RCI set onto the p_x-p_z plane has a major axis equal to 74 cm, and a minor axis equal to 41 cm, which compare quite favorably with the quadrotor wingspan, equal to 50 cm. That is, the size of the RCI tube is rather small and thus the nominal motion planner for the quadrotor is not overly constrained by the tightening of the state constraints. We next discuss an example for which the CCM synthesis

problem can be drastically simplified using non-traditional state representations.

3.3.3 Case Study: Control of Mechanical Lagrangian Systems

In this section we demonstrate how to synthesize CCMs for fully-actuated Lagrangian systems by borrowing concepts from sliding mode control. The analysis presented here demonstrates an alternative method of designing CCMs via elegant state-space descriptions. Indeed the use of contraction theory for *fully* actuated systems is addressed quite recently both in (Manchester et al., 2015) and (Reyes-Báez et al., 2017). In (Manchester et al., 2015) the authors demonstrate that a *constant* CCM can always be constructed for fully actuated systems using the state-space description $x = (q, \dot{q})^T$ where $q \in \mathbb{R}^n$ is the vector of generalized coordinates, thereby reducing the dual metric search problem to a finite-dimensional SDP. In (Reyes-Báez et al., 2017), the authors instead leverage ideas from sliding control (and feedback linearization), and derive a suitable contraction metric for an alternate state-space description and a *given* feedback controller. In the following, we also borrow ideas from sliding control but additionally provide a synthesis procedure for optimizing the controller's disturbance rejection properties and consequently, obtain tighter bounds than those resulting from the procedure in (Manchester et al., 2015), without using feedback linearization as in (Reyes-Báez et al., 2017). To do this, we first introduce the notion of *partial contraction* (Wang and Slotine, 2005).

Partial Contraction using Virtual Systems

Consider the autonomous system $\dot{x}(t) = f(x(t))$ and let $x(t; x_0)$ denote the solution at time t, starting from x_0 at $t = 0$. Define the following system:

$$\dot{y}(t) = \hat{f}(y(t), x(t; x_0)), \quad y(0) = y_0, \tag{3.32}$$

which satisfies the property:

$$\hat{f}(x, x) = f(x),$$

and with solutions denoted as $y(t; y_0)$. Thus setting $y_0 = x_0$ recovers the trajectory $x(t; x_0)$. Then, if there exists a uniformly (in x) stable differential Lyapunov function $V(y, \delta_y, t)$ for the above system, with associated variational dynamics

$$\dot{\delta}_y(t) = \frac{\partial \hat{f}}{\partial y}(y, x)\delta_y,$$

then $y(t; y_0)$ is contracting towards $y(t; x_0)$, i.e., $x(t; x_0)$. The crux of the argument is based on defining the *time-varying* system:

$$\dot{y}(t) = \tilde{f}_{x_0}(t, y(t)) := \hat{f}(y(t), x(t; x_0)), \quad y(0) = y_0,$$

and observing that the contracting properties of $\hat{f}(\cdot,\cdot)$ imply that solutions of the above time-varying system with arbitrary initial conditions y_0 and y_0' converge towards each other. Setting $y_0' = x_0$ proves the desired claim. The system (3.32) is defined as the "virtual" system for $\dot{x} = f(x)$.

Fully Actuated Systems

Consider the fully actuated Lagrangian dynamical system:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} = u,$$

where $q \in \mathbb{R}^n$ is the vector of generalized coordinates, $H(q) \in \mathbb{S}_n^{>0}$ is the inertia matrix, $C(q,\dot{q})$ contains the damping and Coriolis terms, and w.l.o.g., we neglect the potential term. Let $(q^*(t), \dot{q}^*(t))^T$ represent the desired trajectory in generalized coordinates. Instead of using the state representation $x = (q,\dot{q})^T$, consider the change of variables:

$$x(t) := \begin{bmatrix} \tilde{q}(t) \\ \sigma(t) \end{bmatrix} := \begin{bmatrix} q(t) - q^*(t) \\ p(t) - p_r(t) \end{bmatrix},$$

where $p(t) := H(q(t))\dot{q}(t)$ is the generalized momentum, and $p_r(t) := H(q(t))\dot{q}^*(t) - H(q(t))\Lambda\tilde{q}(t)$ where $\Lambda \in \mathbb{S}_n^{>0}$ is a constant, positive definite, diagonal (for simplicity) matrix. Then, our dynamics in this state-representation may be written as:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} H^{-1}(q)(\sigma + p_r) - \dot{q}^* \\ \dot{H}(q)\dot{q} - C(q,\dot{q})\dot{q} + u - \dot{p}_r \end{bmatrix} \\ &= \begin{bmatrix} -\Lambda\tilde{q} + H^{-1}(q)\sigma \\ \left(\dot{H}H^{-1} + H\Lambda H^{-1} - CH^{-1}\right)\sigma + \left(-H\Lambda^2 + C\Lambda\right)\tilde{q} - \left(C\dot{q}^* + H\ddot{q}^*\right) \end{bmatrix} + \begin{bmatrix} O_n \\ I_n \end{bmatrix} u, \end{aligned} \tag{3.33}$$

where $\ddot{q}^*$ is a function of $(q^*, \dot{q}^*, u^*)$. One may interpret σ as a *pseudo*-sliding variable since $\sigma = 0$ implies the stable first order dynamics for $\tilde{q}$. Notice that the state/control trajectory pair $(x^*(t), u^*(t)) = (0, u^*(t))$ is a feasible solution for the dynamics above.

Recall that $C = C(q(t),\dot{q}(t)) = C(q^*(t) + \tilde{q}(t), \dot{q}^*(t) + \dot{\tilde{q}}(t)) =: C_t(\tilde{q}, \dot{\tilde{q}})$ where the subscript $(\cdot)_t$ is used to denote the dependence upon the fixed signal $q^*(t)$. Similarly, write $H = H(q) = H_t(\tilde{q})$. Using the expression above for $\dot{\tilde{q}} = -\Lambda\tilde{q} + H_t^{-1}(\tilde{q})\sigma$, we further delineate $C_t(\tilde{q}, \dot{\tilde{q}})$ as $C_t(\tilde{q}, \sigma)$. Consider, then, the following virtual system for the above dynamics:

$$\dot{y} = \begin{bmatrix} -\Lambda y_1 + H_t^{-1} y_2 \\ \left(-H_t\Lambda^2 + C_t\Lambda\right) y_1 + \left(\dot{H}_t H_t^{-1} + H_t\Lambda H_t^{-1} - C_t H_t^{-1}\right) y_2 - \left(C_t(y_1, y_2)\dot{q}^* + H_t(y_1)\ddot{q}^*\right) \end{bmatrix} + \begin{bmatrix} O_n \\ I_n \end{bmatrix} u,$$

where we omit the explicit dependences $(\tilde{q})$ and $(\tilde{q}, \sigma)$ for clarity. Clearly, setting $y(0) = (\tilde{q}(0), \sigma(0))$

and $u = u(t)$ recovers the true state trajectory $x(t)$ while setting $y(0) = (0, 0)$ and $u(t) = u^*(t)$ recovers the desired nominal state trajectory $x^*(t) = (0, 0)$ for all $t \geq 0$.

We now design a *constant* CCM for the virtual system. The result will also yield a stabilizing controller in the virtual space where as highlighted in the paragraph above, the endpoints of the geodesic in virtual space coincide with the nominal trajectory $x^*(t) = (0, 0)$ and actual trajectory $x(t)$. First, the variational dynamics for the virtual system above are given by:

$$\dot{\delta}_y = \begin{bmatrix} -\Lambda & H_t^{-1} \\ * & * \end{bmatrix} \delta_y + \begin{bmatrix} O_n \\ I_n \end{bmatrix} \delta_u,$$

where the $(*)$ indicate quantities that will become irrelevant in the CCM synthesis problem. Taking

$$B_\perp = \begin{bmatrix} I_n \\ O_n \end{bmatrix},$$

and assuming a *constant* dual metric W with the following block structure:

$$W = \left[\begin{array}{c|c} W_\perp & W_u \\ \hline * & W_\parallel \end{array}\right],$$

where $W_\perp, W_\parallel \in \mathbb{S}_n^{>0}$ and $W_u \in \mathbb{R}^{n \times n}$, the stability condition (3.22) requires that:

$$-\Lambda W_\perp - W_\perp \Lambda + H_t^{-1} W_u^T + W_u H_t^{-1} \preceq -2\lambda W_\perp.$$

Condition (3.21) simply requires that W is not a function of y_2, which is automatically satisfied given W is parameterized as a *constant* matrix. As the CCM should be independent of the desired nominal trajectory, we set $W_u = O_n$, yielding, for given λ, Λ, the following LMI:

$$-\Lambda W_\perp - W_\perp \Lambda \preceq -2\lambda W_\perp.$$

This is an insightful result since the inequality above simply requires that the bandwidth of the pseudo-sliding variable σ, encapsulated by Λ, is faster than the contraction rate λ. With respect to (Manchester et al., 2015), by avoiding the construction of the dual metric in $(q, \dot{q})$ space, the resulting disturbance bound (for input disturbances) will depend upon $\overline{\sigma}\left(W_\parallel^{-1}\right)$ instead of an expression that includes bounding the singular values of $H^{-1}(q)$, which can be very poor indeed[5]. As compared with (Reyes-Báez et al., 2017), we *do not* use feedback linearization to cancel out the nonlinearities for $\dot{\sigma}$ in (3.33), nor do we assume a fixed feedback controller and associated resulting contraction metric. In contrast, our method explicitly allows to balance the robustness properties

[5]This is easily seen since for input disturbances, the matrix $B_w = B = \left[O_n\,, H^{-1}(q)\right]^T$ and thus $\overline{\alpha}_w$ will depend upon $\overline{\sigma}(H^{-1}(q))$.

of the controller and the bandwidth tradeoff between Λ and λ.

Let us examine this design tradeoff. Parameterize $W_\perp = w_\perp I_n$ and $W_\parallel = w_\parallel I_n$ where $w_\perp, w_\parallel \in \mathbb{R}_{>0}$. Then taking $B_w = B$, we have that

$$\bar{d} = \frac{\overline{w}}{\lambda\sqrt{w_\parallel}},$$

where, as before, $\overline{w}$ bounds the Euclidean norm of the disturbance inputs. Projecting the bound into the virtual space (consequently bounding the state x), we obtain:

$$\left|\tilde{q}^i\right| \leq \frac{\overline{w}}{\lambda}\sqrt{\frac{w_\perp}{w_\parallel}}, \quad \text{and} \quad |\sigma^i| \leq \frac{\overline{w}}{\lambda} \quad \forall i = 1, \ldots, n. \tag{3.34}$$

Re-writing σ as $H(q)\left(\dot{\tilde{q}} + \Lambda\tilde{q}\right)$ and leveraging the bounds above on $\tilde{q}$ and σ, one may further deduce bounds on $\dot{\tilde{q}}$, which can be shown to be proportional to $\overline{\sigma}(\Lambda)\|\tilde{q}\| + \bar{\sigma}(H^{-1}(q))\|\sigma\|$. While (3.34) highlights the design insight into reducing the error bounds on $\tilde{q}$ and σ, it is apparent that a more aggressive Λ will result in weaker bounds[6] for $\dot{\tilde{q}}$. Nevertheless, the improvement in the resulting bound for the generalized coordinate and momentum, the simplicity of the synthesis procedure, and the closed-form of the resulting controller are certainly non-trivial consequences of the partial contraction method for designing CCMs.

Remark 5. *One may further customize the dual metric for this class of systems by choosing a set of positive scalars $\{w_\perp^i\}_{i=1}^n$ corresponding to the diagonal matrix $W_\perp$, such that the bound on the i^{th} generalized coordinate q^i may be expressed as:*

$$|\tilde{q}^i| \leq \frac{\overline{w}}{\lambda}\sqrt{\frac{w_\perp^i}{w_\parallel}}.$$

The simplicity of such customization and design freedom highlights the benefits of the virtual CCM method outlined in this section.

3.4 Summary

In this chapter, we introduced contraction theory and control contraction metrics, and demonstrated how to leverage these concepts to construct tracking feedback controllers with strong disturbance rejection guarantees. Additionally, by leveraging SOS programming, we formulated an offline global optimization problem to synthesize controllers that minimize the cross-sectional size of the invariant tube (i.e., the disturbance-to-tracking-error gain). In the next chapter, we demonstrate how to leverage these results within the online phase of our robust motion planning framework.

[6] A strong case can be made, however, for the importance of bounding the error in generalized momentum as opposed to generalized rates.

Chapter 4

CCM-Based Feedback Planning

We now illustrate how to leverage the contributions of Chapter 3, chiefly, the offline synthesis of nonlinear feedback controllers and invariant tubes, to create a robust online planning algorithm. The key feature, as is standard in feedback planning, is to use the invariant tube as a "buffer" when planning nominal trajectories with the unperturbed dynamics. This allows us the flexibility to utilize *any standard motion planner* capable of generating dynamically feasible trajectories for the unperturbed dynamics. We stress, therefore, that the contribution of this chapter is not the nominal planner itself, but rather, the incorporation of CCM-based stabilization into an existing black-box planner. In particular, we demonstrate how to implement the CCM-derived feedback controller online, and leverage online re-planning using model predictive control (MPC). The approach will be illustrated using the planar quadrotor example introduced in Chapter 3.

4.1 Online Computation of CCM Controller

Equation (3.9) provides an example of how to construct a stabilizing feedback controller that guarantees the associated stability and robustness properties. That is, it establishes the *existence* of an exponentially stabilizing feedback controller with respect to the computed CCM. However, this feedback controller is not unique. Given the existence of this controller, we construct an alternative feedback controller that still satisfies the desired stability properties with respect to the computed CCM, but additionally, also minimizes the net control effort in order to curtail the suboptimality introduced by the "nominal plus tracking feedback" parameterization in (2.2). Given a CCM computed offline by solving problem $\mathcal{OPT}_{\widehat{CCM}}$, the feedback controller is computed as a solution to the following *analytical* QP:

> **Optimization Problem $\mathcal{OPT}_{\textbf{online}}$** — At time $t \geq 0$, given a desired/current state pair $(x^*(t), x(t))$ and a minimizing geodesic $\gamma(\cdot, t)$ connecting these two states (i.e., $\gamma(0, t) =$

$x^*(t)$ and $\gamma(1,t) = x(t)$), solve

$$k^*(x^*(t),\ x(t)) = \underset{k\in\mathbb{R}^m}{\operatorname{argmin}} \quad \|k\|^2$$
$$\text{s.t.} \quad 2\delta_\gamma^T(1,t)M(x(t))\hat{\dot{x}}(t) - 2\delta_\gamma^T(0,t)M(x^*(t))\dot{x}^*(t)$$
$$\leq -2\lambda\mathcal{E}(x^*(t),x(t)), \tag{4.1}$$

where $\hat{\dot{x}}(t) = f(x(t)) + B(x(t))(u^*(t)+k)$ represents the nominal dynamics evaluated at $x(t)$ and $\dot{x}^*(t) = f(x^*(t)) + B(x^*(t))u^*(t)$.

A few comments are in order. First, the existence of the dual metric $W(x)$ ensures that there exists a differential feedback controller such that inequality (3.7) holds for all (x,δ_x) along the minimizing geodesic between x^* and x. Then, by the equivalence shown in (3.15), problem $\mathcal{OPT}_{\text{online}}$ is always feasible. Second, the linear inequality (4.1) is essentially a relaxation of (3.7), in that it only enforces contraction *tangent to the given geodesic*. In contrast, the differential controller proposed in (Manchester and Slotine, 2017), obtained by solving a *feasibility* problem, must ensure that the system contracts in all directions with *at least* rate λ (implemented online by integrating the controller along the minimizing geodesic as in (3.9)). Such a relaxation still guarantees IES as only the flow *along the geodesic* affects the convergence of $x(t)$ to $x^*(t)$. On the other hand, one can often dramatically decrease control effort as compared with computing the controller using (3.9). Third, problem $\mathcal{OPT}_{\text{online}}$ is a QP subject to a single linear inequality and thus may be solved analytically (given the geodesic $\gamma(\cdot,t)$). Indeed, the QP above strongly resembles the min-norm formulation of Sontag's generalized formula for CLF-based stabilization (Primbs et al., 1999), thereby underscoring the interpretation of the Riemann energy of the minimizing geodesic as an *incremental* CLF. We now address online computation of the geodesic.

4.1.1 Computing Geodesics Online

Computation of the geodesic between two points $p, q \in \mathcal{X}$ can be framed as the following functional optimization problem:

Optimization Problem $\mathcal{OPT}_\gamma$ — At time $t \geq 0$, given desired state $x^*(t)$ and current state $x(t)$, solve

$$\min_{c\in\Gamma(x^*(t),x(t))} \quad \mathcal{E}(c) \tag{4.2}$$

Following the approach in (Leung and Manchester, 2017), such a problem can be efficiently solved by applying the Chebyshev global pseudospectral method, i.e., by discretizing the interval $[0,1]$ using the Chebyshev-Gauss-Lobatto nodes and using Chebyshev interpolating polynomials up to degree N to approximate the curve. The integral in (4.2) is approximated using the Clenshaw-Curtis

quadrature scheme with $K > N$ nodes. As in (Leung and Manchester, 2017), we choose $K > N$ since the integral involves the inverse of the dual metric W. Thus, the integrand in $\mathcal{E}(c)$ is not guaranteed to be polynomial.

Given the solution to the geodesic problem $\mathcal{OPT}_\gamma$, parameterized by a set of values $\{\gamma(s_k)\}_{k=0}^K$ and $\{\delta_\gamma(s_k)\}_{k=0}^K,\ s_k \in [0,1]$, problem $\mathcal{OPT}_{\text{online}}$ may now be solved as an analytical QP using $\delta_\gamma(s_0)$ and $\delta_\gamma(s_K)$.

4.1.2 Bounding Feedback Control Effort

There are two ways to compute bounds on the CCM feedback controller. For systems that satisfy the stronger Killing field condition given by (3.21), Theorem 5 provides a bound on the magnitude of the optimized tracking controller computed using problem $\mathcal{OPT}_{\text{online}}$. We first require the following technical lemma:

Lemma 4 (Norm Bound for Tracking Controller)**.** *Let S be a symmetric matrix in $\mathbb{R}^{n\times n}$ and Y a full row-rank matrix in $\mathbb{R}^{m\times n}$. Construct matrices $\overline{Y} \in \mathbb{R}^{n\times m}$ and $\overline{Y}_\perp \in \mathbb{R}^{n\times(n-m)}$ such that the columns of $\overline{Y}$ form an orthonormal basis for the column space of Y^T, i.e., Col(Y^T), and the columns of $\overline{Y}_\perp$ form an orthonormal basis for the nullspace of Y, i.e., $\mathcal{N}(Y)$. Suppose, then, that the following conditions hold:*

$$\eta_z^T S \eta_z \leq 0 \quad \forall \eta_z \in \mathcal{N}(Y) \subset \mathbb{R}^n \tag{4.3}$$

$$\kappa \leq \frac{2\bar{\delta}_u}{\theta}, \quad \textit{where} \quad \kappa = \begin{cases} 0 & \textit{if } \bar{\lambda}(S_Y) \leq 0 \\ \sqrt{\left(\dfrac{\bar{\lambda}(S_Y)}{\underline{\sigma}_{>0}(Y\overline{Y})}\right)^2 + 4\left(\dfrac{\bar{\sigma}(\overline{Y}_\perp^T S \overline{Y})}{\underline{\sigma}_{>0}(Y\overline{Y})}\right)^2} & \textit{else} \end{cases}, \tag{4.4}$$

for some constants $\bar{\delta}_u \in \mathbb{R}_{\geq 0}$ and $\theta \in \mathbb{R}_{>0}$, where $\underline{\sigma}_{>0}(\cdot)$ denotes the smallest non-zero singular value, and $S_Y := \overline{Y}^T S \overline{Y}$. Then,

$$\theta\, \eta_z^T S \eta_z \leq 2\bar{\delta}_u \|Y\eta_z\| \quad \forall \eta_z \ \textit{s.t.}\ \|\eta_z\| \leq 1. \tag{4.5}$$

Proof. Decompose η_z as $\eta_{z_Y} + \eta_{z_{Y_\perp}}$, where $\eta_{z_Y} \in \text{Col}(Y^T)$ and $\eta_{z_{Y_\perp}} \in \mathcal{N}(Y)$. Now write $\eta_{z_Y} = \epsilon \hat{\eta}_{z_Y}$ and $\eta_{z_{Y_\perp}} = \epsilon_\perp \hat{\eta}_{z_{Y_\perp}}$ where $\epsilon, \epsilon_\perp \geq 0$, $\epsilon^2 + \epsilon_\perp^2 \leq 1$, and $\hat{\eta}_{z_Y}$ and $\hat{\eta}_{z_{Y_\perp}}$ are (non-zero) unit vectors contained in $\text{Col}(Y^T)$ and $\mathcal{N}(Y)$, respectively. Substituting these expressions into inequality (4.5) above yields

$$\theta\left(\epsilon^2 \hat{\eta}_{z_Y}^T S \hat{\eta}_{z_Y} + \epsilon_\perp^2 \hat{\eta}_{z_{Y_\perp}} S \hat{\eta}_{z_{Y_\perp}} + 2\epsilon\epsilon_\perp \hat{\eta}_{z_{Y_\perp}}^T S \hat{\eta}_{z_Y}\right) \leq 2\bar{\delta}_u \epsilon \|Y \hat{\eta}_{z_Y}\|. \tag{4.6}$$

Now, if $\eta_z \in \mathcal{N}(Y)$, i.e., $\epsilon = 0$, then condition (4.3) is necessary and sufficient for inequality (4.5).

Thus, we consider the case where $\epsilon > 0$. Notice that

$$\max_{\hat{\eta}_{z_{Y_\perp}} \in \mathcal{N}(Y)} \hat{\eta}_{z_{Y_\perp}}^T S \hat{\eta}_{z_Y} = \|\overline{Y}_\perp^T S \hat{\eta}_{z_Y}\|,$$

and by condition (4.3), $\hat{\eta}_{z_{Y_\perp}} S \hat{\eta}_{z_{Y_\perp}} < 0$. Thus, by upper-bounding the left hand side of the inequality in (4.6) and rearranging, we obtain the following *sufficient* condition:

$$\left(\epsilon \frac{\hat{\eta}_{z_Y}^T S \hat{\eta}_{z_Y}}{\|Y \hat{\eta}_{z_Y}\|} + 2\epsilon_\perp \frac{\|\overline{Y}_\perp^T S \hat{\eta}_{z_Y}\|}{\|Y \hat{\eta}_{z_Y}\|} \right) \leq \frac{2\bar{\delta}_u}{\theta}, \tag{4.7}$$

for all $\hat{\eta}_{z_Y} \in \mathrm{Col}(Y^T)$, and $\epsilon, \epsilon_\perp$. Now, given that the columns of $\overline{Y}$ are an orthonormal basis for $\mathrm{Col}(Y^T)$, then $\hat{\eta}_{z_Y}$ may be expressed as $\overline{Y}\overline{\eta}_{z_Y}$ where $\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}$, the $(m-1)$-unit sphere. Substituting this expression into the inequality above yields the following sufficient condition for inequality (4.5):

$$\max_{\substack{\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1} \\ \epsilon, \epsilon_\perp \geq 0 \\ \epsilon^2 + \epsilon_\perp^2 \leq 1}} \quad \epsilon \frac{\overline{\eta}_{z_Y}^T S_Y \overline{\eta}_{z_Y}}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} + 2\epsilon_\perp \frac{\|\overline{Y}_\perp^T S \overline{Y}\,\overline{\eta}_{z_Y}\|}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} \leq \frac{2\bar{\delta}_u}{\theta}.$$

For fixed $(\epsilon, \epsilon_\perp)$, the maximization over $\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}$ above belongs to the class of sum-of-ratios fractional programming and is in general, NP-complete. Recently in (Nguyen et al., 2016), the authors presented a two-stage algorithm using tight SDP relaxations of parameterized subproblems for maximizing the sum of a generalized Rayleigh quotient and another Rayleigh quotient on the unit sphere. For our purposes, we derive a simpler yet suboptimal approximation by decoupling the maximization as

$$\max_{\substack{\epsilon, \epsilon_\perp \geq 0 \\ \epsilon^2 + \epsilon_\perp^2 \leq 1}} \left(\epsilon \max_{\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}} \frac{\overline{\eta}_{z_Y}^T S_Y \overline{\eta}_{z_Y}}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} + 2\epsilon_\perp \max_{\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}} \frac{\|\overline{Y}_\perp^T S \overline{Y}\,\overline{\eta}_{z_Y}\|}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} \right).$$

The two inner maximizations may be further upper-bounded as:

$$\max_{\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}} \frac{\overline{\eta}_{z_Y}^T S_Y \overline{\eta}_{z_Y}}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} \leq \frac{\bar{\lambda}(S_Y)}{\underline{\sigma}_{>0}(Y\overline{Y})} \quad \text{and} \quad \max_{\overline{\eta}_{z_Y} \in \mathcal{S}^{m-1}} \frac{\|\overline{Y}_\perp^T S \overline{Y}\,\overline{\eta}_{z_Y}\|}{\|Y\overline{Y}\,\overline{\eta}_{z_Y}\|} \leq \frac{\bar{\sigma}(\overline{Y}_\perp^T S \overline{Y})}{\underline{\sigma}_{>0}(Y\overline{Y})}.$$

Then, the outer maximization over $(\epsilon, \epsilon_\perp)$ is of an affine expression over a convex set:

$$\max_{\substack{\epsilon, \epsilon_\perp \geq 0 \\ \epsilon^2 + \epsilon_\perp^2 \leq 1}} \epsilon \frac{\bar{\lambda}(S_Y)}{\underline{\sigma}_{>0}(Y\overline{Y})} + 2\epsilon_\perp \frac{\bar{\sigma}(\overline{Y}_\perp^T S \overline{Y})}{\underline{\sigma}_{>0}(Y\overline{Y})}.$$

Now, since the coefficient of $\epsilon_\perp$ is nonnegative, the optimal $(\epsilon, \epsilon_\perp)$ is either $(0, 1)$ (this occurs when the coefficient of ϵ is nonpositive), or lies along the nonnegative quadrant of the circle $\epsilon^2 + \epsilon_\perp^2 = 1$

with $\epsilon > 0$. We can ignore the solution $(0, 1)$ since this corresponds to the case where $\eta_z \in \mathcal{N}(Y)$. For the case $\epsilon > 0$, the maximization above evaluates to

$$\sqrt{\left(\frac{\bar{\lambda}(S_Y)}{\underline{\sigma}_{>0}(Y\overline{Y})}\right)^2 + 4\left(\frac{\bar{\sigma}(\overline{Y}_\perp^T S\overline{Y})}{\underline{\sigma}_{>0}(Y\overline{Y})}\right)^2}.$$

Thus, we arrive at the sufficient condition stated in (4.4). □

We now leverage Lemma 4 to derive the bound on the magnitude of optimized tracking controller.

Theorem 5 (Tracking Control Effort)**.** *Define*

$$F(x) := -\partial_f W(x) + \widehat{\frac{\partial f(x)}{\partial x} W(x)} + 2\lambda W(x).$$

Assume the dual CCM $W(x)$ satisfies conditions (3.21) *and* (3.22)*. Factorize $W(x)$ as $L(x)^T L(x)$ and define $S(x) = L^{-T} F L^{-1}$ and $Y(x) = B^T L^{-1}$. Suppose then that the matrices $S(x)$ and $Y(x)$ satisfy property* (4.4) *for all $x \in \mathcal{X}$ with $\theta = \bar{d} = \overline{\alpha}_w \bar{w}/\lambda$, where $\overline{Y}$ and $\overline{Y}_\perp$ are defined as stated in Lemma 4. Then, the optimized feedback controller $k^*(x^*, x)$ satisfies the bound:*

$$\|k^*(x^*, x)\| \leq \overline{\delta}_u, \tag{4.8}$$

for all $x^*, x \in \mathcal{X}$ such that $x \in \Omega(x^*)$.

Proof. As a consequence of CCM condition (3.22), the matrices $S(x)$ and $Y(x)$ satisfy property (4.3) for all $x \in \mathcal{X}$. Then, in conjunction with property (4.4), it follows from the conclusions of Lemma 4 that

$$\begin{aligned} \bar{d}^2 \eta_z^T \left(L^{-T} F L^{-1}\right) \eta_z \leq 2\bar{d}\,\overline{\delta}_u \|B^T L^{-1} \eta_z\|, \\ \forall \eta_z \ s.t. \ \|\eta_z\| \leq 1, \end{aligned} \tag{4.9}$$

for all $x \in \mathcal{X}$. Let $\eta_x := \bar{d} L^{-1} \eta_z$. Then, the set $\{\eta_z \in \mathbb{R}^n : \|\eta_z\| \leq 1\}$ is equivalent to the set $\{\eta_x \in \mathbb{R}^n : \|\eta_x\|_{W(x)}^2 \leq \bar{d}^2\}$ and inequality (4.9) may be written as

$$a(x, \eta_x) \leq \overline{\delta}_u \|r(x, \eta_x)\|, \quad \forall \eta_x \ s.t. \ \eta_x^T W(x) \eta_x \leq \bar{d}^2, \tag{4.10}$$

for all $x \in \mathcal{X}$, where

$$\begin{aligned} a(x, \eta_x) &:= \eta_x^T F(x) \eta_x, \\ r(x, \eta_x) &:= 2B(x)^T \eta_x. \end{aligned}$$

Notice that statement (4.10) along with the CCM condition (3.22) is equivalent to the feasibility

of the following CLF-like condition (with respect to bounded controls) stated in (Lin and Sontag, 1991):

$$\inf_{\|\delta_u\| \leq \overline{\delta}_u} \left(a(x, \eta_x) + r(x, \eta_x)^T \delta_u \right) \leq 0, \tag{4.11}$$

for all η_x satisfying $\|\eta_x\|^2_{W(x)} \leq \bar{d}^2$. Then, by Theorem 1 in (Lin and Sontag, 1991) there exists an almost-smooth function $\delta_u(x, \eta_x)$, bounded in Euclidean norm by $\overline{\delta}_u$, such that condition (4.11) (equivalently the dual form of inequality (3.7)) is satisfied for all (x, η_x) along the minimizing geodesic connecting any $x^* \in \bar{\mathcal{X}}$ and $x \in \Omega(x^*)$. For completeness, this function is given below:

$$\delta_u(x, \eta_x) = \begin{cases} 0 & \text{if } r = 0, \\ -\dfrac{a + \sqrt{a^2 + \overline{\delta}_u^4 \|r\|^4}}{\overline{\delta}_u \|r\|^2 \left(1 + \sqrt{1 + \overline{\delta}_u^2 \|r\|^2}\right)} r & \text{else,} \end{cases}$$

where we have dropped the parenthesis (x, η_x) for clarity. For each $x \in \mathcal{X}$, the function above is continuous for all η_x (requisite for integrability) and smooth for $\eta_x \neq 0$.

By the equivalence shown through (3.15), the tracking controller given by integrating the function above along the minimizing geodesic connecting x^* and x is indeed a feasible solution to problem $\mathcal{OPT}_{\text{online}}$ that satisfies the bound claimed in (4.8), completing the proof. □

A few comments regarding the computation of the bound $\overline{\delta}_u$ are in order. Note that from Theorem 5, one needs to show that inequality (4.10) holds for all $x \in \mathcal{X}$. Rewriting this inequality as (4.9), one may deduce a simpler, yet loose approximation of $\overline{\delta}_u$ as:

$$\overline{\delta}_u = \frac{\bar{d}}{2} \sup_{x \in \mathcal{X}} \left(\frac{\bar{\lambda}(L^{-T} F L^{-1})}{\underline{\sigma}_{>0}(B^T L^{-1})} \right), \tag{4.12}$$

where we omit the explicit dependence on x for notational clarity. A better approximation may be obtained by leveraging Lemma 4, specifically, inequality (4.4):

$$\overline{\delta}_u = \frac{\bar{d}}{2} \sup_{x \in \mathcal{X}} \kappa(x), \tag{4.13}$$

where κ is as given in (4.4) and $S, Y, \overline{Y}, \overline{Y}_\perp$ are as defined in Theorem 5.

For systems that do not satisfy the strong Killing field condition in (3.21), and instead satisfy the alternative pair of conditions, i.e., (3.22) and (3.23), one may bound the feedback control by augmenting the state with u, and treating $\dot{u}$ as the actual input. The RCI tube obtained via the analysis in Section 3.2 then captures a bound on both x and u. We will shortly illustrate both methods for bounding the control effort.

4.2 Robust Planning

We are finally ready to formalize the robust planning algorithm. The core part of the algorithm relies on computing nominal motion plans (x^*, u^*) for the unperturbed dynamics using tightened constraints. Given the tubes derived in Section 3.2 and the control effort bounds computed in the previous section, these tightened constraints are given as:

$$x^*(\cdot) \in \bar{\mathcal{X}} := \mathcal{X} \ominus \tilde{\Omega}, \tag{4.14a}$$

$$u^*(\cdot) \in \bar{\mathcal{U}} := \{\bar{u} \in \mathcal{U} : \forall x^*(t) \in \bar{\mathcal{X}}, \forall x(t) \in \mathcal{X} \text{ s.t. } x(t) \in \Omega(x^*(t)),\, \bar{u} + k(x^*(t), x(t)) \in \mathcal{U}\}. \tag{4.14b}$$

Notice that the state constraint is shrunk by the fixed-size ellipsoid $\tilde{\Omega}$ to ease computation (e.g., collision-checking). On the other hand, since $x(t)$ is guaranteed to lie in the true RCI tube $\Omega(\cdot)$ defined using the geodesic distance, the control constraint is shrunk based on the bounds computed in the previous section. Having computed such a plan, one may adopt two different frameworks for its robust execution. In the first approach, one could simply execute the controller derived from problem $\mathcal{OPT}_{\text{online}}$ until the robot enters $\mathcal{X}_{\text{goal}}$. In the second approach, provided there exist sufficient online computational resources, one can use a receding-horizon algorithm in which the nominal trajectory is periodically and *locally* re-updated over a short time-horizon $T < T_{\text{goal}}$. This allows one to reduce the tracking cost (as information about the realized disturbances is taken into account). We outline such a receding-horizon strategy next.

4.2.1 Receding Horizon Implementation

Given a robust motion plan (i.e., a nominal state-input trajectory (x^*, u^*) such that the RCI tube centered on x^* does not intersect any obstacles), one can make *local* updates to it using the following MPC problem solved at the discrete time instants $t_i,\ i \in \mathbb{N}_{\geq 0}$:

Optimization Problem MPC — Given current state $x(t_i)$ and a robust motion plan (x^*, u^*) with associated RCI mapping $\Omega(\cdot)$, solve

$$\min_{\substack{\bar{u}(\cdot)\in\mathcal{C}^2([t_i,t_i+T],\bar{\mathcal{U}}) \\ \bar{x}(t_i)\in\bar{\mathcal{X}} \\ T_i\in\mathbb{R}_{>0}}} \quad \int_{t_i}^{t_i+T} \|\bar{u}(\tau)\|_R^2\, d\tau - \mu T_i$$

$$\text{s.t.} \quad x(t_i) \in \Omega(\bar{x}(t_i)), \tag{4.15}$$

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau)) + B(\bar{x}(\tau))\bar{u}(\tau), \tag{4.16}$$

$$\bar{x}(\tau) \in \bar{\mathcal{X}}, \quad \bar{u}(\tau) \in \bar{\mathcal{U}} \qquad \forall \tau \in [t_i, t_i + T] \tag{4.17}$$

$$\bar{x}(t_i + T) = x^*(T_i), \quad t_i + \delta \leq T_i \leq T^*_{\text{goal}}, \tag{4.18}$$

where $\mu > 0$ is a weighting factor. The time T_i (hereby termed as the "re-join" time), also an optimization variable within problem MPC, marks the point where the MPC state trajectory re-joins the initial motion plan and will be used to ensure persistent feasibility (see Lemma 6).

Problem MPC should be understood as a *local* re-optimization step – thus it should be solved using *local* methods such as trajectory optimization techniques (Betts, 2010) or elastic bands (Quinlan and Khatib, 1993) (as opposed to fully-fledged global planners). Notice that the initial value of the updated nominal state trajectory, i.e., $\bar{x}(t_i)$, is also an optimization variable above subject to the RCI constraint (4.15). This permits more drastic updates to the nominal trajectory, e.g., to counteract consistently large disturbances. The terminal constraint given by (4.18) is used to ensure *recursive feasibility* for the MPC problem, as addressed by the next lemma. The optimal re-join time for problem MPC is denoted as T_i^* and the corresponding optimal state-input pair is denoted as $(x_T^*(\cdot; x(t_i)), u_T^*(\cdot; x(t_i))) : [t_i, t_i + T] \to \bar{\mathcal{X}} \times \bar{\mathcal{U}}$, of which the segment $[t_i, t_i + \delta)$ is implemented using (2.2), before MPC is re-solved (with $\delta < T$). This defines the *sampled* MPC strategy commonly employed for continuous-time systems. The following lemma establishes recursive feasibility for problem MPC.

Lemma 6 (Recursive Feasibility for MPC)**.** *Suppose problem MPC is feasible at the initial solve step $t_0 = 0$. Then, the problem is feasible for all $t_i, i \in \mathbb{N}_{\geq 0}$.*

Proof. Let $(x_T^*(t; x(t_i)), u_T^*(t; x(t_i))) : [t_i, t_i + T] \to \bar{\mathcal{X}} \times \bar{\mathcal{U}}$ and T_i^* denote the solution to problem MPC at time-step t_i. Then at solve time $t_{i+1} = t_i + \delta$, due to the RCI property associated with the tracking controller, one is guaranteed that the actual state $x(t_{i+1})$ lies within the set $\Omega(x_T^*(t_{i+1}; x(t_i)))$. Consider then the following feasible, but possibly suboptimal solution to problem MPC at solve time t_{i+1}:

$$\bar{x}(\tau) = \begin{cases} x_T^*(\tau; x(t_i)) & \text{for } \tau \in [t_{i+1}, t_i + T) \\ x^*(\tau) & \text{for } \tau \in [T_i^*, T_i^* + \delta], \end{cases}$$

$$\bar{u}(\tau) = \begin{cases} u_T^*(\tau; x(t_i)) & \text{for } \tau \in [t_{i+1}, t_i + T) \\ u^*(\tau) & \text{for } \tau \in [T_i^*, T_i^* + \delta], \end{cases}$$

The state-input trajectory above is simply a concatenation of the tail portion of the previous solution with the nominal motion plan solution, and represents a feasible solution for problem MPC due to the terminal constraint (4.18) (which guarantees that the end-point of $x_T^*(\cdot, x(t_i))$ re-joins the nominal motion plan x^* at time T_i^*). Hence, the feasible set of the MPC problem at solve time t_{i+1} is not empty, which proves recursive feasibility. □

Remark 6. *For $\mathcal{E}(x^*(0), x(0)) > 0$, the Comparison Lemma gives the following analytical, time-varying bound corresponding to the differential inequality* (3.17)*:*

$$\mathcal{E}(x^*(t), x(t)) \leq \left[\sqrt{\mathcal{E}(x^*(0), x(0))}e^{-\lambda t} + \bar{d}\left(1 - e^{-\lambda t}\right)\right]^2.$$

Thus, for the MPC problem solved at time t_i, one can modify constraint (4.15) *to:*

$$\mathcal{E}(\bar{x}(t_i), x(t_i)) \leq \mathcal{E}(x_T^*(t_i; x(t_{i-1})), x(t_i)). \tag{4.19}$$

This constraint states that the Riemann energy between the start of the new nominal MPC trajectory $\bar{x}(t_i)$ and current state $x(t_i)$ is at most the Riemann energy between the current MPC reference (as computed at the previous solve time t_{i-1}) and current state $x(t_i)$. This constraint may then be leveraged along with the shifted time-varying bound:

$$\mathcal{E}(x^*(t), x(t)) \leq \left[\sqrt{\mathcal{E}(\bar{x}(t_i), x(t_i))}e^{-\lambda(t-t_i)} + \bar{d}\left(1 - e^{-\lambda(t-t_i)}\right)\right]^2, \quad t \in [t_i, t_i + T],$$

within the ellipsoid (3.20) *to obtain a time-varying tube that is smaller than the static ellipsoid and therefore, will be less conservative when incorporated within the tightened constraint set given in* (4.14a)*. We demonstrate such an implementation in Section 4.3.*

4.2.2 Overall Algorithm

Algorithm 1 provides pseudocode for our overall approach.

Algorithm 1 Robust planning algorithm

OFFLINE:
Inputs: dynamics model, $\mathcal{U}$ (control input constraints)
Compute: Ω (RCI set), $k(\cdot, \cdot)$ (controller structure)
ONLINE:
Inputs: $x(0)$ (initial state), $\mathcal{X}$ (state constraints), $\mathcal{X}_{\text{goal}}$
Compute nominal (x^*, u^*), such that $x^*(\cdot) \in \overline{\mathcal{X}}$
Initialize: $t_{\text{plan}} \leftarrow 0$
At each time t:
if New obstacles reported or goal region is changed **then**
    Re-plan nominal (x^*, u^*)
else
    if $t - t_{\text{plan}} = \delta$ **then**
        $(x_T^*(\cdot; x(t)), u_T^*(\cdot; x(t)), T^*) \leftarrow \text{MPC}(x(t), x^*, u^*, \Omega)$
        Update $t_{\text{plan}} \leftarrow t$
    end if
end if
Apply control $u_T^*(t; x(t_{\text{plan}})) + k(x_T^*(t; x(t_{\text{plan}}), x(t))$

4.3 Numerical Experiments

We now verify our approach in simulation by continuing with the 6-state planar-quadrotor system introduced in Chapter 3. A more challenging case-study with hardware experiments on a 10-state 3D quadrotor is presented in the next chapter. All simulation code (MATLAB) is available at `https://asl.github.com/RobustMP`.

The system was subject to horizontal cross-winds with effective acceleration up to 0.1 m/s^2. Notably, this system is underactuated and has unstable zero dynamics, and thus represents a challenging system to benchmark our approach. In addition to the state-space constraints introduced during the metric synthesis (see Example 3.3.2), we imposed limits on the the thrust for each propeller to the range $[0.1, 2]\ mg$. The feedback control effort bound $\bar{\delta}_u$ was determined using (4.12) to be 0.6 N (0.13 mg), a fraction of the overall control range.

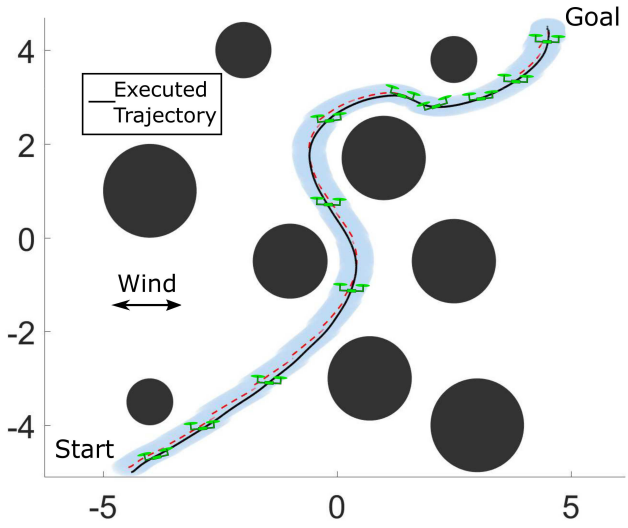


Figure 4.1: A planar quadrotor navigating in real-time through a previously unseen cluttered environment in the presence of a horizontal cross-wind disturbance. A nominal (disturbance-free) trajectory (dashed-red line) is generated online in response to obstacles reported in the environment such that the invariant tube (computed offline) centered around the trajectory does not intersect obstacles. The breakpoints in the tube mark the instances where the nominal path is *locally* re-optimized using MPC as it adjusts to the actual executed trajectory veering to the edges of the tube due to the cross-wind (but still remains within it as guaranteed by our analysis). The spacing between the edge of the tube and the obstacles accounts for the size of the vehicle itself.

Having computed (offline) the RCI mapping, we tested Algorithm 1 on the previously unseen densely cluttered environment in Figure 4.1. The disturbance direction fluctuated (continuously) between left and right to try to push the vehicle into an obstacle. Problem MPC was re-solved every $\delta = 1$s with horizon $T = 2$s using the pseudospectral collocation method and the SNOPT solver, and leveraged the *static* bound given in (3.20). The tracking controller was implemented using zero-order-hold at 200 Hz. Leveraging a suboptimal implementation in MATLAB, the time

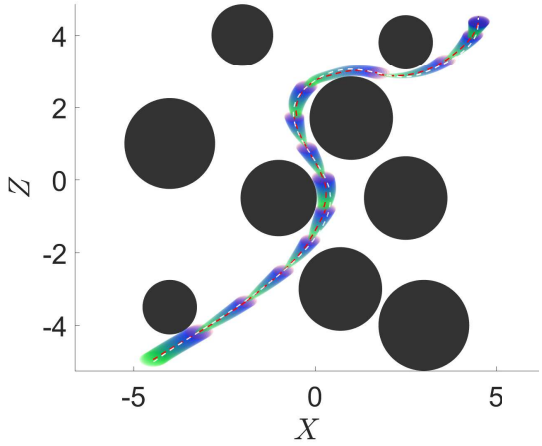
required to compute the tracking controller (on a 3.5 GHz Intel equipped with 16 GB of RAM) is on the order of 3.5 ms. On average, each MPC problem took 0.35 s to solve. This compares favorably with the re-solve time of 1 s. Furthermore, we expect that this can be significantly improved with a more efficient implementation and by using trajectory optimization methods that fully exploit the *local* nature of the problem. We do not report the computation of the nominal trajectory (line 6 in Algorithm 1) since it highly depends on the motion planner used and is not the focus here.

This example provides evidence that Algorithm 1 can be used for the online generation of safe motion plans that can be reliably executed (provided that the nominal motion plan can also be computed in real-time). This example also illustrates the benefits of our method as compared to the funnel library approach (Majumdar and Tedrake, 2017). A pre-computed set of trajectories (as required by (Majumdar and Tedrake, 2017)) would be unlikely to contain a sequence leading from the start to goal while maneuvering through the very tight spaces between obstacles.

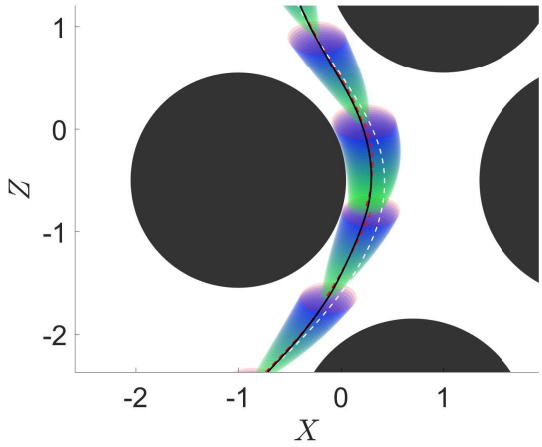
In Figure 4.2, we illustrate the use of time-varying tubes as derived in Remark 6 within the online MPC problem. The MPC lookahead and re-solve times were increased to $T = 4$ s and $\delta = 1.5$ s respectively. We draw attention to the MPC solver making effective use of the time-varying tube to generate tighter nominal trajectories that would have been deemed infeasible using the fixed-size tube.

4.4 Summary

In this chapter we leveraged the strong disturbance rejection properties of CCM-derived feedback controllers to define the robust planning algorithm. In particular, we illustrated how to implement the controller and a receding-horizon planner online, derived bounds on the feedback tracking effort, and validated a base implementation of the algorithm within simulation on a 6-state planar quadrotor model. In the next chapter, we present an extensive numerical and hardware case-study on a more challenging example – a 10-state 3D quadrotor.



(a) Comparison of the initial motion plan (dashed-white line) and the online locally re-optimized trajectory (dashed-red line) leveraging time-varying tube bounds.



(b) Zoomed in view near an obstacle where the locally re-optimized trajectory is able to take a much tighter cut around the obstacle as consequence of significantly tighter time-varying bounds. Also shown is the actual trajectory (solid-black line) which can be seen to remain inside the time-varying tube centered around the re-optimized trajectory (dashed-red line).

Figure 4.2: Implementation of the planar quadrotor example with time-varying tubes. For clarity, the quadrotor graphic has been removed and the obstacles have been inflated by the vehicle size.

Chapter 5

Extended Case Study: Quadrotor

In this chapter we study the limits of the CCM-based robust planning algorithm on a more challenging example – a 3D quadrotor. In particular, we demonstrate the procedure for synthesizing an appropriate dual CCM, provide extensive numerical experiments on the *conservatism* of the computed bounds, discuss computational simplifications of the online controller, and evaluate the performance of the quadrotor in the presence of aerodynamic disturbances induced by unmodeled drag forces.

5.1 Dynamics and Constraints

We adopt the state-space representation $x = (p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z, f, \phi, \theta, \psi)^T$ where position $p = (p_x, p_y, p_z)^T \in \mathbb{R}^3$ and corresponding velocities are expressed in the global inertial (vertical axis pointing down) frame. Adopting the North-East-Down frame convention for the quadrotor body and the XYZ Euler-angle rotation sequence, the attitude (roll, pitch, yaw) is parameterized as (ϕ, θ, ψ) and $f \in \mathbb{R}_{>0}$ is the net (normalized by mass) thrust generated by the four rotors. For the purposes of controller design, we consider as control inputs $u := (\dot{f}, \dot{\phi}, \dot{\theta}, \dot{\psi})^T$. Actual implementation embeds the $\dot{f}$ term within an integrator and the resulting thrust and angular velocity reference are passed to a lower-level controller on the quadrotor that is assumed to operate at a much faster time-scale. Given this parameterization, the dynamics of the quadrotor may be written as:

$$\begin{bmatrix} \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{bmatrix} = ge_3 - f\hat{b}_z = \begin{bmatrix} -f\sin(\theta) \\ f\cos(\theta)\sin(\phi) \\ g - f\cos(\theta)\cos(\phi) \end{bmatrix}, \tag{5.1}$$

where g is the local gravitational acceleration, $e_3 = (0, 0, 1)^T$, and $\hat{b}_z$ is the body-frame z-axis. The dynamics of $(\tau, \phi, \theta, \psi)$ reduce trivially to first-order integrators. We impose the bounds: $(\phi, \theta) \in$

$[-60°, 60°]^2$ and $f \in (0.5, 2)g$, sufficient for executing fairly aggressive maneuvers.

5.2 CCM Computation

Notice that yaw is completely decoupled from these equations of motion. Consider, then, the following block partition model for $W(x)$:

$$W(x) = \left[\begin{array}{c|c|c} W_\perp & W_u & 0_{6\times 1} \\ \hline * & W_\parallel & 0_{3\times 1} \\ \hline * & * & W_\psi \end{array}\right],$$

where $W_\perp \in \mathbb{S}_6^{>0}$, $W_\parallel \in \mathbb{S}_3^{>0}$, $W_\psi \in \mathbb{R}_{>0}$ and $W_u \in \mathbb{R}^{3\times 3}$. Given $B = [0_{4\times 6}\,, I_4]^T$, take $B_\perp = [I_6\,, 0_{4\times 6}]^T$. The weak Killing field condition in (3.23) requires that $W_\perp$ is not a function of (f, ϕ, θ, ψ). Further by noting the translational and yaw invariance of the dynamics, we only parametrize W_u and $W_\parallel$ as functions of (f, ϕ, θ) and pick $W_\perp$ to be a constant positive definite matrix and W_ψ a constant positive scalar. Such a parameterization also reflects the intuition that *differential stabilizability of the quadrotor, as captured by the CCM, should be independent of the quadrotor's position, velocity, and yaw orientation.* Given the block diagonal structure of W, we solve for the top left 9×9 block of the metric using the SOS formulation discussed in Section 3.3 and independently design W_ψ. Once again, the trigonometric terms in the dynamics were approximated using Chebyshev polynomial expansions up to third order.

Figure 5.1 plots the optimal curve for J_{CCM} as a function of λ (for the top left 9×9 block of $W(x)$), using the alternative objective given in Remark 4. The scaling matrix W_s was chosen in order to prioritize the Euclidean error, and set as $\text{diag}(15, 15, 15, 1, 1, 1, 1, 1, 1)$.

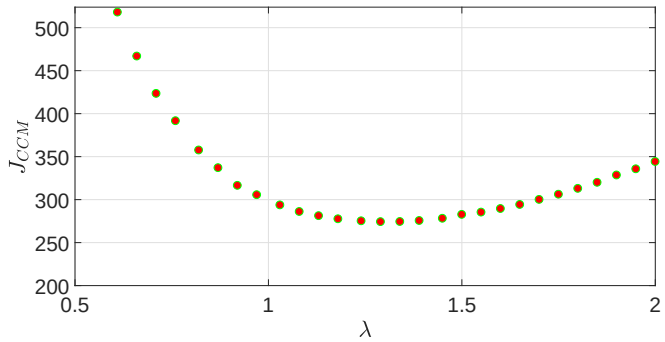


Figure 5.1: Problem $\mathcal{OPT}_{\widehat{CCM}}$ alternative objective as a function of λ.

The optimal contraction rate can be seen to be $\lambda = 1.29$ and the corresponding dual metric $W(x)$ contained 35 unique monomials in (f, ϕ, θ). Assuming a cross-wind acting in *any* direction

with effective acceleration up to 0.1 m/s^2, we used gridding to compute a value of 0.0432 for $\bar{d}$. The resulting projections of ellipsoid (3.20) onto the position coordinates corresponded roughly to a sphere of radius ≈ 8.8 cm. The projection onto the thrust axis yielded the interval $[-0.05, 0.05]g$ and the projections in the (ϕ, θ) axes corresponded to the intervals $[-11.1°, 11.1°]$ and $[-7.45°, 7.455°]$ respectively. Since the projection of the bound along the yaw axis is given by $\bar{d}\sqrt{W_\psi}$ (see Remark 4), we set $W_\psi = 17.3$, yielding the interval $[-10°, 10°]$. The resulting tightened constraints are still permissive of aggressive maneuvers within cluttered obstacle environments, as demonstrated next.

5.3 Simulation under Nominal Disturbances

To verify the controller performance, we randomly initialized obstacle environments for the quadrotor, an example is depicted in Figure 5.2. Trajectory planning was performed by first computing a waypoint path using geometric FMT* (Janson et al., 2015a), and then smoothing this path using polynomial splines with the min-snap algorithm in (Richter et al., 2016). Finally, differential flatness was leveraged to recover the open-loop state and control trajectories. Collision checking was performed by leveraging the configuration space representation of the obstacles, i.e., polytopes, inflated by the size of the quadrotor (approximated as a 20 cm radius ball) and the projection of the tube bound onto position coordinates (a further 8.8 cm radius ball).

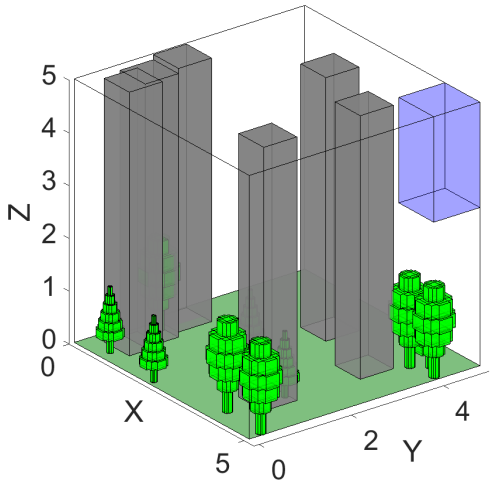
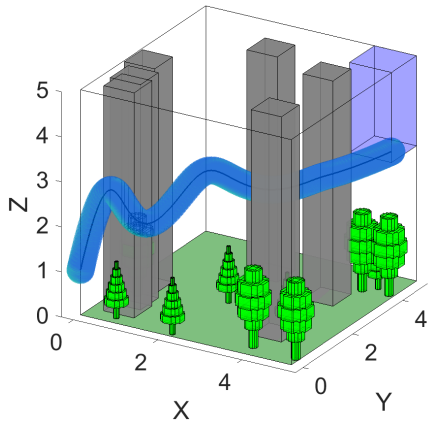


Figure 5.2: Randomly generated obstacle environment with towers and trees; initial position of the quadrotor is $(0, 0, 1)$, corresponding to the leftmost corner. The goal set is depicted as the light blue box.

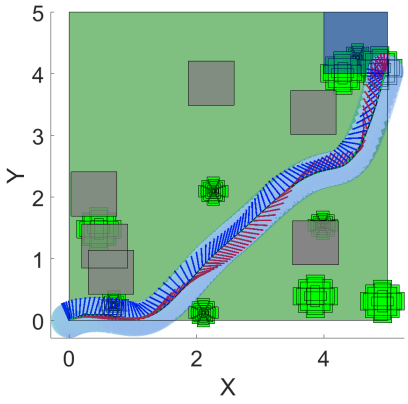
Figure 5.3 shows an example of such a computed trajectory, along with the surrounding tube margin. The maximum speed along this trajectory is 3.00 m/s and the maximum pitch angle is approximately 21.5°. The yaw trajectory was designed to follow the horizontal plane velocity.

Note that both these aspects of planning (waypoint generation using sampling-based planning and polynomial smoothing) are *real-time* algorithms and therefore can be efficiently executed in receding horizon fashion. Robustness is easily accounted for via inflating the obstacles by the tube margin.

Figure 5.4 illustrates the simulation results for the computed trajectory in Figure 5.3 for a variety of disturbance time-series (sinusoidal varying and fixed direction signals with constant magnitude set to $\bar{w}$). The feedback controller is implemented using zero-order-hold at 250 Hz. All errors are observed to respect the theoretically computed bounds.

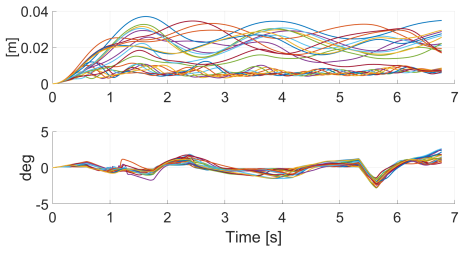


(a) Isometric view.

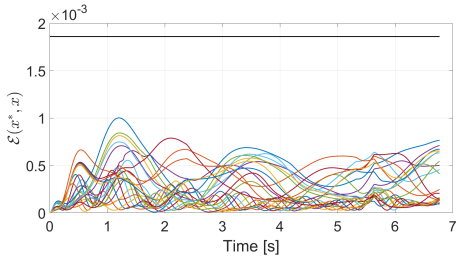


(b) Overhead view.

Figure 5.3: Computed nominal trajectory with attitude depicted using the body-fixed coordinate frame (forward: red, left: blue). The trajectory itself is centered within the depicted invariant ellipsoidal tube (shown inflated by the size of the quadrotor). The overhead view illustrates the tight margins near the beginning and end of the trajectory.



(a) Top: Euclidean norm of translational error (theoretical bound: 8.7 cm); Bottom: Yaw tracking error (theoretical bound: $[-10, 10]^\circ$).



(b) Geodesic energy time-series; theoretical bound ($\bar{d}^2$) illustrated by the flat horizontal line.

Figure 5.4: Time-series tracking error plots for 24 different disturbance time-series for the nominal trajectory in Figure 5.3. As expected, all errors remain within the theoretically computed bounds, ensuring safe execution of the planned path.

5.3.1 Assessing Conservatism

Within this section, we empirically evaluate the conservatism in the computed tracking bounds by planning trajectories assuming the disturbance bound of 0.1 m/s^2 and increasing the *actual* disturbance level during simulation. We randomly generated 100 new trajectories similar to Figure 5.2 with varying obstacle placements. For each of the 100 nominal trajectories, we simulated the 24 disturbance time-series from Figure 5.4 with $\bar{w} \in \{0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$ m/s^2. Figure 5.5 plots the empirical cdf of the tracking errors for different simulation disturbance thresholds. For reference, we also delineate the theoretical tracking bounds computed with the simulation $\bar{w}$ values. The conservatism evaluations, therefore, are: (i) gap between actual tracking errors and the theoretical upper-bound, and (ii) extent of violation of a bound computed with a lower $\bar{w}$.

It can be noted in Figure 5.5(a), that the gap between the largest geodesic energy tracking error observed for a fixed disturbance threshold and the theoretical upper-bound is impressively tight. Indeed, this gap ranged between 1.56 – 30 % of the theoretical bounds. Additionally, in all cases, the tracking errors crossed the theoretical bound computed using a value of $\bar{w}$ that was 0.05 m/s^2 smaller than the simulated level. Both these indicators validate that the analysis in Theorem 2 is in fact practically useful and not overly conservative. The larger gaps between the maximum translational error and the theoretical bound, as observed in Figure 5.5(b), are to be expected since the geodesic energy is a full state error measure.

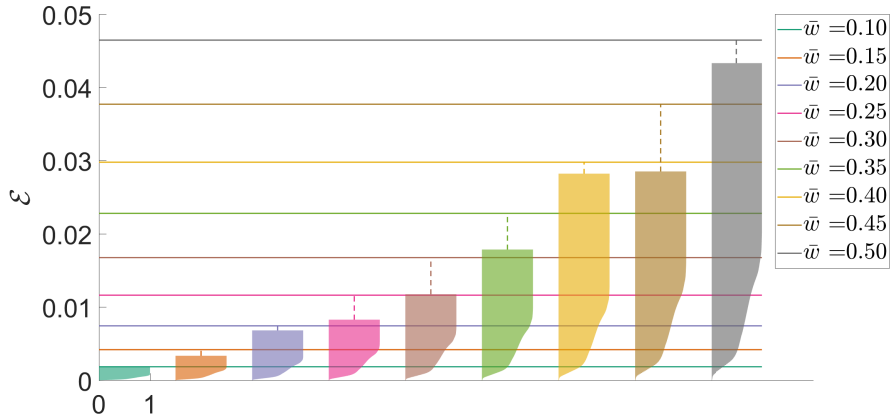
5.4 Hardware Experiments

Finally, we illustrate the approach on an open-source quadrotor platform, shown in Figure 5.6. The quadrotor consists of (i) a standard DJI F330 frame, (ii) Pixhawk autopilot running the estimator and lower-level thrust and angular rate controllers, and (iii) a companion on-board ODROID-XU4 computer running a ROS node for computing the CCM controller (which generates the thrust and angular rate setpoints for Pixhawk). The code for these ROS nodes is available for download at `https://asl.github.com/asl_flight`. There is also an Optitrack motion capture system providing inertial position and yaw estimates at 120Hz, which is fused with the onboard EKF on the Pixhawk.

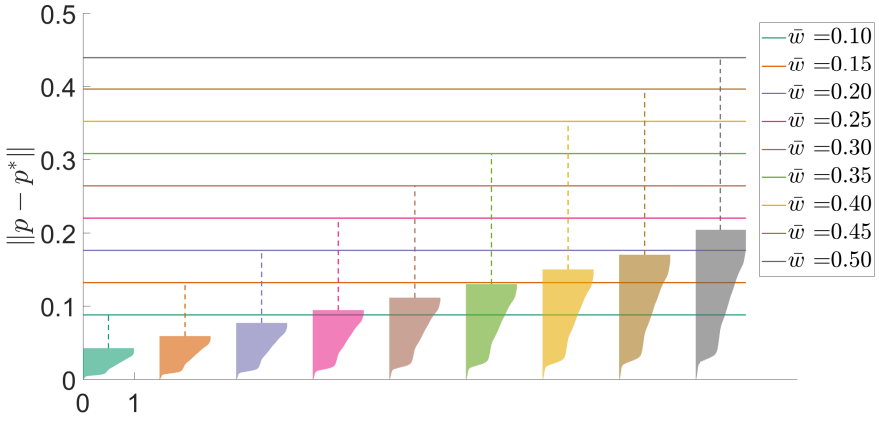
5.4.1 Controller Implementation

CCM Controller

During simulation it was often observed that the computed geodesics at each sampling instant were nearly a straight line between $x^*(t)$ and $x(t)$. Therefore, to further improve the runtime efficiency of the controller (i.e., avoid having to solve problem $\mathcal{OPT}_\gamma$ on the embedded hardware), we investigated the performance of the controller using the straight-line approximation of the geodesic.



(a) Geodesic energy tracking error.



(b) Euclidean norm of translational tracking error.

Figure 5.5: Empirical cdfs (rotated 90° for clarity) of tracking errors with varying simulation disturbance thresholds. Each cdf corresponds to data generated from 100 nominal trajectories and 24 disturbance time-series, for a total of approximately 4.1 million datapoints (equivalently, 4.5 hrs of simulation time) at each disturbance threshold. The horizontal lines indicate the theoretical tracking bounds as a function of $\bar{w}$, linked to the relevant cdfs via the vertical dotted lines. For clarity, the 0-1 range for each cdf is only depicted for the lowest disturbance threshold.

While it is theoretically possible to bound the numerical error resulting from this approximation and the induced error within inequality (4.1) via bounding the Christoffel symbols associated with $M(x)$, such an error analysis is likely to be conservative. Instead, we extracted tuples of $(x^*(t), x(t))$ from the simulations in Section 5.3.1, and performed the following evaluations – see Figure 5.7:

- Compare $\mathcal{E}(\gamma)$ and $\mathcal{E}(c_{\text{sl}})$, where $c_{\text{sl}}(\cdot, t)$ is the straight-line connecting $x^*(t)$ and $x(t)$.
- Magnitude of the induced error (i.e., violation) in inequality (4.1) with k set to the feedback computed using c_{sl}. This is the main evaluation criterion, from a stability performance perspective.

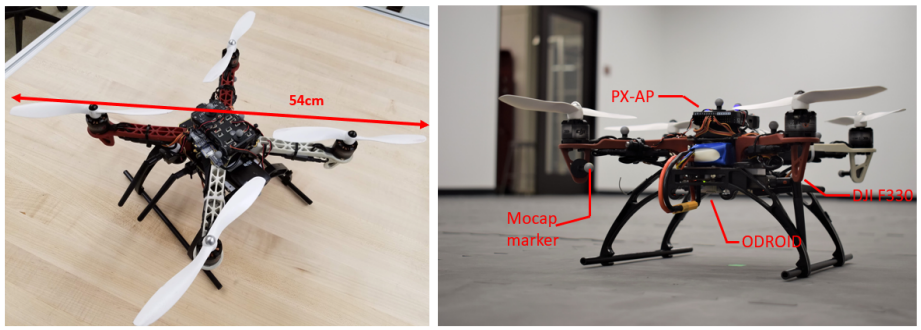
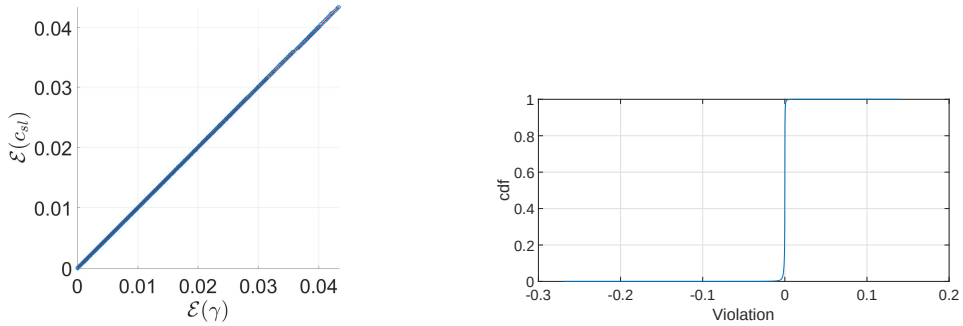


Figure 5.6: Quadrotor experimental platform, equipped with Pixhawk autopilot (PX-AP) for low-level (thrust and angular rate) control, and ODROID companion computer for planning and CCM controller.

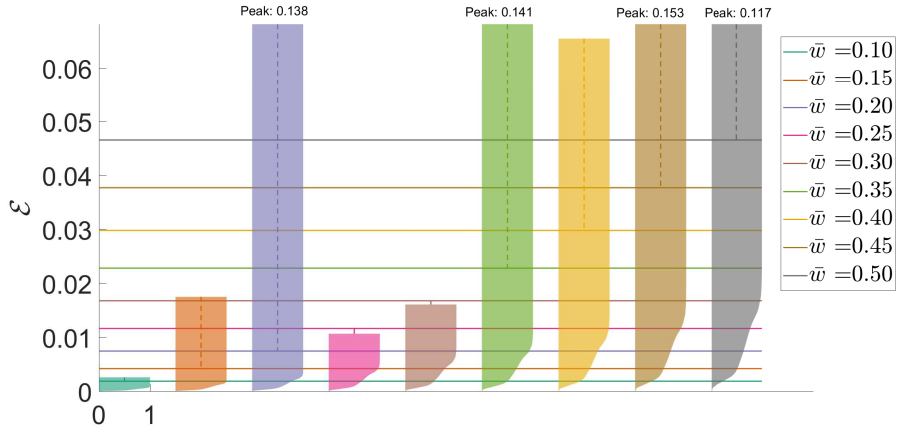


(a) Compare $\mathcal{E}(\gamma)$ and $\mathcal{E}(c_{\text{sl}})$. The datapoints lie almost perfectly along the $\mathcal{E}(c_{\text{sl}}) = \mathcal{E}(\gamma)$ diagonal line.

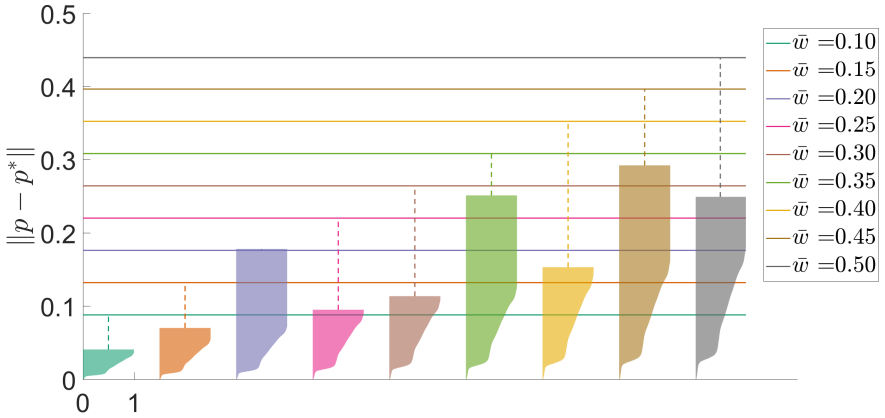
(b) cdf of violation of inequality (4.1) with k set to the feedback computed using c_{sl}.

Figure 5.7: Validating use of straight line approximation of geodesic to compute the online tracking controller. In the right subfigure, a negative value indicates slack, while a positive value indicates violation of the stability inequality (4.1). The steep saturation of the curve just past 0 indicates relatively inconsequential implications for the violation of the stability inequality.

From Figure 5.7(a), we observe that $\mathcal{E}(c_{\text{sl}})$ almost perfectly matches $\mathcal{E}(\gamma)$, while Figure 5.7(b) illustrates that the resulting violation in the stability inequality (4.1) is practically negligible. To account for potential time-compounding effects of using the straight-line approximation (the evaluations in Figure 5.7 are pointwise in time), we repeated all simulations from Section 5.3.1 using c_{sl} in place of the geodesic to compute the tracking controller. Figure 5.8 presents the resulting tracking error cdfs. From Figure 5.8, one notices that the theoretical tracking bounds are indeed violated due to the compounding effect of the error induced within the stability inequality (4.1). However, the cdfs evaluated at the theoretical tracking bounds ranged from 0.999940 – 1.0. Equivalently, the cumulative time spent in violation of the geodesic energy theoretical bound, over all the simulations represented within Figure 5.8, was 7.044 s. Compared to the 40.71 hrs of simulation time represented in this figure, this is equivalent to a proportional violation of 0.004%. The translational tracking error however, remained always below the theoretical upper bound. Therefore, the straight-line approximation of the geodesic is a relatively mild simplification for this quadrotor example.



(a) Geodesic energy tracking error.



(b) Euclidean norm of translational tracking error.

Figure 5.8: Empirical cdfs (rotated 90° for clarity) of tracking errors with varying simulation disturbance thresholds, from using a straight-line approximation of the geodesic for computing the tracking controller. As before, the horizontal lines indicate the theoretical tracking bounds as a function of $\bar{w}$, linked to the relevant cdfs via the vertical dotted lines.

Lower-Level Pixhawk Controller

The lower-level controller on the Pixhawk autopilot is tasked with tracking the thrust and angular rate control setpoints generated by the CCM controller. The Euler rate commands from the CCM controller are converted into desired body-rates onboard Pixhawk and fed into the stock PID control loop. For thrust control, the commanded normalized thrust f_c from the CCM controller and the estimated inertial acceleration $\hat{\dot{p}}$ (obtained via moving-average finite-differencing) are used to first compute an error:

$$e_f = (f_c - ge_3 \cdot \hat{b}_z) - (-\hat{\dot{p}} \cdot \hat{b}_z),$$

which is simply the difference in the desired and actual inertial accelerations, projected along the instantaneous $\hat{b}_z$ axis. This error is then converted into a normalized throttle command $\tau \in [0, 1]$ using the following discrete-time recursion:

$$\tau(t_k) = \tau(t_{k-1}) + k_p e_f(t_k),$$

where the time indices $t_k, k = 0, 1, 2, ...$ delineate the Pixhawk sampling times. The CCM controller on board the ODROID is run at 250 Hz, while the Pixhawk control loop runs between 350-400 Hz. The controller above is similar to the thrust controller proposed in (Tal and Karaman, 2018), and corresponds to an *iterative* scheme where the proportional error term is used to simply *correct* the previous throttle command, precluding the need to estimate the complex mapping between desired thrust and throttle, or rely on the popular yet oversimplifying quadratic model of this relationship.

5.4.2 Calibrating Disturbance Bound

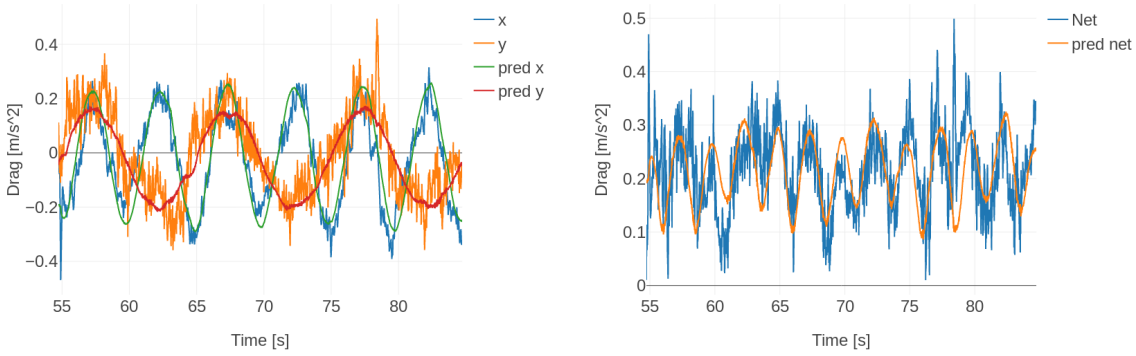
The role of aerodynamic disturbance for experiments was played by the (neglected in eq. (5.1)) drag force. To calibrate an upper bound for planning, we flew two calibration trajectories: a figure-eight (presented in this section) and a "race-course" (similar to the one in the next section). We describe the figure-eight trajectory here. The nominal trajectory was set as: $p_x(t) = r_x - r_x \cos(\omega t)$, $p_y(t) = r_y \sin(2\omega t)$, $p_z(t) = h$, where we set $r_x = 1$ m, $r_y = 0.7$ m, $\omega = 2\pi/10$, and a constant altitude $h = 1.5$ m. The drag-force was estimated based on the following augmented translational dynamics:

$$\ddot{p} = ge_3 - f\hat{b}_z + R\underbrace{(d_0 - DR^T\dot{p})}_{:=f_d}, \tag{5.2}$$

where R is the rotation matrix, $D = \text{diag}(\mu_x, \mu_y, 0)$, and $d_0 = (d_{0_x}, d_{0_y}, 0)^T$. Here d_0 plays the role of a fixed perturbing force stemming from propeller misalignment, and D models the linear velocity drag coefficients, commonly accepted to be the dominant component of drag within our flight regime; for instance, see (Kai et al., 2017; Faessler et al., 2018). As in (Faessler et al., 2018), we neglect the body z component of drag.

To estimate the net body-frame drag f_d, we leveraged a smoothed finite-differenced estimate of $\ddot{p}$, the estimated rotation matrix R, and the *commanded* normalized thrust f_c in place of f. This last substitution is justified courtesy of the lower-level thrust controller presented in the preceding section. Figure 5.10 shows a series of plots from the figure-eight experiment, including (a) XY desired and actual traces, (b) translational tracking errors, (c) body-frame velocity, (d) estimated body-frame drag, (e) Euclidean norm of net drag, and (f) geodesic energy.

Concatenating the data from the figure-eight and race-course trajectories, we used least-squares to compute the following estimates: $\mu_x = 0.302, \mu_y = 0.288, d_{0_x} = -0.008, d_{0_y} = -0.026$. Figure 5.9 plots an overlap of the estimated and predicted drag for the figure-eight trajectory, and illustrates good agreement. In the next section, we present additional plots of estimated and predicted drag from the planning experiments to further validate the use of this drag model as a viable disturbance model.

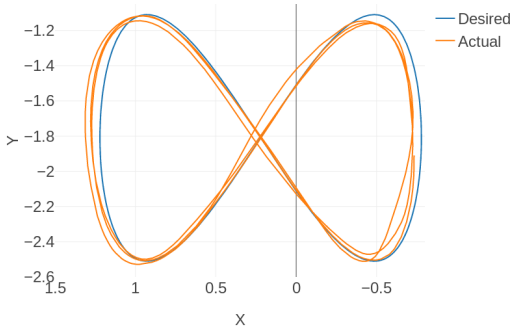


(a) Predicted and estimated drag in the body frame.

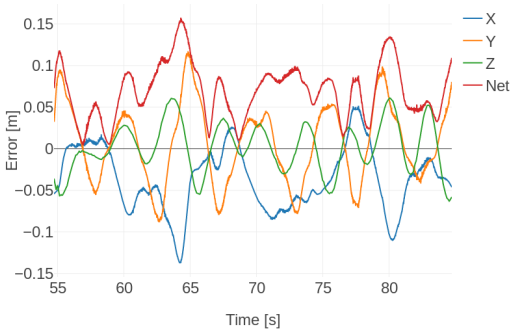
(b) Net Euclidean norm of predicted and estimated drag.

Figure 5.9: Comparison of linear drag model predictions and estimated drag for the figure-eight calibration flight. Note that the entire flight (including the cycles at slower speed) and the race-course trajectory were used for estimating the parameters of the linear drag model.

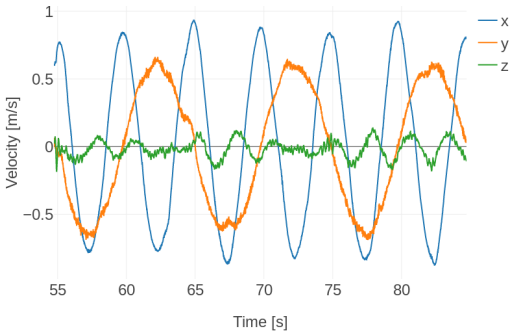
Remark 7. *In these experiments, the unknown but bounded perturbing force was taken to be the linear drag model. However, given the recent advancements in leveraging drag models within planning, for instance, as in (Faessler et al., 2018), one may also incorporate the learned drag model within the nominal dynamics and compute a drag-compensated CCM. While this would necessitate including coupled velocity and yaw dependence within the CCM, it would allow the separation of drag from the unmodeled aerodynamic disturbances, thereby reserving the disturbance bound margin for purely exogenous effects. For the purposes of illustrating the methodology, we reserve this extension for future work.*



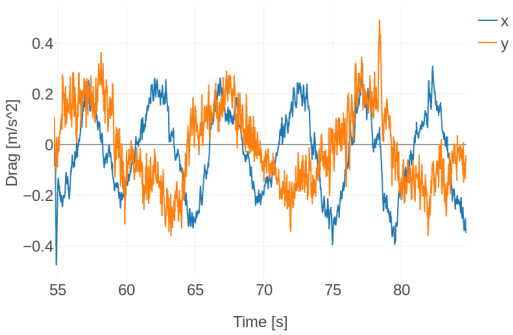
(a) XY trace of desired vs actual trajectories.



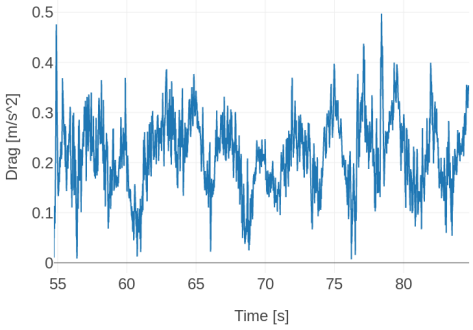
(b) Translational errors. Net (Euclidean norm) error statistics: 7.66 cm RMS, 15.76 cm max.



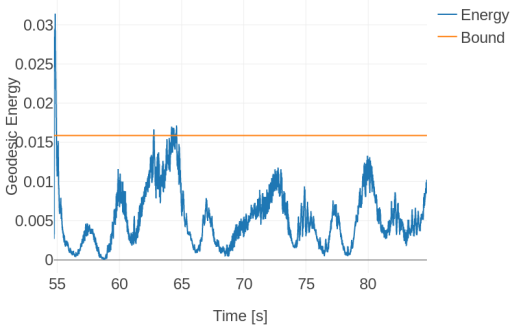
(c) Velocity in body frame.



(d) Estimated drag along body axes.



(e) Net (Euclidean norm) estimated body drag.



(f) Geodesic energy, along with adjusted bound computed using a max drag of 0.3 m/s^2.

Figure 5.10: Experiment results for drag calibration flight using fixed-yaw figure-eight trajectory. The experiment was performed by ramping up speed over two cycles; the plots shown correspond to three cycles at the final speed (period of 10s). In subplot (f), corresponding to geodesic energy, the initial spike around 55 s corresponds to a jump in speed for the nominal (reference) trajectory. Following the spike, the error stays below the (drag-adjusted) theoretical bound illustrated by the horizontal line.

5.4.3 Robust Planning

Equipped with a calibrated disturbance model, we ran the robust trajectory planner (geometric FMT* plus polynomial spline smoothing) introduced in the simulation section on the "race-course" test environment shown below in Figure 5.11. The computed trajectory was setup to create a challenging loop through the obstacle course through intermediate waypoints placed, for example, at the center of gates and in between the poles. To ensure robustness with respect to the drag model, a single constant was used to scale time (effectively, speed) along the trajectory such that the drag-adjusted invariant tube was collision free. Importantly, *the drag model was not re-estimated for these test trajectories*. The computed trajectory along with the invariant tube is shown in Figure 5.13. Figure 5.12 plots the nominal body-frame velocity and expected drag.

Figure 5.11: Quadrotor "race-course" test environment.

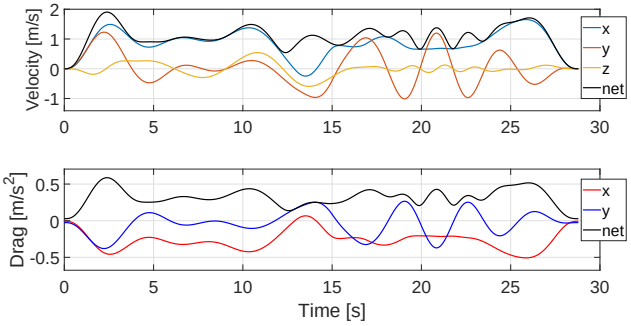
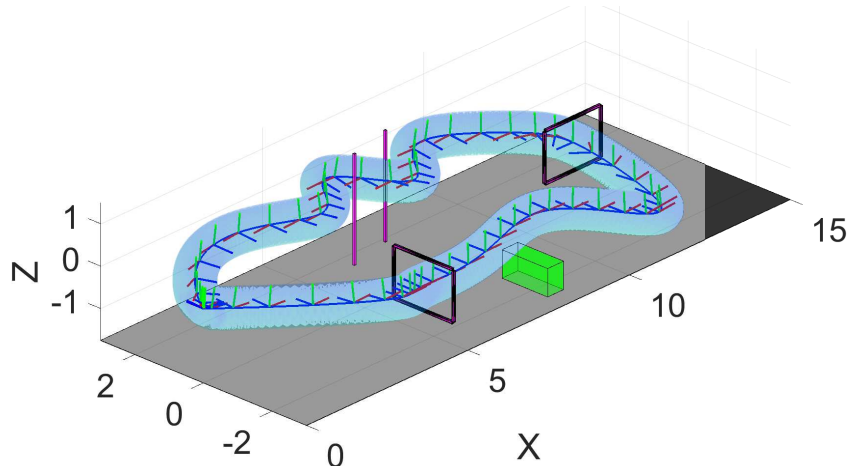


Figure 5.12: Top: Body-frame and net velocity; Bottom: Predicted body-frame and net drag force, along nominal trajectory. Maximum expected drag: 0.59 m/s^2.

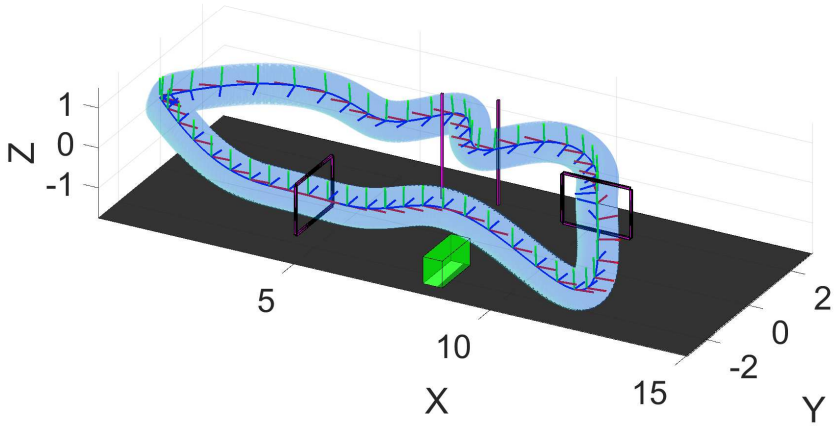
The maximum expected drag along the nominal trajectory is 0.59 m/s^2, corresponding to a top speed of 1.9 m/s. This results in an adjusted translation error bound of 52 cm and a geodesic energy bound of $\bar{d}^2 = 0.0648$. The trajectory in Figure 5.13 was computed with a combined collision margin of 52 cm plus an additional 27 cm to account for the size of the quadrotor (see Figure 5.6).

Following the computation of the nominal trajectory, the quadrotor was flown through the obstacle loop three times, with a 25% increase in speed (respectively, decrease in lap time) with each

successive lap. A time-lapse[1] during the fastest lap is shown in Figure 5.14.



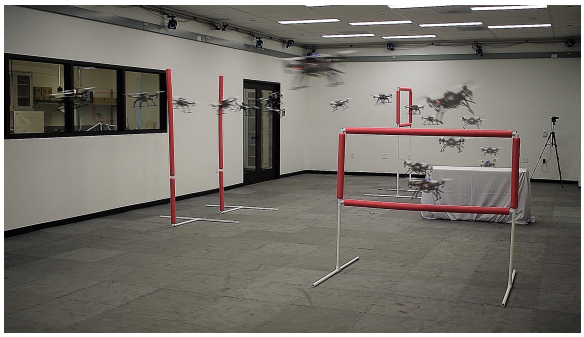
(a)



(b)

Figure 5.13: Computed nominal trajectory with attitude depicted using the body-fixed coordinate frame (forward: red, left: blue). The trajectory itself is centered within the depicted invariant ellipsoidal tube. The views illustrate the tight margins through the obstacles.

[1]Videos of the flights can be found at `https://www.youtube.com/playlist?list=PL8-2mtIlFIJo5n4J4wgP6zdtEGKYeTYO1`.



(a) Time-lapse through the full obstacle course.

(b) Time-lapse through the slalom maneuver past the poles.

Figure 5.14: Time-lapse of quadrotor through the obstacle course during the fastest lap. Top speed achieved: 3.81 m/s.

Note that the trajectory was not re-computed during the experiment since the purpose here was to (i) illustrate that the theoretically computed tracking error bounds are indeed physically meaningful and achievable on an actual hardware testbed, and (ii) evaluate the conservatism in the tracking bounds via greater experienced disturbance than planned for. In particular, only the first speed setting is theoretically robustly collision-free, in that the invariant tube computed using the max expected drag is collision-free. The subsequent laps executed at higher speeds have significantly higher expected drag and the resulting scaled tubes are not collision-free. Given the simulation results presented in Figure 5.5(b) however, we anticipated (and observed) collision-free execution at the higher speeds as well.

The experiment results for the three laps are presented in Figures 5.15– 5.17, and comprise (i) XY desired and actual traces, (ii) translational and geodesic energy tracking errors, and (iii) estimated and predicted body-frame and net drag. The statistics for the three laps are presented in Table 5.1.

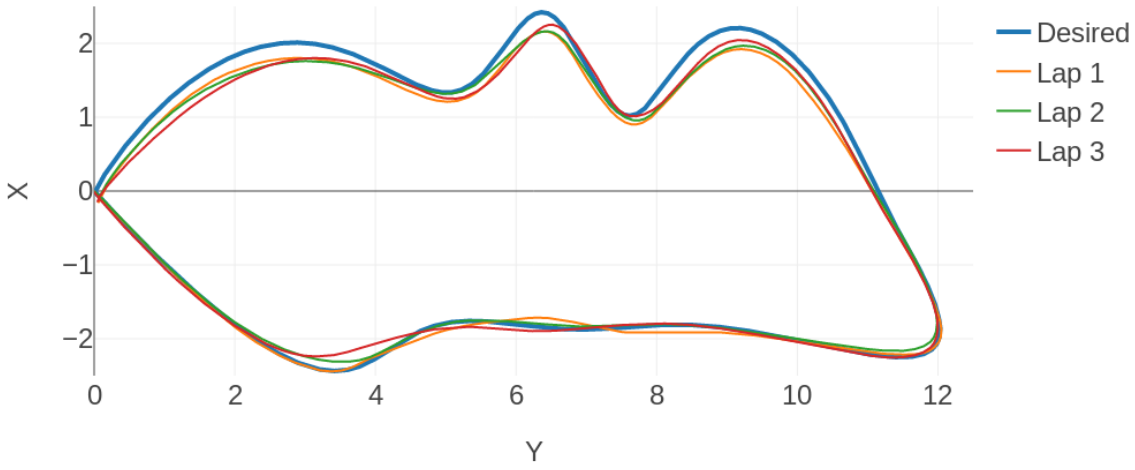
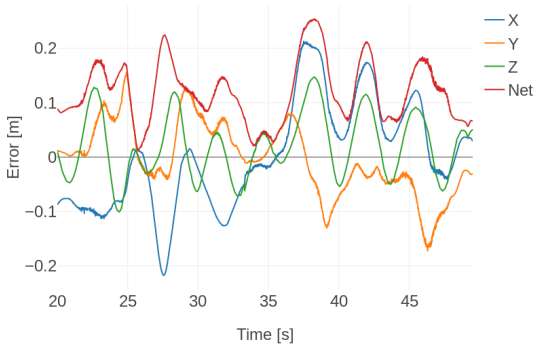


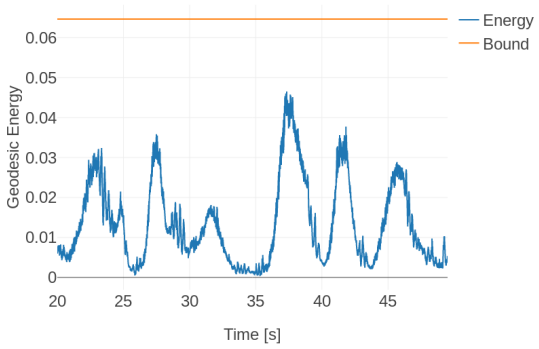
Figure 5.15: XY Trace of desired and actual followed trajectories. Lap direction: counter-clockwise.

	Flight Extremes			**Bounds**		**Errors rms (max)**	
	Speed [m/s]	Bank [deg]	Drag [m/s^2]	$\mathcal{E}$ -	$\|p - p^*\|$ [cm]	$\mathcal{E}$ -	$\|p - p^*\|$ [cm]
Lap 1	1.91	11.46	0.59	0.0648	52	0.017 (0.044)	13.18 (25.4)
Lap 2	2.54	19.8	0.78	0.112	68	0.025 (0.074)	14.51 (30.17)
Lap 3	3.81	39.04	1.15	0.247	101	0.036 (0.124)	16.24 (38.91)

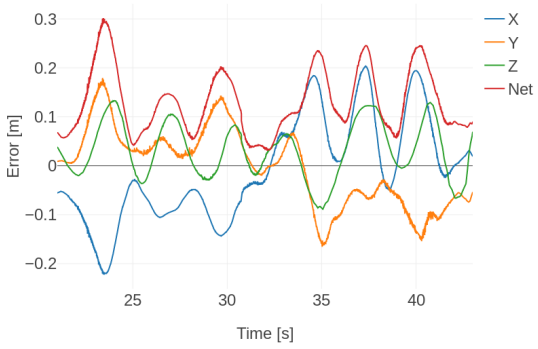
Table 5.1: Nominal trajectory extremes (max drag corresponds to the prediction from the linear drag model), drag-adjusted bounds, and actual flight error statistics. All three laps respect the theoretical upper-bounds computed prior to the flights.



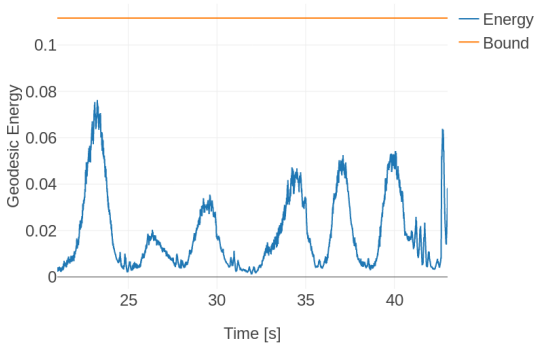
(a) Lap 1 translational error. Error rms: 13.18 cm, max: 25.4 cm, bound: 52 cm.



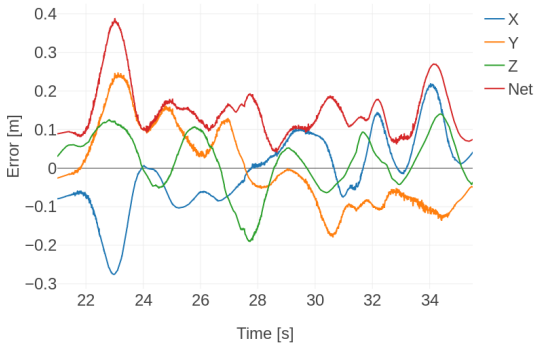
(b) Lap 1 geodesic energy. Error rms: 0.017, max: 0.044, bound: 0.0648.



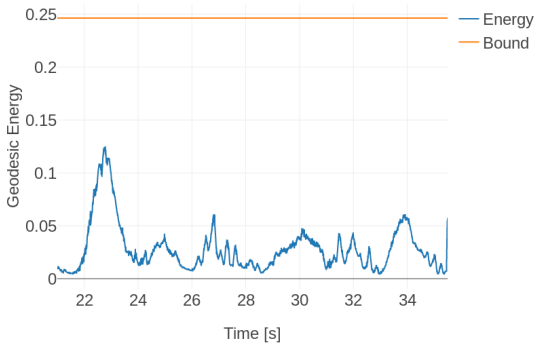
(c) Lap 2 translational error. Error rms: 14.51 cm, max: 30.17 cm, bound: 68 cm.



(d) Lap 2 geodesic energy. Error rms: 0.025, max: 0.074, bound: 0.112.

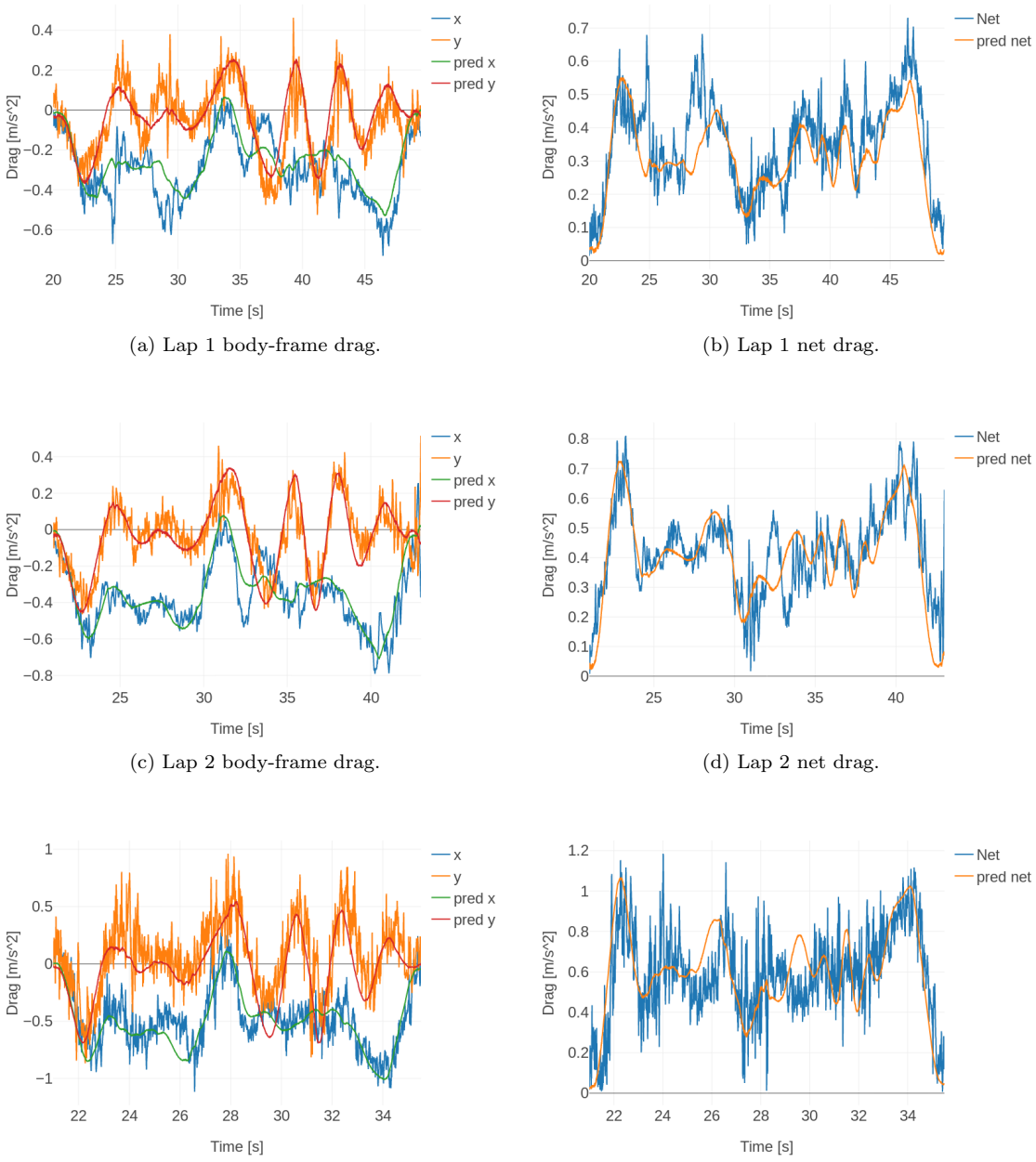


(e) Lap 3 translational error. Error rms: 16.24 cm, max: 38.91 cm, bound: 101 cm.



(f) Lap 3 geodesic energy. Error rms: 0.036, max: 0.124, bound: 0.247.

Figure 5.16: Validation of translational (left column) and geodesic energy (right column) error upper-bounds.



(a) Lap 1 body-frame drag.

(b) Lap 1 net drag.

(c) Lap 2 body-frame drag.

(d) Lap 2 net drag.

(e) Lap 3 body-frame drag.

(f) Lap 3 net drag.

Figure 5.17: Validation of predicted drag model with parameters fixed *a priori to all laps*.

The key takeaway from the preceding numbers and plots is that all theoretical (a priori) geodesic energy and translational error tracking bounds computed using the calibrated drag model are validated within all three flights. Additionally, we note the following interesting observations. First, the

peaks in the geodesic energy curves line up quite well with the local peaks in aerodynamic disturbance due to either drag model or ground/interaction effects. Indeed, as the video illustrates, there are notable perturbations in the vicinity of the box (see also Figure 5.18), ground, and poles. Second, on each successive lap, as the trajectory becomes more aggressive, the geodesic energy tracking error becomes more concentrated at smaller fractions of the theoretical upper bounds. This is potentially due to the fact that the min-norm formulation of the feedback controller in Section 4.1 possesses a "minimally invasive" property akin to trigger or event-based control, in that, the optimal feedback can be zero. However, for the faster trajectories, the feedback is non-zero for a greater proportion of the time, thereby resulting in proportionally tighter tracking. Further, we observe that the geodesic energy at higher velocities would violate the bounds generated for lower velocities. This suggests that the bounds computed during CCM synthesis are not overly conservative, which confirms the numerical results presented in Section 5.3.1.

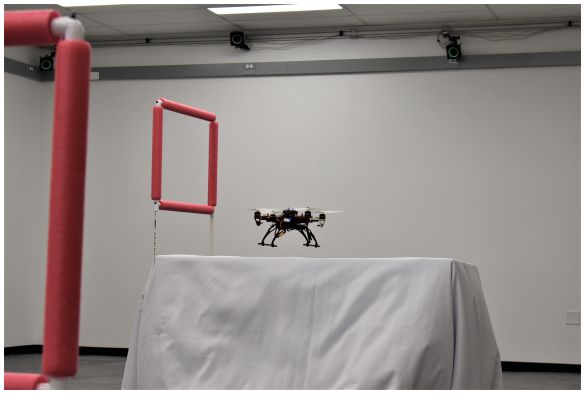


Figure 5.18: Close-up of quad passing near the box obstacle and experiencing complex (unmodeled) aerodynamic disturbances due to the downwash interaction.

Finally, we stress that the purpose of this experiment was to illustrate the feasibility of converting the rich theoretical analysis of our framework into a practical tool capable of running on modern hardware, and obtain true validation of the complete planning methodology. Further performance improvements may be obtained through incorporating richer dynamic models and leveraging higher rate dedicated controllers.

5.5 Summary

In this chapter, we validated the rich theoretical analysis developed using CCMs on a challenging example, and in the process, thoroughly investigated the limits of performance of the controller and the conservatism in the computed invariance bounds. We believe that the modular approach presented herein for automatically synthesizing feedback controllers that are optimized for robust performance, and accompanied by guarantees readily integrable into existing planning algorithms,

serves as a valuable tool for a practitioner to systematically balance performance and safety.

With this chapter, we close our discussion of the robust feedback control design portion of the robust planning framework, and move our attention to the real-time generation of the nominal trajectory itself, for the unperturbed dynamics. Specifically, the examples discussed so far leveraged direct trajectory optimization using collocation (an approach that is not scalable for real-time implementation) and differential flatness. In the next chapter, we illustrate a more general methodology for generating nominal motion plans using model-mismatch.

Chapter 6

Real-Time Planning using Model Mismatch

In this chapter we address the challenge of enabling fast real-time planning for complex nonlinear systems using simplified lower-dimensional models, *in the absence of disturbances*. Essentially, the material presented within this chapter concerns the task of generating the dynamically feasible nominal trajectories that act as the *reference* for the robust CCM-based feedback tracking controller outlined in the previous few chapters. In this chapter however, a *nominal plan* is a state-control trajectory associated with a simplified lower-dimensional model, which is "tracked" using a feedback controller derived for the full system dynamics. The result of this tracking is a state-control trajectory for the *full system dynamics* that is dynamically feasible and collision-free (and suitable as a reference for tracking in the presence of disturbances with a robust feedback controller). Specifically, we make three main contributions. First, we provide an SOS-based formulation of tracking a nominal motion plan computed via a simplified lower-dimensional model. Second, leveraging the SOS formulation, we devise a bilinear optimization algorithm to jointly compute, offline, a *trajectory-independent* feedback tracking controller and the corresponding tracking error bound (TEB). Finally, we demonstrate our approach within simulation on four examples, two of which feature models that are beyond the reach of Hamilton-Jacobi (HJ) techniques.

6.1 Problem Formulation

Consider a relatively high-fidelity model of a robotic system, referred to as the "tracking model," and a lower-fidelity model, referred to as the "planning model," described, respectively, by the ordinary differential equations (ODEs)

$$\dot{s} = f^s(s, u^s), \quad u^s \in \mathcal{U}^s, \qquad\qquad \dot{p} = f^p(p, u^p), \quad u^p \in \mathcal{U}^p, \tag{6.1}$$

where $s \in \mathbb{R}^{n_s}$ is the tracking model's state (or "tracking state"), and $u^s \in \mathbb{R}^{m_s}$ is its control input (or "tracking control"), constrained to lie within the compact set $\mathcal{U}^s$. Similarly, $p \in \mathbb{R}^{n_p}$ is the planning state and $u^p \in \mathbb{R}^{m_p}$ is the planning control input, constrained to lie within the compact set $\mathcal{U}^p$. The planning model can be thought of as a coarser or lower-fidelity model for which motion planning can be done effectively in real time. For both ODEs in equation (6.1), we assume that the dynamics are Lipschitz continuous in the state for fixed controls, so that given any measurable control function, a unique trajectory exists for each system (Coddington and Levinson, 1955). Furthermore, we assume that *the planning state p is a strict subset of the tracking state s*, in particular $n_p < n_s$ (this assumption allows one to clearly relate the tracking and planning state, as discussed in Section 6.2.1), and that the dynamics for the tracking model are *control affine* (a rather mild condition that significantly simplifies the SOS formulation, as discussed in Section 6.2.3).

Crucially, trajectories generated by using the planning model may not be dynamically feasible with respect to the tracking model. Thus, a tracking controller that accounts for the high-fidelity dynamics of the robotic system may not be able to track these trajectories exactly and some (possibly large) tracking error may be incurred. Figure 6.1 illustrates this behavior, for the case of a 4D car model used to track a 2D single integrator model. Specifically, a planning algorithm is used to "quickly" find a nominal motion plan for the planning model, depicted by the dashed blue line. The robot, represented more accurately by the tracking model, strives to track the nominal motion plan as closely as possible, giving rise to the red trajectory. The presence of a minimum turning radius constraint in the tracking model prevents the robot from tracking the sharp corners exactly, thereby yielding a tracking error.

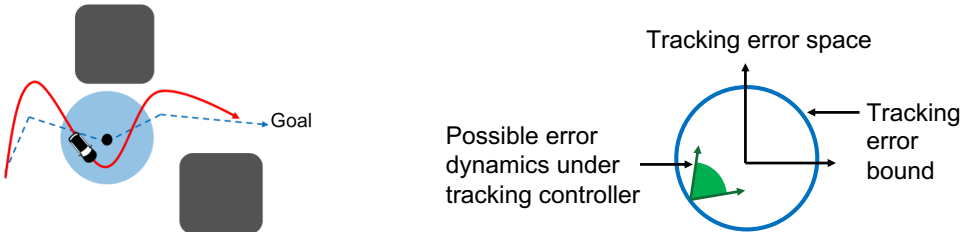


Figure 6.1: *Left: Planning with model mismatch.* The tracking system (car, red trajectory) follows a motion plan computed via a low-fidelity dynamical model (blue trajectory). Tracking with model mismatch while maintaining safety requires keeping the maximum tracking error bound (TEB) below a certifiable value. *Right: TEB certification.* The TEB is characterized by the property that on its boundary, all possible error dynamics resulting from the nominal trajectory point inwards under some tracking controller, so that the TEB is never exceeded.

Such a decoupling approach is quite common in the field of motion planning (LaValle, 2011), yet very few results exist on how to account for the resulting tracking error. In particular, we advocate

that the computation of the nominal motion plan via the planning model should account for the tracking error in order to ensure safe execution (i.e., tracking) by the real system, described by the tracking model. Accordingly, the objective of this chapter is to devise an efficient and scalable algorithm to compute (i) a *trajectory-independent* TEB (represented by the light blue circle in Fig. 6.1, left), which the planner can use as a "safety buffer" to ensure that the actual trajectory is collision free, and (ii) a feedback tracking controller to track such a nominal motion plan while respecting the TEB. Intuitively, the tracking controller should have the property that on the boundary of the TEB, the error dynamics are driven "inwards," *regardless of the nominal motion plan* (depicted in Fig. 6.1, right).

6.2 SOS-Based Bounded Tracking

6.2.1 Relative System

Given the tracking and planning models as in equation (6.1), we define the *relative system state* $r \in \mathbb{R}^{n_r}$ as $r := \phi(s,p)(s - Qp)$, where $\phi(\cdot,\cdot) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_r \times n_s}$, with $n_r \leq n_s$, is a matrix-valued map with bounded derivatives, and $Q = \left[I_{n_p}, O_{n_p \times (n_s - n_p)}\right]^\top$ is a projection matrix from $\mathbb{R}^{n_p}$ to $\mathbb{R}^{n_s}$ (I and O denote the identity and zero matrices, respectively). In the definition of Q, we leveraged the assumption that the planning state p is a strict subset of the tracking state s. We make the following assumption on the dynamics of the relative system state, which is central for our approach:

Assumption 1 (Relative System Dynamics)**.** *The relative system dynamics can be written as a Lipschitz continuous function f^r of (r, u^s, u^p), that is, $\dot{r} = f^r(r, u^s, u^p)$.*

Lipschitz continuity is necessary for the existence and uniqueness of solutions to the ODEs. The structural property in Assumption 1 is satisfied for a number of dynamical models of mobile robotic systems. For example, for ubiquitous cases where the planning model has integrator dynamics, Assumption 1 is satisfied by simply selecting $\phi(\cdot,\cdot)$ as the identity map. However, in general, it is difficult to characterize the conditions on the planning/tracking models such that Assumption 1 is satisfied. Therefore, in Table 6.1, presented at the end of this chapter, we provide a "catalog" of tracking and planning models fulfilling Assumption 1 for a representative set of robotic systems, along with the map ϕ that guarantees fulfillment of Assumption 1 and the resulting relative system dynamics. Specifically, the relative system states in the 6th column are obtained by combining the tracking state in the 2nd column, planning state in the 4th column, and the transformation map $\phi(\cdot,\cdot)$ in the 5th column. Henceforth, we will consider planning/tracking models that fulfill Assumption 1, and refer to the system $\dot{r} = f^r(r, u^s, u^p)$ as the *relative system*.

Finally, we define the *error state* e as the relative system state *excluding* the absolute states of the tracking model, and the *auxiliary state* η as the relative system state *excluding* the error state.

Hence, $r = (e, \eta)$. For instance, for the case of a 5D car model used to track a 3D Dubins car model (second example in Table 6.1), $e = (x_r, y_r, \theta_r)$ and $\eta = (v, \omega)$. Roughly speaking, considering relative system dynamics allows one to reformulate the tracking problem as a stabilization problem to the origin.

6.2.2 Optimization Problem

In order to define a notion of *tracking error bound* (TEB), let $c : r \in \mathbb{R}^{n_r} \mapsto \mathbb{R}^{n_{\tilde{r}}}$ be a smooth mapping representing a vector of quantities we wish to remain bounded, with $n_{\tilde{r}} \geq n_r$; henceforth, we will refer to $c(r)$ as the *bounded state*. The simplest example for $c(\cdot)$ is the identity map. However, for more complex planning/tracking models, the mapping $c(\cdot)$ can be more sophisticated to better capture the structure of the planner/tracker dynamics pair – some examples are provided in Table 6.1. The goal then is to find a closed and bounded set $\mathcal{B}$ in the bounded state such that $c(r(t)) \in \mathcal{B}$ for all $t \geq 0$ (as an illustration, see the light blue ball in Figure 6.1). Such an invariance condition is captured by the implication

$$c(r(0)) \in \mathcal{B} \Rightarrow c(r(t)) \in \mathcal{B}, \quad \forall t \geq 0. \tag{6.2}$$

To parameterize this set, let ρ be a positive constant and $\tilde{V} : \tilde{r} \in \mathbb{R}^{n_{\tilde{r}}} \mapsto \mathbb{R}$ be a smooth function with a bounded $\rho-$sublevel set. Also, define the function $V : r \in \mathbb{R}^{n_r} \mapsto \mathbb{R}$ as simply the composition $\tilde{V}(c(\cdot))$. Suppose that the following implication is true:

$$V(r) = \rho \quad \Rightarrow \quad \dot{V} = \frac{\partial \tilde{V}(\tilde{r})}{\partial \tilde{r}}\bigg|_{\tilde{r}=c(r)}^{T} \frac{\partial c(r)}{\partial r} f^r(r, u^s, u^p) < 0.$$

That is, there exists some tracking control u^s such that the time-derivative of V on the boundary $V(r) = \rho$ is negative. It follows that the set $\{r : V(r) \leq \rho\}$ is invariant. Then, the set $\{c(r) : \tilde{V}(c(r)) \leq \rho\}$ is a valid error bound $\mathcal{B}$ in the sense of equation (6.2). An illustration of this idea is provided in Fig. 6.1 (right).

Remark 8. *Note that the set $\{c(r) : \tilde{V}(c(r)) \leq \rho\}$ is equivalent, up to a translation and scaling factor, to the set $\{c(r) : \alpha\left(\tilde{V}(c(r))) + \beta\right) \leq \alpha(\rho + \beta)\}$, where $\beta \in \mathbb{R}$ and $\alpha > 0$. To eliminate this redundancy, we impose, without loss of generality, the conditions $\tilde{V}(0) = 0$ and $\rho = 1$. The choice $\tilde{V}(0) = 0$ essentially corresponds to a translational adjustment that "centers" the set $\mathcal{B}$ on a zero bounded state.*

In the context of this chapter, set $\mathcal{B}$ is used as a "buffer" that a planner should consider along each candidate nominal motion plan, to ensure that the *realized* tracking trajectory is safe, that is, collision free. Clearly, the search for set $\mathcal{B}$ is intertwined with the search for a tracking controller that can keep the realized tracking trajectory within it. We parameterize the tracking controller u^s

as a function of the relative system state and planning control signal, that is $u^s = K(r, u^p)$. Denote the closed-loop dynamics as $f^r_K(r, u^p) := f^r(r, K(r, u^p), u^p)$ and define

$$\dot{V}_K(r, u^p) := \left.\frac{\partial \tilde{V}(\tilde{r})}{\partial \tilde{r}}\right|^T_{\tilde{r}=c(r)} \frac{\partial c(r)}{\partial r} f^r(r, K(r, u^p), u^p).$$

The search for a suitable function V and controller $K(\cdot, \cdot)$ is then formalized by the optimization problem:

$$\min_{K(\cdot,\cdot),\mathcal{B}} \quad \text{volume}(\mathcal{B}) \tag{6.3a}$$

$$\text{s.t.} \quad \left.\begin{array}{l} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \dot{V}_K(r, u^p) < 0 \tag{6.3b}$$

$$\left.\begin{array}{l} V(r) \leq 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow K(r, u^p) \in \mathcal{U}^s. \tag{6.3c}$$

We will next encode the constraints and objective of problem (6.3) as SOS certificates.

6.2.3 Reformulating the Constraints as SOS Certificates

We consider the constraints first. Under the assumption that the dynamics for the tracking model are control affine (common for most robotic systems), the function f^r is control affine in u^s. That is, f^r takes the form $f^r(r, u^p, u^s) = h(r, u^p) + B(r)u^s$. Thus,

$$\dot{V} = \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \frac{\partial V(r)}{\partial r}^T B(r)u^s.$$

Hence, the invariance condition requires the existence of a tracking controller $K(r, u^p)$ such that

$$\left.\begin{array}{l} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \frac{\partial V(r)}{\partial r}^T B(r)K(r, u^p) < 0. \tag{6.4}$$

We can *equivalently* state inequality (6.4) as:

$$\left.\begin{array}{l} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \min_{K(r,u^p)\in\mathcal{U}^s} \left(B(r)^T \frac{\partial V(r)}{\partial r}\right)^T K(r, u^p) < 0. \tag{6.5}$$

The equivalence between inequalities (6.4) and (6.5) follows from the observation that inequality (6.4) holds for *some* $u^s \in \mathcal{U}^s$ if and only if it holds for a $u^s \in \mathcal{U}^s$ minimizing the left hand side in inequality (6.5). Importantly, the minimization in inequality (6.5) is *independent* of u^p, as the constraint set

$\mathcal{U}^s$ and quantity $B^T\partial V/\partial r$ do not depend on u^p. Thus, we can simplify K by making it a function of r only (with a slight abuse of notation, we still refer to such a simplified tracking controller with K).

We can now define SOS certificates for the constraints in problem (6.3). Suppose set $\mathcal{U}^p$ can be written as the semialgebraic set $\{u^p \in \mathbb{R}^{m_p} : g_i^p(u^p) \leq 0,\ i = 1, \ldots, N_p\}$ and let the controller K be a *polynomial* function in r. Expanding $\dot{V}_K$, one can encode constraint (6.3b) using the multiplier polynomials $L^{\text{Lya}}(r, u^p), \{L_i^p(r, u^p)\}_{i=1}^{N_p}$ such that

$$-\dot{V}_K + L^{\text{Lya}} \cdot (V - 1) + \sum_{i=1}^{N_p} L_i^p \cdot g_i^p \text{ is SOS}, \qquad \{L_i^p\} \text{ are SOS}, \qquad i = 1, \ldots, N_p. \tag{6.6}$$

(In the following we drop the summation notation and write the collective sum as a "dot-product:" $L^p \cdot g^p$). To capture the control constraint in (6.3c), one may leverage two different techniques. The first one is to compose the tracking controller K with a saturation function and write $\dot{V}$ in case form corresponding to the unsaturated and saturated regimes, respectively. However, this approach scales exponentially in the dimension of the control input, as one needs to capture all combinations of saturation regimes (Majumdar and Tedrake, 2017). Instead, we assume that the tracking control set $\mathcal{U}^s$ is polytopic, i.e., $\mathcal{U}^s = \{u^s \in \mathbb{R}^{m_s} : g^s(u^s) \leq 0\}$, where $g^s(u^s) \leq 0$ is a set of linear inequalities in u^s of the form $g_i^s = a_{s_i}^T u^s - b_{s_i} \leq 0$, $i = 1, \ldots, N_s$. Note that this is not an overly restrictive assumption as the control constraints are often taken to be box constraints. Then, we encode constraint (6.3c) using the multipliers $\{L_i^s(r, u^p)\}$:

$$-g_i^s(K) + L_i^s \cdot (V - 1) \text{ is SOS}, \qquad L_i^s \text{ are SOS}, \qquad i = 1, \ldots, N_s. \tag{6.7}$$

Crucially, this approach scales linearly with respect to the number of inputs.

6.2.4 Reformulating the Objective as SOS Certificates

Minimizing the volume of $\mathcal{B}$, which itself is a nonlinear mapping of the $1-$sublevel set of V, is a difficult task if one reasons in terms of arbitrary functions V. One approach is to approximate the volume by the integral of V over an Euclidean ball of radius R such that the ball is contained within the $1-$sublevel set of V – an approach taken, for example, in (Posa et al., 2017). Alternatively, one can minimize the volume of an encapsulating ellipsoid, easily captured using an additional constraint and a convex pseudo-objective.

Toward this end, we define the ellipsoid $\mathcal{E} := \{r : c(r)^\top E c(r) \leq 1\}$ for some positive definite matrix $E \succeq \delta_E I$, where $\delta_E > 0$ is a small tolerance parameter. Then, the inclusion constraint $\mathcal{B} \subseteq \mathcal{E}$ is captured using the SOS multiplier $L^{\mathcal{E}}(r)$:

$$1 - c^\top E c + L^{\mathcal{E}}(V - 1) \text{ is SOS}, \qquad L^{\mathcal{E}} \text{ is SOS}, \qquad E \succeq \delta_E I. \tag{6.8}$$

As the volume of $\mathcal{E}$ is inversely proportional to the square root of its determinant, the minimum volume objective reduces to maximizing the determinant of E, and can be written in the form of (2.6) (Ben-Tal and Nemirovski, 2001, Chapter 4). For non-identity mappings $c(\cdot)$, the first SOS constraint in equation (6.8) can quickly become computationally challenging due to the quadratic terms in $c(r)$ and the complexity of the relation $V(\cdot) = \tilde{V}(c(\cdot))$. In this case, one may consider the following simplification. Since

$$\mathcal{B} = \{c(r) \in \mathbb{R}^{n_{\tilde{r}}} : \tilde{V}(c(r)) \leq 1\} \subseteq \{\tilde{r} \in \mathbb{R}^{n_{\tilde{r}}} : \tilde{V}(\tilde{r}) \leq 1\}, \tag{6.9}$$

the simplification is to outer bound the set on the right by using an ellipsoid. Specifically, let the ellipsoid $\mathcal{E}$ be given by $\mathcal{E} = \{\tilde{r} : \tilde{r}^\top E \tilde{r} \leq 1\}$ for some positive definite matrix $E \succeq \delta_E I$. The inclusion condition $\tilde{V}(\tilde{r}) \leq 1 \Rightarrow \tilde{r}^\top E \tilde{r} \leq 1$, is captured using the SOS multiplier $L^{\mathcal{E}}(\tilde{r})$:

$$1 - \tilde{r}^\top E \tilde{r} + L^{\mathcal{E}}(\tilde{V} - 1) \text{ is SOS}, \qquad L^{\mathcal{E}} \text{ is SOS}, \qquad E \succeq \delta_E I. \tag{6.10}$$

Crucially, the constraints above are deliberately written with respect to the variable $\tilde{r}$, which is treated as an *independent indeterminate* from r. This is beneficial when using a mapping $c(\cdot)$ other than the trivial identity map, as it allows one to approximate the otherwise complex set $\mathcal{B}$ with potentially simpler functions in $\tilde{r}$.

6.2.5 SOS Formulation of Optimization Problem

Collecting all the results so far, our approach entails conservatively (as we rely on sufficient SOS certificates) solving the optimization problem (6.3) as a SOS program:

$$\max_{K, \tilde{V}, E, \mathbf{L}} \quad \log\det(E) \tag{6.11a}$$

$$\text{s.t.} \quad \text{eqs. (6.6), (6.7), (6.10)} \tag{6.11b}$$

$$\tilde{V}(0) = 0, \tag{6.11c}$$

where $\mathbf{L} := \{L^{\text{Lya}}, L^p, L^s, L^{\mathcal{E}}\}$. Reinforcing the ideas in Section 6.2.4, we iterate that $\tilde{V}$ is considered as a function in the independent indeterminate $\tilde{r}$, while constraints (6.6), (6.7) are encoded using the indeterminate r via the definition $V(r) = \tilde{V}(c(r))$.

Constraints in (6.11b) are bilinear in the decision variables. Consequently, similar to the bilinear solution algorithms in (Majumdar and Tedrake, 2017) and (Posa et al., 2017), one must alternate between the decision variable sets $\{E, K, \mathbf{L}\}$ and $\{\tilde{V}, E, L^p\}$, each time holding the other variable set fixed. Specifically, we refer to the sub-problem where $K(\cdot)$ is part of the decision variable set as the K sub-problem, and the sub-problem where $\tilde{V}$ is part of the decision variable set as the V sub-problem. Direct implementation of alternations is hindered by two challenges. First, one

requires a feasible initial guess for $\tilde{V}$ to begin the alternations. In (Majumdar and Tedrake, 2017) and (Posa et al., 2017), the authors leverage locally linearized models and the solution to the Riccati equation to generate such a guess. However, we address a fundamentally more challenging problem as we consider the controllability of a relative dynamical system between two different dynamical models. Consequently, the relative system is often not linearly controllable at $r = 0$ (even if the individual models are) and thus one requires an additional procedure to generate a feasible function $\tilde{V}$ for problem (6.11) from an initially infeasible guess. Second, alternating optimization is prone to numerical instabilities as the solutions of the individual convex problems frequently lie on boundaries of the respective feasible sets. We address these challenges next.

6.3 Solving the Bilinear Optimization Problem

The general idea to tackle both of the aforementioned challenges entails using iterative slack minimization. We first discuss the solutions to the K and V sub-problems (Sections 6.3.1 and 6.3.2, respectively) and then present the general solution algorithm in Section 6.3.3.

6.3.1 The K Sub-Problem

A naïve implementation of the K sub-problem would entail solving problem (6.11) with respect to $\{E, K, \mathbf{L}\}$ for a given $\tilde{V}$. However, constraint (6.6) as written may generate controllers that are numerically unstable when held fixed in the V sub-problem. Given the polytopic constraint set $\mathcal{U}^s$ and a fixed $\tilde{V}$, one can exactly characterize the "most stabilizing controller" as the function K that minimizes the left hand side in inequality (6.5) for all r such that $V(r) = 1$. Thus, we propose the following two-step method for solving the K sub-problem, *for a given function* $\tilde{V}$:

1. Find the "most stabilizing controller" $K(\cdot)$ as the solution to:

$$\min_{K, L^{\text{Lya}}, L^p, L^s, \gamma} \quad \gamma \tag{6.12a}$$

$$\text{s.t.} \quad -\dot{V}_K + \gamma + L^{\text{Lya}} \cdot (V - 1) + L^p \cdot g^p \quad \text{is SOS} \tag{6.12b}$$

$$-g^s(K) + L^s \cdot (V - 1) \text{ is SOS}, \qquad \{L_i^p\}, \{L_i^s\} \quad \text{are SOS}, \tag{6.12c}$$

where γ is a slack variable for the invariance constraint. Notice that when $V(r) = 1$ and $g^p(u^p) \leq 0$, then $\dot{V}_K(r, u^p) \leq \gamma$. Denote the optimal slack as γ_c^*.

2. Compute the tightest bounding ellipsoid $\mathcal{E}$:

$$\max_{L^{\mathcal{E}},E} \quad \log\det(E) \tag{6.13a}$$

$$\text{s.t.} \quad 1-\tilde{r}^\top E\tilde{r}+L^{\mathcal{E}}(\tilde{V}-1) \quad \text{is SOS} \tag{6.13b}$$

$$E \succeq \delta_E I, \qquad L^{\mathcal{E}} \text{ is SOS.} \tag{6.13c}$$

These two steps comprise the K sub-problem. The benefit of decomposing the K sub-problem in this fashion is that we independently search for the most stabilizing controller while simultaneously relaxing the invariance constraint by using the slack variable γ; in particular, γ accounts for the suboptimality of the computed controller with respect to the left hand side of inequality (6.6).

6.3.2 The V Sub-Problem

Given a controller $K(\cdot)$, and multiplier polynomials $\{L_i^s\}$, $L^{\mathcal{E}}$, and L^{Lya} from the K sub-problem, the V sub-problem is defined as

$$\max_{\tilde{V},E,L^p,\gamma,\epsilon} \quad \lambda\log\det(E)-\gamma-\|\epsilon\|_1 \tag{6.14a}$$

$$\text{s.t.} \quad -\dot{V}_K+\gamma+L^{\text{Lya}}\cdot(V-1)+L^p\cdot g^p \quad \text{is SOS}, \quad \{L_i^p\} \quad \text{are SOS} \tag{6.14b}$$

$$-g^s(K)+\epsilon+L^s\cdot(V-1) \text{ is SOS} \tag{6.14c}$$

$$1-\tilde{r}^\top E\tilde{r}+L^{\mathcal{E}}(\tilde{V}-1) \quad \text{is SOS} \tag{6.14d}$$

$$E\succeq\delta_E I, \quad \tilde{V}(0)=0, \tag{6.14e}$$

where $\epsilon \in \mathbb{R}^{N_s}_{\geq 0}$ is a slack vector for the control constraints and $\lambda \in \mathbb{R}_{>0}$ is a Pareto trade-off parameter. Notice that the control slack ϵ is only necessary in the V sub-problem since the controller is held fixed within this problem. In contrast, in problem (6.12), we directly optimize over the set of strictly *feasible* controllers. Given these slack-based relaxed sub-problems, we now provide a solution algorithm for problem (6.11).

6.3.3 Solution Algorithm

Before providing the full solution algorithm, we describe an initialization procedure that generates a numerically stable initial guess for $\tilde{V}$. The procedure is detailed in Algorithm 2. In particular, the algorithm may be initialized by using any polynomial guess for $\tilde{V}$ such that $\tilde{V}(0)=0$ (e.g., $\tilde{r}^T\tilde{r}$).

Notice that in line 6, when problem (6.14) is solved, the Pareto parameter λ is set to 0 since the objective for the initialization algorithm is to simply generate a feasible initial guess for problem (6.11). Furthermore, we impose the additional constraint $\gamma \leq \gamma_c^*$ to ensure monotonic improvement in solution quality between sub-problems. As the original problem is still non-convex, the convergence

Algorithm 2 Generating Feasible Guess

1: **Input:** Initial guess $\tilde{V}$ satisfying $\tilde{V}(0) = 0$; slack tolerance $\delta \in [0, 1)$; max # of iterations N.
2: $i \leftarrow 0$.
3: **while** $i < N$ **do**
4: $\{K, L^{\text{Lya}}, L^s, L^{\mathcal{E}}, E, \gamma_c^*\} \leftarrow$ SOLVE (6.12)–(6.13).
5: **if** $\gamma_c^* \leq \delta$ **then return** $(\tilde{V}, E, K)$
6: $\{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ SOLVE (6.14) with $\lambda = 0$ and subject to additional constraint $\gamma \leq \gamma_c^*$.
7: **if** $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then return** $(\tilde{V}, E, K)$
8: $i \leftarrow i + 1$.
9: **end while**
10: **return** Failure.

of the slack variables below the tolerance threshold is not guaranteed for every initial guess of $\tilde{V}$. However, typical guesses such as $\tilde{r}^T Q \tilde{r}$ for a positive diagonal matrix Q appear to work quite well, as the experiment section will illustrate.

Given an initial guess generated by Algorithm 2, we are now ready to solve problem (6.11). To do so, we will make use of a slightly modified version of the V sub-problem, given below:

$$\max_{\tilde{V}, E, L^p, \gamma, \epsilon} \quad \log\det(E) - \lambda(\gamma + \|\epsilon\|_1) \tag{6.15a}$$

$$\text{s.t.} \quad \text{eqs. (6.14b) – (6.14e)} \tag{6.15b}$$

$$\underline{\alpha} E^* \preceq E \preceq (1 + \alpha) E^*, \tag{6.15c}$$

where E^* is the previous numerically stable solution (i.e., with optimal slack values $\gamma^*, \|\epsilon^*\|_\infty \leq \delta$), $\underline{\alpha} \in (0, 1)$ is a fixed parameter, and $\alpha \in (0, 1)$ is a backtracking search parameter that is adjusted in an iterative fashion. Note that the Pareto parameter λ now multiplies the slack terms in the objective. Specifically, we iteratively solve problem (6.15), *each time* checking the slack tolerances to ensure numerical stability, while using constraint (6.15c) to enforce a *trust region* around the current numerically stable solution. The full solution algorithm is summarized in Algorithm 3.

The backtrack search procedure in line 8 is summarized in Algorithm 4. Within this procedure, we first iteratively maximize $\log\det(E)$ within the trust region (6.15c) centered on the previous stable numerical solution, using the slack values as a check on solution quality (lines 3—6). Once solution quality degrades, we backtrack (shrink) the trust region in lines 7—14 until we again fall below the slack tolerances. Note that Algorithm 4 will either return an updated $\{\tilde{V}, E\}$ that is numerically stable (with respect to slack tolerances), or it will simply return the numerically stable solution from line 5 in Algorithm 3 if unable to make progress. Thus, Algorithm 3 terminates if either (1) improvement in $\log\det(E)$ stalls, or (2) the function $\tilde{V}^*$ returned by the backtrack procedure is not *strictly* feasible (i.e., $\gamma \leq 0$) with respect to line 5 in Algorithm 3, but is still acceptable according to the slack tolerances.

The key difference between the alternating method described here and a similar procedure

Algorithm 3 Solving Problem (6.11)

Input: Output tuple $(\tilde{V}, E, K)$ from Algorithm 2; slack tolerance δ; termination tolerance $\theta_1 \in (0, 1)$.
$i \leftarrow 1$, `converged` $\leftarrow$ `false`.
Initialize: $(\tilde{V}^*, E^*, K^*, c_0^*) \leftarrow (\tilde{V}, E, K, \log\det(E))$.
while $i < N \wedge \neg$ `converged` **do**
    $\{K, L^{\text{Lya}}, L^s, L^{\mathcal{E}}, E\} \leftarrow$ SOLVE (6.12)– (6.13) subject to additional constraint: $\gamma \leq 0$.
    if Line 5 successfully solved **then**
        $(E^*, K^*) \leftarrow (E, K)$.
        $(\tilde{V}^*, E^*, c_i^*) \leftarrow$ `Backtrack`$\Big(\{K^*, L^{\text{Lya}}, L^s, L^{\mathcal{E}}\}, E^*, \tilde{V}^*\Big)$.
        if $|c_i^* - c_{i-1}^*| \leq \theta_1 |c_{i-1}^*|$ **then** `converged` $\leftarrow$ `true`. **else** $i \leftarrow i + 1$.
    else
        `converged` $\leftarrow$ `true`.
    end if
end while
return $(\tilde{V}^*, E^*, K^*)$

Algorithm 4 Backtrack

Input: Solution $\{K^*, L^{\text{Lya}}, L^s, L^{\mathcal{E}}\}$ and E^* from line 5 in Algorithm 3 and current best solution $\tilde{V}^*$; slack tolerance δ; backtrack parameters $\overline{\alpha}$, β, $\theta_2 \in (0, 1)$.
Initialize: $\alpha \leftarrow \overline{\alpha}$, `bt`$\leftarrow$ `false`, $c^* \leftarrow \log\det(E^*)$.
while $\neg$`bt` **do**
    $\{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ SOLVE (6.15).
    if $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then** $(\tilde{V}^*, E^*, c^*) \leftarrow (\tilde{V}, E, \log\det(E))$. **else** `bt`$\leftarrow$ `true`.
end while
while `bt` $\wedge\ \alpha > \theta_2 \overline{\alpha}$ **do**
    $\alpha \leftarrow \beta\alpha$.
    $\{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ SOLVE (6.15).
    if $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then**
        $(\tilde{V}^*, E^*, c^*) \leftarrow (\tilde{V}, E, \log\det(E))$.
        `bt` $\leftarrow$`false`.
    end if
end while
return $(\tilde{V}^*, E^*, c^*)$

in (Posa et al., 2017) is that the authors in (Posa et al., 2017) minimize only the slack variable in both sub-problems and use binary search to iteratively refine an upper bound on the cost function within the V sub-problem. On the other hand, our algorithm maintains numerical stability by using the slack variables *solely as a check* on the allowable change in the solution, while taking advantage of minimizing the true objective (i.e., $\log\det(E)$) within *both* sub-problems. This is especially useful in situations where the second phase of the algorithm struggles to make notable improvements on the objective (e.g., due to a smaller set of decision variables), thereby allowing the K sub-problem to take over.

6.4 Numerical examples

In this section we numerically validate our proposed approach. Specifically, in Section 6.4.1, we compare our approach with the HJ method in (Herbert et al., 2017), while in Section 6.4.2, we study higher-dimensional systems that are beyond the reach of HJ analysis. The code is available at `https://github.com/StanfordASL/Model-Mismatch`.

6.4.1 Comparison with the HJ Method

In this section, we compare our method with the HJ-based approach in (Herbert et al., 2017) for two cases: (i) 5D car model tracking a 3D Dubins car (second example in Table 6.1), and (ii) 4D dynamically extended Dubins car tracking a 2D single integrator (fist example in Table 6.1).

5D tracking 3D: This example was chosen as it is the current limit for standard grid-based HJ reachability methods when techniques such as decomposition cannot be applied. The system dynamics and bounded state definition are provided in the second row of Table 1. The model parameters were chosen as $|a| \leq 1$ m/s^2, $|\alpha| \leq 3$ rad/s^2, $\hat{v} = 1.0$ m/s, $|\hat{\omega}| \leq 0.1$ rad/s.

For the SOS method, we parameterized K and $\tilde{V}$ as 2nd order polynomials in r and $\tilde{r}$, respectively. The trigonometric terms $\cos\theta_r$ and $\sin\theta_r$ were approximated with Chebyshev polynomials up to degree 2, over the range $\theta_r \in [-\pi/6, \pi/6]$ rad. To ensure the validity of these approximations, an additional constraint was appended to problems (6.13) and (6.14):

$$-g + L_g(\tilde{V} - 1) \text{ is SOS}, \qquad L_g \text{ is SOS}, \tag{6.16}$$

where $g = \theta_r^2 - (\pi/6)^2$ and L_g is a SOS polynomial in $\tilde{r}$. The initial guess for $\tilde{V}$ was simply $\tilde{r}^T Q \tilde{r}$, where Q is a diagonal matrix with randomly sampled positive entries. In order to ensure a fair comparison, the cost function in the min-max game for the HJ method was $x_r^2 + y_r^2 + \theta_r^2 + (v\cos\theta_r - \hat{v})^2 + v^2\sin^2\theta_r + \omega^2$, which is the closest analogue to the SOS objective of minimizing the entire volume of $\mathcal{B}$ and not simply its projection onto the position coordinates.

Figure 6.2 plots the projections of the boundary of set $\mathcal{B}$ onto the (x_r, y_r, θ_r) components (i.e., the dimensions most relevant for collision checking) for the HJ and SOS solutions, respectively. The right-most panel in this figure provides a top-down view onto the (x_r, y_r) plane. As expected, the HJ solution provides a tighter error bound – approximately 42% smaller than the SOS error bound, or about 0.2 m in absolute terms.

The main reason behind this is that by using a grid-based approach to solve the min-max HJ game, one *exactly* computes the *non-smooth* "optimal controller" in (6.5), whereas the SOS approach attempts to find the best polynomial approximation. On the other hand, the computation time difference for the two solutions is *substantial* – approximately 25 hours for the HJ solution versus 5

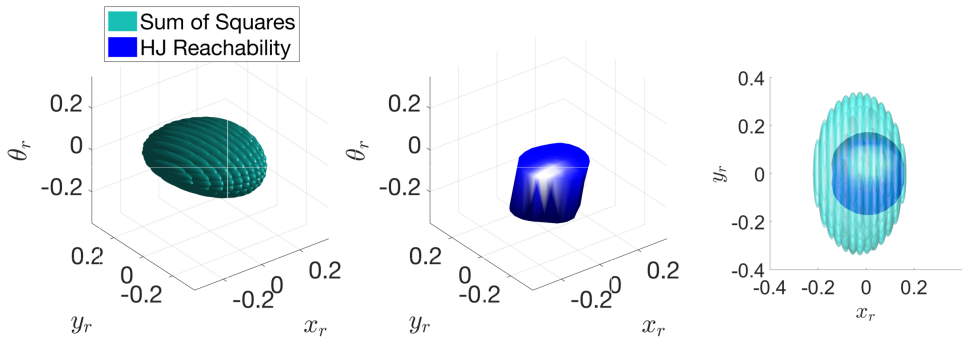


Figure 6.2: Projection of the boundary of $\mathcal{B}$ onto (x_r, y_r, θ_r). *Left*: SOS. *Middle*: HJ. *Right*: Top-down view (i.e., onto (x_r, y_r)) for both solutions. The HJ positional error bound is smaller (42% of SOS bound), but the SOS solution requires only 0.3% of the computation time required by HJ analysis.

minutes for the SOS solution. Specifically, the HJ solution was obtained by solving a corresponding HJ PDE (Herbert et al., 2017) using a C++ implementation of the Lax Friedrichs numerical scheme (Tanabe and Chen, 2018). The SOS programs were solved using the Spotless polynomial optimization toolbox (Tobenkin et al., 2013) and MOSEK SDP solver (ApS, 2017). Computations were done on a desktop computer with an Intel Core i7-2600K CPU and 16 GB of RAM. Clearly, for this example, the SOS approach appears to provide a high-quality approximation to the optimal (HJ-based) solution, in a fraction of the time required by the HJ-based approach.

4D tracking 2D: This example highlights a case in which the HJ method is efficient enough to warrant its use instead of the SOS method (i.e. when the dynamics are less than five dimensions and/or can be decomposed into subsystems that are each less than five dimensions).

The error and auxiliary states are $e = (x_r, y_r)$, and $\eta = v$, respectively. Note that the map $\phi(\cdot)$ for this system, as given in Table 6.1, exploits the rotational invariance of the tracking model, thereby yielding a dimensionality reduction. In addition, given a 2-norm bound on $\|(\hat{u}_{\hat{x}}, \hat{u}_{\hat{y}})\|$ and the norm-preservation property of rotations, the same bound holds for $(\tilde{u}_{\hat{x}}, \tilde{u}_{\hat{y}})$. Finally, since the rotation angle θ is collapsed in the relative system state definition through the rotation matrix $R(\theta)$, following the computation of the bound $\mathcal{B}$ for the relative system state (x_r, y_r, v), we can project this bound back to the space (e', v) where $e' = (x - \hat{x}, y - \hat{y})$ is the absolute error in relative position, by simply taking the union $\mathcal{B}' := \bigcup_v \bigcup_{\theta \in [-\pi, \pi]} \left\{ \begin{bmatrix} R(\theta)^T e \\ v \end{bmatrix} : \begin{bmatrix} e \\ v \end{bmatrix} \in \mathcal{B} \right\}$. For a fixed v, this is a rotational sweep of the projection of the set $\mathcal{B}$ onto e. Since the true error bound we care about is e', all results below will be displayed with respect to $\mathcal{B}'$.

We imposed the control bounds: $(u_a, u_\omega) \in [-1, 1]$ m/s^2 $\times$ $[-1, 1]$ rad/s, and $\|(\hat{u}_{\hat{x}}, \hat{u}_{\hat{y}})\| \leq 0.1$ m/s. The bounded state mapping $c(\cdot)$ is the identity. We parameterized the feedback controller and function $\tilde{V}$ up to degree 4 and initialized the infeasible start algorithm (i.e., Algorithm 2) with $\tilde{V}(\tilde{r}) =$

$[\tilde{r}, \tilde{r}^2]Q[\tilde{r}, \tilde{r}^2]^T$ where $\tilde{r}^2$ denotes component-wise square of $\tilde{r}$, and Q was a randomly generated psd matrix. Over 10 initializations, on average, 7 iterations were needed to exit Algorithm 2. The optimization loop in Algorithm 3 took < 90 s on all runs for convergence.

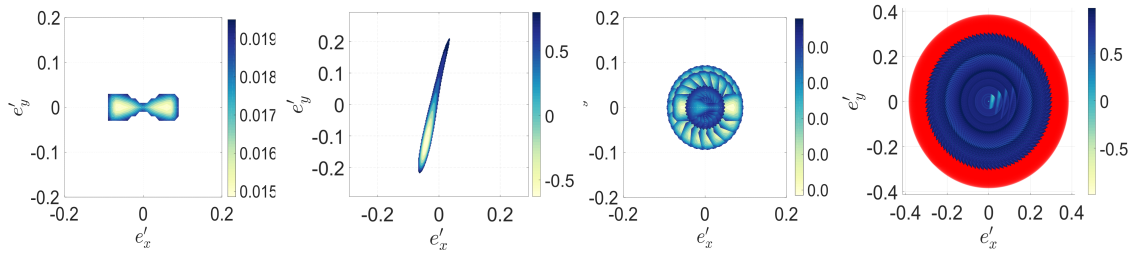


Figure 6.3: Left two plots: Projection of the invariant set $\mathcal{B}'$ onto e' for the fixed slice $(v, \theta) = (0.013, 0)$ (left: HJ, right: SOS); Right two plots: Projection of the invariant set $\mathcal{B}'$ onto e' for all (v, θ) (left: HJ, right: SOS). In red is the projection of the bounding ellipsoid $\mathcal{E}$ onto e'.

Figure 6.3 illustrates the results of the HJ and SOS solutions. Specifically, in the left two plots of Figure 6.3, we plot the projection of the set $\mathcal{B}'$ onto e' for a fixed value of $(v, \theta) = (\bar{v}, \bar{\theta})$, i.e. $\left\{ R(\bar{\theta})^T e : \begin{bmatrix} e \\ \bar{v} \end{bmatrix} \in \mathcal{B} \right\}$. In the right two plots, we plot the projection of $\mathcal{B}'$ onto e' over all (v, θ), i.e. $\bigcup_v \bigcup_{\theta \in [-\pi, \pi]} \left\{ R(\theta)^T e : \begin{bmatrix} e \\ v \end{bmatrix} \in \mathcal{B} \right\}$. For the SOS solution, the range of v for this union was extracted from the projection of the bounding ellipsoid $\mathcal{E}$ onto v. The projection of this ellipsoid onto e' is shown in red in the background of the rightmost panel of Figure 6.3, and can be seen to envelop $\mathcal{B}'$ as expected.

We note that the SOS solution is within a reasonable ballpark of the optimal solution from HJ. In particular, the projection of $\mathcal{B}'$ onto e' is approximately a 0.1 m radius l_2 ball, while the corresponding solution from SOS is a 0.3 m radius l_2 ball. The computation time comparison was on the order of approximately 2 minutes for SOS versus 5 minutes for HJ. Similar to the first example, in order to ensure a fair comparison, the cost function for the min-max game for the HJ method was $e_x + e_y^2 + v^2$, which is the closest analogue to the SOS objective of minimizing the entire volume of $\mathcal{B}$.

The projection of the optimal $\mathcal{B}$ onto the translational plane from the HJ method, as shown in the left-most sub-figure in Figure 6.3 clearly illustrates the limitation of SOS in its ability to approximate non-smooth quantities. Indeed, herein lies the primary weakness of the SOS method – tight approximations require high degree polynomials which can rapidly increase the size of the SOS problems. Despite this limitation inherent in the SOS method, the approach allows us to establish a conservative over-approximation of the tracking error bound (when one exists) for a given planner/tracker pair and gauge the sensitivity of the expected tracking error to changing model parameters such as inertial properties and control authorities. Furthermore, as the next section illustrates, the SOS method allows us to perform this analysis for higher dimensional systems that

are severely challenging (albeit, not impossible) for HJ. As such, the HJ and SOS methods should really be viewed as complementary to one another.

6.4.2 Higher Dimensional Examples

In this section, we present numerical results for two higher dimensional systems for which the HJ approach is intractable: (i) 6D planar quadrotor model tracking a 4D double integrator (third example in Table 6.1), and (ii) 8D plane model tracking a 4D decoupled Dubins plane model (fourth example in Table 6.1). To ensure that these example cannot be solved using standard grid-based HJ reachability, we must first determine that decomposition techniques would not be applicable. Both the 6D and 8D relative system dynamics are highly coupled; this implies that approximate decomposition techniques would lead to results that are overly conservative (Chen et al., 2018b). The dynamics also do not form self-contained subsystems, making exact decomposition impossible (Chen et al., 2016a).

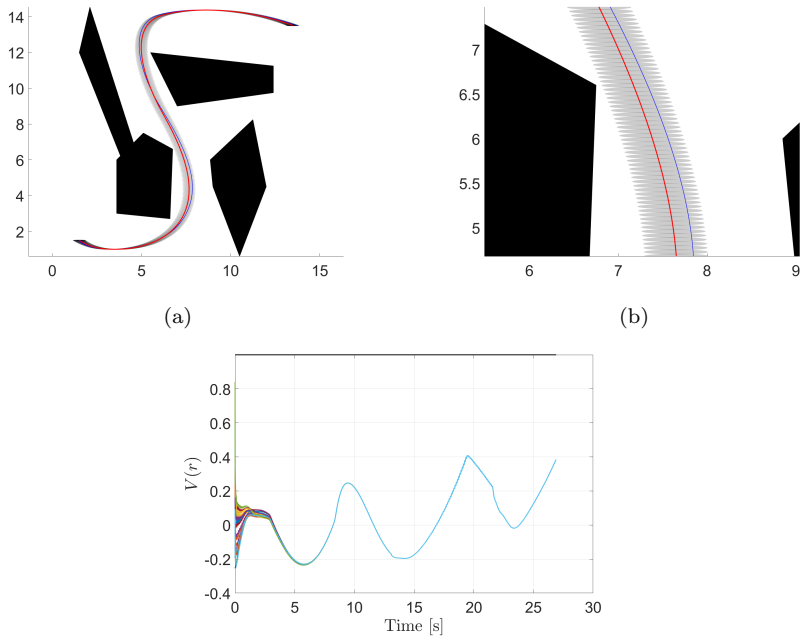
6D tracking 4D: The bounds on the planar quadrotor's inputs were generated by linearly transforming the bounds for each thruster, chosen to be the range $[0.1, 1.0]g$ (where g is the gravitational acceleration), by using the model parameters in (Steinhardt and Tedrake, 2012), namely thruster moment arm 0.25 m, mass 0.486 kg, and rotational inertia 0.00383 kg m^2. The control input for the double integrator is the acceleration $(\hat{u}_{\hat{x}}, \hat{u}_{\hat{z}})$, with bounds chosen as $\|(\hat{u}_{\hat{x}}, \hat{u}_{\hat{z}})\| \leq 0.5$ m/s^2.

The bounded state mapping $c(\cdot)$ is the identity. We parameterized the K and V functions as polynomials up to degree 2. The trigonometric terms were again approximated using Chebyshev expansions up to degree 3 over the range $\theta \in [-\pi/3, \pi/3]$ rad and enforced using the additional constraint as in (6.16). The algorithm was initialized using the solution to the Riccati equation for the linearized system at $r = 0$; the overall computation time was just under 3 minutes.

The projection onto the (x_r, z_r) dimensions was an ellipse with span $[-0.36, 0.36] \times [-0.03, 0.03]$ m, which is quite tight. The projections onto the other dimensions are: $(\dot{x}_r, \dot{z}_r) : [-0.42, 0.42] \times [-0.03, 0.03]$ m/s, and $(\theta, \omega) : [-5.16, 5.16]^\circ \times [-18.11, 18.11]^\circ$/s.

Using the projection of the set $\mathcal{B}$ onto (x_r, z_r) as a safety margin, we planned a nominal motion plan for the obstacle environment depicted in Figure 6.4a, by using kinodynamic (double-integrator) FMT* (Schmerling et al., 2015a) (red curve). Note that planning using the simple double-integrator model is orders of magnitude (indeed, efficiently implementable in real-time) faster than using nonlinear programming methods to find feasible trajectories for the actual planar quadrotor system. Superimposed on the plot are the realized trajectories (in blue). The feedback tracking controller ensures that the realized trajectory lies within the swept ellipsoidal tube centered at the nominal motion plan for all time. Figure 6.4c plots the time series $V(t) = V(r(t))$; as expected, $V(r(t)) \leq 1$ for all $t \geq 0$.

8D tracking 4D: The planning model (i.e., decoupled Dubins plane) is a benchmark planning



(a) (b)

(c)

Figure 6.4: (a) Nominal double-integrator motion plan (red), and projection of the set $\mathcal{B}$ onto the (x_r, z_r) dimensions as a safety margin; (b) Zoomed-in view of the tightness of the realized trajectories (blue) with respect to the nominal plan (red) while navigating closely next to an obstacle; (c) Verification that $V(r(t))$ stays below 1 as required.

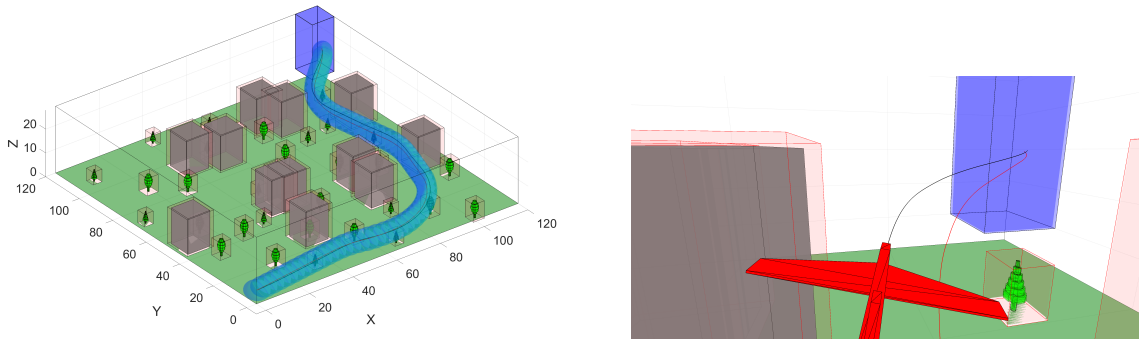
example with well characterized optimal trajectories (Chitsaz and LaValle, 2007). The full 8D dynamics of the plane, however, are considerably more difficult to plan with online. The system dynamics and bounded state definition are provided in the fourth row of Table 1. The planner model parameters are as follows. The constant speed $\hat{v}$ was set to be the nominal speed for the plane in straight level flight conditions (lift equals gravitational force) at a nominal angle of attack $\alpha_0 = 5^o$, which corresponds to $\hat{v} = 6.5$ m/s (see (Schmerling and Pavone, 2017) for the relevant constants used to compute this quantity). The maximum magnitude of the turning rate for the planner, $|\hat{\omega}|$, was set to be 20% of the plane's maximum turning rate in a horizontal coordinated turn at nominal speed $\hat{v}$, and was computed to be 0.21 rad/s (equivalently, a minimum turning radius of 30 m for the horizontal Dubins model). Finally, the planner's vertical velocity was limited to the range $[-0.1, 0.1]\,\hat{v}$.

Taking advantage of the structure of the dynamics, the normalized acceleration control u_a was chosen to exactly cancel drag, plus an additional 2nd degree polynomial in r as determined by the SOS program. The rest of the controller components and $\tilde{V}$ were also parameterized as 2nd order polynomials in r and $\tilde{r}$, respectively. The plane's (tracking) control limits were chosen to be $u_a \in [-8, 8]$ m/s^2 (≈ 10 times the acceleration needed to cancel drag at level trim conditions), $u_{\dot{\phi}} \in [-120, 120]^\circ$/s, and $u_{\dot{\alpha}} \in [-60, 60]^\circ$/s. To enforce the validity of the Chebyshev approximation

for the trigonometric and $1/v$ terms in the dynamics, we enforce the additional constraints $\phi \in [-\pi/4, \pi/4]$ rad, $\gamma \in [-\pi/6, \pi/6]$ rad, $\psi_r \in [-\pi/6, \pi/6]$ rad, and $v \in [3, 10]$ m/s. The overall (initialization plus main optimization) computation time was under 2 hours.

The projection of the set $\mathcal{B}$ onto (x_r, y_r, z_r) (i.e., the position errors *in the frame of the Dubins car*) had span $[-5.6, 5.6] \times [-3.8, 3.8] \times [-4.5, 4.5]$ m. The bound is naturally more loose in the (x_r, z_r) dimensions since the planning model's horizontal velocity is equal to the plane's trim conditions at level flight, limiting the plane's ability to use its remaining velocity range for both tracking and ascending/descending. Thus, for such a choice of planning and tracking models, the bound appears reasonable. To obtain tighter bounds, one could alternatively use the coupled kinematic plane model proposed in (Owen et al., 2015) as the planning model.

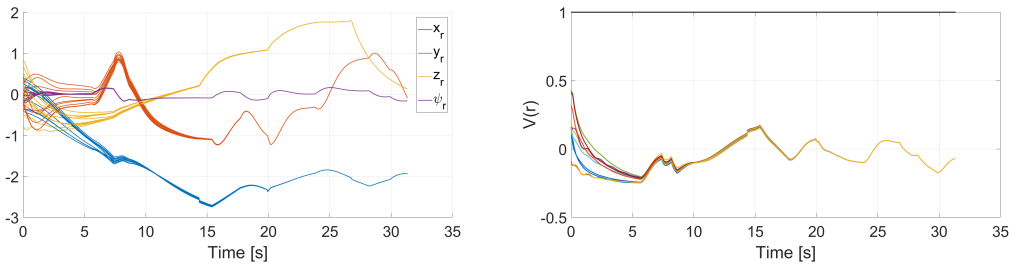
For an application of this bound to online motion planning, we created a cluttered obstacle environment with trees and buildings (see Figure 6.5a). The goal region is the blue shaded box in the corner, and the plane starts at position $(1, 1, 5)$. The nominal motion plan was computed using kinodynamic FMT* (Schmerling et al., 2015b) with the locally optimal trajectories in (Chitsaz and LaValle, 2007) as steering connections. Having computed all steering connections offline (< 2 minutes computation time for 5000 samples), the online computation time for the trajectory was on the order of 100 ms. Collision checking was performed using obstacles inflated by the size of the plane (roughly a $2 \times 2 \times 0.5$m box envelope) and the projection of the ellipsoidal bound $\mathcal{B}$ onto (x_r, y_r, z_r) (rotated and translated by the orientation and position of the Dubins plane). In Figure 6.5a, the nominal motion plan (i.e., the plan for the decoupled Dubins model) is shown in red, while the actual trajectory followed by the plane is shown in black. We overlay the projection of set $\mathcal{B}$ onto position space, and its sweep along the nominal Dubins airplane motion plan. The snapshot confirms that this "tube" remains obstacle free, and highlights the necessity of using a $\mathcal{B}$-based buffer as the plane negotiates difficult turns (see Figure 6.5b). A video of this simulation can be found at `https://www.youtube.com/watch?v=UsfaWsjectQ`.



(a) (b)

Figure 6.5: Left: snapshot of the nominal Dubins motion plan (red), along with the bounding ellipsoidal tube. The actual trajectory (black) stays within the ellipsoidal tube at all times. Right: Close-up of a tight turn at the end towards the goal.

Figure 6.6a plots the error states time series $e = (x_r, y_r, z_r, \psi_r)$ for the nominal planned path in Figure 6.5a for a set of varying initial conditions. As expected, the majority of the error is in the x_r direction where the bound is weakest. Figure 6.6b plots the time series $V(r(t))$, and confirms that $V(r(t)) \leq 1$ for all $t \geq 0$.



(a) (b)

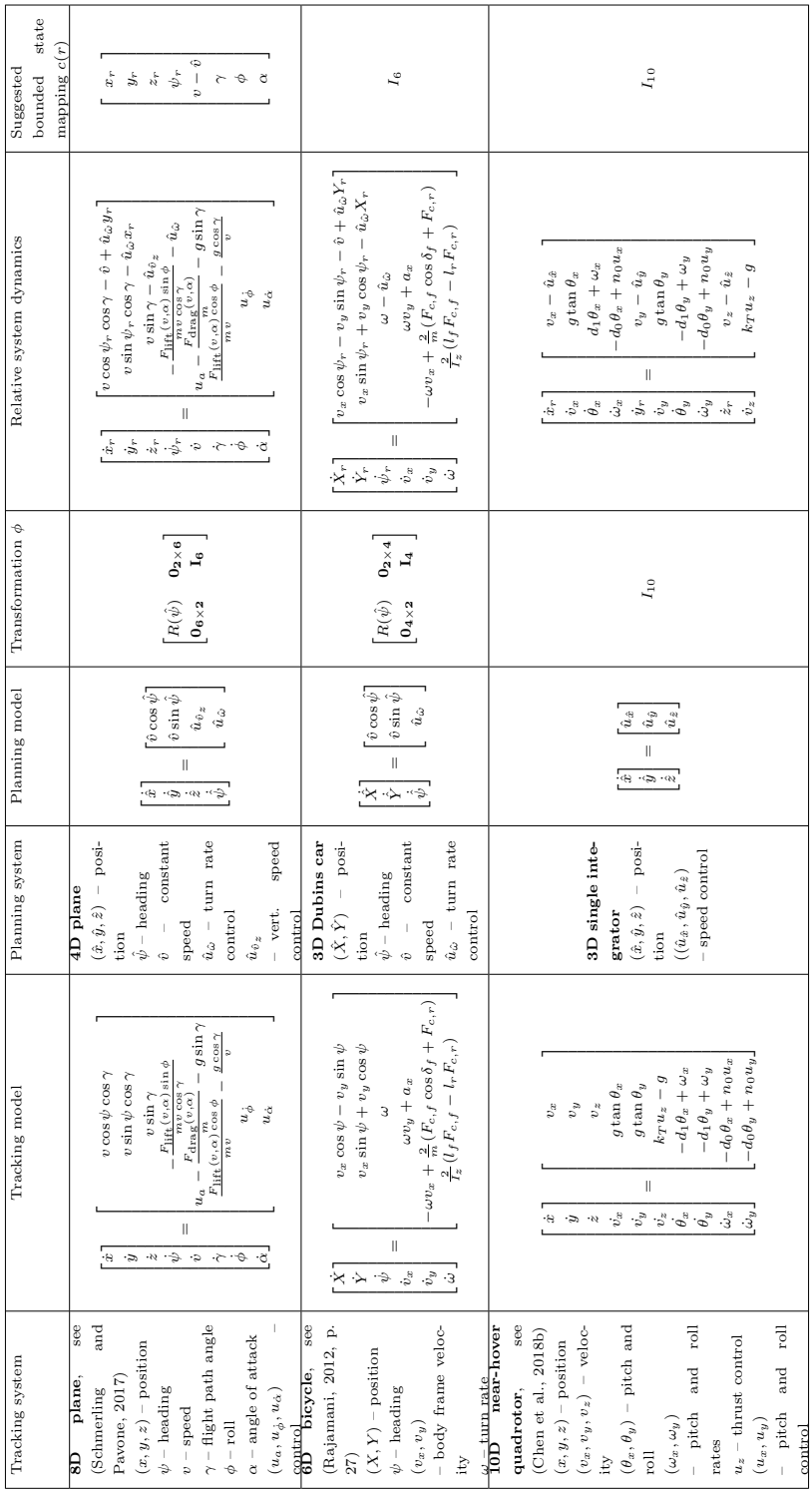
Figure 6.6: (a) Error states $e = (x_r, y_r, z_r, \psi_r)$; (b) Verification of $V(r(t)) \leq 1$ for all $t \geq 0$, for the nominal planned path in Figure 6.5a for a variety of initial conditions.

6.5 Summary

In this chapter we presented a principled and scalable approach for real-time planning with model mismatch. Specifically, by harnessing the tool of SOS programming, we designed an algorithmic framework to compute, offline, for a given pair of planning and tracking models, a feedback tracking controller and associated tracking bound. This bound can then used by the planning algorithm as a safety-margin when generating motion plans. The efficacy of the approach was verified via several illustrative examples, including comparions with a state-of-the-art method based on reachability analysis.

Tracking system	Tracking model	Planning system	Planning model	Transformation ϕ	Relative system dynamics	Suggested bounded state mapping $c(r)$
4D car (x, y) – position θ – heading v – speed u_ω – turn rate control u_a – accel. control	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{v}\end{bmatrix} \equiv \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ u_\omega\\ u_a\end{bmatrix}$	**2D single integrator** $(\hat{x}, \hat{y})$ – position $(\hat{u}_{\hat{x}}, \hat{u}_{\hat{y}})$ – speed control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{y}}\end{bmatrix} \equiv \begin{bmatrix}\hat{u}_{\hat{x}}\\ \hat{u}_{\hat{y}}\end{bmatrix}$	$\begin{bmatrix}R(\theta) & O_{2\times 2}\\ O_{1\times 2} & \begin{bmatrix}0 & 1\end{bmatrix}\end{bmatrix}$	$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{v}\end{bmatrix} \equiv \begin{bmatrix}v + u_\omega y_r - \tilde{u}_{\hat{x}}\\ -u_\omega x_r - \tilde{u}_{\hat{y}}\\ u_a\end{bmatrix}, \begin{bmatrix}\tilde{u}_{\hat{x}}\\ \tilde{u}_{\hat{y}}\end{bmatrix} = R(\theta)\begin{bmatrix}\hat{u}_{\hat{x}}\\ \hat{u}_{\hat{y}}\end{bmatrix}$	I_3
5D car (x, y) – position θ – heading v – speed ω – turn rate u_a – accel. control u_α – ang. accel. control	$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} \equiv \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ \omega\\ u_a\\ u_\alpha\end{bmatrix}$	**3D Dubins car** $(\hat{x}, \hat{y})$ – position $\hat{\theta}$ – heading $\hat{v}$ – constant speed $\hat{u}_\omega$ – turn rate control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{y}}\\ \dot{\hat{\theta}}\end{bmatrix} \equiv \begin{bmatrix}\hat{v}\cos\hat{\theta}\\ \hat{v}\sin\hat{\theta}\\ \hat{u}_{\hat{\omega}}\end{bmatrix}$	$\begin{bmatrix}R(\hat{\theta}) & \mathbf{0_{2\times 3}}\\ \mathbf{0_{3\times 2}} & \mathbf{I_3}\end{bmatrix}$	$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{\theta}_r\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}-\hat{v} + v\cos\theta_r + \hat{u}_\omega y_r\\ v\sin\theta_r - \hat{u}_{\hat{\omega}} x_r\\ \omega - \hat{u}_{\hat{\omega}}\\ u_a\\ u_\alpha\end{bmatrix}$	$\begin{bmatrix}x_r\\ y_r\\ \theta_r\\ v\cos\theta_r - \hat{v}\\ v\sin\theta_r\\ \omega\end{bmatrix}$
6D planar quadrotor (x, z) – position (v_x, v_z) – velocity θ – pitch ω – pitch rate u_T – thrust control u_τ – ang. accel. control	$\begin{bmatrix}\dot{x}\\ \dot{z}\\ \dot{v}_x\\ \dot{v}_z\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v_x\\ v_z\\ -u_T\sin\theta\\ u_T\cos\theta - g\\ \omega\\ u_\tau\end{bmatrix}$	**4D double integrator** $(\hat{x}, \hat{z})$ – position $(\hat{v}_x, \hat{v}_z)$ – velocity $(\hat{u}_{\hat{x}}, \hat{u}_{\hat{z}})$ – accel. control	$\begin{bmatrix}\dot{\hat{x}}\\ \dot{\hat{z}}\\ \dot{\hat{v}}_x\\ \dot{\hat{v}}_z\end{bmatrix} = \begin{bmatrix}\hat{v}_x\\ \hat{v}_z\\ \hat{u}_{\hat{x}}\\ \hat{u}_{\hat{z}}\end{bmatrix}$	I_6	$\begin{bmatrix}\dot{x}_r\\ \dot{z}_r\\ \dot{v}_{x,r}\\ \dot{v}_{z,r}\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}v_{x,r}\\ v_{z,r}\\ -u_T\sin\theta - \hat{u}_{\hat{x}}\\ u_T\cos\theta - g - \hat{u}_{\hat{z}}\\ \omega\\ u_\tau\end{bmatrix}$	I_6

Table 6.1: Catalog of tracking and planning system models (part 1); $R(\cdot) = \begin{bmatrix}\cos(\cdot) & \sin(\cdot)\\ -\sin(\cdot) & \cos(\cdot)\end{bmatrix}$ denotes the rotation matrix.



Tracking system	Tracking model	Planning system	Planning model	Transformation ϕ	Relative system dynamics	Suggested bounded state mapping $c(r)$
8D plane, see (Schmerling and Pavone, 2017) – (x, y, z) – position – ψ – heading – v – speed – γ – flight path angle – ϕ – roll – α – angle of attack – $(u_a, u_\phi, u_\dot{\alpha})$ – control	$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{v} \\ \dot{\gamma} \\ \dot{\phi} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} v\cos\psi\cos\gamma \\ v\sin\psi\cos\gamma \\ v\sin\gamma \\ \frac{-F_{\text{lift}}(v,\alpha)\sin\phi}{mv\cos\gamma} \\ u_a - \frac{F_{\text{drag}}(v,\alpha)}{m} - g\sin\gamma \\ \frac{F_{\text{lift}}(v,\alpha)\cos\phi}{mv} - \frac{g\cos\gamma}{v} \\ u_\phi \\ u_\dot{\alpha} \end{bmatrix}$	**4D plane** $(\hat{x}, \hat{y}, \hat{z})$ – position $\hat{\psi}$ – heading $\hat{v}$ – constant speed $\hat{u}_\dot{\alpha}$ – turn rate control $\hat{u}_{\dot{v}_z}$ – vert. speed control	$\begin{bmatrix} \dot{\hat{x}} \\ \dot{\hat{y}} \\ \dot{\hat{z}} \\ \dot{\hat{\psi}} \end{bmatrix} = \begin{bmatrix} \hat{v}\cos\hat{\psi} \\ \hat{v}\sin\hat{\psi} \\ \hat{u}_{\dot{v}_z} \\ \hat{u}_\dot{\alpha} \end{bmatrix}$	$\begin{bmatrix} R(\hat{\psi}) & \mathbf{0_{2\times 6}} \\ \mathbf{0_{6\times 2}} & \mathbf{I_6} \end{bmatrix}$	$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{z}_r \\ \dot{\psi}_r \\ \dot{v} \\ \dot{\gamma} \\ \dot{\phi} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} v\cos\psi_r\cos\gamma - \hat{v} + \hat{u}_\dot{\alpha} y_r \\ v\sin\psi_r\cos\gamma - \hat{u}_\dot{\alpha} x_r \\ v\sin\gamma - \hat{u}_{\dot{v}_z} \\ \frac{F_{\text{lift}}(v,\alpha)\sin\phi}{mv\cos\gamma} - \hat{u}_\dot{\alpha} \\ u_a - \frac{F_{\text{drag}}(v,\alpha)}{m} - g\sin\gamma \\ \frac{F_{\text{lift}}(v,\alpha)\cos\phi}{mv} - \frac{g\cos\gamma}{v} \\ u_\phi \\ u_\dot{\alpha} \end{bmatrix}$	$\begin{bmatrix} x_r \\ y_r \\ z_r \\ \psi_r \\ v - \hat{v} \\ \gamma \\ \phi \\ \alpha \end{bmatrix}$
6D bicycle, see (Rajamani, 2012, p. 27) – (X, Y) – position – ψ – heading – (v_x, v_y) – body frame velocity – ω – turn rate	$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x\cos\psi - v_y\sin\psi \\ v_x\sin\psi + v_y\cos\psi \\ \omega \\ \omega v_y + a_x \\ -\omega v_x + \frac{2}{m}(F_{c,f}\cos\delta_f + F_{c,r}) \\ \frac{2}{I_z}(l_f F_{c,f} - l_r F_{c,r}) \end{bmatrix}$	**3D Dubins car** $(\hat{X}, \hat{Y})$ – position – $\hat{\psi}$ – heading – $\hat{v}$ – constant speed – $\hat{u}_\dot{\alpha}$ – turn rate control	$\begin{bmatrix} \dot{\hat{X}} \\ \dot{\hat{Y}} \\ \dot{\hat{\psi}} \end{bmatrix} = \begin{bmatrix} \hat{v}\cos\hat{\psi} \\ \hat{v}\sin\hat{\psi} \\ \hat{u}_\dot{\alpha} \end{bmatrix}$	$\begin{bmatrix} R(\hat{\psi}) & \mathbf{0_{2\times 4}} \\ \mathbf{0_{4\times 2}} & \mathbf{I_4} \end{bmatrix}$	$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\psi}_r \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x\cos\psi_r - v_y\sin\psi_r - \hat{v} + \hat{u}_\dot{\omega} Y_r \\ v_x\sin\psi_r + v_y\cos\psi_r - \hat{u}_\dot{\omega} X_r \\ \omega - \hat{u}_\dot{\omega} \\ \omega v_y + a_x \\ -\omega v_x + \frac{2}{m}(F_{c,f}\cos\delta_f + F_{c,r}) \\ \frac{2}{I_z}(l_f F_{c,f} - l_r F_{c,r}) \end{bmatrix}$	I_6
10D near-hover quadrotor, see (Chen et al., 2018b) – (x, y, z) – position – (v_x, v_y, v_z) – velocity – (θ_x, θ_y) – pitch and roll – (ω_x, ω_y) – pitch and roll rates – u_z – thrust control – (u_x, u_y) – pitch and roll control	$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\omega}_x \\ \dot{\omega}_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ g\tan\theta_x \\ g\tan\theta_y \\ k_T u_z - g \\ -d_1\theta_x + \omega_x \\ -d_1\theta_y + \omega_y \\ -d_0\theta_x + n_0 u_x \\ -d_0\theta_y + n_0 u_y \end{bmatrix}$	**3D single integrator** $(\hat{x}, \hat{y}, \hat{z})$ – position $((\hat{u}_\dot{x}, \hat{u}_\dot{y}, \hat{u}_\dot{z})$ – speed control	$\begin{bmatrix} \dot{\hat{x}} \\ \dot{\hat{y}} \\ \dot{\hat{z}} \end{bmatrix} = \begin{bmatrix} \hat{u}_\dot{x} \\ \hat{u}_\dot{y} \\ \hat{u}_\dot{z} \end{bmatrix}$	I_{10}	$\begin{bmatrix} \dot{x}_r \\ \dot{v}_x \\ \dot{\theta}_x \\ \dot{\omega}_x \\ \dot{y}_r \\ \dot{v}_y \\ \dot{\theta}_y \\ \dot{\omega}_y \\ \dot{z}_r \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} v_x - \hat{u}_\dot{x} \\ g\tan\theta_x \\ d_1\theta_x + \omega_x \\ -d_0\theta_x + n_0 u_x \\ v_y - \hat{u}_\dot{y} \\ g\tan\theta_y \\ -d_1\theta_y + \omega_y \\ -d_0\theta_y + n_0 u_y \\ v_z - \hat{u}_\dot{z} \\ k_T u_z - g \end{bmatrix}$	I_{10}

Table 6.2: Catalog of tracking and planning system models (part 2); $R(\cdot) = \begin{bmatrix} \cos(\cdot) & \sin(\cdot) \\ -\sin(\cdot) & \cos(\cdot) \end{bmatrix}$ denotes the rotation matrix.

Part II

Control-Theoretic Regularization for Model-Based RL

Chapter 7

Planning with Unknown Dynamics

In the previous part of the thesis, we assumed knowledge of the dynamics used within planning. However, the problem of efficiently and accurately estimating an unknown dynamical system,

$$\dot{x}(t) = F(x(t), u(t)), \tag{7.1}$$

from a small set of sampled trajectories, is a central task in model-based Reinforcement Learning (RL). In this setting, a robotic agent strives to pair an estimated dynamics model with a feedback policy in order to act optimally in a dynamic and uncertain environment. The model of the dynamical system can be continuously updated as the robot experiences the consequences of its actions, and the improved model can be leveraged for different tasks, affording a natural form of transfer learning. When it works, model-based RL typically offers major improvements in sample efficiency in comparison to state-of-the-art model-free methods such as Policy Gradients, that do not explicitly estimate the underlying system. Yet, all too often, when standard supervised learning with powerful function approximators such as Deep Neural Networks and Kernel Methods are applied to model complex dynamics, the resulting controllers do not perform on par with model-free RL methods in the limit of increasing sample size, due to compounding errors across long time horizons. The main goal of this part of the thesis is to develop a new control-theoretic regularizer for dynamics fitting rooted in the notion of *stabilizability*, which guarantees that the existence of a robust tracking controller for arbitrary open-loop trajectories generated with the learned system.

7.1 Problem Statement and Solution Approach

The primary task we wish to solve is the motion planning problem, as stated in Section 2.1, in the absence of disturbances. That is, we wish to compute a (possibly non-stationary) policy mapping state and time to control that drives any given initial state to a desired compact goal region, while

satisfying state and control input constraints, and minimizing some task specific performance cost (e.g., control effort and time to completion). However, in this part of the thesis, we assume that the dynamics function $F(x, u)$ is unknown to us and we are instead provided with a dataset of tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ taken from a collection of observed trajectories (e.g., expert demonstrations) on the robot. Thus, the *problem* we wish to address is how to solve the motion planning task, given this dataset of pointwise samples.

The solution approach adopts the model-based RL paradigm, whereby one first estimates a model of the dynamical system $\hat{F}(x, u)$ using some form of regression, and then uses the learned model to solve the motion planning task with traditional planning algorithms. We will continue to adopt the "nominal plus tracking feedback" parameterization of the policy introduced in Section 2.2, composed of a nominal state-control trajectory $(x^*(\cdot), u^*(\cdot))$ and a tracking feedback controller $k(x^*, x)$. In this part of the thesis however, we do *not* present a new methodology for solving the planning task. Specifically, it is assumed that there exists an algorithm for computing (i) the open-loop state and control trajectories $(x^*(t), u^*(t))$ that minimize the open-loop cost:

$$J(u(\cdot)) = \int_0^{T_{\text{goal}}} 1 + \|u(t)\|_R^2 \, dt,$$

and (ii) the feedback tracking controller $k(\cdot, \cdot)$, *given* a dynamical model. The focus here is on how to design the regression algorithm for computing the model estimate $\hat{F}$.

Specifically, we illustrate that naïve regression techniques used to estimate the dynamics model from a small set of sample trajectories can yield model estimates that are severely ill-conditioned for trajectory generation and feedback control. Instead, we advocate for the use of a *constrained* regression approach in which one attempts to solve the following problem:

$$\min_{\hat{F} \in \mathcal{H}} \quad \sum_{i=1}^{N} \left\| \hat{F}(x_i, u_i) - \dot{x}_i \right\|^2 + \mu \|\hat{F}\|_{\mathcal{H}}^2 \tag{7.2}$$

$$\text{s.t.} \quad \hat{F} \text{ is stabilizable}, \tag{7.3}$$

where $\mathcal{H}$ is an appropriate normed function space and $\mu > 0$ is a regularization parameter. We will demonstrate that for systems that are indeed stabilizable, enforcing such a constraint drastically *prunes the hypothesis space*, and therefore plays the role of a *"control-theoretic" regularizer* that is potentially more powerful and ultimately, more pertinent for the downstream control task of generating and tracking new trajectories.

7.2 Model-Based RL: State-of-the-Art

Model-based RL has enjoyed considerable success in various application domains within robotics such as underwater vehicles (Cui et al., 2017), soft robotic manipulators (Thuruthel et al., 2019), and control of agents with non-stationary dynamics (Ohnishi et al., 2019). While the literature on model-based RL is substantial; see (Polydoros and Nalpantidis, 2017) for a recent review, we focus our attention on five broad categories relevant to the problem we address in this work. Namely, these are: (i) direct regression for learning the full dynamics, where one ignores any control-theoretic notions tied to the learning task and treats dynamics estimation as a standard regression problem; (ii) residual learning, where one only attempts to learn *corrections* to a nominal prediction model that may have been derived, for example, from physics-based reasoning; (iii) uncertainty-aware model-based RL, where one tries to additionally represent the uncertainty in the learned model using probabilistic representations that are subsequently leveraged within the planning phase using robust or stochastic control techniques; (iv) hybrid model-based/model-free methods; and (v) imitation learning, where one learns dynamical representations of stable closed-loop behavior for a set of outputs (e.g., the end-effector on a robotic arm), and assumes knowledge of the robot controlled dynamics to realize the learned closed-loop motion, for instance, using dynamic inversion.

The simplest approach to learning dynamics is to ignore stabilizability and treat the problem as a standard one-step time series regression task (Punjani and Abbeel, 2015; Bansal et al., 2016; Nagabandi et al., 2017; Polydoros and Nalpantidis, 2017). However, coarse dynamics models trained on limited training data typically generate trajectories that rapidly diverge from expected paths, inducing controllers that are ineffective when applied to the true system. This divergence can be reduced by expanding the training data with corrections to boost multi-step prediction accuracy (Venkatraman et al., 2015, 2016). Despite being effective, these methods are still heuristic in the sense that the existence of a stabilizing feedback controller is not explicitly guaranteed. Alternatively, one can leverage strong physics-based priors and use learning to only regress the unmodeled dynamics. For instance, (Mohajerin et al., 2019; Shi et al., 2019; Punjani and Abbeel, 2015) aim to capture the unmodeled aerodynamic disturbance terms as corrections to a prior rigid body dynamics model. (Punjani and Abbeel, 2015) accomplish this for helicopter dynamics using a deep neural network, but then do not use the learned model for control. (Shi et al., 2019) attempt to capture the unmodeled ground-effect forces on quadrotors to build better controllers for near-ground tracking and precision landing. (Mohajerin et al., 2019) leverage a residual RNN in combination with a rigid-body model to generate time-series predictions for linear and angular velocities of a quadrotor as a function of current state and candidate future motor inputs, but do not use the model for closed-loop control. Finally, (Zhou et al., 2017) adopt a different perspective to learning "corrections" in that they attempt to learn the inverse dynamics (output to reference) for a system and pre-cascade the resulting predictions to correct an existing controller's reference signal in order to improve trajectory tracking performance. The approach relies on the existence of a stabilizing controller and the

stability of the system's zero dynamics, thereby decoupling the effects of learning from stability. In similar spirit, (Taylor et al., 2019) leverage input-output feedback linearization to derive a Control Lyapunov Function (CLF) for a nominal dynamics model, assume that this function is a CLF for the actual dynamics as well, and regress only the correction terms in the derivative of this CLF. While leveraging physics-based priors can certainly be powerful, especially when the residual errors to be learned are small enough such that the system is feedback stabilizable with a controller derived from the physics model, in this work we are interested in the far more challenging scenario when such priors are unavailable and the full dynamics model must be learned from scratch. While exemplified using quadrotor models that can certainly be accurately stabilized even in the absence of learning, the insights provided in this work shed light on fundamental topics in the context of control-theoretic learning, which hopefully may influence dynamics-learning methods in more complex settings where priors are unavailable or too simple to be useful for adequate control.

An alternative strategy to cope with error in the learned dynamics model is to use uncertainty-aware model-based RL where control policies are optimized with respect to stochastic rollouts from probabilistic dynamics models (Kocijan et al., 2004; Kamthe and Deisenroth, 2018; Deisenroth and Rasmussen, 2011; Chua et al., 2018). For instance, PILCO (Deisenroth and Rasmussen, 2011) leverages a Gaussian Process (GP) state transition model and moment matching to analytically estimate the expected cost of a rollout with respect to the induced distribution. (Kamthe and Deisenroth, 2018) extend this formulation using nonlinear model predictive control (MPC) to incorporate chance constraints. (Chua et al., 2018) leverage an ensemble of probabilistic models to capture both epistemic (i.e., model) and aleatoric (i.e., intrinsic) uncertainty, and compute their control policy in receding horizon fashion through finite sample approximation of the random cost. Probabilistic models such as GPs may also be used to capture the residual error between a nominal physics-based model and the true dynamics. In (Ostafew et al., 2016), a GP is incrementally learned over multiple trials to capture unmodeled disturbances. The 3σ prediction range is subsequently leveraged to formulate chance constraints as a robust nonlinear MPC problem. The goal of (Fisac et al., 2017) and (Berkenkamp et al., 2017) is motivated from a safety perspective, where one wishes to actively learn a control policy while remaining "safe" in the presence of unmodeled dynamics, represented as GPs. The authors in (Fisac et al., 2017) leverage Hamilton-Jacobi reachability analysis to give high-probability invariance guarantees for a region of the state-space within which the learning controller is free to explore. On the other hand, (Berkenkamp et al., 2017) utilize Lyapunov analysis and smoothness arguments to incrementally grow the Lyapunov function's region of attraction while simultaneously updating the GP. For the special case where the underlying dynamics are linear-time-invariant, (Dean et al., 2019) derive high-probability convergence rates for the estimated model and leverage system-level robust control techniques (Wang et al., 2019) for guaranteeing state and control constraint satisfaction.

While utilizing probabilistic prediction models along with a control strategy that incorporates

this uncertainty, such as robust or approximate stochastic MPC, can certainly help guard against imperfect dynamics models, large uncertainty in the dynamics can lead to overly conservative strategies. This is true especially when the learned model is not merely a correction or residual term, or if the probabilistic model is computationally intractable to use within planning (e.g., GPs without additional sparsifying simplifications), thereby forcing conservative approximations. Finally, with the exception of the "safe" RL methods mentioned above, the learning algorithms themselves do not incorporate knowledge of the downstream application of the function being regressed, in that learning is viewed purely from a *statistical point-of-view, rather than within a control-theoretic context.*

More recently, hybrid combinations of model-based and model-free techniques have gained attention within the learning community. The authors in (Bansal et al., 2017a) use Bayesian optimization to find an optimal linear dynamics model whose induced MPC policy minimizes the task-specific cost. In similar spirit, (Amos et al., 2018) differentiate through the fixed-point solutions of a parametric MPC problem to find optimal MPC cost and dynamics functions in order to minimize the actual task-specific cost. (Nagabandi et al., 2017) use behavioral cloning with respect to an MPC policy generated from a learned dynamics model to initialize model-free policy fine-tuning. The works in (Levine et al., 2016; Finn et al., 2016; Chebotar et al., 2017) leverage subroutines where local time-varying dynamics are fitted around a set of policy rollouts, and then used to perform trajectory optimization via an LQR backward pass. The induced local linear-time-varying policy from this rollout is then used as a supervisory signal for global policy optimization. While these lines of work try to frame dynamics fitting within the downstream context of the task, thereby imbuing the resulting learning algorithm with a more closed-loop flavor, the learned dynamics may be substantially different from the actual dynamics of the robot since, with the exception of the local time-varying dynamics fitting, the true goal is to optimize the task-specific cost. This can yield distorted dynamic models whose induced policies are more cost-optimal than policies extracted from the true dynamics. Thus, while the work presented herein espouses a closed-loop learning ideology, it does so from the control-theoretic perspective of trajectory stabilizability, i.e., the true objective is dynamics fitting which will subsequently be used to derive optimal trajectories and tracking controllers.

Finally, we address lines of work closest in spirit to this work. Learning dynamical systems satisfying some desirable stability properties (such as asymptotic stability about an equilibrium point, e.g., for point-to-point motion) has been studied in the autonomous case, $\dot{x}(t) = f(x(t))$, in the context of imitation learning. In this line of work, one assumes perfect knowledge and invertibility of the robot's *controlled* dynamics to solve for the input that realizes this desirable closed-loop motion (Lemme et al., 2014; Khansari-Zadeh and Khatib, 2017; Ravichandar et al., 2017; Khansari-Zadeh and Billard, 2011; Medina and Billard, 2017). In particular, for a vector-valued RKHS formulation in the autonomous case with constant (identity) contraction metric, see (Sindhwani et al., 2018). Crucially, in our work, we *do not* require knowledge or invertibility of the robot's controlled dynamics. We seek

to learn the full controlled dynamics of the robot, under the constraint that the resulting learned dynamics generate dynamically feasible and most importantly, stabilizable trajectories. Thus, this work generalizes existing literature by additionally incorporating the controllability limitations of the robot within the learning problem.

The tools we develop may also be used to extend standard adaptive robot control design, such as (Slotine and Li, 1987) – a technique which achieves stable concurrent learning and control using a combination of physical basis functions and general mathematical expansions, e.g. radial basis function approximations (Sanner and Slotine, 1992). Notably, our work allows us to handle complex underactuated systems – a consequence of the significantly more powerful function approximation framework developed herein, as well as of the use of a differential (rather than classical) Lyapunov-like setting, as we shall detail.

7.3 Summary

Stabilizability of trajectories is not only a complex task in nonlinear control, but also a difficult notion to capture (in an algebraic sense) within a unified control theory. In this work, we leverage contraction theory, specifically CCMs, developed within the first part of the thesis, to represent stabilizability, and formulate the learning stabilizable dynamics problem by enforcing the existence of an *approximate* CCM. The resulting optimization problem is not only infinite-dimensional, as it is formulated over function spaces, but also infinitely-constrained due to the state-dependent LMI representing the stabilizability constraint (as was the case in the first part of this thesis). We make the following primary contributions:

- Under an arguably weak assumption on the structural form of the true dynamics model and a relaxation of the functional constraints to sampling-based constraints, we derive a Representer Theorem (Scholköpf and Smola, 2001) specifying the form of the optimal solutions for the dynamics functions and the certificate of stabilizability by leveraging the powerful framework of vector-valued Reproducing Kernel Hilbert Spaces. We motivate the sampling-based relaxation of the functional constraints from a standpoint of viewing the stabilizability condition as a novel control-theoretic *regularizer* for dynamics learning.

- By leveraging theory from randomized matrix feature approximations, we derive a tractable algorithm leveraging alternating convex optimization problems and adaptive sampling to iteratively solve a *finite-dimensional* optimization problem.

- We perform an extensive set of numerical simulations on the planar quadrotor model and provide a comprehensive study of various aspects of the iterative algorithm. Specifically, we demonstrate that naïve regression-based dynamics learning can yield estimated models that generate completely unstabilizable trajectories. In contrast, the control-theoretic regularized

model generates vastly superior quality trackable trajectories, especially when learning from small supervised datasets.

- We validate our algorithm on a quadrotor testbed with partially closed control loops to emulate a planar quadrotor, where we verify that the stabilizability regularization effects in low-data regimes observed in simulations does indeed generalize to real-world noisy data. In particular, with just 150 noisy tuples of $(x, u, \dot{x})$, we are able to stably track a challenging test trajectory, which is generated with the learned model and substantially different from any of the training data. In contrast, a model learned using traditional regression techniques leads to consistently unstable behavior and eventual failure as the quadrotor repeatedly flips out of control and crashes.

Chapter 8

Learning Stabilizable Models

In this chapter, we provide motivation for incorporating stabilizability as a constraint within model-based RL using the canonical planar quadrotor system, and formulate the stabilizable dynamics learning problem through the lens of CCMs. Leveraging a relaxation of the infinite-dimensional constraints associated with CCMs, we derive the form of the optimal solution to the learning problem by harnessing recent statistical learning results for vector-valued RKHS. Finally, using random matrix feature approximations, we further distill the stabilizable dynamics learning problem into *finite-dimensional* non-convex optimization.

8.1 Motivating Example

Consider the 6-state planar quadrotor example introduced in Section 3.3.2. The dynamics assume the general control-affine form:

$$\dot{x}(t) = f(x(t)) + B(x(t))u(t), \tag{8.1}$$

where $f : \mathcal{X} \to \mathbb{R}^n$, and $B : \mathcal{X} \to \mathbb{R}^{n\times m}$ is the input matrix mapping, depicted in column-stacked form as $(b_1, \ldots, b_m)$. Let us define the model estimate also in control-affine form as $\dot{x} = \hat{f}(x)+\hat{B}(x)u$, where $\hat{B} = (\hat{b}_1, \ldots, \hat{b}_m)$. Consider, as a first solution attempt, the following linear parameterization for the vector-valued functions $\hat{f}$ and $\hat{b}_j$:

$$\hat{f}(x) = \Phi_f(x)^T\alpha, \tag{8.2a}$$

$$\hat{b}_j(x) = \Phi_b(x)^T\beta_j \quad j \in \{1, \ldots, m\}, \tag{8.2b}$$

where $\alpha \in \mathbb{R}^{d_f}$, $\beta_j \in \mathbb{R}^{d_b}$ are constant vectors to be optimized over, and $\Phi_f : \mathcal{X} \to \mathbb{R}^{d_f\times n}$, $\Phi_b : \mathcal{X} \to \mathbb{R}^{d_b\times n}$ are a priori chosen feature mappings. To replicate the sparsity structure of the

input matrix, the feature matrix Φ_b has all zeros in its first $n - m$ columns.

The justification for a linear model and the construction of the feature mappings will be elaborated upon later. At this moment, we wish to study the quality of the learned models obtained from solving the following convex optimization problem:

$$\min_{\alpha,\{\beta_j\}} \quad \sum_{i=1}^{N} \|\hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i\|^2 + \mu_f \|\alpha\|^2 + \mu_b \sum_{j=1}^{m} \|\beta_j\|^2, \tag{8.3}$$

where $\mu_f, \mu_b > 0$ are given regularization constants. Note that the above optimization corresponds to the ubiquitous ridge-regression problem and is therefore a viable solution approach.

To evaluate the feasibility of this solution approach, we extracted a collection of training tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^{N}$ from simulations of the planar quadrotor without any noise (for further details, please see Section 9.1.2). We learned three models: (i) **N-R**: un-regularized model[1] ($\mu_f = 0$, $\mu_b = 10^{-6}$), (ii) **R-R**: standard ridge-regularized model with $\mu_f = 10^{-4}, \mu_b = 10^{-6}$, and (iii) **CCM-R**: control-theoretic regularized model, corresponding to the algorithm proposed within this work and elaborated upon in the remaining chapter.

We learned four versions of the model corresponding to varying training dataset sizes with $N \in \{100, 250, 500, 1000\}$. The dimensions of α and β_j were both 576 (corresponding to 96 parameters per state dimension). The feature mappings themselves are described in Section 9.1.2. The regularization constants were held fixed for all N.

Evaluation

The evaluation corresponded to the motion planning task of generating and tracking trajectories using the learned models. We gridded the (p_x, p_z) plane to create a set of 120 initial conditions between 4 m and 12 m away from $(0, 0)$, and randomly sampled the other states for the rest of the initial conditions. These conditions were *held fixed* for all models and for all training dataset sizes to evaluate model improvement.

For each model at each value of N, the evaluation task was to (i) solve a trajectory optimization problem to compute a dynamically feasible trajectory for the learned model to go from initial state x_0 to the goal state - a stable hover at $(0, 0)$ at near-zero velocity; and (ii) track this trajectory with a feedback controller computed using time-varying LQR (TV-LQR). Note that all simulations without any feedback controller (i.e., open-loop control rollouts) led to the PVTOL crashing. This is understandable since the dynamics fitting objective does not optimize for *multi-step* error. The trajectory optimization step was solved as a fixed-endpoint, fixed-final time optimal control problem using the Chebyshev pseudospectral method (Fahroo and Ross, 2002) with the objective of minimizing $\int_0^T \|u(t)\|_R^2 \, dt$. The final time T for a given initial condition was held fixed between all models.

[1] A small penalty on μ_b was necessary since the input feature matrix Φ_b was chosen to be constant (and degenerate) to reflect the fact that the input matrix for the planar quadrotor is constant.

Note that 120 trajectory optimization problems were solved for each model and each value of N.

In Figure 8.1, we highlight two trajectories that start from the *same initial conditions* – one generated and tracked using the R-R model, the other using the CCM-R model, for $N = 250$. Overlaid on the plot are snapshots of the vehicle outline itself, illustrating the aggressive flight-regime of the trajectories (the initial bank angle is 40°). While tracking the R-R model generated trajectory eventually ends in complete loss of control, the system successfully tracks the CCM-R model generated trajectory to the stable hover at $(0, 0)$.

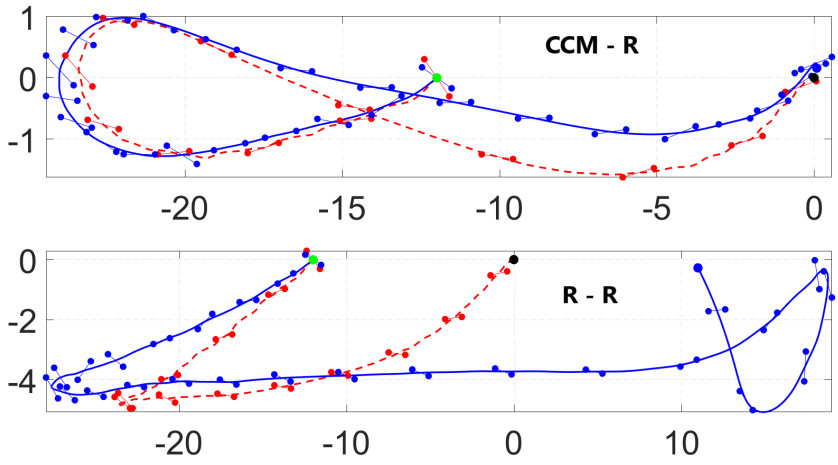


Figure 8.1: Comparison of reference and tracked trajectories in the (p_x, p_z) plane for R-R and CCM-R models starting at the same initial conditions with $N = 250$. Red (dashed): nominal, Blue (solid): actual, Green dot: start, Black dot: nominal endpoint, blue dot: actual endpoint. The vehicle successfully tracks the CCM-R model generated trajectory to the stable hover at $(0, 0)$ while losing control when attempting to track the R-R model generated trajectory.

Figure 8.2 shows a boxplot comparison of the trajectory RMS full state errors ($\|x(t) - x^*(t)\|$ where $x^*(t)$ is the reference trajectory obtained from the optimizer and $x(t)$ is the actual realized trajectory). As N increases, the spread of the RMS errors decreases for both R-R and CCM-R models as expected. However, we see that the N-R model generates *several* unstable trajectories for $N \in \{100, 500, 1000\}$, indicating the need for a form of regularization.

For $N = 100$ (the extreme lower limit of the necessary number of samples as there are 96 features for each dimension of the dynamics function), both N-R and R-R models generate a large number of unstable trajectories. In contrast, *all* trajectories generated with the CCM-R model were successfully tracked with bounded error. Indeed, the CCM-R model consistently achieves a lower RMS error distribution than both the N-R and R-R models *for all training dataset sizes*. Most notable, however, is its performance when the number of supervision training samples is small (i.e., $N \in \{100, 250\}$) and there is considerable risk of overfitting. It appears the stabilizability constraints leveraged to compute the CCM-R model have a notable regularizing effect on the resulting model trajectories (recall that the initial conditions of the trajectories are held fixed between the models).

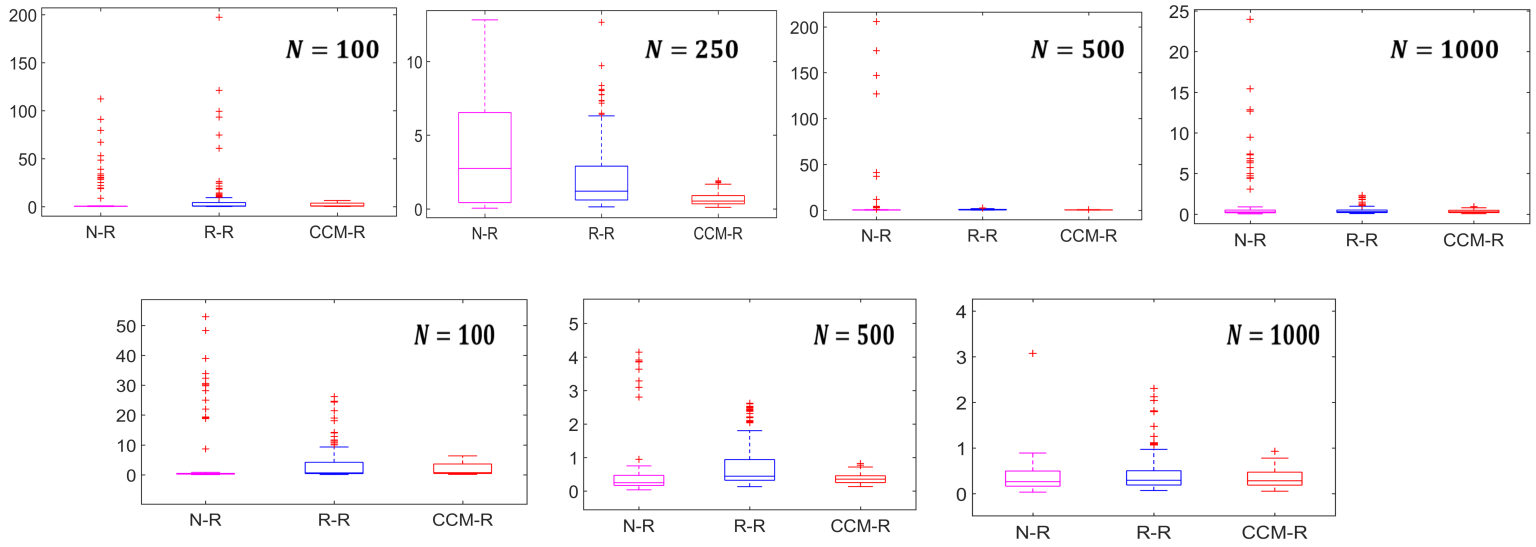


Figure 8.2: Box-whisker plot comparison of trajectory-wise RMS state-tracking errors over all 120 trajectories for each model and all training dataset sizes. *Top row, left-to-right:* $N = 100, 250, 500, 1000$; *Bottom row, left-to-right:* $N = 100, 500, 1000$ (zoomed in). The box edges correspond to the 25th, median, and 75th percentiles; the whiskers extend beyond the box for an additional 1.5 times the interquartile range; outliers, classified as trajectories with RMS errors past this range, are marked with red crosses. Notice the presence of unstable trajectories for N-R at all values of N and for R-R at $N = 100, 250$. The CCM-R model dominates the other two *at all values of* N, particularly for $N \in \{100, 250\}$.

Effect of Regularization

At this point, one might wonder if the choice of the regularization parameter μ_f may be sub-optimal for the R-R model. Traditionally, such parameters are tuned using regression error on a validation dataset. In Figure 8.3 we plot the mean regression error $\|\hat{f} + \hat{B}u - \dot{x}\|$ over an independently sampled validation dataset of 2000 demonstration tuples, as a function of the regularization parameter μ_f, for all R-R models.

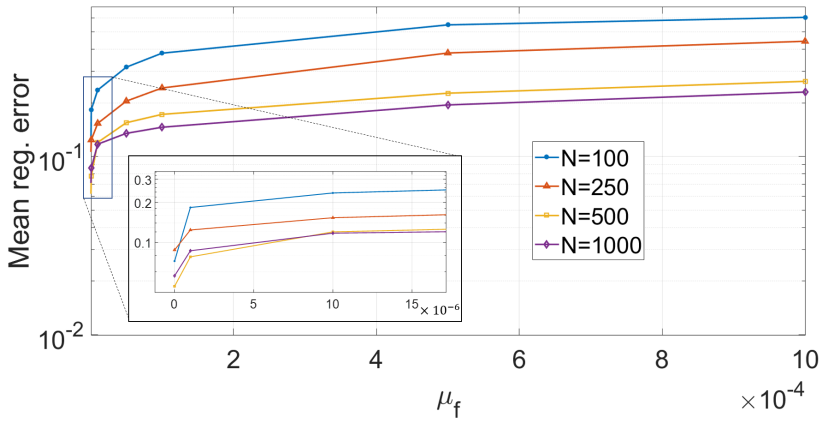


Figure 8.3: Mean regression error over an independent validation dataset as a function of μ_f for the RR model learned using (8.3), with varying training set size N. The best out of sample performance is achieved with $\mu_f = 0.0$. The constant μ_b is fixed at 10^{-6}.

The plot illustrates that the best out-of-sample performance is achieved with $\mu_f = 0.0$. However, this corresponds to the N-R model which, as we learned in the previous section, generated several unstable trajectories for all training dataset sizes. This is not a surprising result; the feature mapping used as the basis for the dynamics model corresponds to the randomized matrix approximation of a reproducing kernel (see Section 8.4.4). Recent results, as in (Liang and Rakhlin, 2018) and references within, corroborate such a pattern and even advocate for "ridgeless" regression.

Given that the CCM-R model uses the *same* feature mapping as the R-R and N-R models (i.e., the model capacity of all three models is the same), and is given the same set of demonstration tuples, it appears that traditional model-fitting techniques such as ridge-regression and associated hyper-parameter tuning rules are ill-suited to learn representations of dynamics that are appropriate for planning and control. This motivates the need for *constrained* dynamics learning, where the notion of *model stabilizability* is encoded as a constraint within the learning algorithm (as opposed to the unconstrained optimization in (8.3)).

8.2 CCM Constrained Dynamics Learning

Leveraging the characterization of stabilizability via CCMs, consider the following formulation of stabilizable dynamics learning. Given a supervised dataset of demonstration tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$, we wish to learn the dynamics functions $f(x)$ and $B(x)$ in (8.1), subject to the constraint that there exists a CCM $M(x)$ for the learned dynamics. That is, the CCM $M(x)$ plays the role of a *certificate* of stabilizability for the learned dynamics.

As discussed in Section 3.3, a necessary and sufficient characterization of a CCM $M(x)$ for the system $\{\hat{f}, \hat{B}\}$ is given in terms of its dual $W(x) := M(x)^{-1}$. For ease of presentation, we repeat these conditions here:

$$\hat{B}_\perp^T \left(\partial_{\hat{b}_j} W(x) - \widehat{\frac{\partial \hat{b}_j(x)}{\partial x} W(x)} \right) \hat{B}_\perp = 0, \ j = 1, \dots, m \quad \forall x \in \mathcal{X}, \tag{8.4}$$

$$\underbrace{\hat{B}_\perp(x)^T \left(-\partial_{\hat{f}} W(x) + \widehat{\frac{\partial \hat{f}(x)}{\partial x} W(x)} + 2\lambda W(x) \right) \hat{B}_\perp(x)}_{:=\mathcal{F}_\lambda(x;\hat{f},W)} \preceq 0, \quad \forall x \in \mathcal{X}, \tag{8.5}$$

where $\hat{B}_\perp$ is the annihilator matrix for $\hat{B}$, i.e., $\hat{B}(x)^T \hat{B}_\perp(x) = 0$ for all x. In the definition above, we write $\mathcal{F}_\lambda(x; \hat{f}, W)$ since $\{\hat{f}, W\}$ will be optimization variables in our formulation while λ is treated as a hyper-parameter. Thus, the learning task reduces to finding the functions $\{\hat{f}, \hat{B}, W\}$ that jointly satisfy the above constraints, while minimizing an appropriate regularized regression loss function. Formally, problem (7.2) can be re-stated as:

$$\min_{\substack{\hat{f}\in\mathcal{H}^f, \hat{b}_j\in\mathcal{H}^B, j=1,\dots,m \\ W\in\mathcal{H}^W \\ \underline{w},\overline{w}\in\mathbb{R}_{>0}}} \quad \overbrace{\sum_{i=1}^N \left\| \hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i \right\|^2 + \mu_f \|\hat{f}\|_{\mathcal{H}^f}^2 + \mu_b \sum_{j=1}^m \|\hat{b}_j\|_{\mathcal{H}^B}^2}^{:=J_d(\hat{f},\hat{B})} +$$

$$+ \underbrace{(\overline{w} - \underline{w}) + \mu_w \|W\|_{\mathcal{H}^W}^2}_{:=J_m(W,\underline{w},\overline{w})} \tag{8.6a}$$

$$\text{s.t.} \quad \text{eqs. (8.4), (8.5)} \quad \forall x \in \mathcal{X}, \tag{8.6b}$$

$$\underline{w} I_n \preceq W(x) \preceq \overline{w} I_n \quad \forall x \in \mathcal{X}, \tag{8.6c}$$

where $\mathcal{H}^f$ and $\mathcal{H}^B$ are appropriately chosen $\mathbb{R}^n$-valued function classes on $\mathcal{X}$ for $\hat{f}$ and $\{\hat{b}_j\}_{j=1}^m$ respectively, and $\mathcal{H}^W$ is a suitable $\mathbb{S}_n^{>0}$-valued function space on $\mathcal{X}$. The objective is composed of a dynamics term J_d – consisting of regression loss and regularization terms, and a metric term J_m – consisting of a condition number surrogate loss on the metric $W(x)$ and a regularization term. The metric cost term $\overline{w} - \underline{w}$ is motivated by the observation that the state tracking error (i.e.,

$\|x(t) - x^*(t)\|^2$) in the presence of bounded additive disturbances is proportional to the ratio $\overline{w}/\underline{w}$; see Section 3.2.

Notice that the coupling constraint (8.5) is a bi-linear matrix inequality in the decision variables $\hat{f}$ and W. Thus, at a high-level, a solution algorithm must consist of alternating between two convex sub-problems, defined by the objective and decision variable pairs $(J_d, \{\hat{f}, \hat{B}\})$ and $(J_m, \{W, \underline{w}, \overline{w}\})$. We refer to the first sub-problem as the *dynamics* sub-problem, and the second as the *metric* sub-problem. These sub-problems are described in full in Section 9.1.

Remark 9. *While one may include λ as part of the overall optimization problem; indeed this is the case within the robust planning setting, doing so would require either (i) adding $-\lambda$ to the dynamics sub-problem cost function, or (ii) line-searching over λ within the metric sub-problem. The first option detracts from the primary objective of the dynamics sub-problem, which is to minimize regression error, while the second option induces needless complexity. Thus, in this work, λ is held fixed as a tuning parameter.*

8.3 CCM Regularized Dynamics Learning

When performing dynamics learning on a system that is a priori *known* to be exponentially stabilizable with rate at least λ, the constrained problem formulation in (8.6) follows naturally given the assured *existence* of a CCM. Alternatively, for systems that are not globally exponentially stabilizable in $\mathcal{X}$, one can imagine that such a constrained formulation may lead to adverse bias effects on the learned dynamics model.

Thus, in this section we propose a relaxation of problem (8.6) motivated by the concept of regularization. Specifically, constraints (8.4) and (8.5) capture this notion of stability of infinitesimal deviations *at all points* in the space $\mathcal{X}$. They stem from requiring that $\dot{V} \leq -2\lambda V(x, \delta_x)$ in (3.3) when $\delta_x^T M(x) B(x) = 0$, i.e., when δ_u can have no effect on $\dot{V}$. This is nothing but the standard control Lyapunov inequality, applied to the differential setting. Constraint (8.4) sets to zero, the terms in (3.3) affine in u, while constraint (8.5) enforces this "natural" stability condition.

The simplifications we make are (i) relax constraints (8.5) and (8.6c) to hold pointwise over some *finite* constraint set $X_c \in \mathcal{X}$, and (ii) assume a specific sparsity structure for the input matrix estimate $\hat{B}(x)$. We discuss the pointwise relaxation here; the sparsity assumption on $\hat{B}(x)$ is discussed in the following section.

First, from a purely mathematical standpoint, the pointwise relaxation of (8.5) and (8.6c) is motivated by the observation that, as the CCM-based controller is exponentially stabilizing, we only require the differential stability condition to hold locally (in a tube-like region) with respect to the provided demonstrations. By continuity of eigenvalues for continuously parameterized entries of a matrix, it is sufficient to enforce the matrix inequalities at a sampled set of points (Lax, 2007).

Second, enforcing the existence of such an "approximate" CCM seems to have an impressive

regularization effect on the learned dynamics that is more meaningful than standard regularization techniques used in for instance, ridge or lasso regression. Specifically, problem (8.6), and more generally, problem (7.2) can be viewed as the *projection* of the best-fit dynamics onto the set of stabilizable systems. This results in dynamics models that jointly balance regression performance and stabilizablity, ultimately yielding systems whose generated trajectories are notably easier to track. This effect of regularization is apparent in the simulation results presented in Section 8.1, and the hardware testbed results in Section 9.2.

Practically, the finite constraint set X_c, with cardinality N_c, includes all x_i in the supervised regression training set of $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ tuples. However, as the LMI constraints are *independent* of u_i and $\dot{x}_i$, the set X_c is chosen as a strict superset of $\{x_i\}_{i=1}^N$ (i.e., $N_c > N$) by randomly sampling additional points within $\mathcal{X}$, drawing parallels with *semi-supervised learning*.

8.3.1 Sparsity of Input Matrix

While a pointwise relaxation for the matrix inequalities is reasonable, one cannot apply such a relaxation to the exact equality condition in (8.4). Thus, the second simplification made is the following assumption, reminiscent of control normal form equations.

Assumption 2. *Assume $\hat{B}(x)$ to take the following sparse representation:*

$$\hat{B}(x) = \begin{bmatrix} O_{(n-m)\times m} \\ \mathfrak{b}(x) \end{bmatrix}, \tag{8.7}$$

where $\mathfrak{b}(x)$ is an invertible $m \times m$ matrix for all $x \in \mathcal{X}$.

For the assumed structure of $\hat{B}(x)$, a valid $\hat{B}_\perp$ matrix is then given by:

$$\hat{B}_\perp = \begin{bmatrix} I_{n-m} \\ O_{m\times(n-m)} \end{bmatrix}. \tag{8.8}$$

Therefore, constraint (8.4) simply becomes:

$$\partial_{\hat{b}_j} W_\perp(x) = 0, \quad j = 1, \ldots, m,$$

where $W_\perp$ is the upper-left $(n-m) \times (n-m)$ block of $W(x)$. Assembling these constraints for the (p, q) entry of $W_\perp$, i.e., $w_{\perp_{pq}}$, we obtain:

$$\begin{bmatrix} \dfrac{\partial w_{\perp_{pq}}(x)}{\partial x^{(n-m)+1}} & \cdots & \dfrac{\partial w_{\perp_{pq}}(x)}{\partial x^n} \end{bmatrix} \mathfrak{b}(x) = 0.$$

Since the matrix $\mathfrak{b}(x)$ in (8.7) is assumed to be invertible, the *only* solution to this equation is $\partial w_{\perp_{pq}}/\partial x^i = 0$ for $i \in \{(n-m)+1, \ldots, n\}$, and all $(p, q) \in \{1, \ldots, (n-m)\}$. That is, $W_\perp$ cannot

be a function of the last m components of x – an elegant simplification of constraint (8.4).

Let us justify the structural assumption in (8.7). Physically, this assumption is not as mysterious as it appears. Indeed, consider the standard dynamics form for mechanical systems:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = B(q)u,$$

where $q \in \mathbb{R}^{n_q}$ is the configuration vector, $H \in \mathbb{S}^{>0}_{n_q}$ is the inertia matrix, $C(q,\dot{q})\dot{q}$ contains the centrifugal and Coriolis terms, g are the gravitational terms, $B \in \mathbb{R}^{n_q \times m}$ is the (full rank) input matrix mapping, and $u \in \mathbb{R}^m$ is the input; for fully actuated systems, $\text{rank}(B) = m = n_q$. For underactuated systems, $m < n_q$. By rearranging the configuration vector (Spong, 1998; Olfati-Saber, 2001; Reyhanoglu et al., 1999), one can partition q as (q_u, q_a) where $q_u \in \mathbb{R}^{n_q - m}$ represents the unactuated degrees of freedom and $q_a \in \mathbb{R}^m$ represents the actuated degrees of freedom. Applying this partitioning to the dynamics equation above yields:

$$\begin{bmatrix} H_{uu}(q) & H_{ua}(q) \\ H_{ua}(q) & H_{aa}(q) \end{bmatrix} \begin{bmatrix} \ddot{q}_u \\ \ddot{q}_a \end{bmatrix} + \begin{bmatrix} \tau_u(q,\dot{q}) \\ \tau_a(q,\dot{q}) \end{bmatrix} = \begin{bmatrix} O_{(n_q-m)\times m} \\ \mathfrak{b}(q) \end{bmatrix} u,$$

where $\mathfrak{b} \in \mathbb{R}^{m \times m}$ is an invertible square matrix. As observed in (Reyhanoglu et al., 1999), a substantial class of underactuated systems can be represented in this manner. Of course, fully actuated systems (i.e., $m = n_q$) also take this form. Thus, by taking as state $x = (q,p) \in \mathbb{R}^n$ where $p = H(q)\dot{q}$ is momentum (so that $n = 2n_q$), the dynamics can be written as:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} H^{-1}(q)p \\ \dot{H}(q)\dot{q} - \tau(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} O_{(n-m)\times m} \\ \mathfrak{b}(q) \end{bmatrix} u. \tag{8.9}$$

Notice that the input matrix takes the desired normal form in (8.7). To address the apparent difficult of working with the state representation of (q,p) (when usually only measurements of $(q,\dot{q},\ddot{q})$ are typically available *and* the inertia matrix $H(q)$ is unknown), one may leverage the following result from (Manchester and Slotine, 2017):

Theorem 7 (CCM Invariance to Diffeomorphisms)**.** *Suppose there exists a valid CCM $M_x(x)$ with respect to the state x. Then, if $z = \psi(x)$ is a diffeomorphism, then the CCM conditions also hold with respect to state z with metric $M_z(z) = \Psi(z)^{-T} M_x(x) \Psi(z)^{-1}$, where $\Psi(z) = \partial\psi(x)/\partial x$ is evaluated at $x = \psi^{-1}(z)$.*

Thus, for the (substantial) class of underactuated systems of the form (8.9), one would solve problem (8.6) by fitting the dynamics terms $\{H, \tau, b\}$ using the state representation $x = (q,\dot{q})$, and leveraging Theorem 7 and the previous estimate of H to enforce the matrix inequality constraints using the state representation (q,p). This allows us to borrow several existing results from adaptive control on estimating mechanical system by leveraging the known linearity of $H(q)$ in terms of

unknown mass property parameters multiplying known physical basis functions. We leave this extension however, to future work, and from hereon assume the structural form given in (8.8) for the estimated input matrix.

8.3.2 Finite-dimensional Optimization

We now present a tractable finite-dimensional optimization for solving problem (8.6) under the two simplifying assumptions introduced in the previous sections. For ease of presentation, we outline the final optimization formulation here and provide the derivation, which relies extensively on vector-valued RKHS, in the following section.

Parameterization: Model $\hat{f}, \{\hat{b}_j\}_{j=1}^m$ and $\{w_{ij}\}_{i,j=1}^n$ as a linear combination of features. That is,

$$\hat{f}(x) = \Phi_f(x)^T \alpha, \tag{8.10}$$

$$\hat{b}_j(x) = \Phi_b(x)^T \beta_j \quad j \in \{1, \dots, m\}, \tag{8.11}$$

$$w_{ij}(x) = \begin{cases} \hat{\phi}_w(x)^T \hat{\theta}_{ij} & \text{if} \quad (i,j) \in \{1, \dots, n-m\}, \\ \phi_w(x)^T \theta_{ij} & \text{else}, \end{cases} \tag{8.12}$$

where $\alpha \in \mathbb{R}^{d_f}$, $\beta_j \in \mathbb{R}^{d_b}$, $\hat{\theta}_{ij}, \theta_{ij} \in \mathbb{R}^{d_w}$ are constant vectors to be optimized over, and $\Phi_f : \mathcal{X} \to \mathbb{R}^{d_f \times n}$, $\Phi_b : \mathcal{X} \to \mathbb{R}^{d_b \times n}$, $\hat{\phi}_w : \mathcal{X} \to \mathbb{R}^{d_w}$ and $\phi_w : \mathcal{X} \to \mathbb{R}^{d_w}$ are a priori chosen feature mappings. To enforce the sparsity structure in (8.7), the feature matrix Φ_b must have all 0s in its first $n-m$ columns. The features $\hat{\phi}_w$ are distinct from ϕ_w in that the former are only a function of the first $n-m$ components of x (as per Section 8.3.1). While one can use any function approximator (e.g., neural nets), we motivate this parameterization from a perspective of RKHS – see Section 8.4.

Optimization: Given positive regularization constants μ_f, μ_b, μ_w and positive tolerances $\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}$, solve:

$$\min_{\alpha, \beta_j, \hat{\theta}_{ij}, \theta_{ij}, \underline{w}, \overline{w}} \quad \overbrace{\sum_{i=1}^N \|\hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i\|^2 + \mu_f \|\alpha\|^2 + \mu_b \sum_{j=1}^m \|\beta_j\|^2}^{:=\hat{J}_d} + \underbrace{(\overline{w} - \underline{w}) + \mu_w \sum_{i,j} \|\tilde{\theta}_{ij}\|^2}_{:=\hat{J}_m} \tag{8.13a}$$

$$\text{s.t.} \quad \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha, \tilde{\theta}_{ij}) \preceq 0 \quad \forall x_i \in X_c, \tag{8.13b}$$

$$(\underline{w} + \epsilon_{\underline{w}}) I_n \preceq W(x_i) \preceq \overline{w} I_n \quad \forall x_i \in X_c, \tag{8.13c}$$

$$\theta_{ij} = \theta_{ji}, \ \hat{\theta}_{ij} = \hat{\theta}_{ji} \tag{8.13d}$$

$$\underline{w} \geq \delta_{\underline{w}}, \tag{8.13e}$$

where $\tilde{\theta}_{ij}$ is used as a placeholder for θ_{ij} and $\hat{\theta}_{ij}$ to simplify notation, and we use $\hat{J}_d$ and $\hat{J}_m$ to distinguish them from the functional equivalents J_d and J_m in problem (8.6). We wish to highlight the following key points regarding problem (8.13). Constraints (8.13b) and (8.13c) are the pointwise relaxations of (8.5) and (8.6c) respectively; (8.13d) enforces the symmetry of $W(x)$, and (8.13e) imposes some tolerance requirements to ensure a well conditioned solution. Additionally, the tolerances ϵ_λ and $\epsilon_{\underline{w}}$ are used to account for the pointwise relaxations of the matrix inequalities. A key challenge is to efficiently solve this constrained optimization problem, given a potentially large number of constraint points in X_c. In Section 9.1, we present an iterative algorithm and an adaptive constraint sampling technique to solve problem (8.13).

8.4 Derivation of a Finite Dimensional Problem

To go from the general problem definition in (8.6) to the finite dimensional problem in (8.13), we first must define appropriate function classes for f, b_j, and W. We will do this using the framework of RKHS. We first provide a brief introduction to RKHS theory.

8.4.1 Reproducing Kernel Hilbert Spaces

Scalar-valued RKHS: Kernel methods (Scholköpf and Smola, 2001) constitute a broad family of non-parametric modeling techniques for solving a range of problems in machine learning. A scalar-valued positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ generates a RKHS $\mathcal{H}_k$ of functions, with the nice property that if two functions are close in the distance derived from the norm (associated with the Hilbert space), then their pointwise evaluations are close at all points. This continuity of evaluation functionals has the far-reaching consequence that norm-regularized learning problems over RKHSs admit finite dimensional solutions via Representer Theorems. The kernel k may be (non-uniquely) associated with a higher-dimensional embedding of the input space via a feature map, $\phi : \mathbb{R}^n \mapsto \mathbb{R}^D$, such that $k(x, z) = \phi(x)^T \phi(z)$, where D is infinite for universal kernels associated with RKHSs that are dense in the space of square-integrable functions. Standard regularized linear statistical models in the embedding space implicitly provide nonlinear inference with respect to the original input representation. In a nutshell, kernel methods provide a rigorous algorithmic framework for deriving nonlinear counterparts of a whole array of linear statistical techniques, e.g., for classification, regression, dimensionality reduction, and unsupervised learning. For further details, we point the reader to (Hearst et al., 1998).

Vector-valued RKHS: Dynamics estimation is a vector-valued learning problem. Such problems can be naturally formulated in terms of vector-valued generalizations of RKHS concepts. The theory and formalism of vector-valued RKHS can be traced as far back as the work of Laurent Schwartz (Schwartz, 1964), with applications ranging from solving partial differential equations to machine learning. Informally, we say that that $\mathcal{H}$ is an RKHS of $\mathbb{R}^n$-valued maps if for any $v \in \mathbb{R}^n$,

the linear functional that maps $f \in \mathcal{H}$ to $v^T f(x)$ is continuous.

More formally, denote the standard inner product on $\mathbb{R}^n$ as $\langle \cdot, \cdot \rangle$, and let $\mathbb{R}^n(\mathcal{X})$ be the vector space of all functions $f : \mathcal{X} \to \mathbb{R}^n$ and let $\mathcal{L}(\mathbb{R}^n)$ be the space of all bounded linear operators on $\mathbb{R}^n$, i.e., $n \times n$ matrices. A function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ is an *operator-valued positive definite kernel* if for all $(x, z) \in \mathcal{X} \times \mathcal{X}$: $K(x, z)^T = K(z, x)$, and for all finite set of points $\{x_i\}_{i=1}^N \in \mathcal{X}$ and $\{y_i\}_{i=1}^N \in \mathbb{R}^n$,

$$\sum_{i,j=1}^{N} \langle y_i, K(x_i, x_j) y_j \rangle \geq 0.$$

Given such a K, we define the unique $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K \subset \mathbb{R}^n(\mathcal{X})$ with reproducing kernel K as follows. For each $x \in \mathcal{X}$ and $y \in \mathbb{R}^n$, define the function $K_x y = K(\cdot, x)y \in \mathbb{R}^n(\mathcal{X})$. That is, $K_x y(z) = K(z, x)y$ for all $z \in \mathcal{X}$. Then, the Hilbert space $\mathcal{H}_K$ is defined to be the completion of the linear span of functions $\{K_x y \mid x \in \mathcal{X}, y \in \mathbb{R}^n\}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ between functions $f = \sum_{i=1}^N K_{x_i} y_i,\ g = \sum_{j=1}^{N'} K_{z_j} w_j \in \mathcal{H}_K$, defined as:

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_{i=1}^{N} \sum_{j=1}^{N'} \langle y_i, K(x_i, z_j) w_j \rangle ,$$

and function norm $\|f\|^2_{\mathcal{H}_K}$ given as $\langle f, f \rangle_{\mathcal{H}_K}$. Analogous to the canonical reproducing property of scalar-valued RKHS kernels, i.e.,

$$h(x) = \langle h, k(x, \cdot) \rangle_{\mathcal{H}_k} \quad \forall h \in \mathcal{H}_k, \tag{8.14}$$

the matrix-valued kernel K satisfies the following reproducing property:

$$\langle f(x), y \rangle = \langle f, K_x y \rangle_{\mathcal{H}_K} \quad \forall (x, y) \in \mathcal{X} \times \mathbb{R}^n, \forall f \in \mathcal{H}_K, \tag{8.15}$$

and is thereby referred to as the *reproducing kernel* for $\mathcal{H}_K$. As our learning problem involves the Jacobian of f, we will also require a representation for the derivative of the kernel matrix. Accordingly, suppose the kernel matrix K lies in $\mathcal{C}^2(\mathcal{X} \times \mathcal{X})$. For any $j \in \{1, \ldots, n\}$, define the matrix functions $\frac{\partial}{\partial s^j} K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ and $\frac{\partial}{\partial r^j} K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ as

$$\frac{\partial}{\partial s^j} K(z, x) := \frac{\partial}{\partial s^j} K(r, s)\bigg|_{r=z, s=x} \quad \frac{\partial}{\partial r^j} K(z, x) := \frac{\partial}{\partial r^j} K(r, s)\bigg|_{r=z, s=x},$$

where the derivative of the kernel matrix is understood to be element-wise. Define the $\mathcal{C}^1$ function $\partial_j K_x y \in \mathbb{R}^n(\mathcal{X})$ as

$$\partial_j K_x y(z) := \frac{\partial}{\partial s^j} K(z, x) y \quad \forall z \in \mathcal{X}.$$

The following result provides a useful reproducing property for the derivatives of functions in $\mathcal{H}_K$

that will be instrumental in deriving the solution algorithm.

Theorem 8 (Derivative Properties on $\mathcal{H}_K$ (Micheli and Glaunés, 2014))**.** *Let $\mathcal{H}_K$ be a RKHS in $\mathbb{R}^n(\mathcal{X})$ with reproducing kernel $K \in \mathcal{C}^2(\mathcal{X} \times \mathcal{X})$. Then, for all $(x, y) \in \mathcal{X} \times \mathbb{R}^n$:*

(a) $\partial_j K_x y \in \mathcal{H}_K$ for all $j = 1, \dots, n$.

(b) The following derivative reproducing property holds for all $j = 1, \dots, n$:

$$\left\langle \frac{\partial f(x)}{\partial x^j}, y \right\rangle = \langle f, \partial_j K_x y \rangle_{\mathcal{H}_K} \quad \forall f \in \mathcal{H}_K.$$

As mentioned in Section 8.2, the bi-linearity of constraint (8.5) forces us to adopt an alternating solution strategy whereby in the dynamics sub-problem, W is held fixed and we minimize J_d with respect to $\{\hat{f}, \hat{B}\}$. In the metric sub-problem, $\{\hat{f}, \hat{B}\}$ are held fixed and we minimize J_m with respect to $\{W, \underline{w}, \overline{w}\}$.

In the following we derive several useful representer theorems to characterize the solution of the two sub-problems, under the two simplifying assumptions introduced in Section 8.3.2.

8.4.2 Dynamics Sub-Problem

Let K^f be the reproducing $\mathcal{C}^2$ kernel for an $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K^f$ and let K^B be another reproducing kernel for an $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K^B$. Define the finite-dimensional subspaces:

$$\mathcal{V}_f := \left\{ \sum_{i=1}^{N_c} K^f_{x_i} a_i + \sum_{i=1}^{N_c} \sum_{p=1}^{n} \partial_p K^f_{x_i} a'_{ip}, \quad a_i, a'_{ip} \in \mathbb{R}^n \right\} \subset \mathcal{H}_K^f \ , \tag{8.16}$$

$$\mathcal{V}_B := \left\{ \sum_{i=1}^{N_c} K^B_{x_i} c_i + \sum_{i=1}^{N_c} \sum_{p=1}^{n} \partial_p K^B_{x_i} c'_{ip}, \quad c_i, c'_{ip} \in \mathbb{R}^n \right\} \subset \mathcal{H}_K^B \ . \tag{8.17}$$

Note that all x_i taken from the training dataset of $(x_i, u_i, \dot{x}_i)$ tuples are a subset of X_c.

Theorem 9 (Representer Theorem for f, B)**.** *Suppose the reproducing kernel K^B is chosen such that all functions $g \in \mathcal{H}_K^B$ satisfy the sparsity structure $g^j(x) = 0$ for $j = 1, \dots, n - m$. Consider then the pointwise-relaxed dynamics sub-problem for problem* (8.6)*:*

$$\begin{aligned} \min_{\hat{f} \in \mathcal{H}_K^f,\ \hat{b}_j \in \mathcal{H}_K^B, j=1,\dots,m} \quad & J_d(\hat{f}, \hat{B}) \\ \text{s.t.} \quad & \mathcal{F}_\lambda(x_i; \hat{f}, W) \preceq 0 \quad \forall x_i \in X_c \ . \end{aligned} \tag{8.18}$$

Suppose the feasible set for the LMI constraint is non-empty. Denote f^* and $b_j^*, j = 1, \dots, m$ as the optimizers for this sub-problem. Then, $f^* \in \mathcal{V}_f$ and $\{b_j^*\}_{j=1}^M \in \mathcal{V}_B$.

Proof. For a fixed W, the constraint is convex in $\hat{f}$ by linearity of the matrix $\mathcal{F}_\lambda$ in $\hat{f}$ and $\partial\hat{f}/\partial x$. By assumption (2), and the simplifying form for $B_\perp$, the matrix $\mathcal{F}_\lambda$ is additionally independent of B. Then, by strict convexity of the objective functional, there exists unique minimizers $f^*, \{b_j\}^* \in \mathcal{H}_K$, provided the feasible region is non-empty (Kurdila and Zabarankin, 2006).

Since $\mathcal{V}_f$ and $\mathcal{V}_B$ are closed, by the Projection Theorem, $\mathcal{H}_K^f = \mathcal{V}_f \oplus \mathcal{V}_f^\perp$ and $\mathcal{H}_K^B = \mathcal{V}_B \oplus \mathcal{V}_B^\perp$. Thus, any $g \in \mathcal{H}_K^f$ may be written as $g^{\mathcal{V}_f} + g^\perp$ where $g^{\mathcal{V}_f} \in \mathcal{V}_f$, $g^\perp \in \mathcal{V}_f^\perp$, and $\left\langle g^{\mathcal{V}_f}, g^\perp \right\rangle_{\mathcal{H}_K^f} = 0$. Similarly, any $g \in \mathcal{H}_K^B$ may be written as $g^{\mathcal{V}_B} + g^\perp$ where $g^{\mathcal{V}_B} \in \mathcal{V}_B$, $g^\perp \in \mathcal{V}_B^\perp$, and $\left\langle g^{\mathcal{V}_B}, g^\perp \right\rangle_{\mathcal{H}_K^B} = 0$.

Let $f = f^{\mathcal{V}_f} + f^\perp$ and $b_j = b_j^{\mathcal{V}_B} + b_j^\perp$, and define

$$H^{\mathcal{V}}(x,u) = f^{\mathcal{V}_f}(x) + \sum_{j=1}^m u^j b_j^{\mathcal{V}_B}(x)$$
$$H^{\perp}(x,u) = f^{\perp}(x) + \sum_{j=1}^m u^j b_j^{\perp}(x).$$

We can re-write $J_d(f,B)$ as:

$$\begin{aligned} J_d(f,B) = \sum_{i=1}^N &\left\langle H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i, H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right\rangle + 2\left\langle H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i, H^{\perp}(x_i,u_i) \right\rangle \\ &+ \left\langle H^{\perp}(x_i,u_i), H^{\perp}(x_i,u_i) \right\rangle + \mu_f \left(\|f^{\mathcal{V}_f}\|^2_{\mathcal{H}_K^f} + \|f^{\perp}\|^2_{\mathcal{H}_K^f} \right) \\ &+ \mu_b \left(\sum_{j=1}^m \|b_j^{\mathcal{V}_B}\|^2_{\mathcal{H}_K^B} + \|b_j^{\perp}\|^2_{\mathcal{H}_K^B} \right). \end{aligned}$$

Leveraging the reproducing property in (8.15),

$$\begin{aligned} &\left\langle H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i, H^{\perp}(x_i,u_i) \right\rangle \\ &\quad = \left\langle f^{\perp}(x_i) + \sum_{j=1}^m u_i^j b_j^{\perp}(x_i), H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right\rangle \\ &\quad = \underbrace{\left\langle f^{\perp}, K_{x_i}^f \left(H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right) \right\rangle_{\mathcal{H}_K^f}}_{=0} + \underbrace{\left\langle \sum_{j=1}^m u_i^j b_j^{\perp}, K_{x_i}^B \left(H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right) \right\rangle_{\mathcal{H}_K^B}}_{=0} \end{aligned}$$

since $K_{x_i}^f \left(H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right) \in \mathcal{V}_f$, $\mathcal{V}_B^\perp$ is closed under addition, and $K_{x_i}^B \left(H^{\mathcal{V}}(x_i,u_i) - \dot{x}_i \right) \in \mathcal{V}_B$.

Thus, $J_d(f, B)$ simplifies to

$$\sum_{i=1}^{N} \left\| H^{\mathcal{V}}(x_i, u_i) - \dot{x}_i \right\|^2 + \mu_f \|f^{\mathcal{V}_f}\|^2_{\mathcal{H}_K^f} + \mu_b \sum_{j=1}^{n} \|b_j^{\mathcal{V}_B}\|^2_{\mathcal{H}_K^B} + \\ + \left[\sum_{i=1}^{N} \left\| H^{\perp}(x_i, u_i) \right\|^2 + \mu_f \|f^{\perp}\|^2_{\mathcal{H}_K^f} + \mu_b \sum_{j=1}^{m} \|b_j^{\perp}\|^2_{\mathcal{H}_K^B} \right]. \tag{8.19}$$

Now, the (p, q) element of $\partial_f W(x_i)$ takes the form

$$\left\langle \frac{\partial w_{pq}(x_i)}{\partial x}, f(x_i) \right\rangle = \left\langle f, K_{x_i}^f \frac{\partial w_{pq}(x_i)}{\partial x} \right\rangle_{\mathcal{H}_K^f} = \left\langle f^{\mathcal{V}_f}, K_{x_i}^f \frac{\partial w_{pq}(x_i)}{\partial x} \right\rangle_{\mathcal{H}_K^f}.$$

Column p of $\frac{\partial f(x_i)}{\partial x} W(x_i)$ takes the form

$$\sum_{j=1}^{n} w_{jp}(x_i) \frac{\partial f(x_i)}{\partial x^j} = \begin{bmatrix} \sum_{j=1}^{n} w_{jp}(x_i) \left\langle \frac{\partial f(x_i)}{\partial x^j}, e_1 \right\rangle \\ \vdots \\ \sum_{j=1}^{n} w_{jp}(x_i) \left\langle \frac{\partial f(x_i)}{\partial x^j}, e_n \right\rangle \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} w_{jp}(x_i) \left\langle f, \partial_j K_{x_i}^f e_1 \right\rangle_{\mathcal{H}_K^f} \\ \vdots \\ \sum_{j=1}^{n} w_{jp}(x_i) \left\langle f, \partial_j K_{x_i}^f e_n \right\rangle_{\mathcal{H}_K^f} \end{bmatrix} \\ = \begin{bmatrix} \sum_{j=1}^{n} w_{jp}(x_i) \left\langle f^{\mathcal{V}_f}, \partial_j K_{x_i}^f e_1 \right\rangle_{\mathcal{H}_K^f} \\ \vdots \\ \sum_{j=1}^{n} w_{jp}(x_i) \left\langle f^{\mathcal{V}_f}, \partial_j K_{x_i}^f e_n \right\rangle_{\mathcal{H}_K^f} \end{bmatrix},$$

where e_i is the i^{th} standard basis vector in $\mathbb{R}^n$. Thus, $f^{\perp}$ plays no role in pointwise relaxation of constraint (8.5) and thus does not affect problem feasibility. Given the assumed structure of functions in $\mathcal{H}_K^B$ as provided in the theorem statement, the matrix in (8.8) is a valid annihilator matrix for $\hat{B}(x)$. Consequently, $b^{\perp}$ also has no effect on problem feasibility. Thus, by non-negativity of the term in the square brackets in (8.19), we have that the optimal f lies in $\mathcal{V}_f$ and all optimal $\{b_j\}_{j=1}^m$ lie within $\mathcal{V}_B$. □

The key consequence of this theorem is the reduction of the infinite-dimensional search problem for the functions f and $b_j, j = 1, \ldots, m$ to a finite-dimensional *convex* optimization problem for the constant vectors $a_i, a'_{ip}, \{c_i^{(j)}, c_{ip}^{(j)'}\}_{j=1}^m \in \mathbb{R}^n$, by choosing the function classes $\mathcal{H}^f = \mathcal{H}_K^f$ and $\mathcal{H}^B = \mathcal{H}_K^B$. Next, we characterize the optimal solution to the metric sub-problem.

8.4.3 Metric Sub-Problem

By the simplifying assumption in Section 8.3.1, constraint (8.4) requires that $W_{\perp}$ be only a function of the first $(n - m)$ components of x. Thus, define $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as a *scalar* reproducing kernel with associated real-valued scalar RKHS $\mathcal{H}_\kappa$. Additionally, define $\hat{\kappa} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as another scalar

reproducing kernel with associated real-valued scalar RKHS $\mathcal{H}_{\hat{\kappa}}$. In particular, $\hat{\kappa}$ is only a function of the first $(n-m)$ components of $x \in \mathcal{X}$ in both arguments. Let κ_x and $\hat{\kappa}_x$ denote $\kappa(x,\cdot)$ and $\hat{\kappa}(x,\cdot)$ respectively.

Define the kernel derivative functions:

$$\partial_j \kappa_x(z) := \frac{\partial \kappa}{\partial r^j} \kappa(r,s)\Big|_{(r=x,s=z)} \qquad \partial_j \hat{\kappa}_x(z) := \frac{\partial \hat{\kappa}}{\partial r^j} \kappa(r,s)\Big|_{(r=x,s=z)} \qquad \forall z \in \mathcal{X}.$$

From (Zhou, 2008), it follows that the kernel derivative functions satisfy the following two properties, similar to Theorem 8:

$$\begin{aligned} &\partial_j \kappa_x \in \mathcal{H}_\kappa \quad \forall j = 1,\dots,n,\ x \in \mathcal{X} \\ &\frac{\partial h}{\partial x^j}(x) = \langle h, \partial_j \kappa_x \rangle_{\mathcal{H}_\kappa} \quad \forall h \in \mathcal{H}_\kappa. \end{aligned}$$

A similar property holds for $\partial_j \hat{\kappa}_x$ and $\mathcal{H}_{\hat{\kappa}}$. Consider then the following finite-dimensional spaces:

$$\mathcal{V}_\kappa := \left\{ \sum_{i=1}^{N_c} a_i \kappa_{x_i} + \sum_{i=1}^{N_c} \sum_{p=1}^{n} a'_{ip}\, \partial_p \kappa_{x_i}, \quad a_i, a'_{ip} \in \mathbb{R} \right\} \subset \mathcal{H}_\kappa \tag{8.20}$$

$$\mathcal{V}_{\hat{\kappa}} := \left\{ \sum_{i=1}^{N_c} c_i \hat{\kappa}_{x_i} + \sum_{i=1}^{N_c} \sum_{p=1}^{n-m} c'_{ip}\, \partial_p \hat{\kappa}_{x_i}, \quad c_i, c'_{ip} \in \mathbb{R} \right\} \subset \mathcal{H}_{\hat{\kappa}} \tag{8.21}$$

and the proposed representation for $W(x)$:

$$\begin{aligned} W(x) = &\sum_{i=1}^{N_c} \hat{\Theta}_i \hat{\kappa}_{x_i}(x) + \sum_{i=1}^{N_c} \sum_{j=1}^{n-m} \hat{\Theta}'_{ij} \partial_j \hat{\kappa}_{x_i}(x) \\ &+ \sum_{i=1}^{N_c} \Theta_i \kappa_{x_i}(x) + \sum_{i=1}^{N_c} \sum_{j=1}^{n} \Theta'_{ij} \partial_j \kappa_{x_i}(x), \end{aligned} \tag{8.22}$$

where $\hat{\Theta}, \hat{\Theta}' \in \mathbb{S}_n$ are constant symmetric matrices with non-zero entries only in the top-left $(n-m)\times(n-m)$ block, and $\Theta, \Theta' \in \mathbb{S}_n$ are constant symmetric matrices with zero entries in the top-left $(n-m)\times(n-m)$ block.

Theorem 10 (Representer Theorem for W)**.** *Consider the pointwise-relaxed metric sub-problem for problem* (8.6)*:*

$$\begin{aligned} \min_{\substack{w_{pq} \in \mathcal{H}_{\hat{\kappa}}, (p,q) \in \{1,\dots,(n-m)\} \\ w_{pq} \in \mathcal{H}_\kappa \ else \\ \underline{w}, \overline{w} \in \mathbb{R}_{>0}}} \quad & J_m(W, \underline{w}, \overline{w}) \\ \text{s.t.} \quad & \mathcal{F}_\lambda(x_i; \hat{f}, W) \preceq 0, \quad \forall x_i \in X_c, && (8.23) \\ & \underline{w} I_n \preceq W(x_i) \preceq \overline{w} I_n, \quad \forall x_i \in X_c, && (8.24) \end{aligned}$$

Suppose the feasible set of the above LMI constraints is non-empty. Denote W^* as the optimizer for this sub-problem. Then, W^* takes the form given in (8.22).

Proof. Notice that while the regularizer term is strictly convex, the surrogate loss function for the condition number is affine. However, provided the feasible set is non-empty, there still exists a minimizer (possibly non-unique) for the above sub-problem.

Since $\mathcal{V}_\kappa$ and $\mathcal{V}_{\hat{\kappa}}$ are closed, by the Projection Theorem, $\mathcal{H}_\kappa = \mathcal{V}_\kappa \oplus \mathcal{V}_\kappa^\perp$ and $\mathcal{H}_{\hat{\kappa}} = \mathcal{V}_{\hat{\kappa}} \oplus \mathcal{V}_{\hat{\kappa}}^\perp$. Thus, any $h \in \mathcal{H}_\kappa$ may be written as $h^{\mathcal{V}_\kappa} + h^\perp$ where $h^{\mathcal{V}_\kappa} \in \mathcal{V}_\kappa$, $h^\perp \in \mathcal{V}_\kappa^\perp$ and $\left\langle h^{\mathcal{V}_\kappa}, h^\perp \right\rangle_{\mathcal{H}_\kappa} = 0$. A similar decomposition property holds for $\mathcal{H}_{\hat{\kappa}}$.

Now, consider the following chain of equalities for the $(p,q)^{\text{th}}$ element of $\partial_f W_\perp(x_i)$:

$$\begin{aligned}
\partial_f w_{\perp_{pq}}(x_i) &= \sum_{j=1}^{n-m} \frac{\partial w_{\perp_{pq}}(x_i)}{\partial x^j} f^j(x_i) \\
&= \sum_{j=1}^{n-m} \left\langle w_{\perp_{pq}}, \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} f^j(x_i) \\
&= \left\langle w_{\perp_{pq}}, \sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} \\
&= \left\langle w_{\perp_{pq}}^{\mathcal{V}_{\hat{\kappa}}}, \sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}},
\end{aligned}$$

since $\sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \in \mathcal{V}_{\hat{\kappa}}$. Trivially, the (p,q) element of $W(x_i)$ takes the form

$$w_{pq}(x_i) = \begin{cases} \left\langle w_{pq}^{\mathcal{V}_{\hat{\kappa}}}, \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} & \text{if } (p,q) \in \{1, \ldots, (n-m)\} \\ \left\langle w_{pq}^{\mathcal{V}_\kappa}, \kappa_{x_i} \right\rangle_{\mathcal{H}_\kappa} & \text{else.} \end{cases}$$

Thus, constraints (8.5) and uniform definiteness at the constraint points in X_c can be written in terms of functions in $\mathcal{V}_\kappa$ and $\mathcal{V}_{\hat{\kappa}}$ alone. By strict convexity of the regularizer, and recognizing that $W(x)$ is symmetric, the result follows. □

Similar to the previous section, the key property here is the reduction of the infinite-dimensional search over $W(x)$ to the finite-dimensional convex optimization problem over the constant symmetric matrices $\Theta_i, \Theta'_{ij}, \hat{\Theta}_i, \hat{\Theta}'_{ij}$ by choosing the function class for the entries of W using the scalar-valued RKHS.

At this point, both sub-problems are finite-dimensional convex optimization problems. Crucially, the only simplifications made are those given in Section 8.3.2. However, a final computational challenge here is that the number of parameters scales with the number of training (N) and constraint (N_c) points. This is a fundamental challenge in all non-parametric methods. In the next section we

present the dimensionality reduction techniques used to alleviate these issues.

8.4.4 Approximation via Random Matrix Features

The size of the problem using full matrix-valued kernel expansions grows rapidly in $N_c \cdot n$, the number of constraint points times the state dimensionality. This makes training slow for even moderately long demonstrations in low-dimensional settings. The induced dynamical system is slow to evaluate and integrate at inference time. Random feature approximations to kernel functions have been extensively used to scale up training complexity and inference speed of kernel methods (Rahimi and Recht, 2007; Avron et al., 2016) in a number of applications (Huang et al., 2014). The quality of approximation can be explicitly controlled by the number of random features. In particular, it has been shown (Rahimi and Recht, 2008) that any function in the RKHS associated with the exact kernel can be approximated to arbitrary accuracy by a linear combination of a sufficiently large number of random features.

These approximations have only recently been extended to matrix-valued kernels (Minh, 2016; Brault et al., 2016). Given a matrix-valued kernel K, one defines an appropriate matrix-valued feature map $\Phi : \mathcal{X} \to \mathbb{R}^{d \times n}$ with the property

$$K(x, z) \approx \Phi(x)^T \Phi(z),$$

where d controls the quality of this approximation. A canonical example is the Gaussian separable kernel $K_\sigma(x, z) := e^{\frac{-\|x-z\|_2^2}{\sigma^2}} I_n$ with feature map:

$$\Phi(x) = \frac{1}{\sqrt{s}} \begin{bmatrix} \cos(\omega_1^T x) \\ \sin(\omega_1^T x) \\ \vdots \\ \cos(\omega_s^T x) \\ \sin(\omega_s^T x) \end{bmatrix} \otimes I_n,$$

where $\omega_1, \ldots, \omega_s$ are i.i.d. draws from $\mathcal{N}(0, \sigma^{-2} I_n)$ and $\otimes$ denotes the Kronecker product.

By construction, the linear span $\{K_z y \mid z \in \mathcal{X}, y \in \mathbb{R}^n\}$ is dense in $\mathcal{H}_K$. Thus, any function g in the associated RKHS $\mathcal{H}_K$ can be arbitrarily well-approximated by the expansion $\sum_j^{N'} K_{z_j} y_j$, which, in turn, may be approximated as

$$g(x) \approx \sum_{j=1}^{N'} K(x, z_j) y_j \approx \sum_{j=1}^{N'} \Phi(x)^T \Phi(z_j) y_j = \Phi(x)^T \alpha,$$

where $\alpha = \sum_{j=1}^{N'} \Phi(z_j) y_j \in \mathbb{R}^d$. Applying this approximation for the spaces[2] $\mathcal{H}_K^f$, $\mathcal{H}_K^B$, $\mathcal{H}_\kappa$ and $\mathcal{H}_{\hat{\kappa}}$,

[2]To apply this decomposition to $W(x)$, we simply leverage vector-valued feature maps for each entry of $W(x)$.

we obtain finite-dimensional representations of the optimal $f, \{b_j\}, W$ that scale with the order of the matrix feature map, instead of $N_c \cdot n$. Indeed, these are the representations defined in eqs. (8.10)–(8.12). To complete the derivation of the cost terms $\hat{J}_d$ and $\hat{J}_m$ in problem (8.13), we address the functional regularization terms as follows. From the definition of the inner product in $\mathcal{H}_K$:

$$\|g\|^2_{\mathcal{H}_K} = \sum_{i,j=1}^{N'} \langle y_i, K(z_i, z_j) y_j \rangle \approx \sum_{i,j=1}^{N'} \langle \Phi(z_i) y_i, \Phi(z_j) y_j \rangle = \left\langle \sum_{i=1}^{N'} \Phi(z_i) y_i, \sum_{j=1}^{N'} \Phi(z_j) y_j \right\rangle = \|\alpha\|^2 \ .$$

8.5 Summary

Prior to proceeding to the solution algorithm in the next chapter, we summarize the content since Section 8.2 until now. First, starting from the infinite-dimensional *non-convex* constrained optimization over function spaces, i.e., problem (8.6), we split the problem into alternating between two infinite-dimensional *convex* sub-problems. Second, we derived the representation of the optimal solution of each sub-problem, i.e., Theorems 9 and 10, by leveraging the mild structural assumption on the input matrix, relaxation of the infinite-dimensional constraints to sampling-based constraints, and restriction of the function spaces to RKHS. Third, we used the universal approximation property of random matrix feature maps to reduce the dimensionality of the representation of the optimal solutions. Fourth, and finally, problem (8.13) gives the final finite-dimensional re-formulation of problem (8.6) using this feature representation and the sample-based constraints.

Chapter 9

Solving Stabilizable Dynamics Learning

The fundamental structure of the solution algorithm consists of alternating between the dynamics and metric sub-problems derived from problem (8.13). However, there are three fundamental challenges in solving these problems. First, enforcing the constraints exactly requires a feasible initialization (either with the dynamics or metric function parameters). Given our use of random matrix feature approximations, this is simply intractable. Second, even with an initial feasible guess, due to the alternation, one may lose feasibility within either of the sub-problems stated in Theorems 9 and 10. This is a fundamental numerical challenge of bilinear optimization. Third, since the constraint set X_c includes at least the state samples from the demonstration tuples (plus additional random samples from $\mathcal{X}$), enforcing LMIs over all constraint points at each iteration is again computationally intractable. To address these challenges, we present a solution algorithm that (i) eliminates the need for a feasible initial guess, thereby permitting cold-starts, (ii) leverages sub-sampling and *dynamic* updating of the constraint set for each iteration sub-problem to maintain tractability, and (iii) performs feasibility corrections using efficient, unconstrained second-order optimization. Additionally, we present an extensive empirical study of the numerical performance of the algorithm on the motivating example introduced in the previous chapter, and finally, evaluate the algorithm on a quadrotor testbed.

9.1 Solution Algorithm

We first give the full formulation of the relevant optimization sub-problems and then provide a pseudocode summary of the algorithm at the end. Let $X_c^{(k)}$ denote the finite sample constraint set at iteration k. In particular, $X_c^{(k)} \subset X_c$ with $N_c^{(k)} := |X_c^{(k)}|$ being ideally much less than N_c, the

cardinality of the full constraint set X_c. Formally, each major iteration k is characterized by four minor steps (sub-problems):

1. **Finite-dimensional dynamics sub-problem**:

$$\begin{aligned}
\min_{\substack{\alpha,\beta_j,j=1,\dots,m,\\ s\geq 0}} \quad & \sum_{i=1}^{N} \|\hat{f}(x_i)+\hat{B}(x_i)u_i-\dot{x}_i\|^2+\mu_f\|\alpha-\alpha^{(k-1)}\|^2 \\
& +\mu_b\sum_{j=1}^{m}\|\beta-\beta_j^{(k-1)}\|^2+\mu_s\|s\|_1 && \text{(9.1a)}\\
\text{s.t.} \quad & \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i;\alpha,\tilde{\theta}_{ij}^{(k-1)}) \preceq s(x_i)I_{n-m} \quad \forall x_i\in X_c^{(k)} && \text{(9.1b)}\\
& s(x_i)\leq \bar{s}^{(k-1)} \quad \forall x_i\in X_c^{(k)}, && \text{(9.1c)}
\end{aligned}$$

   where $\{\alpha^{(k-1)},\beta_j^{(k-1)},\tilde{\theta}_{ij}^{(k-1)}\}$ are the dynamics and metric parameters obtained from the previous major iteration[1], and $\mu_s\in(0,1)$ is an additional regularization parameter for s, a non-negative slack vector in $\mathbb{R}^{N_c^{(k)}}$. The quantity $\bar{s}^{(k-1)}$ is defined as

$$\begin{aligned}
\bar{s}^{(k-1)} &:= \max_{x_i\in X_c}\bar{\lambda}\left(\mathcal{F}_{\lambda+\epsilon_\lambda}^{(k-1)}(x_i)\right), \quad \text{where}\\
\mathcal{F}_{\lambda+\epsilon_\lambda}^{(k-1)}(x_i) &:= \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i;\alpha^{(k-1)},\tilde{\theta}_{ij}^{(k-1)}).
\end{aligned}$$

   That is, $\bar{s}^{(k-1)}$ captures the worst violation for the stability LMI over the entire constraint set X_c, given the parameters at the end of iteration $k-1$. Denote $\alpha^{(k)},\{\beta_j^{(k)}\}_{j=1}^m$ to be the optimizers of this sub-problem.

2. **Compute an upper bound $\bar{s}^{(k)'}$ on the maximum allowed violation of the stability LMI for the current constraint set $X_c^{(k)}$**:

$$\begin{aligned}
&\texttt{while} \quad \min_{x_i\in X_c^{(k)}} \ \underline{\lambda}\left(W(x_i)\right) < \delta_{\underline{w}}+\epsilon_{\underline{w}} && \text{(9.2a)}\\
&\quad \texttt{do} \quad \min_{\tilde{\theta}_{ij}} \ \sum_{x_i\in X_c^{(k)}}\Big[\psi_w\left(\bar{\lambda}\left((\delta_{\underline{w}}+\epsilon_{\underline{w}})I_n-W(x_i)\right)\right)+\\
&\qquad\qquad +\mu_w'\psi_{\mathcal{F}}\left(\bar{\lambda}\left(\mathcal{F}_{\lambda+\epsilon_\lambda}(x_i;\alpha^{(k)},\tilde{\theta}_{ij})\right)\right)\Big]+\\
&\qquad +\mu_w'\sum_{i,j}\|\tilde{\theta}_{ij}-\tilde{\theta}_{ij}^{(k-1)}\|^2, && \text{(9.2b)}\\
&\mu_w'\leftarrow 0.5\mu_w' && \text{(9.2c)}
\end{aligned}$$

[1] We set $\alpha^{(0)},\{\beta_j^{(0)}\}_{j=1}^m,\tilde{\theta}_{ij}^{(0)}=0$.

where $\mu'_w > 0$ is a regularization parameter that is exponentially decayed until the `while` loop termination condition is met. The functions $\psi_w(\cdot)$ and $\psi_{\mathcal{F}}(\cdot)$ are penalty functions. Let $\tilde{\theta}'_{ij}$ be the metric parameters when the termination condition is met. We then set

$$\bar{s}^{(k)'} := \max_{x_i \in X_c^{(k)}} \bar{\lambda}\left(\mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha^{(k)}, \tilde{\theta}'_{ij})\right). \tag{9.3}$$

The primary objective of this step is to compute an upper bound on the stability LMI over the constraint set $X_c^{(k)}$ with respect to a dual metric that satisfies the positive definiteness condition at all points in this set. This upper-bound will then be used as a constraint within the metric sub-problem, described next. The reason one cannot use $\bar{s}^{(k-1)}$ as in the dynamics sub-problem is because this value is computed over the entire constraint set at the end of the previous iteration, with a metric that is potentially *not* positive definite at all points. Since the metric sub-problem strictly enforces the positive definiteness constraint on W, $\bar{s}^{(k-1)}$ is not a valid upper bound on the stability LMI constraint, and can lead to infeasibility for the metric sub-problem. As long as there exists a set of $\tilde{\theta}_{ij}$ such that the dual metric satisfies the positive definiteness constraint at all points, the `while` loop above will always terminate given a sufficiently small μ'_w. In our implementation, we initialize μ'_w to be equal to μ_w. In Section 9.1.1 we provide an efficient second-order method for solving this *unconstrained* optimization.

3. **Finite-dimensional metric sub-problem**:

$$\min_{\tilde{\theta}_{ij}, \underline{w}, \overline{w}, s \geq 0} \quad (\overline{w} - \underline{w}) + \mu_w \sum_{i,j} \|\tilde{\theta}_{ij} - \tilde{\theta}_{ij}^{(k-1)}\|^2 + (1/\mu_s)\|s\|_1 \tag{9.4a}$$

$$\text{s.t.} \quad \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha^{(k)}, \tilde{\theta}_{ij}) \preceq s(x_i) I_{n-m} \quad \forall x_i \in X_c^{(k)} \tag{9.4b}$$

$$s(x_i) \leq \bar{s}^{(k)'} \quad \forall x_i \in X_c^{(k)} \tag{9.4c}$$

$$(\underline{w} + \epsilon_{\underline{w}}) I_n \preceq W(x_i) \preceq \overline{w} I_n \quad \forall x_i \in X_c^{(k)}, \tag{9.4d}$$

$$\underline{w} \geq \delta_{\underline{w}}. \tag{9.4e}$$

4. **Update $X_c^{(k)}$ sub-problem**.

Choose a tolerance parameter $\delta > 0$. Then, define

$$\nu^{(k)}(x_i) := \max\left\{\bar{\lambda}\left(\mathcal{F}^k_{\lambda+\epsilon_\lambda}(x_i)\right), \bar{\lambda}\left((\delta_{\underline{w}} + \epsilon_\delta) I_n - W(x_i)\right)\right\} \quad \forall x_i \in X_c, \tag{9.5}$$

and set

$$X_c^{(k+1)} := \left\{x_i \in X_c^{(k)} : \nu^{(k)}(x_i) > -\delta\right\} \bigcup \left\{x_i \in X_c \setminus X_c^{(k)} : \nu^{(k)}(x_i) > 0\right\}. \tag{9.6}$$

Thus, in the update $X_c^{(k)}$ step, we balance addressing points where constraints are being violated ($\nu^{(k)} > 0$) and discarding points where constraints are satisfied with sufficient strict inequality ($\nu^{(k)} \leq -\delta$). This prevents overfitting to any specific subset of the constraint points. A potential variation to the union above is to only add up to L constraint violating points from $X_c \setminus X_c^{(k)}$ (e.g., corresponding to the L worst violators), where L is a fixed positive integer. Indeed this is the variation used in our experiments and was found to be extremely efficient in balancing the size of the set $X_c^{(k)}$ and thus, the complexity of each iteration. This adaptive sampling technique is inspired by *exchange algorithms* for semi-infinite optimization, as the one proposed in (Zhang et al., 2010) where one is trying to enforce the constraints at *all* points in a compact set $\mathcal{X}$.

The use of the modified regularization terms above (i.e., penalizing change in the parameters from the previous iteration as opposed to absolute penalty) is, in the spirit of trust region methods, to prevent large updates to the parameters due to the dynamically updating constraint set $X_c^{(k)}$. At the first major iteration of the problem however, with $\alpha^{(0)}, \{\beta_j^{(0)}\}_{j=1}^m, \tilde{\theta}_{ij} = 0$, this is the same as an absolute regularization term. To ensure non-degeneracy in the stability LMI constraint within the first dynamics sub-problem iteration, we initialize the algorithm with $W(x_i) = I_n$. The full pseudocode for the CCM-Regularized (CCM-R) dynamics learning algorithm is summarized in Algorithm 5.

Algorithm 5 CCM - Regularized (CCM-R) Dynamics Learning

Input: Dataset $\{x_i, u_i, \dot{x}_i\}_{i=1}^N$, constraint set X_c, regularization constants $\{\mu_f, \mu_b, \mu_w, \mu_s\}$, constraint tolerances $\{\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}\}$, discard tolerance parameter δ, Initial # of constraint points: $N_c^{(0)}$, Max # iterations: $N_{\max}$, termination tolerance ε.
$k \leftarrow 1$, `converged` $\leftarrow$ **false**, $W(x) \leftarrow I_n$, $\{\alpha^{(0)}, \{\beta_j^{(0)}\}_{j=1}^m, \tilde{\theta}_{ij}\} \leftarrow 0$.
$X_c^{(0)} \leftarrow$ RandSample$(X_c, N_c^{(0)})$
while $\neg$`converged` $\wedge\, k < N_{\max}$ **do**
  $\{\alpha^{(k)}, \{\beta_j^{(k)}\}\} \leftarrow$ Solve (9.1)
  $\bar{s}^{(k)'} \leftarrow$ Solve (9.2)
  $\{\tilde{\theta}_{ij}^{(k)}, \underline{w}, \overline{w}\} \leftarrow$ Solve (9.4)
  $X_c^{(k+1)}, \bar{s}^{(k)}, \nu^{(k)} \leftarrow$ Update $X_c^{(k)}$ using (9.6)
  $\Delta \leftarrow \max\left\{\|\alpha^{(k)} - \alpha^{(k-1)}\|_\infty, \|\beta_j^{(k)} - \beta_j^{(k-1)}\|_\infty, \|\tilde{\theta}_{ij}^{(k)} - \tilde{\theta}_{ij}^{(k-1)}\|_\infty\right\}$
  if $\Delta < \varepsilon$ **or** $\nu^{(k)}(x_i) < \varepsilon \quad \forall x_i \in X_c$ **then**
    `converged` $\leftarrow$ **true**.
  end if
  $k \leftarrow k + 1$.
end while

Some comments are in order. First, convergence in Algorithm 5 is declared if either progress in the solution variables stalls or all constraints are satisfied within tolerance. Using dynamic subsampling for the constraint set at each iteration implies that (i) the matrix function $W(x)$ at iteration

k resulting from variables $\tilde{\theta}_{ij}^{(k)}$ does *not* have to correspond to a valid dual metric for the interim learned dynamics at iteration k, and (ii) a termination condition based on constraint satisfaction at all N_c points is justified by the fact that at each iteration, we are solving relaxed sub-problems that collectively generate a sequence of lower-bounds on the overall objective.

Second, as a consequence of this iterative procedure, the dual metric and contraction rate pair $\{W(x), \lambda\}$ do not possess any sort of "control-theoretic" optimality, as was the case in the first part of this thesis. Within the learning context, these quantities are used solely as *regularizers* to *promote* stabilizability of the learned model. Following the recovery of a fixed dynamics model from the algorithm, one may leverage global CCM optimization algorithms as the one presented in part I of this thesis to compute an optimal *robust* CCM.

9.1.1 Solving the Upper Bound Sub-Problem

Step 2 in the iterative algorithm outlined in the previous section entails solving a non-smooth, but convex, unconstrained optimization to compute a viable upper bound on the stability LMI for its inclusion as a constraint within the metric sub-problem. While the problem can be formulated as an SDP through the use of epigraph LMIs, this would result in yet another computationally intensive step. Instead, recognizing that the purpose of this part of the algorithm is *not* to update any problem variables, one can use faster unconstrained optimization techniques to approximately solve this problem.

Specifically, we employ Newton descent to solve this problem with backtracking line search. The issue however, lies in taking the second order derivatives of the maximum eigenvalue function of an affinely parameterized matrix (recall W is linear in $\tilde{\theta}_{ij}$ and and $\mathcal{F}_\lambda$ is linear in $\tilde{\theta}_{ij}$ for fixed α). For a given affinely parameterized symmetric matrix $G(\theta) \in \mathbb{R}^{d\times d}$, the gradient and Hessian of $\bar{\lambda}(G(\theta))$ with respect to θ has the components (Overton and Womersley, 1995):

$$\frac{\partial \bar{\lambda}(G)}{\partial \theta^i}(\theta) = \bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^i} \bar{v}_1,$$
$$\frac{\partial^2 \bar{\lambda}(G)}{\partial \theta^i \partial \theta^j}(\theta) = 2\sum_{k>1}^{d} \frac{\bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^i} \bar{v}_k \cdot \bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^j} \bar{v}_k}{\lambda_1 - \lambda_k},$$

where we assume the eigenvalue ordering $\lambda_1 \geq \cdots \geq \lambda_d$, and $\bar{v}_1, \ldots, \bar{v}_d$ are the associated normalized eigenvectors. Clearly, the Hessian is undefined when the multiplicity of the max eigenvalue is greater than one. While there exist several works within the literature on modified parameterizations of the matrix to allow the use of second order methods, these techniques require knowing the multiplicity of the max eigenvalue at the optimal parameters – which is impossible to guess in this case.

Instead, we borrow a stochastic smoothing approximation from (D'Aspremont and Karoui, 2014),

whereby one approximates the max eigenvalue of $G(\theta)$ as

$$\max_{i=1,\ldots,k} \bar{\lambda}\left(G(\theta) + \frac{\sigma}{d} z_i z_i^T\right),$$

where $\sigma > 0$ is a small noise parameter, and z_i are i.i.d. samples from $\mathcal{N}(0, I_d)$. In particular, $\bar{\lambda}\left(G(\theta) + \frac{\varepsilon}{n} z_i z_i^T\right)$ is unique with probability one, for any $z_i \sim \mathcal{N}(0, I_d)$ (see (D'Aspremont and Karoui, 2014), Prop. 3.3 and Lemma 3.4). While the algorithm in (D'Aspremont and Karoui, 2014) leverages a finite-sample expectation of the randomized function above and its gradient, for our implementation, we simply leverage the uniqueness property induced by the use of the random rank one perturbation, and compute the gradient and Hessian expressions using the eigenvalue decomposition of the matrix $G(\theta) + (\varepsilon/d) z z^T$, where z is a single Gaussian vector sample. Additionally, we employ early termination of the Newton iterations if the `while` loop termination condition is met by the current iterate.

9.1.2 Revisiting Planar Quadrotor

We now revisit the results presented in Section 8.1 and shed additional light on the performance of the CCM-R model.

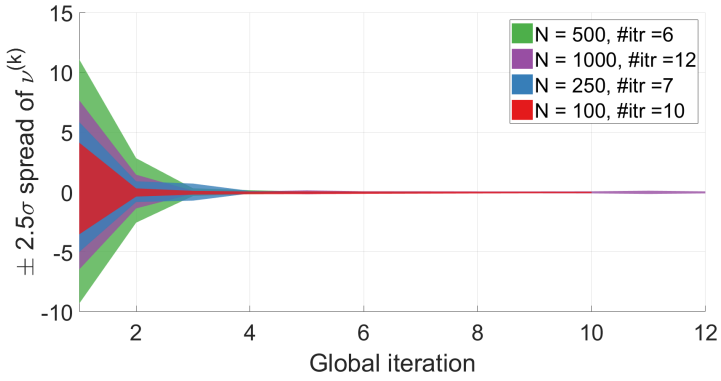
The training dataset was generated in three steps. First, a fixed set of waypoint paths in (p_x, p_z) was randomly generated. Second, for each waypoint path, multiple smooth polynomial splines were fitted using a minimum-snap algorithm. To create variation amongst the splines, the waypoints were perturbed within Gaussian balls and the time durations for the polynomial segments were also randomly perturbed. Third, the planar quadrotor was simulated with perturbed initial conditions and the polynomial trajectories as references, and tracked using a sub-optimally tuned PD controller; thereby emulating a noisy/imperfect demonstrator. These final simulated paths were sub-sampled at 0.1 s resolution to create the datasets. The variations created at each step of this process were sufficient to generate a rich exploration of the state-space for training.

The matrix feature maps used for all models (N-R, R-R, and CCM-R) are derived from the random matrix feature approximation of the Gaussian separable kernel; the relevant parameters are provided at the end of this chapter. We enforced the CCM regularizing constraints for the CCM-R model at $N_c = 2426$ points in the state-space, composed of the N demonstration points in the training dataset and randomly sampled points from $\mathcal{X}$ (recall that the CCM constraints do not require samples of $u, \dot{x}$). The contraction rate λ was set to 0.1 and the termination tolerance ε was set to 0.01. All other tolerance parameters are provided at the end of the chapter.

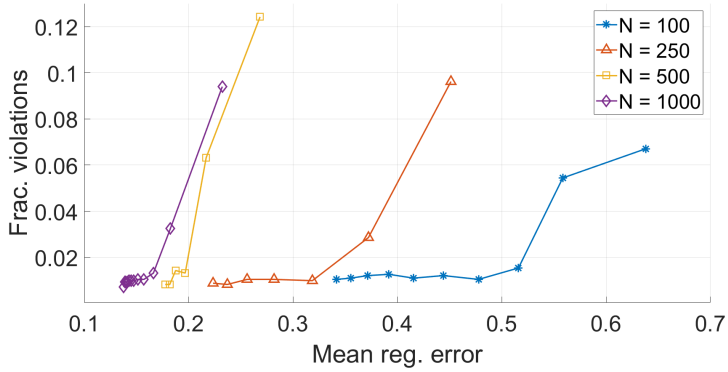
As the CCM constraints were relaxed to hold pointwise on the finite constraint set X_c as opposed to everywhere on $\mathcal{X}$, in the spirit of viewing these constraints as regularizers for the model, we simulated both the R-R and CCM-R models using the time-varying Linear-Quadratic-Regulator (TV-LQR) feedback controller. This also helped ensure a more direct comparison of the quality of

the learned models themselves, independently of the tracking feedback controller. The results are virtually identical using a tracking MPC controller and yield no additional insight.

Figure 9.1 plots the training curves generated by the CCM-R algorithm for varying demonstration dataset sizes N. In particular, we plot (i) the evolution of the distribution of the constraint violation vector ν defined in (9.5) over the training constraint set X_c, and (ii) evolution of the regression error and fraction of violations (i.e., average number of points with $\nu > 0$) over an independent validation set, as a function of global iteration k. The dynamic constraint set was initialized with 250 points (sampled randomly from the 2426 constraint points in the full training constraint set), and over all iterations, the size of $X_c^{(k)}$ remained below 400 points – a drastic factor of improvement over an intractable brute-force approach. For $N \in \{100, 250, 500\}$, the algorithm terminated with all constraints satisfied below the termination threshold of $\varepsilon = 0.01$. For $N = 1000$, we manually terminated the algorithm after 12 global iterations after observing a stall in the number of violations.



(a) $\pm 2.5\sigma$ spread of the constraint violation vector ν over *training set* as a function of global iteration. `#itr` denotes the total number of global iterations.



(b) Evolution of mean regression error norm and fraction of violations over *validation* set with global iteration number. The markers delineate the iterations. The curves proceed from right to left.

Figure 9.1: Simulation data training curves for all CCM-R models.

From Figure 9.4a, we see the effectiveness of the sub-sampling constraint set method in efficiently eliminating all training violations. Several comments are in order regarding Figure 9.1b. First, as the number of demonstration tuples N goes down, the curves shift further to the right, indicating higher validation error, as expected. Second, the iterates seem to clearly exhibit a two-phase convergent behavior. During the first phase, the fraction of violations on the validation set drops rapidly and approaches a steady-state. During the second phase, the fraction of violations remains roughly constant as the validation error drops monotonically, with decreasing drop rate. The training and final validation errors for all models are summarized in Table 9.1.

	N-R		**R-R**		**CCM-R**		
N	Train err.	Val err.	Train err.	Val err.	Train err.	Val err.	Frac. viol.
100	0.001	0.072	0.125	0.380	0.012	0.342	0.0008 (0.010)
250	0.003	0.088	0.125	0.243	0.010	0.223	0.0004 (0.009)
500	0.004	0.047	0.092	0.172	0.006	0.178	0.0016 (0.008)
1000	0.01	0.056	0.08	0.146	0.003	0.138	0.0021 (0.007)

Table 9.1: Comparison of average (over N tuples) training and validation (over 2000 tuples) regression error norms for all 3 models. Also shown are the fraction of violations ($\nu > 0$) on the training (validation) sets for the CCM-R models. The termination threshold for CCM-R training was $\nu < \varepsilon = 0.01$ for all points in the training constraint set.

An interesting trend to observe here is that the training errors for the N-R and CCM-R models are significantly closer (by an order of magnitude) together than R-R and N-R. Consistent with the trend in Figure 8.3, N-R features the smallest validation errors. Thus, while the CCM-R model manages to perform well in terms of regression error on the training set (almost on par with N-R), the resulting model does not suffer from the over-fitting trend observed with the N-R model.

Surprisingly, the final validation errors of the CCM-R models and the R-R models are almost identical. *Despite this however, the CCM-R model significantly outperforms the R-R model, especially when learned from smaller demonstration datasets.* What sets the CCM-R model apart is the manner in which the dynamics sub-problem improves the regression error over multiple iterations – through the joint minimization of regression error and stability LMI violations. Effectively, as seen in Figure 9.1b, the iterates essentially move along a level set of the constraint violations, decreasing regression error while maintaining control over the *stabilizability of the learned model.* This validates the benefit of using a control-theoretic regularization technique that is *tailored* to the motion planning task, and the fragility of *only* using traditional measures of performance (such as regression error) for evaluating learned dynamical models[2].

While the analytical planar quadrotor model studied in simulation is indeed an example of a system that is CCM stabilizable, in the next section we deploy the algorithm on a full 3D quadrotor

[2]Code for data generation, training, and evaluation is provided at `https://github.com/StanfordASL/SNDL`.

testbed, shown in Figure 5.6, with partially closed control loops to approximately emulate the planar dynamics.

9.2 Validation on Quadrotor Testbed

The objective of the flight experiments was to verify whether the performance trends observed in simulation for a relatively simple dynamics model, that was a priori known to be CCM stabilizable, carried over to a far more complex setting where this is unknown. The dynamics to be learned are for the motion of the quadrotor *within the vertical plane*. Accordingly, the training data consisted of samples from trajectories flown within the plane autonomously using a nonlinear trajectory tracking controller. For evaluation, the generated planar trajectories will be tracked using a combination of an *in-plane* controller, derived from the learned dynamics model, and an *out-of-plane* controller derived from known principles (detailed in Section 9.2.2). The purpose of the out-of-plane controller is simply to keep the quad aligned and fixed within the vertical plane, thereby permitting an evaluation of the quality of the learned planar dynamics.

This is a complex system subject to noise and complex nonlinear disturbances stemming from aerodynamic effects (e.g., ground-effect) and residual out-of-plane motion, and therefore constitutes a challenging evaluation benchmark.

State and control parameterization: We adopt the NED body-frame convention and the XYZ-Euler angle sequence parameterized by, in order, roll (ϕ), pitch (θ), and yaw (ψ) angles. For simplicity, we fixed the nominal yaw angle to 0 and inertial X-position to be constant so that the planar motion is aligned with the inertial Y-Z plane. The state representation used for the planar dynamics is therefore: $x = (p_y, p_z, \phi, v_y, v_z, \omega_x)$ where (p_y, p_z) are the inertial position in the Y-Z plane, (v_y, v_z) are the body-frame velocities, and ω_x is the body-frame angular rate. The control inputs are set to be the desired net normalized thrust τ and desired body rate $\dot{\phi}_c$. These inputs are fed into the PX4 autopilot which uses an on-board faster rate control loop to realize these commands.

The supervisory signal $\dot{x}$ is provided by $(\dot{p}_y, \dot{p}_z, \omega_x, \dot{v}_y, \dot{v}_z, \dot{\omega}_x)$. Note that by kinematics, $\omega_x = \dot{\phi}\cos(\theta)\cos(\psi) + \dot{\theta}\sin(\psi)$. Thus, any non-zero ψ, and/or non-zero $\dot{\theta}$ with non-zero ψ can introduce bias errors into the training data. The signals $\dot{\omega}_x$ and $(\dot{v}_y, \dot{v}_z)$ are obtained by convolving a finite-difference derivative estimate through a moving-average filter.

9.2.1 Model Training

Training data collection: To collect the training data, we created a set of training trajectories within the plane consisting of clockwise and counter-clockwise circles at varying speeds and "swoop" trajectories (see Figure 9.2) where the quadrotor swoops towards the ground in a parabolic arc and travels in a straight line close to the ground. The goal of the swoops was to learn data pertaining to ground-effect aerodynamic disturbances. All trajectories were tracked using a CCM-based feedback

controller derived from a nominal model of a quadrotor that *neglects any aerodynamic disturbances.* That is, the tracking controller and designed trajectories do not pre-compensate for any aerodynamic disturbances. Instead, we would like for such strategies to evolve directly from the learned model and subsequently derived tracking controller.

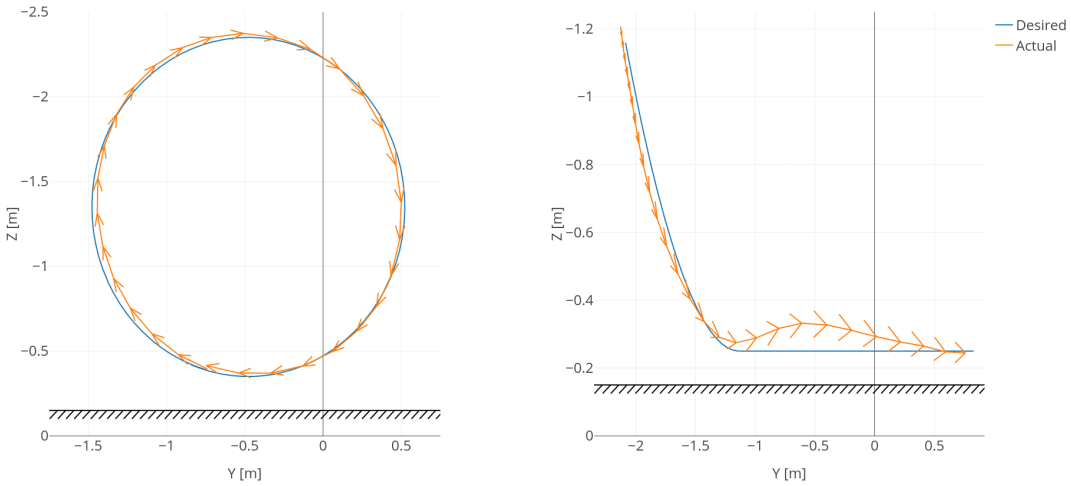


Figure 9.2: Examples of flown trajectories with a 3D nonlinear tracking feedback controller to create the training dataset. Left: clockwise circle with radius 1 m, period 6 s, and a nominal speed of 1.05 m/s. Right: Swoop trajectory to excite the ground-effect aerodynamic disturbances. Notice the deviation of the quadrotor from the desired trajectory as it approaches the ground. Ideally, a model learned from such data should generate nominal control signals that compensate for such effects.

We collected data from 3 clockwise and 3 counter-clockwise 1 m radius circles at periods of $10, 8$ and 6 seconds (corresponding to nominal speeds of 0.63, 0.79, and 1.05 m/s), and two symmetric swoop trajectories. Figure 9.3 shows the yaw and inertial X tracking errors for the fastest clockwise training circle. Both these quantities pertain to the out-of-plane motion and need to be small in order to minimize any induced bias within the planar motion, which is significantly coupled with the out-of-plane motion. This is indeed observed to be the case, validating the suitability of the collected data to train a planar dynamics model. Figure 9.2 shows a trace of a swoop training trajectory. Note the clear influence of ground-effect on the Z-tracking error as the quadrotor dips towards the ground and settles into the straight line path. These are exactly the sort of effects we wish to learn.

The complete dataset consists of approximately 2 minutes of flight time, logged at 250 Hz. After filtering the finite-difference derivative estimates, we further sub-sampled the data to avoid aliasing effects and extracted training and validation datasets of 1000 and 3000 $(x, u, \dot{x})$ tuples respectively.

Models Learned: To investigate the hypothesized trend where the control-theoretic regularization from the CCM approach dominates for small supervised training dataset sizes, we learned R-R and CCM-R models at $N = 150$ and $N = 1000$. For the CCM-R models, the full training constraint set

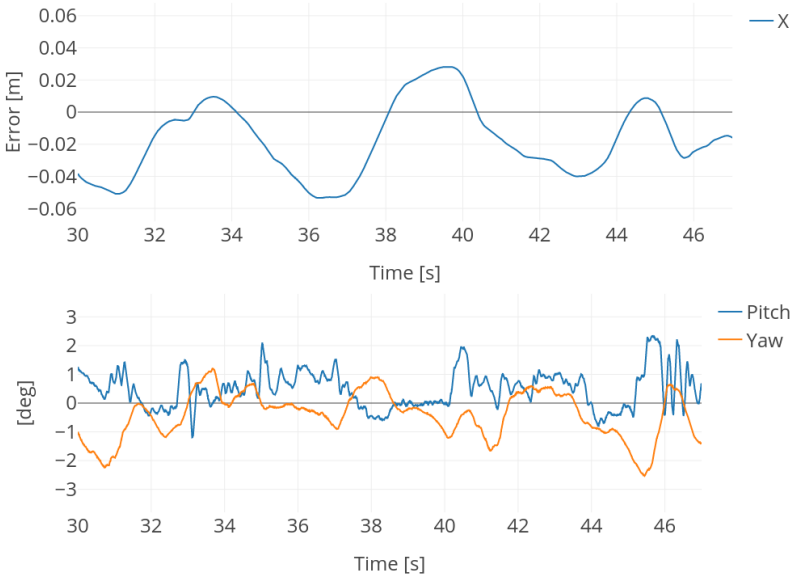
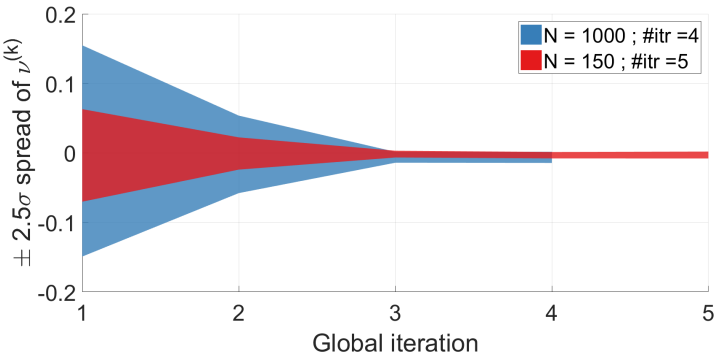


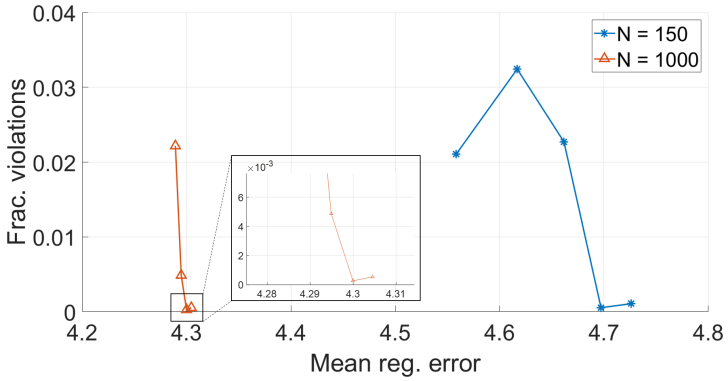
Figure 9.3: Out-of-plane errors, i.e., inertial X motion and pitch and yaw angles. All errors are sufficiently small, validating the use of the remaining data to train a planar dynamics model.

X_c comprised of the state samples from the demonstration tuples, plus up to 1600 additional state samples from the collected trajectories, and 500 independently sampled Halton points within the state-space. In total, for both the $N = 150$ and $N = 1000$ models, a total of 3100 constraint points were used.

The dimensionality of the feature space for $\hat{f}, \{\hat{b}_j\}_{j=1}^2$, and W were held the same as from the PVTOL simulations (corresponding to $\alpha, \beta_j \in \mathbb{R}^{576}$, and $\tilde{\theta}_{ij} \in \mathbb{R}^{72}$). The constant μ_f was held fixed at 10^{-3} for both R-R and CCM-R models and both N, and μ_b was set to 10^{-2}. The need for a higher regularization constant for β_j stemmed from the high noise content in the $\dot{\phi}_c$ and $\dot{\omega}_x$ signals. The constant μ_w was set to 10^{-3} for $N = 150$ and 10^{-4} for $N = 1000$, as in the PVTOL simulations. Figure 9.4 gives the training curve plots for the $N = 150$ and $N = 1000$ CCM-R models, and Table 9.2 summarizes the final measures of performance. The contraction rate λ was again set to 0.1 and all other tolerances from the simulations were held the same. CCM-R models converged in 5 iterations for $N = 150$ and 4 iterations for $N = 1000$.



(a) $\pm 2.5\sigma$ spread of the constraint violation vector ν over *training set* as a function of global iteration. `#itr` denotes the total number of global iterations.



(b) Evolution of mean regression error norm and fraction of violations over *validation* set with global iteration number. The markers delineate the iterations. The curves proceed from left to right.

Figure 9.4: Testbed data training curves for both CCM-R models.

	R-R		**CCM-R**		
N	Train err.	Val err.	Train err.	Val err.	Frac. viol.
150	4.077	4.549	0.399	4.726	0.0013 (0.0011)
1000	4.149	4.289	0.167	4.305	0.0048 (0.0005)

Table 9.2: Comparison of average (over N tuples) training and validation (over 3700 tuples) regression error norms for all 3 models. Also shown are the fraction of violations ($\nu > 0$) on the training (validation) sets for the CCM-R models. The termination threshold for CCM-R training was $\nu < \varepsilon = 0.01$ for all points in the training constraint set.

From Table 9.2, we note that the final regression validation errors for R-R and CCM-R are again close together, as in the simulation case. However, the training curves in Figure 9.4 appear to go in the opposite direction, i.e., validation regression error for CCM-R grows with iterations, starting from around 4.558 (almost identical to the R-R model), and growing to a final value of 4.726 as the

number of violations converges to 0. A potential reason could be that the noise in the demonstration tuples (i.e., from $\dot{x}$) forces more drastic updates in the parameters in order to find a compatible dual metric W. Nevertheless, as will be seen in the evaluation results, this small sacrifice in validation error performance works in the CCM-R model's favor.

9.2.2 Out-of-Plane Control for Enabling Evaluation

The out-of-plane errors in the training data were quite small (since the tracking controller was a full 3D state feedback controller). The learned planar dynamics will be used to generate a planar desired trajectory and LQR feedback tracking controller. To ensure that the quadrotor remains within the plane however, we need to separately close the loops on inertial X and yaw motion.

The nominal (ignoring disturbances) equation for p_x is given by:

$$\ddot{p}_x = -\tau \sin(\theta),$$

which is completely decoupled from the planar dynamics. In order to regulate p_x to an arbitrary constant $\bar{p}_x$ (without loss of generality, assume $\bar{p}_x = 0$), given a desired $\tau > 0$ computed using the planar control loop, we choose a desired inertial acceleration $\ddot{p}_{x,des} := -k_x p_x - k_v \dot{p}_x$, where $k_x, k_v > 0$ are control gains. Inverting the dynamics above, we obtain a desired pitch command θ_c:

$$\sin(\theta_c) = -\frac{\ddot{p}_{x,des}}{\tau} = \frac{k_x p_x + k_v \dot{p}_x}{\tau}.$$

From the above equation, we deduce a desired nominal pitch rate $\dot{\theta}_{des}$:

$$\dot{\theta}_{des} \cos(\theta_c) = \frac{\tau(k_x \dot{p}_x + k_v \ddot{p}_x) - \dot{\tau}(k_x p_x + k_v \dot{p}_x)}{\tau^2} \approx \frac{k_x \dot{p}_x}{\tau},$$

where we neglect the $\dot{\tau}$ and $\ddot{p}_x$ terms. The actual pitch rate command $\dot{\theta}_c$ sent to the quadrotor is then given by:

$$\dot{\theta}_c = \dot{\theta}_{des} + k_\theta(\theta_c - \theta),$$

which is a combination of the feedforward desired pitch rate plus a proportional feedback term on the error with respect to the desired pitch angle with gain $k_\theta > 0$. Finally, to keep yaw ψ at zero, we implement a simple proportional controller to generate a desired yaw rate command $\dot{\psi}_c$ given as:

$$\dot{\psi}_c = -k_\psi \psi,$$

with gain $k_\psi > 0$. In the next section we provide plots of p_x, θ, and ψ to check how well the planar assumption holds.

9.2.3 Evaluation

Test Trajectory Generation: The evaluation task is similar to that used for simulations, except that instead of initializing the quadrotor at random initial configurations and asking it to stabilize to a hover (an arguably dangerous test), we designed a desired nominal trajectory in the Y-Z plane consisting of segments of a continuous, smooth figure-eight, denoted as $(p_y^{\text{ref}}(t), p_z^{\text{ref}}(t))$. The nominal state and control trajectory for the quadrotor was then computed as the solution to the following trajectory optimization problem:

$$(x^*(\cdot), u^*(\cdot)) = \underset{x(\cdot),u(\cdot)}{\text{argmin}} \int_0^T \begin{bmatrix} p_y(t) - p_y^{\text{ref}}(t) \\ p_z(t) - p_z^{\text{ref}}(t) \end{bmatrix}^T Q \begin{bmatrix} p_y(t) - p_y^{\text{ref}}(t) \\ p_z(t) - p_z^{\text{ref}}(t) \end{bmatrix} + \begin{bmatrix} \tau(t) - g \\ \dot{\phi}_c(t) \end{bmatrix}^T R \begin{bmatrix} \tau(t) - g \\ \dot{\phi}_c(t) \end{bmatrix} dt,$$

where g is the gravitational acceleration, $T = 10$ s, and Q, R are cost weighting matrices in $\mathbb{S}_2^{>0}$. To avoid the initial jump in desired velocity for a quadrotor starting from rest, we designed the trajectory in three segments (see Figure 9.5). The quadrotor accelerates along half of the figure-eight path to full speed (point A to B in Figure 9.5) in the first segment. This is followed by a full figure-eight loop (back to B) in the second segment. Finally the quadrotor decelerates along the second half of the figure-eight back to zero velocity during the third segment. Each segment was designed to overlap with the desired figure-eight shape, and the desired speed was modulated along the path using a smooth time-varying phase function for the first and third segments. These two segments along with the speed along the trajectory are depicted in Figure 9.5.

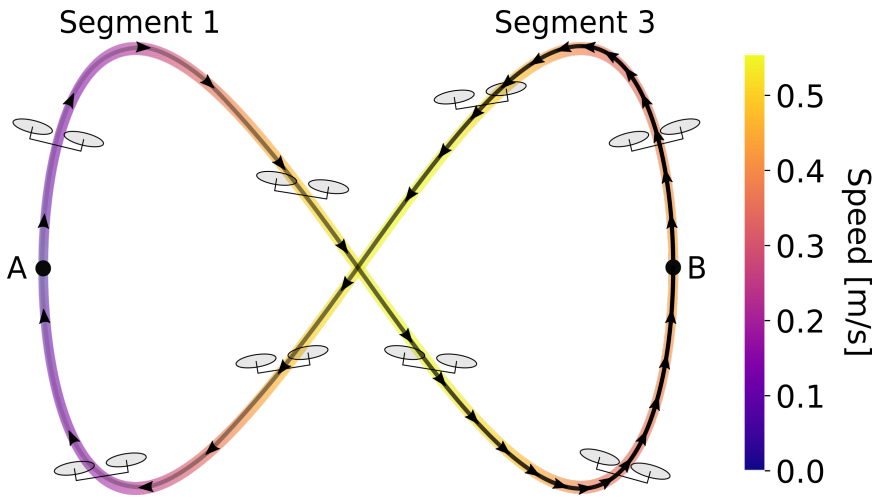


Figure 9.5: Illustration of the desired Y-Z plane trajectory to be flown for evaluation of the learned models. Shown here are the first (accelerating from A to B) and third (decelerating from B to A) segments of the trajectory with the indicated speed profile. The middle segment (not shown) is a complete figure eight maneuver that overlaps with the above path. The actual reference state/control trajectories for each model are computed using trajectory optimization where the cost to be minimized is a combination of control effort and the deviation from the desired figure-eight path above. All computed trajectories, projected onto the Y-Z plane overlap with the desired figure eight maneuver.

For each model tested, three trajectory optimization problems were solved corresponding to the three segments, each with the same time-span of 10 s. The middle segment therefore represents the most challenging and aggressive part of the trajectory, executed following the acceleration segment. The planar tracking controller was set as the steady-state LQR controller computed from the TV-LQR solution (to avoid having to additionally compute and store the true time-varying gain solution from the backward Riccati differential equation).

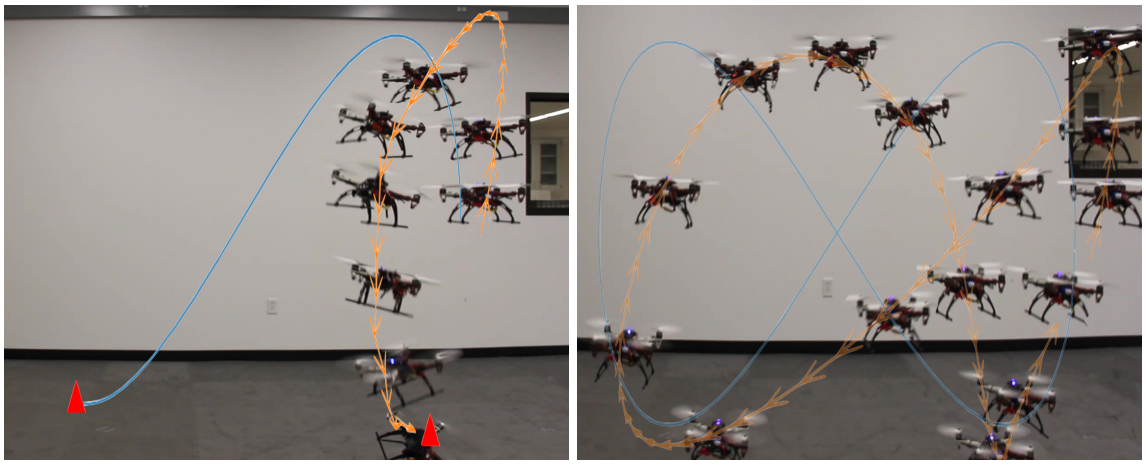


Figure 9.6: Time-lapse of a quadrotor trying to execute a figure-eight maneuver (blue curve) using a reference trajectory and an LQR feedback tracking controller generated using the learned dynamical system. *Left*: Model learned using traditional ridge-regression; *Right*: Model learned using control-theoretic regularization proposed within this work. The models were trained with the same, extremely limited (150 points) set of $(x, u, \dot{x})$ supervisory tuples. The quadrotor consistently failed and crashed into the floor with the trajectory and controller generated by the model learned with ridge-regression; the red triangles mark the points along the reference and actual trajectories at moment of crash – a separation of 1.6 m. In contrast, despite imperfect tracking (not unexpected given the extremely limited amount of supervision given to the learning algorithm), which leads to a slight graze along the floor at one point during the maneuver, the quadrotor manages to maintain bounded tracking error while using the model learned with control-theoretic regularization.

Results: We present tracking results[3] for all 4 models: (i) CCM-R $N = 1000$, (ii) R-R $N = 1000$, (iii) CCM-R $N = 150$, and (iv) R-R $N = 150$. Figure 9.7 plots tracking errors for the $N = 150$ case and Table 9.3 summarizes the tracking performance for all models in terms of RMS and maximum translational tracking errors. The quadrotor is unable to track the R-R $N = 150$ model generated trajectory and quickly becomes unstable as it accelerates into the middle segment of the trajectory. A human pilot needed to take control just as the quadrotor crashed into the ground. At the point of the takeover, the net Y tracking error was approximately 1.5 m.

In contrast, while the tracking performance for the CCM-R model is rather far from what could be achieved with an accurate model, all errors remain bounded. There is a point where the quadrotor grazes the ground as it passes through the fastest part of the middle segment, traveling downwards. The margin of error (distance between the lowest point on the trajectory and the ground) is 15 cm,

[3]Videos of the flight experiments can be found at `https://youtu.be/SK1tsYrXXUY`.

which is a challenging tracking constraint to meet for a quadrotor operating using a model trained with just 150 samples of supervision. Despite this, the quadrotor recovers and successfully completes the remaining portion of the trajectory. Figure 9.6 shows a time-lapse of the quadrotors during this middle segment of the maneuver.

From Table 9.3, as expected for the $N = 1000$ case, the tracking numbers for CCM-R and R-R are on par, with remaining differences at the noise level. For the $N = 150$ case, we only present numerics up until 14 s, which is when the R-R case crashes. For this range, CCM-R significantly outperforms R-R, let alone the fact that the quadrotor operating with the CCM-R model manages to complete the entire 30 s trajectory while maintaining stability. The results confirm the trend observed within simulations that, at low sample regimes, the stabilizability constraints enforced during the CCM-R model learning process have a dramatic context-driven regularization effect.

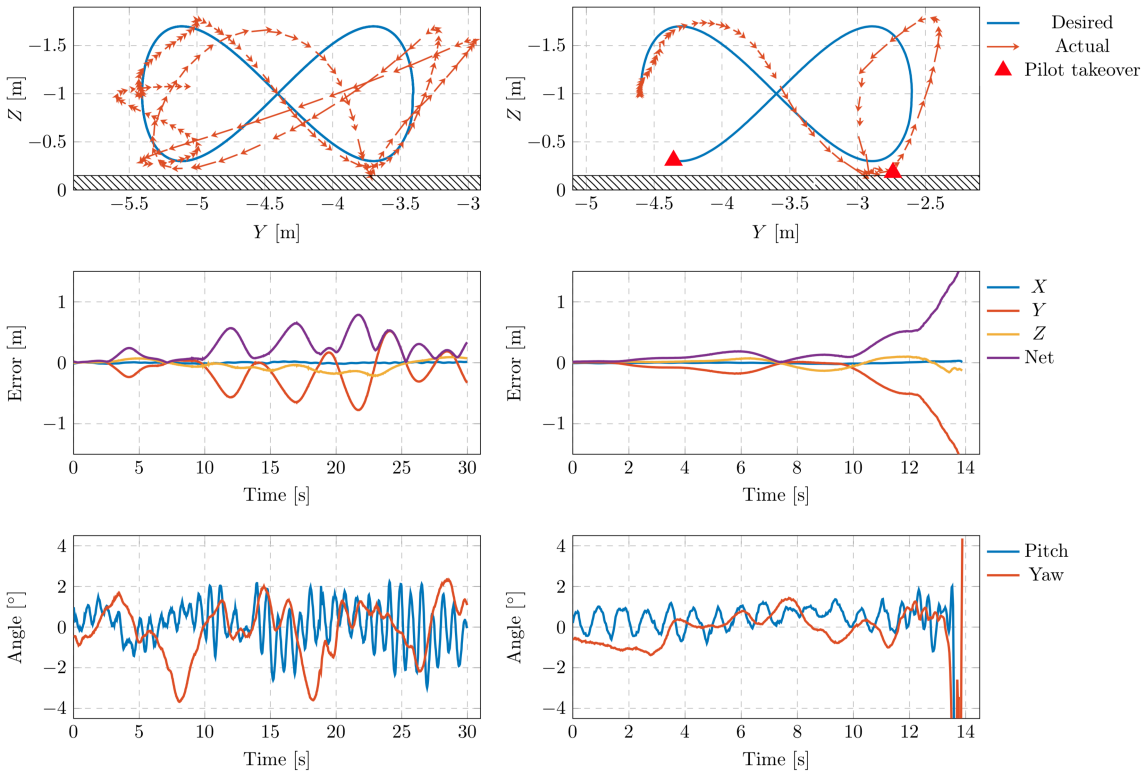


Figure 9.7: Visualization of tracking results for the learned models CCM-R $N = 150$ (left column) and R-R $N = 150$ (right column). The top row shows the desired and actual trajectory traces in the inertial Y-Z plane, while the middle row shows the error over time in the inertial X, Y, and Z coordinates. The quadrotor is unable to track the R-R $N = 150$ model generated trajectory and quickly becomes unstable as it accelerates into the middle segment of the trajectory. A human pilot needed to take control just as the quadrotor crashed into the ground. At the point of the takeover, the net Y tracking error was 1.5 m. Tracking the CCM-R $N = 150$ model generated trajectory, the quadrotor achieves bounded tracking errors and successfully completes the trajectory. The bottom row shows the pitch and yaw angles over time, which remain close to zero, thereby ensuring our planar motion assumption is valid.

	CCM-R $N = 1000$		**R-R $N = 1000$**		**CCM-R $N = 150$**		**R-R $N = 150$**	
Error	RMS	Max.	RMS	Max.	RMS	Max.	RMS	Max.
Y	0.150	0.445	0.126	0.290	0.208	0.567	0.423	1.618
Z	0.066	0.167	0.076	0.196	0.047	0.090	0.068	0.157
Net	0.165	0.446	0.148	0.308	0.213	0.570	0.429	1.623

Table 9.3: Numerical tracking results for all learned models. These include both root mean square (RMS) and maximum error values (in meters) over the entire trajectory. At $N = 1000$ training points, both the CCM-R and R-R models yield similar tracking performance with remaining differences at the noise level. The tracking numbers for $N = 150$ are presented up until 14 s, which is when the crash occured with the R-R model. CCM-R clearly outperforms R-R during those first 14 s, let alone the fact that the quadrotor operating with the CCM-R model successfully completes the entire 30 s trajectory while maintaining bounded errors.

9.3 Summary

The previous three chapters presented a framework for learning *controlled* dynamics from demonstrations for the purpose of trajectory optimization and control for continuous robotic tasks. By leveraging tools from nonlinear control theory, chiefly, contraction theory, we introduced the concept of learning *stabilizable* dynamics, a notion which guarantees the existence of feedback controllers for the learned dynamics model that ensures trajectory trackability. Borrowing tools from Reproducing Kernel Hilbert Spaces and convex optimization, we proposed a bi-convex semi-supervised algorithm for learning the dynamics and provided a substantial numerical study of its performance and comparison with traditional regression techniques. In particular, we validated the algorithm within simulations for a planar quadrotor system, and on a full quadrotor hardware testbed with partially closed loops to emulate planar quadrotor dynamics. The results lend credence to the hypothesis that enforcing stabilizability constraints during the learning process can have a dramatic regularization effect upon the learned dynamics in a manner that is tailored to the downstream task of trajectory generation and feedback control. This effect is most apparent when learning from smaller supervised training datasets, where we showed, both in simulation and on hardware, the quadrotor losing control and crashing when using a model learned with traditional ridge-regression. In contrast, the quadrotor is able to maintain bounded tracking performance when using the stabilizability regularized model in the low sample learning regime, and is on par with the ridge regularized model in the large sample regime.

We believe that the dynamics learning framework presented herein provides compelling motivation from both a stability and data efficiency standpoint, for incorporating control-theoretic notions within learning as a means of context-driven hypothesis pruning.

Solution Parameters for Planar Quadrotor Simulations

Notice that the true input matrix for the PVTOL system satisfies Assumption 2. Furthermore, it is a constant matrix. Thus, the feature mapping Φ_b is therefore just a constant matrix with the necessary sparsity structure.

The feature matrix for f was generated using the random matrix feature approximation to the Gaussian separable matrix-valued kernel with $\sigma = 6$ and $s = 8n = 48$ sampled Gaussian directions, yielding a feature mapping matrix Φ_f with $d_f = 576$ (96 features for each component of f). The scalar-valued reproducing kernels for the entries of W were taken to be the Gaussian kernel with $\sigma = 15$. To satisfy condition (8.4), the kernel for $w_{ij},\ (i,j) \in \{1,\ldots,(n-m)\}$ was only a function of the first $n-m$ components of x. A total of $s = 36$ Gaussian samples were taken to yield feature vectors ϕ_w and $\hat{\phi}_w$ of dimension $d_w = 72$. Furthermore, by symmetry of $W(x)$ only $n(n+1)/2$ functions were actually parameterized. Thus, the learning problem in (8.13) comprised of $d_f + d_w n(n+1)/2 + 4 = 508$ parameters for the functions, plus the extra scalar constants $\lambda, \underline{w}, \overline{w}$.

The learning parameters used were: model N-R (all N): $\mu_f = 0, \mu_b = 10^{-6}$, R-R (all N): $\mu_f = 10^{-4}, \mu_b = 10^{-6}$, CCM-R (all N): $\mu_f = 10^{-3}, \mu_b = 10^{-6}$; $N \in \{100, 250, 500\} : \mu_w = 10^{-3}$, $N = 1000 : \mu_w = 10^{-4}$. Tolerance parameters: constraints: $\{\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}\} = \{0.1, 0.1, 0.1\}$; discard tolerance $\delta = 0.05$. Note that a small penalization on μ_b was necessary for all models due to the fact that feature matrix Φ_b is rank deficient.

Part III

Risk-Sensitive Decision-Making

Chapter 10

Risk Sensitivity in Human Decision-Making

While the previous two parts of the thesis addressed decision-making at the lowest level of the autonomy stack, namely, optimization over continuous trajectories, this section of the thesis is concerned with higher-level sequential decision-making, specifically, in the context of human-robot interaction. As a motivating example, consider the driving scenario visualized in Figure 10.1.

Figure 10.1: A driving scenario where two cars are simultaneously attempting to negotiate a highway on-ramp/off-ramp. The intent of the two cars is shown by the colored arrows.

The snapshot illustrates a scenario where two cars, roughly side-by-side, are attempting to simultaneously negotiate a highway on-ramp/off-ramp. The car entering the highway has less than 200 ft of space in front of it before it is stuck taking the upcoming off-ramp. This is a test case that is notoriously challenging even for human drivers, let alone an autonomous or self-driving car. There are several different ways in how the interaction may proceed, i.e., the behavior is inherently *multi-modal*. For an autonomous car to successfully negotiate such an interaction, it needs the ability

to actively *infer* the intent of the other (human-driven) car, and potentially *influence* their actions.

In this section of the thesis, we describe a framework for learning a predictive model for human agents in safety-critical scenarios such as the one presented here. In particular, we adopt the *inverse reinforcement learning* (IRL) paradigm where it is assumed that the human agent makes decisions by minimizing an appropriate statistical measure of a stochastic cost. The objective then, is to learn the structure and parameters of this decision problem in order to make accurate predictions for the human agent's actions, and subsequently use this prediction model within the autonomous robot's planning algorithm. We will focus here purely on the inference problem.

Within this chapter, we provide an overview of the state-of-the-art in IRL and introduce a theory of *risk measures*.

10.1 Inverse Reinforcement Learning

In order to enable robots and humans operating in close vicinity of each other, robots should, among other things, be able to (i) accurately predict the actions of humans in their environment, (ii) quickly learn the preferences of human agents in their proximity and act accordingly, and (iii) learn how to accomplish new tasks from human demonstrations. IRL (Russell, 1998; Ng and Russell, 2000; Abbeel and Ng, 2005; Levine and Koltun, 2012; Ramachandran and Amir, 2007; Ziebart et al., 2008; Englert and Toussaint, 2015) is a powerful and flexible framework for tackling these challenges and has been previously used for a wide range of tasks, including modeling and mimicking human driver behavior (Abbeel and Ng, 2004; Kuderer et al., 2015; Sadigh et al., 2016a), pedestrian trajectory prediction (Ziebart et al., 2009; Mombaur et al., 2010; Kretzschmar et al., 2016), and legged robot locomotion (Zucker et al., 2010; Kolter et al., 2007; Park and Levine, 2013). More recently, the popular technique of Max-Entropy (MaxEnt) IRL, an inspiration for some of the techniques leveraged in this thesis, has been adopted in a deep learning framework (Wulfmeier et al., 2015), and embedded within the guided policy optimization algorithm (Finn et al., 2016). The underlying assumption behind IRL is that humans act optimally with respect to an (unknown) cost function. The goal of IRL is then to infer this cost function from observed actions of the human. By learning the human's underlying preferences (in contrast to, e.g., directly learning a policy for a given task), IRL allows one to generalize one's predictions to novel scenarios and environments.

The prevalent modeling assumption made by existing IRL techniques is that humans take actions in order to minimize the *expected value* of a random cost. Such a model, referred to as the expected value (EV) model, implies that humans are *risk neutral* with respect to the random cost; yet, humans are often far from being risk neutral. A generalization of the EV model is represented by the expected utility (EU) theory in economics (von Neumann and Morgenstern, 1944), whereby one assumes that a human is an optimizer of the expected value of a disutility function of a random cost. Despite the historical prominence of EU theory in modeling human behavior, a large body

of literature from the theory of human decision-making strongly suggests that humans behave in a manner that is *inconsistent* with the EU model. At a high level, the EU model has two main limitations: (i) experimental evidence consistently confirms that this model is lacking in its ability to describe human behavior in risky scenarios (Allais, 1953; Ellsberg, 1961; Kahneman and Tversky, 1979), and (i) the EU model assumes that humans make no distinction between scenarios in which the probabilities of outcomes are known and ones in which they are unknown, which is often not the case. Consequently, a robot interacting with a human in a safety-critical setting (e.g., as pictured in Figure 10.1), while leveraging such an inference model, could make incorrect assumptions about the human agent's behavior, potentially leading to catastrophic outcomes.

The known and unknown probability scenarios are referred to as *risky* and *ambiguous* respectively in the decision theory literature. An elegant illustration of the role of ambiguity is provided by the *Ellsberg paradox* (Ellsberg, 1961). Imagine an urn (Urn 1) containing 50 red and 50 black balls. Urn 2 also contains 100 red and black balls, but the relative composition of colors is unknown. Suppose that there is a payoff of \$10 if a red ball is drawn (and no payoff for black). In human experiments, subjects display an overwhelming preference towards having a ball drawn from Urn 1. However, now suppose the subject is told that a black ball has \$10 payoff (and no payoff for red). Humans *still* prefer to draw from Urn 1. This is a *paradox*, since choosing to draw from Urn 1 in the first case (payoff for red) indicates that the human assesses the proportion of red in Urn 1 to be higher than in Urn 2, while choosing Urn 1 in the second case (payoff for black) indicates that the human assesses a lower proportion of red in Urn 1 than in Urn 2. Indeed, there is no utility function for the two outcomes that can resolve such a contradictory assessment of underlying probabilities since it stems from a subjective distortion of outcome *probabilities* rather than *rewards*.

The limitations of EU theory in modeling human behavior has prompted substantial work on various alternative theories such as rank-dependent expected utility (Quiggin, 1982), expected uncertain utility (Gul and Pesendorfer, 2014), dual theory of choice (distortion risk measures) (Yaari, 1987), prospect theory (Kahneman and Tversky, 1979; Barberis, 2013), and many more (see (Majumdar and Pavone, 2017) for a recent review of the various axiomatic underpinnings of these risk measures). Further, one way to interpret the Ellsberg paradox is that humans are not only risk averse, but are also *ambiguity averse* – an observation that has sparked an alternative set of literature in decision theory on "ambiguity-averse" modeling; see, e.g., the recent review (Gilboa and Marinacci, 2016). It is clear that the assumptions made by EU theory thus represent significant restrictions from a modeling perspective in an IRL context since a human expert is likely to be both risk- and ambiguity- averse, especially in safety critical applications such as driving where outcomes are inherently ambiguous and can possibly incur very high cost.

The key insight of our work is to address these challenges by modeling humans as evaluating costs according to an (unknown) *risk measure*. A risk measure is a function that maps an uncertain cost to a real number (the expected value is thus a particular risk measure and corresponds to risk

neutrality). In particular, we will consider the class of *coherent risk measures* (CRMs) (Artzner et al., 1999; Shapiro, 2009; Ruszczyński, 2010). CRMs were proposed within the operations research community and have played an influential role within the modern theory of risk in finance (Rockafellar and Uryasev, 2000; Acerbi and Tasche, 2002; Acerbi, 2002; Rockafellar, 2007). This theory has also recently been adopted for risk-sensitive (RS) MPC and decision-making (Chow and Pavone, 2014; Chow et al., 2015; Singh et al., 2018), and guiding autonomous robot exploration for maximizing information gain in time-varying environments (Axelrod et al., 2016).

Coherent risk measures enjoy a number of useful properties that jointly provide key advantages over EV and EU theories in the context of IRL. First, they capture an entire spectrum of risk assessments from risk-neutral to worst-case and thus offer a significant degree of modeling flexibility. Second, they capture risk sensitivity in an *axiomatically justified* manner; specifically, they formally capture a number of intuitive properties that one would expect any risk measure should satisfy. Third, a representation theorem for CRMs implies that they can be interpreted as computing the expected value of a cost function in a worst-case sense over a *set* of probability distributions (referred to as the *risk envelope*). Thus, CRMs capture both risk and ambiguity aversion within the *same modeling framework* since the risk envelope can be interpreted as capturing uncertainty about the underlying probability distribution that generates outcomes in the world. Finally, they are tractable from a computational perspective; the representation theorem allows us to solve both the inverse and forward problems in a computationally tractable manner for a rich class of static and dynamic decision-making settings. An introduction to risk measures and CRMs is provided in the next section.

There is a large body of work on RS decision-making. For instance, in (Howard and Matheson, 1972) the authors leverage the exponential (or entropic) risk. This has historically been a very popular technique for parameterizing risk-attitudes in decision theory but suffers from the usual drawbacks of the EU framework such as the calibration theorem (Rabin, 2000). The latter states that very little risk aversion over moderate costs leads to unrealistically high degrees of risk aversion over large costs, which is undesirable from a modeling perspective. Other RS Markov Decision Process (MDP) formulations include Markowitz-inspired mean-variance (Filar et al., 1989; Tamar et al., 2012), percentile criteria on objectives (Wu and Yuanlie, 1999) and constraints (Geibel and Wysotzki, 2005), and cumulative prospect theory (Prashanth et al., 2016). This has driven research in the design of learning-based solution algorithms, i.e., RS reinforcement learning (Mihatsch and Neuneier, 2002; Bäuerle and Ott, 2011; Tamar et al., 2012; Petrik and Subramanian, 2012; Shen et al., 2014; Tamar et al., 2016; Chow et al., 2018). Ambiguity in MDPs is also well studied via the robust MDP framework, see, e.g., (Nilim and El Ghaoui, 2005; Xu and Mannor, 2010), as well as (Osogami, 2012; Chow et al., 2015) where the risk and ambiguity duality of CRMs is exploited. We refer the reader to the recent PhD thesis of Yinlam Chow (Chow, 2017) for a more extensive review of incorporating risk-sensitivity within the *design* of decision-making algorithms. The key

difference between this literature and the present work is that we consider the *inverse* reinforcement learning problem.

Results in the risk-sensitive IRL (RS-IRL) setting are more limited and have largely been pursued in the *neuroeconomics* literature (Glimcher and Fehr, 2014). For example, (Hsu et al., 2005) performed Functional Magnetic Resonance Imaging (FMRI) studies of humans making decisions in risky and ambiguous settings and modeled risk and ambiguity aversion using parametric utility and weighted probability models. In a similar vein, (Shen et al., 2014) models risk aversion using utility based shortfalls (with utility functions fixed a priori) and presents FMRI studies on humans performing a sequential investment task. While this literature may be interpreted in the context of IRL, the models used to predict risk and ambiguity aversion are quite limited. Risk in (Sadigh et al., 2016b) is captured via a *single* parameter to represent the aggressiveness of the expert driver – a fairly limited model that additionally does not account for probabilistic uncertainty. More recently, the authors in (Ratliff and Mazumdar, 2017) leverage the shortfall-risk model and associated $Q-$value decomposition introduced in (Shen et al., 2014) to devise a gradient-based RS-IRL algorithm. The model again assumes an a priori known risk measure and parameterized utility function and the learning loss function is taken to be the likelihood of the observed actions assuming the Boltzmann distribution fit to the optimal $Q-$values. There are two key limitations of this approach. First, learning is performed assuming a known utility function and risk measure – both of which, in general, are difficult to fix *a priori* for a given application. Second, computing gradients involves taking expectations with respect to the optimal policy as determined by the current value of the parameters. This must be determined by solving the fixed-point equations defining the "forward" RL problem – a computationally demanding task for large or infinite domains. This limitation is not an artifact of RS-IRL but in fact a standard complexity issue in any MaxEnt IRL-based algorithm.

Contributions: In this part of the thesis, we present an IRL algorithm that explicitly takes into account risk sensitivity under *general* axiomatically-justified risk models that jointly capture risk and ambiguity within the same modeling framework. In particular, we make four main contributions. First, we propose a flexible modeling framework for capturing risk sensitivity in humans by assuming that the human demonstrator (hereby referred to as the "expert") acts according to a CRM. This framework allows us to capture an entire spectrum of risk assessments from risk-neutral to worst-case. Second, we develop efficient algorithms based on Linear Programming (LP) for inferring an expert's underlying risk measure for a broad range of static decision-making settings, including a proof of convergence of the predictive capability of the algorithm in the case where we only attempt to learn the risk measure. We additionally consider cases where both the cost and risk measure of the expert are unknown. Third, we develop a maximum likelihood based model for inferring the expert's risk measure and cost function for a rich class of dynamic decision-making settings. Fourth, we demonstrate our approach on a simulated driving game using a state-of-the-art commercial driving simulator and present results on ten human participants. We show that our approach is able to infer

and mimic qualitatively different driving styles ranging from highly risk-averse to risk-neutral using only a minute of training data from each participant. We also compare the predictions made by our RS-IRL approach with one that models the expert using expected value theory and demonstrate that the RS-IRL framework more accurately captures observed participant behavior both qualitatively and quantitatively, especially in scenarios involving significant risk to the participant-driven car. With regards to the aforementioned literature on RS-IRL, within our formulation, we harnesses the elegant dual representation results for CRMs to avoid having to *assume* a known risk measure, and solve a significantly less complex forward problem by leveraging a receding-horizon planning model for the expert – a technique used to great effect also in (Sadigh et al., 2016a).

10.2 Introduction to Risk Measures

In this section we briefly review the theory of risk measures for both static and dynamic decision-making, focusing in particular on CRMs and their associated axiomatic and computational properties.

10.2.1 Static, Coherent Measures of Risk

Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the set of outcomes (sample space), $\mathcal{F}$ is a σ-algebra over Ω representing the set of events we are interested in, and $\mathbb{P}$ is a probability measure over $\mathcal{F}$. In this section of the thesis we will focus on disturbance models characterized by probability *mass* functions (pmfs), hence we restrict our attention to finite probability spaces (i.e., Ω has a finite number of elements or, equivalently, $\mathcal{F}$ is a finitely generated algebra). Denote with $\mathcal{Z}$ the space of random variables $Z : \Omega \mapsto (-\infty, \infty)$ defined over the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. In our setting, the random variable $Z \in \mathcal{Z}$ is interpreted as a cost, i.e., the smaller the realization of Z, the better. For Z, W, we denote by $Z \leq W$ the pointwise partial order, i.e., $Z(\omega) \leq W(\omega)$ for all $\omega \in \Omega$.

By a *risk measure* we understand a function $\rho(Z)$ that maps an uncertain outcome Z into the extended real line $\bar{\mathbb{R}} := \mathbb{R} \cup \{+\infty\} \cup \{-\infty\}$. In particular, we restrict our analysis to *coherent risk measures*, defined as follows:

Definition 3 (Coherent Risk Measures)**.** *A coherent risk measure (CRM) is a mapping $\rho : \mathcal{Z} \to \mathbb{R}$, satisfying the following four axioms: for all $Z, W \in \mathcal{Z}$,*

A1. Monotonicity: $Z \leq W \Rightarrow \rho(Z) \leq \rho(W)$*;*

A2. Translational Invariance: $\forall a \in \mathbb{R}$, $\rho(Z + a) = \rho(Z) + a$*;*

A3. Positive Homogeneity: $\forall \lambda \geq 0$, $\rho(\lambda Z) = \lambda \rho(Z)$*;*

A4. Subadditivity: $\rho(Z + W) \leq \rho(Z) + \rho(W)$*.*

These axioms were originally proposed in (Artzner et al., 1999) to ensure the "rationality" of risk assessments. For example, A1 states that if a random cost Z is less than or equal to a random

cost W *regardless of the disturbance realizations*, then Z must be considered less risky (one may think of the cost distributions Z and W stemming from different control policies). A4 reflects the intuition that a risk-averse agent should prefer to *diversify*. We refer the reader to (Artzner et al., 1999; Majumdar and Pavone, 2017) for a thorough justification of these axioms.

One of the main properties for CRMs is a universal representation theorem, which in the context of *finite* probability spaces takes the following form:

Theorem 11 (Representation Theorem for Coherent Risk Measures (Artzner et al., 1999))**.** *Consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is finite with cardinality $|\Omega|$, $\mathcal{F} = 2^{\Omega}$, and $\mathbb{P} = (p(1), p(2), \ldots, p(|\Omega|))$, with all probabilities positive. Denote by $\mathcal{C}$ the set of probability densities:*

$$\mathcal{C} := \left\{ \zeta \in \mathbb{R}^{|\Omega|} \mid \sum_{i=1}^{|\Omega|} p(i)\zeta(i) = 1,\, \zeta \geq 0 \right\}. \tag{10.1}$$

Define $q_\zeta \in \mathbb{R}^{|\Omega|}$ where $q_\zeta(i) = p(i)\zeta(i)$, $i = 1, \ldots, |\Omega|$. A risk measure $\rho : \mathcal{Z} \to \mathbb{R}$ with respect to the space $(\Omega, \mathcal{F}, \mathbb{P})$ is a CRM if and only if there exists a compact convex set $\mathcal{B} \subset \mathcal{C}$ such that for any $Z \in \mathcal{Z}$:

$$\rho(Z) = \max_{\zeta \in \mathcal{B}} \mathbb{E}_{q_\zeta}[Z] = \max_{\zeta \in \mathcal{B}} \sum_{i=1}^{|\Omega|} p(i)\zeta(i)Z(i). \tag{10.2}$$

This theorem is important for two reasons. Conceptually, it gives us an interpretation of CRMs as computing the worst-case expectation of the cost with respect to a *set* of *distorted* distributions $q_\zeta = p \cdot \zeta$. Coherent risk measures thus allow us to consider risk and ambiguity in a unified framework since one may interpret an agent acting according to a coherent risk model as being *uncertain about the underlying probability density*. Practically, estimating this set of distributions provides us with an algorithmic handle for inferring the expert's risk preferences, and indeed will form the basis of our IRL methodology.

Polytopic Risk Measures

In this work, we will take the set $\mathcal{B}$ in (10.2) to be a polytope. Let $\Delta^{|\Omega|}$ denote the $|\Omega|-$dimensional probability simplex, defined as:

$$\Delta^{|\Omega|} := \{ q \in \mathbb{R}^{|\Omega|} \mid \sum_{i=1}^{|\Omega|} q(i) = 1,\; q \geq 0 \}.$$

By absorbing the density ζ into the pmf p in (10.2), we can represent (without loss of generality) a polytopic CRM as:

$$\rho(Z) = \max_{q \in \mathcal{P}} \mathbb{E}_q[Z], \tag{10.3}$$

where $\mathcal{P}$ is a polytopic subset of the probability simplex $\Delta^{|\Omega|}$:

$$\mathcal{P} = \left\{ q \in \Delta^{|\Omega|} \mid A_{\text{ineq}} q \leq b_{\text{ineq}} \right\}, \tag{10.4}$$

where the matrix $A_{\text{ineq}} \in \mathbb{R}^{d \times |\Omega|}$ and vector $b_{\text{ineq}} \in \mathbb{R}^d$ define a set of d halfspace constraints. The polytope $\mathcal{P}$ is hereby referred to as the *risk envelope*. Polytopic CRMs constitute a rich class of risk measures, encompassing a spectrum ranging from risk neutrality ($\mathcal{P} = \{p\}$) to worst-case assessments ($\mathcal{P} = \Delta^{|\Omega|}$); see also (Eichhorn and Römisch, 2005; Chow and Pavone, 2014; Shapiro et al., 2014). A particularly well-known example of a polytopic CRM is the Conditional Value-at-Risk (CVaR) risk measure, defined as follows.

For an integrable cost random variable $Z \in \mathcal{Z}$, let the quantity $v_{1-\alpha}(Z) := \inf\{z \in \mathbb{R} \,|\, \mathbb{P}(Z \leq z) \geq 1-\alpha\}$ denote its $(1-\alpha)$-quantile (also referred to as the Value-at-Risk, or VaR). For continuous distributions[1], $\text{CVaR}_\alpha(Z)$ is defined as:

$$\text{CVaR}_\alpha(Z) := \mathbb{E}[Z \,|\, Z \geq v_{1-\alpha}(Z)].$$

That is, $\text{CVaR}_\alpha(Z)$ is the expected value of the α-tail distribution of Z (see Figure 10.2). Equivalently, it may be expressed in the form (10.3) where the polytopic risk envelope $\mathcal{P}$ is given as

$$\left\{ q \in \Delta^{|\Omega|} \mid 0 \leq q(j) \leq \frac{p(j)}{\alpha} \quad \forall j \in \{1, \ldots, |\Omega|\} \right\}.$$

In particular, one can show that when $\alpha = 1$, $\text{CVaR}_\alpha(Z)$ reduces to the standard expected value $\mathbb{E}[Z]$. Thus, the expected value is a *special case* of CVaR. See (Shapiro et al., 2014, Chapter 6) for additional examples of CRMs, which include CVaR, mean absolute semi-deviation, spectral risk measures, optimized certainty equivalent, and the distributionally robust risk.

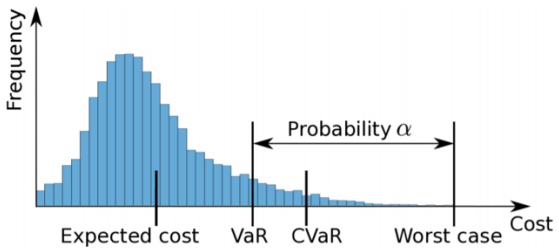


Figure 10.2: Illustration of the CVaR_α CRM. $\text{CVaR}_\alpha(Z)$ quantifies the mean of the α-tail of the cost distribution of Z.

[1] More general definitions can be found in (Rockafellar and Uryasev, 2002).

10.2.2 Dynamic, Time-Consistent Measures of Risk

This section provides a multi-period generalization of the concepts presented in the previous section and follows closely the discussion in (Ruszczyński, 2010). Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, a filtration $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \mathcal{F}_2 \cdots \subset \mathcal{F}_T \subset \mathcal{F}$, and an adapted sequence of real-valued random variables Z_t, $t \in \{0, \ldots, T\}$. We assume that $\mathcal{F}_0 = \{\Omega, \emptyset\}$, i.e., Z_0 is deterministic. The variables Z_t can be interpreted as stage-wise costs. For each $t \in \{0, \ldots, T\}$, denote with $\mathcal{Z}_t$ the space of random variables defined over the probability space $(\Omega, \mathcal{F}_t, \mathbb{P})$; also, let $\mathcal{Z}_{t,T} := \mathcal{Z}_t \times \cdots \times \mathcal{Z}_T$. Given sequences $Z = \{Z_t, \ldots, Z_T\} \in \mathcal{Z}_{t,T}$ and $W = \{W_t, \ldots, W_T\} \in \mathcal{Z}_{t,T}$, we interpret $Z \leq W$ component-wise, i.e., $Z_j \leq W_j$ for all $j \in \{t, \ldots, T\}$.

A dynamic risk measure is a sequence of mappings $\rho_{t,T} : \mathcal{Z}_{t,T} \to \mathcal{Z}_t$, $t \in \{0, \ldots, T\}$, obeying the following monotonicity property: $\rho_{t,T}(Z) \leq \rho_{t,T}(W)$ for all $Z, W \in \mathcal{Z}_{t,T}$ such that $Z \leq W$. This monotonicity property is an intuitive extension of the monotonicity property for single-step risk assessments, and an arguably defensible axiom for all risk assessments.

To give dynamic risk measures a concrete functional form, we need to generalize the CRM axioms presented in Definition 3 to the dynamic case.

Definition 4 (Coherent One-Step Conditional Risk Measures)**.** *A* coherent one-step conditional risk measure *is a mapping $\rho_t : \mathcal{Z}_{t+1} \to \mathcal{Z}_t$, for all $t \in \mathbb{N}$, that obeys four axioms, namely, for all $Z_{t+1}, Y_{t+1} \in \mathcal{Z}_{t+1}$ and $Z_t \in \mathcal{Z}_t$:*

A1. Monotonicity: $Z_{t+1} \leq Y_{t+1} \Rightarrow \rho_t(Z_{t+1}) \leq \rho_t(Y_{t+1})$.

A2. Translation invariance: $\rho_t(Z_{t+1} + Z_t) = \rho_t(Z_{t+1}) + Z_t$.

A3. Positive homogeneity: $\forall \lambda \geq 0$, $\rho_t(\lambda Z_{t+1}) = \lambda \rho_t(Z_{t+1})$.

A4. Subadditivity: $\rho_t(Z_{t+1} + Y_{t+1}) \leq \rho_t(Z_{t+1}) + \rho_t(Y_{t+1})$.

Note that each ρ_t is a random variable on the space $\mathcal{Z}_t$ and given the discrete underlying probability space, each component of ρ_t is uniquely identified by the sequence of disturbances preceding stage t (hence the term *conditional*). Furthermore, it is readily observed that a mapping $\rho_t : \mathcal{Z}_{t+1} \to \mathcal{Z}_t$ is a coherent one-step conditional risk measure if and only if each component of ρ_t is a CRM.

As investigated in (Ruszczyński, 2010), in order for dynamic risk assessments to satisfy the intuitive monotonicity condition and to ensure rationality of evaluations over time, a dynamic risk measure must have a compositional form:

$$\begin{aligned} \rho_{t:t'}(Z_{t:t'}) :=& Z_t + \rho_t \left(Z_{t+1} + \rho_{t+1} \left(Z_{t+2} + \ldots + \rho_{t'-1}(Z_{t'}) \cdots \right) \right) \\ =& \rho_t \circ \cdots \circ \rho_{t'-1}(Z_t + \cdots + Z_{t'}), \end{aligned} \tag{10.5}$$

where each ρ_t is a coherent one-step conditional risk measure, and the second equality follows by the translational invariance property. Figure 10.3 provides a helpful visualization of such a compounded functional form.

When discussing IRL in the context of multi-period decisions (Chapter 12), we will leverage

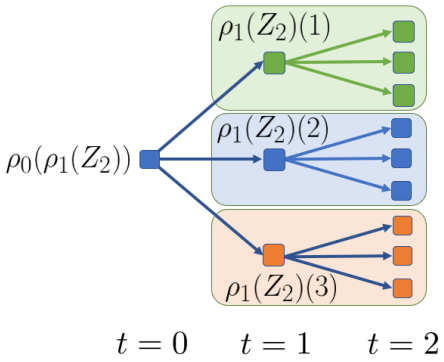


Figure 10.3: A scenario tree with three uncertain outcomes at each stage. The one-step risk mapping $\rho_1(Z_2) \in \mathcal{Z}_1$ maps the random cost $Z_2 \in \mathcal{Z}_2$ to a risk assessment at stage 1, i.e., is a random variable on $\mathcal{Z}_1$ and is thus isomorphic to the space $\mathbb{R}^3$. Here, each component j of $\rho_1(Z_2)$, i.e., $\rho_1(Z_2)(j)$, associated with node j at stage 1 (e.g., for $j = 1$, the green node), is a CRM over the children of node j at stage 2. The mapping $\rho_0(\rho_1(Z_2))$ subsequently maps the risk-assessments at stage 1 (i.e., $\rho_1(\cdot)$) back to stage 0.

dynamic risk measures with compositional structure as shown in (10.5). In the next chapter however, we first consider risk-sensitive IRL for static decision-making.

10.3 Summary

This chapter reviewed some of the state-of-the-art literature in IRL, a popular framework for capturing human decision-making in algorithmic form. We motivated the need for *risk-sensitivity* within an IRL formulation from a viewpoint of how different agents interpret *probabilities*. Risk measures allow us to capture exactly this phenomenon, indeed the dual representation theorem (Theorem 11) makes clear this notion of a *subjective* view of probabilities. As our experiments in Chapter 12 will illustrate, such a representation is in fact necessary to be able to generalize across a wide range of human decision-making styles – something pure reward/cost shaping cannot achieve. Having introduced risk measures, in the next chapter we provide algorithms for risk-sensitive IRL in the static decision-making case.

Chapter 11

Static Risk-Sensitive IRL

In this chapter we setup the IRL problem by defining the environment dynamics and a RS decision-model for the human agent (expert) for *static* decision-making. That is, we assume that the expert chooses a single action at the current time-step in order to minimize a RS evaluation of a cost incurred at the next time-step, following a stochastic state transition. After introducing the decision model, we provide two algorithms for inferring both the internal cost function and risk measure of the expert, given a dataset of demonstrations. Additionally, for the case where only the risk measure is unknown, we provide a proof of convergence for the predictions generated using the learned risk measure.

11.1 System Dynamics and Decision Model

Consider the following discrete-time dynamical system:

$$x_{k+1} = f(x_k, u_k, w_k), \tag{11.1}$$

where $k \in \mathbb{N}$ is the time index, $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the control input, and $w_k \in \mathcal{W}$ is the disturbance. The control input is assumed to be bounded component-wise: $u_k \in \mathcal{U} := \{u : u^- \leq u \leq u^+\}$. We take $\mathcal{W}$ to be a finite set $\{w^{[1]}, \ldots, w^{[L]}\}$ with probability mass function (pmf) $p := [p(1), p(2), \ldots, p(L)]$, where $\sum_{i=1}^{L} p(i) = 1$ and $p(i) > 0, \forall i \in \{1, \ldots, L\}$.

The static, or single-step decision problem, corresponds to a setting where given a current (known) state x_0, the expert chooses a single control action u_0 to minimize a coherent risk assessment of a random cost Z, represented by a non-negative cost function $C(x_1, u_0)$, where $x_1 = f(x_0, u_0, w_0)$. Thus, the uncertain cost Z is a random variable on the discrete probability space $(\mathcal{W}, 2^{\mathcal{W}}, p)$. Chapter 12 will discuss the more challenging case of multi-step decision problems.

We assume that we are given demonstrations from the expert in the form of a dataset $\mathcal{D}$ of

state-control pairs $\{(x^{*,d}, u^{*,d})\}_{d=1}^{D}$, corresponding to the expert taking action $u^{*,d}$ at state $x^{*,d}$. It is assumed that the expert has knowledge of the underlying dynamics (11.1) and disturbance realizations $\mathcal{W}$, but not the disturbance pmf p.

Formally, a coherent risk model implies that the expert is solving the following optimization problem at state x_0 in order to compute an optimal action:

$$\tau^* := \min_{u_0 \in \mathcal{U}} \ \rho(C(x_1, u_0)) = \min_{u_0 \in \mathcal{U}} \max_{q \in \mathcal{P}} \ \mathbb{E}_q[C(x_1, u_0)] \tag{11.2}$$

$$:= \min_{u_0 \in \mathcal{U}} \max_{q \in \mathcal{P}} \ g(x_0, u_0)^T q, \tag{11.3}$$

where $g(x_0, u_0)(j)$ is the cost when the disturbance $w_0 = w^{[j]} \in \mathcal{W}$ is realized, and $\rho(\cdot)$ is a CRM with respect to the space $(\mathcal{W}, 2^{\mathcal{W}}, p)$, with risk envelope $\mathcal{P}$ a subset of the probability simplex Δ^L.

The objective of RS-IRL therefore is to devise an algorithmic framework whereby an expert's cost function and "risk preferences" will be estimated using the given dataset $\mathcal{D}$ of observed state-control pairs. Under the coherent risk model formulation, estimating risk preferences entails finding an approximation of the true risk envelope $\mathcal{P}$. We first consider the case where the cost function C is known.

11.2 Known Cost Function

We first consider the static decision-making setting where the expert's cost function is known but the risk measure is unknown. Since the inner maximization problem in (11.3) is linear in q, the optimal value is achieved at a vertex of the polytope $\mathcal{P}$. Let $\text{vert}(\mathcal{P}) = \{v_i\}$ denote the set of vertices of $\mathcal{P}$ and let N_V be the cardinality of this set. Then, we can rewrite problem (11.2) as:

$$\begin{aligned} \min_{u_0 \in \mathcal{U}, \tau} \quad & \tau \\ \text{s.t.} \quad & \tau \geq g(x_0, u_0)^T v_i, \quad i \in \{1, \ldots, N_V\}. \end{aligned} \tag{11.4}$$

If the cost function $C(\cdot, \cdot)$ is convex in the control input u, the resulting optimization problem is convex. Given the dataset $\mathcal{D} = \{(x^{*,d}, u^{*,d})\}_{d=1}^{D}$ of optimal state-control pairs, our goal is to deduce an approximation $\mathcal{P}_o$ of $\mathcal{P}$ from the given data. The key idea of our technical approach is to examine the Karush-Kuhn-Tucker (KKT) conditions for Problem (11.4). The use of KKT conditions for Inverse Optimal Control is a technique also adopted in (Englert and Toussaint, 2015). The KKT conditions are necessary for optimality in general and are also sufficient in the case of convex problems. We can thus use the KKT conditions along with the dataset $\mathcal{D}$ to *constrain the constraints* of problem (11.4). In other words, the KKT conditions will allow us to constrain where the vertices of $\mathcal{P}$ must lie in order to be consistent with the fact that the state-control pairs represent optimal solutions to problem (11.4). Importantly, we will *not* assume access to the number

of vertices N_V of $\mathcal{P}$.

Specifically, let (x^*, u^*) be an optimal state-control pair and let $\mathcal{J}^+$ and $\mathcal{J}^-$ denote the sets of components of the control input u^* that are saturated above and below respectively (i.e., $u^*(j) = u^+(j)$ for all $j \in \mathcal{J}^+$ and $u^*(j) = u^-(j)$ for all $j \in \mathcal{J}^-$).

Theorem 12 (KKT-Based Inference)**.** *Consider the following optimization problem:*

$$\begin{aligned} \max_{\substack{v\in\Delta^L \\ \sigma_+,\sigma_-\geq 0}} \quad & g(x^*,u^*)^T v \\ \text{s.t.} \quad & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^*,u^*} + \sigma_+(j), \forall j \in \mathcal{J}^+ \\ & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^*,u^*} - \sigma_-(j), \forall j \in \mathcal{J}^- \\ & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^*,u^*}, \forall j \notin \mathcal{J}^+, j \notin \mathcal{J}^- \\ & \sigma_+(j) = 0,\ \sigma_-(j) = 0, \quad \forall j \notin \mathcal{J}^+, j \notin \mathcal{J}^- \end{aligned} \tag{11.5}$$

Denote the optimal value of this problem by τ' and define the halfspace:

$$\mathcal{H}_{(x^*,u^*)} := \{v \in \mathbb{R}^L \mid \tau' \geq g(x^*,u^*)^T v\}. \tag{11.6}$$

Then, the risk envelope $\mathcal{P}$ satisfies $\mathcal{P} \subset (\mathcal{H}_{(x^*,u^*)} \cap \Delta^L)$.

Proof. The KKT conditions for Problem (11.4) are:

$$1 = \sum_{i=1}^{N_V} \lambda_i, \tag{11.7}$$

$$0 = \lambda_i [g(x^*,u^*)^T v_i - \tau],\ i = 1,\dots,N_V, \tag{11.8}$$

and for $j = 1,\dots,m$:

$$0 = \sigma_+(j) - \sigma_-(j) + \sum_{i=1}^{N_V} \lambda_i\ \nabla_{u(j)} g(x,u)^T v_i\big|_{x^*,u^*}, \tag{11.9}$$

$$0 = \sigma_+(j)[u^*(j) - u^+(j)],\ 0 = \sigma_-(j)[u^-(j) - u^*(j)], \tag{11.10}$$

where $\lambda_i, \sigma_+(j), \sigma_-(j) \geq 0$ are multipliers. Now, suppose there are multiple optimal vertices $\{v_i\}_{i\in\mathcal{I}}$ for Problem (11.4) in the sense that $\tau^* = g(x^*,u^*)^T v_i$, for all $i \in \mathcal{I}$. Defining $\bar{v} := \sum_{i\in\mathcal{I}} \lambda_i v_i$, we see that $\bar{v}$ satisfies:

$$0 = \nabla_{u(j)} g(x^*,u^*(j))^T \bar{v} + \sigma_+(j) - \sigma_-(j), \quad j = 1,\dots,m, \tag{11.11}$$

and $\tau^* = g(x^*, u^*)^T \bar{v}$ since $\sum_{i \in \mathcal{I}} \lambda_i = 1$. Now, since $\bar{v}$ satisfies the constraints of Problem (11.5) (which are implied by the KKT conditions), it follows that $\tau' \geq \tau^*$. From problem (11.4), we see that $\tau' \geq \tau^* \geq g(x^*, u^*)^T v_i$ for all $v_i \in \text{vert}(\mathcal{P})$ and thus $\mathcal{P} \subset (\mathcal{H}_{(x^*, u^*)} \cap \Delta^L)$. □

Problem (11.5) is a *Linear Program (LP)* and can thus be solved efficiently. For each demonstration $(x^{*,d}, u^{*,d}) \in \mathcal{D}$, Theorem 12 provides a halfspace constraint on the risk envelope $\mathcal{P}$. By aggregating these constraints, we obtain a *polytopic* outer approximation $\mathcal{P}_o$ of $\mathcal{P}$. This is summarized in Algorithm 6. Note that Algorithm 6 operates sequentially through the data $\mathcal{D}$ and is thus directly applicable in *online* settings. An illustration of the sequential pruning process in Algorithm 6 is provided in Figure 11.1.

Algorithm 6 Sequential Halfspace Pruning

1: Initialize $\mathcal{P}_o = \Delta^L$
2: **for** $d = 1, \ldots, D$ **do**
3: Solve Linear Program (11.5) with $(x^{*,d}, u^{*,d})$ to obtain a hyperplane $\mathcal{H}_{(x^{*,d}, u^{*,d})}$
4: Update $\mathcal{P}_o \leftarrow \mathcal{P}_o \cap \mathcal{H}_{(x^{*,d}, u^{*,d})}$
5: **end for**
6: Return $\mathcal{P}_o$

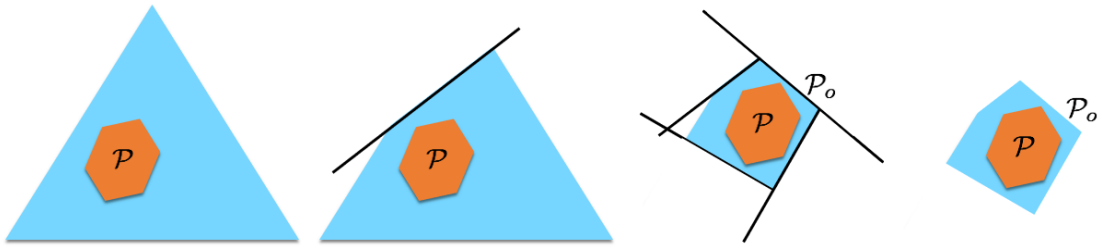


Figure 11.1: Schematic illustration of Algorithm 6. Probability simplex (3 scenarios) is shown in blue while the true (unknown) risk envelope is shown in orange. Algorithm 6 sequentially prunes portions of probability simplex that are inconsistent with the observed actions by leveraging the *necessary* conditions for optimality (KKT conditions) for problem (11.2). The end result is an outer approximation $\mathcal{P}_o$ of the true risk envelope $\mathcal{P}$.

Remark 10. *Algorithm 6 is a* non-parametric *algorithm for inferring the expert's risk measure; i.e., we are not fitting parameters for an a priori chosen risk measure. Instead, by leveraging the dual representation of CRMs as provided by Theorem 11 and reasoning directly over the risk envelope $\mathcal{P}$, Algorithm 6 can recover any polytopic risk measure within the class of CRMs, that best explains the expert's demonstrations.*

Remark 11. *As we collect more half-space constraints in Algorithm 6, the constraint $v \in \Delta^L$ in Problem* (11.5) *above can be replaced by $v \in \mathcal{P}_o$, where $\mathcal{P}_o$ is the current outer approximation of the risk envelope. It is easily verified that the results of Theorem 12 still hold. This allows us to obtain a tighter (i.e., lower) upper bound τ' for τ^*, thus resulting in tighter halfspace constraints for each*

new demonstration processed by the algorithm.

Denote by $\mathcal{P}_D$ the output of Algorithm 6 after processing sequentially the first D demonstrations $\{(x^{*,d}, u^{*,d})\}_{d=1}^{D}$. Observe that for all $D \geq 1$, $\mathcal{P}_{D+1} \subseteq \mathcal{P}_D$. We can then define the limiting set as $\mathcal{P}_\infty := \bigcap_{d=1}^{\infty} \mathcal{P}_d$. An important consideration for this algorithm is whether it is possible to recover, at least from an imitation perspective, the risk envelope $\mathcal{P}$ from sufficiently many optimal demonstrations. In other words, we are specifically interested in the question of whether the limiting set $\mathcal{P}_\infty$ (whenever such a limit exists) allows one to *exactly* predict the actions of a decision maker that operates under a risk model characterized by the set $\mathcal{P}$. In the following theorem we establish, under some restrictive technical conditions, that this is indeed possible. The proof is provided at the end of this chapter.

Theorem 13 (Convergence of Algorithm 6)**.** *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a convex, compact subset of the state space. Let $\{(x^{*,d}, u^{*,d})\}_{d=1}^{\infty}$ be a set of infinitely many optimal demonstrations such that the sequence $\{x^{*,d}\}$ is dense in $\mathcal{S}$. Assume that the following technical conditions hold:*

A.1 *The expert's cost vector $g(x, u)$ is strictly convex with respect to the control input u and continuous with respect to the state variable x.*

A.2 *For all $j \in \{1, \ldots, L\}$ and any state $x \in \mathcal{S}$, the cost function associated with the j-th disturbance $u \mapsto g(x, u)(j)$ has bounded level sets.*

Finally, for any risk envelope $\mathcal{P}' \subseteq \Delta^L$ and any state $x \in \mathcal{S}$, define

$$u(\mathcal{P}', x) := \operatorname*{argmin}_{u \in \mathcal{U}} \max_{v \in \mathcal{P}'} v^T g(x, u),$$

as the (unique) optimal control action of an expert with risk envelope $\mathcal{P}'$ at state x. Then, for any state $x \in \mathcal{S}$,

$$u(\mathcal{P}_\infty, x) = u(\mathcal{P}, x). \tag{11.12}$$

That is, for any state $x \in \mathcal{S}$, the optimal action predicted using the limiting envelope $\mathcal{P}_\infty$ matches that computed using the true expert polytope $\mathcal{P}$.

Remark 12. *The technical condition **A.1** assumes convexity of the cost vector with respect to the control input, and the proof of Theorem 13 heavily relies on this assumption. Finding conditions under which Algorithm 6 is guaranteed to be consistent for the general case of non-convex cost functions is an open problem left for future research; we re-emphasize, though, that Algorithm 6 is guaranteed to provide a conservative outer approximation regardless of the convexity of the cost vectors.*

Once we have recovered an approximation $\mathcal{P}_o$ of $\mathcal{P}$, we can solve the "forward" problem (i.e., compute actions at a given state x) by solving the optimization problem (11.2) with $\mathcal{P}_o$ as the risk envelope.

11.2.1 Example: Linear-Quadratic System

As a simple illustrative example to gain intuition for the convergence properties of Algorithm 6, consider a linear dynamical system with multiplicative uncertainty of the form $f(x_k, u_k, w_k) = A(w_k)x_k + B(w_k)u_k$. We consider the one-step decision-making process with a quadratic cost on state and action: $C := u_0^T R u_0 + x_1^T Q x_1$, where $x_1 = A(w_0)x_0 + B(w_0)u_0$. Here, $R \succ 0$ and $Q \succeq 0$. We consider a 10-dimensional state space with a 5-dimensional control input space. The number of realizations is taken to be $L = 3$ for ease of visualization. The L different $A(w_k)$ and $B(w_k)$ matrices corresponding to each realization are generated randomly by independently sampling elements of the matrices from the standard normal distribution $\mathcal{N}(0, 1)$. The cost matrix Q is a randomly generated positive semi-definite matrix and R is the identity. The initial states x^* were drawn randomly from the standard normal distribution $\mathcal{N}(0, I)$ where I denotes the identity matrix. The true envelope was generated by taking the convex hull of a set of random samples in the probability simplex Δ^L.

Figure 11.2 shows the outer approximations of the risk envelope obtained using Algorithm 6. We observe rapid convergence (approximately 20 sampled states x^*) of the outer approximations $\mathcal{P}_o$ (red) to the true risk envelope $\mathcal{P}$ (green). Figure 11.3 shows the mean squared error (on an independent test set with 30 demonstrations) between actions predicted using the sequentially refined polytope approximations generated by Algorithm 6 and the expert's true actions, as a function of the number of training demonstrations. One can observe rapid convergence in prediction performance after just 10 demonstration samples, further highlighting the data efficiency of the proposed algorithm.

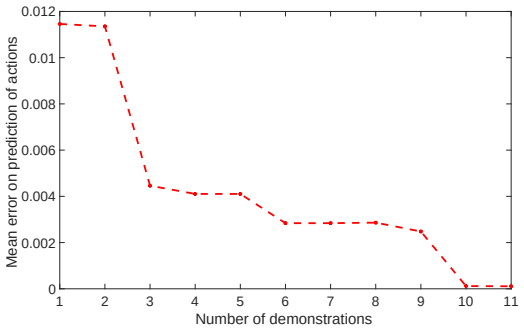
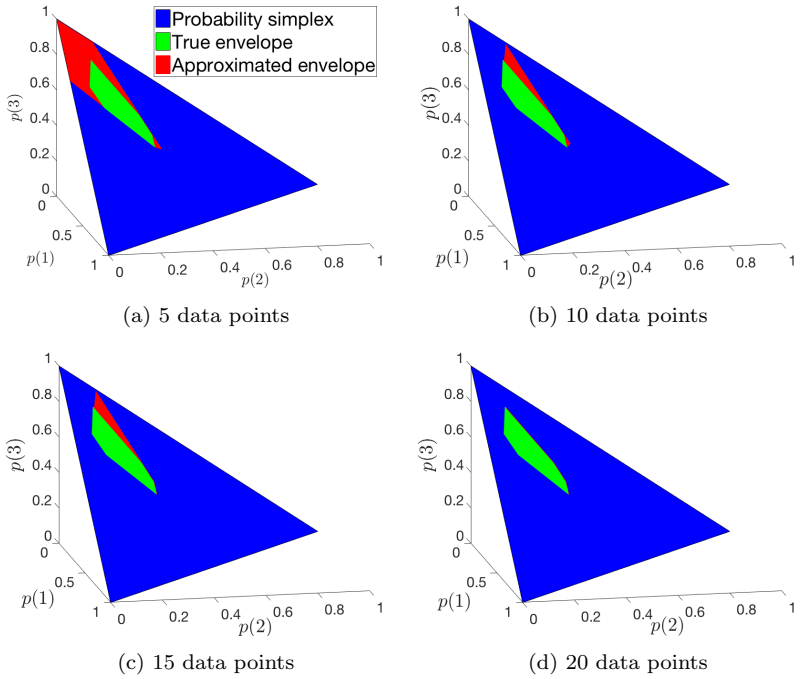


Figure 11.3: Rapid decrease of the mean squared error between predicted and expert's actions on an independent test set, as a function of the number of training demonstrations.

11.3 Unknown Cost Function

We next consider the more general case where both the expert's cost function $C(x_1, u_0)$ and risk measure are unknown. We parameterize the cost function as a linearly weighted combination of cost features, i.e.,

$$C(x_1, u_0) = c^T \phi(x_1, u_0),$$



(a) 5 data points (b) 10 data points

(c) 15 data points (d) 20 data points

Figure 11.2: Rapid convergence of the outer approximation of the risk envelope.

where $c \in \mathbb{R}^H_{\geq 0}$ is a vector of *unknown* weights and $\phi : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^H$ denotes the cost feature mapping from state and control input to an $H-$dimensional real vector. Since the solution of problem (11.2) solved by the expert is invariant to (i) constant shifts (Axiom A2 in Definition 3) and one can thus absorb negative signs into the cost features, and (ii) positive scalings (Axiom A3 in Definition 3), one can assume without loss of generality that the feature weights c are nonnegative and sum to one. Extending the notation in (11.3), let $\phi^{[j]}$ denote the feature vector when disturbance $w_0 = w^{[j]}$ is realized so that

$$g(x_0, u_0)(j) = c^T \phi^{[j]}(x_0, u_0), \quad j = 1, \ldots, L. \tag{11.13}$$

Thus, problem (11.4) now takes the form:

$$\begin{aligned} \min_{u_0 \in \mathcal{U}, \tau} \quad & \tau \\ s.t. \quad & \tau \geq \sum_{j=1}^{L} \sum_{h=1}^{H} v_i(j) c(h)\, \phi_h^{[j]}(x_0, u_0), \quad i \in \{1, \ldots, N_V\}. \end{aligned}$$

With this cost structure, we see that the KKT conditions derived in Section 11.2 now involve *products* of the feature weights c and the vertices v_i of $\mathcal{P}$. Thus, an analogous version of optimization problem (11.5) can be used to bound the optimal value. This problem will now contain products of the feature weights c and probability vector v. The key idea here is to introduce new *matrix* decision variables z

that replace each product $v(j)c(h)$ by a new variable z_{jh} which allows us to re-write problem (11.5) as an LP in (z, σ_+, σ_-), with the addition of the following two simple constraints: $0 \leq z_{jh} \leq 1$, for all j, h, and $\sum_{j,h} z_{jh} = 1$.

In a manner analogous to Theorem 12, this optimization problem allows us to obtain bounding hyperplanes *in the space of product variables* z which can then be aggregated as in Algorithm 6. Denoting this polytope as $\mathcal{P}_z$, we can then proceed to solve the "forward" problem (i.e., computing actions at a given state x) by solving the following optimization problem:

$$\min_{u_0 \in \mathcal{U}} \max_{z \in \mathcal{P}_z} \sum_{j,h} z_{jh} \phi_h^{[j]}(x_0, u_0). \tag{11.14}$$

This problem can be solved by enumerating the vertices of the polytope $\mathcal{P}_z$ in a manner similar to problem (11.4). Similar to the case where the cost function is known, this provides us with a way to conservatively approximate the expert's decision-making process (in the sense that we are considering a larger risk envelope).

11.3.1 Approximate Recovery of Cost and Risk Measure

While the procedure described above operates in the space of product variables z and does not require explicitly recovering the cost function and risk envelope separately, it may nevertheless be useful to do so for two reasons. First, the number of vertices of $\mathcal{P}_z$ may be quite large (since the space of product variables may be high dimensional) and thus solving the forward problem (11.14) may be computationally expensive. Recovering the cost and risk envelope separately allows us to solve a smaller optimization problem (since the risk envelope is lower dimensional in this case). Second, recovering the cost and risk measure separately may provide additional intuition and insights into the expert's decision-making process and may also allow us to make useful predictions in novel settings (e.g., where we expect the expert's risk measure to be the same but not the cost function or vice versa).

Here we describe a procedure for approximately recovering the feature weights and the risk envelope from the polytope $\mathcal{P}_z$. The key observation that makes this possible is to note that the matrix z containing the variables z_{jh} is equal to the outer product vc^T by definition. Hence, for $h = 1, \ldots, H$, we have:

$$\sum_{j=1}^{L} z_{jh} = \sum_{j=1}^{L} v(j)c(h) = c(h) \sum_{j=1}^{L} v(j) = c(h). \tag{11.15}$$

The last equality follows from the fact v is a probability vector and sums to 1. Similarly, for $j = 1, \ldots, L$, we have:

$$\sum_{h=1}^{H} z_{jh} = \sum_{h=1}^{H} v(j)c(h) = v(j) \sum_{h=1}^{H} c(h) = v(j). \tag{11.16}$$

The last equality follows from the fact that we assumed without loss of generality that the feature weights sum to 1.

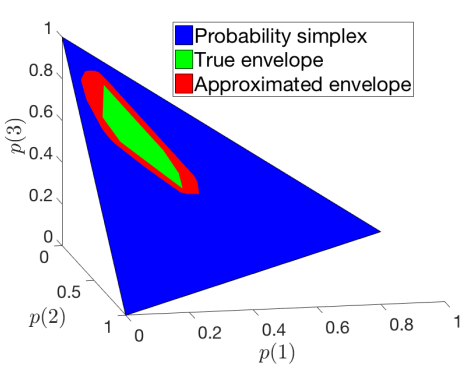
Let $\{\hat{z}_i\}$ be the set of (matrix-valued) vertices of the polytope $\mathcal{P}_z$. Then, by applying equations (11.15) and (11.16) to each vertex $\hat{z}_i$, we obtain a set of estimates of the feature weight vector c and a set of vectors in the probability simplex Δ^L, the convex hull of which gives an approximation of the risk envelope $\mathcal{P}$. If we have exactly recovered the polytope $\mathcal{P}_z$ in the space of product variables and the vertices $\hat{z}_i$ each have rank one, then it follows from problem (11.14) that the feature weight estimates will coincide and the convex hull of the probability vectors extracted from the vertices $\hat{z}_i$ will match the true risk envelope $\mathcal{P}$. In general however, this will not be the case since (i) there is no guarantee of exactly recovering the product polytope $\mathcal{P}_z$ (similar to how there is no guarantee of recovering the true risk envelope $\mathcal{P}$ in Algorithm 6), and (ii) $z = vc^T$ is a non-convex rank constraint that is not enforced in the KKT-based LP.

In light of these limitations, it is important to be able to gauge the quality of the estimates we obtain from the procedure above. We can do this in two ways. First, if the estimates of the weight vector are tightly clustered, this is a good indication that we have an accurate recovery. Second, if each vertex $\hat{z}_i$ of the polytope is close to a rank one matrix, then this is again a good indication (since the true product variables z equal vc^T).

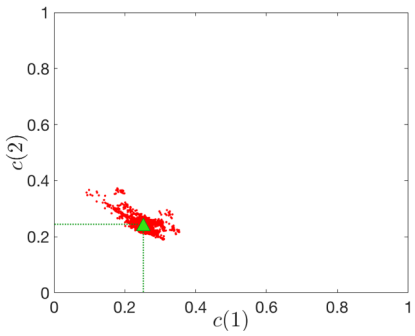
11.3.2 Example: Linear-Quadratic System

Consider the same system as in Section 11.2.1, but now we assume that the cost function is unknown. We take the true cost function as the weighted sum of three quadratic features (i.e., $H = 3$). The quadratic features are generated randomly by taking them to be equal to SS^T, where the elements of S are sampled from the standard normal distribution. The corresponding weights are drawn uniformly between 0 and 1 and are normalized to sum to 1.

Figure 11.4 a) illustrates the tightness of the approximate envelope as compared with the true polytope, while Figure 11.4 b) is a scatter plot of the first two feature weights (the third is uniquely determined given the first two) as recovered from applying eq. (11.15) to each vertex of the compound polytope $\mathcal{P}_z$. Notice that the cost feature weight estimates are tightly clustered near the true weights.



(a) Polytope estimate.



(b) Feature weight estimates (red), true weights marked by green triangle.

Figure 11.4: Approximated risk envelope and cost feature weights from 200 state-control pair demonstrations.

11.4 Summary

In this chapter we formulated a risk-sensitive single-step decision problem for the expert and derived IRL algorithms for estimating their cost function and CRM risk envelope. For the case where the cost function is known (e.g., for settings where this is set externally) but the risk measure is unknown, we derived a consistency result for the algorithm and proved that the learned model's prediction error converges to zero. In the next chapter, we extend our formulation to allow for the agent planning over multiple time-steps and multiple disturbance sampling events.

Appendix

In this appendix we provide a proof of consistency for Algorithm 6.

Appendix 11.A Proof of Algorithm Consistency

In order to prove the algorithm's consistency, i.e., Theorem 13, we need to establish a few intermediate results. Since $\mathcal{P}_\infty$ is an intersection of convex (respectively, compact) sets, it is also convex (respectively, compact). Moreover, since each $\mathcal{P}_D$ contains the expert's polytope $\mathcal{P}$, then $\mathcal{P}_\infty$ is also an outer approximation of $\mathcal{P}$. In particular, $\mathcal{P}_\infty$ is not empty.

Denote with d_H the Hausdorff distance between subsets of $\mathbb{R}^L$ associated with the Euclidean norm $||.||$, i.e., for two subsets A and B of $\mathbb{R}^L$,

$$d_H(A,B) := \max\left\{\sup_{b\in B}\inf_{a\in A}\|a-b\|,\ \sup_{a\in A}\inf_{b\in B}\|a-b\|\right\}. \tag{11.17}$$

The Hausdorff distance defines a metric on the set of non-empty compact subsets of $\mathbb{R}^L$, that we

use to measure the distance between risk envelopes. The sequence $\mathcal{P}_d$ is non-increasing (for set inclusion). Therefore, $\{d_H(\mathcal{P}_d, \mathcal{P}_\infty)\}_{d\geq 1}$ is a non-increasing sequence of non-negative real numbers. In particular, we have the following result:

Lemma 14 (Convergence in Hausdorff metric)**.** *The sequence $\{d_H(\mathcal{P}_d, \mathcal{P}_\infty)\}$ goes to 0 when $d \to \infty$.*

Lemma 15 (Compact uniform convergence)**.** *Consider a sequence $\{\mathcal{Q}_n\}_{n\geq 1}$ of compact convex subsets of Δ^L such that for all $n \geq 1$, $\mathcal{P}_\infty \subseteq \mathcal{Q}_n$ and $\lim_{n\to\infty} d_H(\mathcal{Q}_n, \mathcal{P}_\infty) = 0$. Consider also a sequence of states $\{x_n\}_{n\geq 1}$ such that $x_n \to x^*$ when $n \to \infty$. Define the functions $\varphi_n(u) := \max_{v\in\mathcal{Q}_n} v^T g(x_n, u)$, $\varphi_{n,\infty}(u) := \max_{v\in\mathcal{P}_\infty} v^T g(x_n, u)$ and $\varphi(u) := \max_{v\in\mathcal{P}_\infty} v^T g(x^*, u)$. Then, for any compact set $\mathcal{K} \subseteq \mathcal{U}$:*

$$\lim_{n\to\infty} \sup_{u\in\mathcal{K}} |\varphi_n(u) - \varphi(u)| = 0. \tag{11.18}$$

Proof. Fix $u \in \mathcal{U}$. For any $n \geq 1$, denote with $v_n \in \mathcal{Q}_n$ a point such that $\varphi_n(u) = v_n^T g(x_n, u)$, and with $v_{n,\infty} \in \mathcal{P}_\infty$ a point such that $\varphi_{n,\infty}(u) = v_{n,\infty}^T g(x_n, u)$. Let Γ_∞ be the projection operator onto the compact convex set $\mathcal{P}_\infty$. Then,

$$\Gamma_\infty(v_n)^T g(x_n, u) \leq v_{n,\infty}^T g(x_n, u) \leq v_n^T g(x_n, u). \tag{11.19}$$

The second inequality in (11.19) results from the fact that $\mathcal{P}_\infty \subseteq \mathcal{Q}_n$. By Cauchy-Schwarz inequality,

$$|(\Gamma_\infty(v_n) - v_n)^T g(x_n, u)| \leq ||\Gamma_\infty(v_n) - v_n||_2 ||g(x_n, u)||_2. \tag{11.20}$$

Since $||\Gamma_\infty(v_n) - v_n||_2 \leq d_H(\mathcal{Q}_n, \mathcal{P}_\infty)$ and $\lim_{n\to\infty} x_n = x^*$, we get that the left-hand side (LHS) in (11.20) tends to 0 when $n \to \infty$, which implies that

$$\lim_{n\to\infty} \varphi_n(u) - \varphi_{n,\infty}(u) = 0. \tag{11.21}$$

Since g is continuous with respect to the state variable x and $\mathcal{P}_\infty$ is compact, we get that $x \mapsto \max_{v\in\mathcal{P}_\infty} v^T g(x, u)$ is also continuous with respect to x. Therefore, $\lim_{n\to\infty} \varphi_{n,\infty}(u) = \varphi(u)$. Using (11.21), $\lim_{n\to\infty} \varphi_n(u) = \varphi(u)$. But, by Theorem 10.8 in (Rockafellar, 2007), pointwise convergence of a sequence of convex functions over $\mathcal{U}$ implies uniform convergence over any compact set $\mathcal{K} \subseteq \mathcal{U}$, which is the desired result. □

Since we assumed the cost functions to be strictly convex with respect to u, the risk-sensitive optimization problem admits a unique optimal control.

Lemma 16 (Strict convexity of risk)**.** *For any compact subset $\mathcal{B} \subseteq \Delta^L$ and at any state x, the function $u \mapsto \max_{v\in\mathcal{B}} v^T g(x, u)$ is strictly convex. In particular, it admits a unique minimizer.*

Proof. Fix a state x. Take $u_1, u_2 \in \mathcal{U}$ and $\alpha \in [0, 1]$. Denote $u_\alpha = (1-\alpha)u_1 + \alpha u_2$. Since $\mathcal{B}$ is compact, there exists $\bar{v} \in \mathcal{B}$ such that: $\bar{v}^T g(x, u_\alpha) = \max_{v\in\mathcal{B}} v^T g(x, u_\alpha)$. By strict convexity

of $u \mapsto \bar{v}^T g(x,u)$, it follows that: $\bar{v}^T g(x,u_\alpha) < (1-\alpha)\bar{v}^T g(x,u_1) + \alpha \bar{v}^T g(x,u_2)$. Taking the worst-case for both terms of the previous inequality's right-hand side, we get that: $\bar{v}^T g(x,u_\alpha) < (1-\alpha)\max_{v\in\mathcal{B}} v^T g(x,u_1) + \alpha \max_{v\in\mathcal{B}} v^T g(x,u_2)$, which proves strict convexity of the function $u \mapsto \max_{v\in\mathcal{B}} v^T g(x,u)$. Since the latter function has, by assumption, bounded level sets, it admits a minimizer, which is unique by strict convexity. □

Lemma 17 (Optimal control convergence)**.** *Define the sequence of functions φ_n and φ as in Lemma 15. Denote $u_n := \operatorname{argmin}_{u\in\mathcal{U}} \varphi_n(u)$ and $u^* := \operatorname{argmin}_{u\in\mathcal{U}} \varphi(u)$ (each of these minima are unique by strict convexity). Then:*

$$\lim_{n\to\infty} u_n = u^*. \tag{11.22}$$

Proof. Let $\tau_n := \min_{u\in\mathcal{U}} \varphi_n(u)$ and $\tau := \min_{u\in\mathcal{U}} \varphi(u)$. By construction, the control set $\mathcal{U}$ is a compact convex set. Thus, by Lemma 15, the function φ_n converges to φ uniformly over $\mathcal{U}$. Thus, for any given $\epsilon > 0$, there exists an $n_0(\epsilon) \in \mathbb{N}$ such that for all $n > n_0(\epsilon)$,

$$|\varphi_n(u) - \varphi(u)| \leq \epsilon \quad \forall u \in \mathcal{U}.$$

It follows that for $n > n_0(\epsilon)$, we have:

$$\begin{aligned} |\varphi_n(u^*) - \varphi(u^*)| &= |\varphi_n(u^*) - \tau| \leq \epsilon \\ |\varphi_n(u_n) - \varphi(u_n)| &= |\tau_n - \varphi(u_n)| \leq \epsilon. \end{aligned} \tag{11.23}$$

Furthermore,

$$\tau_n - \varphi(u_n) \leq \tau_n - \tau \leq \varphi_n(u^*) - \tau,$$

since $\varphi(u) \geq \tau$ for all $u \in \mathcal{U}$ and $\tau_n \leq \varphi_n(u)$ for all $u \in \mathcal{U}$. Combining this with eq. (11.23), we have:

$$-\epsilon \leq \tau_n - \tau \leq \epsilon, \quad \forall n > n_0(\epsilon).$$

Thus, $\tau_n \to \tau$ as $n \to \infty$.

We proceed by contradiction to prove (11.22). Assume that $\{u_n\}_{n\geq 1}$ does not converge to u^*. Without loss of generality, assume that there exists some $\eta > 0$ such that for all n, $||u_n - u^*|| \geq \eta$. Define u'_n and $\alpha_n \in [0,1]$ such that $u'_n = \alpha_n u_n + (1-\alpha_n)u^*$, and $||u^* - u'_n|| = \frac{\eta}{2}$. By convexity of φ_n:

$$\varphi_n(u'_n) \leq \alpha_n \varphi_n(u_n) + (1-\alpha_n)\varphi_n(u^*) = \alpha_n \tau_n + (1-\alpha_n)\varphi_n(u^*). \tag{11.24}$$

By the pointwise convergence property of φ_n, we have that $\lim_{n\to\infty} \varphi_n(u^*) = \varphi(u^*) = \tau$. Furthermore, we have also established that $\lim_{n\to\infty} \tau_n = \tau^*$. Thus, the RHS in eq. (11.24) converges to $\varphi(u^*)$. For the LHS, assume that the sequence $\{u'_n\}_{n\geq 1}$ converges to u' (or consider a converging

subsequence, which is allowed since $\mathcal{U}$ is compact). Then, by continuity of φ and uniform convergence of φ_n, it follows that: $\lim_{n\to\infty} \varphi_n(u'_n) = \varphi(u')$. Using (11.24), we get: $\varphi(u') \leq \varphi(u^*)$. But $||u' - u^*|| = \frac{\eta}{2}$ and by uniqueness of the minimum, this is a contradiction. □

We are now ready to prove Theorem 13.

Proof. (Theorem 13). Fix any $x^* \in \mathcal{S}$ and choose a subsequence of $\{\mathcal{P}_d\}_{d\geq 1}$ (that we still denote $\{\mathcal{P}_d\}$ for simplicity) such that $x^{*,d} \to x^*$. For any $d \geq 1$ and corresponding demonstration $(x^{*,d}, u^{*,d})$, according to the update of the outer approximation $\mathcal{P}_d$ in Algorithm 6,

$$u(\mathcal{P}_d, x^{*,d}) = u(\mathcal{P}, x^{*,d}). \tag{11.25}$$

We justify (11.25). Given the demonstration pair $(x^{*,d}, u^{*,d})$ where by definition, $u^{*,d} = u(\mathcal{P}, x^{*,d})$, let $\bar{v} \in \mathcal{P}_{d-1}$, $\bar{\sigma}_+$, $\bar{\sigma}_-$ be solutions of:

$$\begin{aligned} \max_{\substack{v\in\mathcal{P}_{d-1}\\ \sigma_+,\sigma_-\geq 0}} \quad & g(x^{*,d}, u^{*,d})^T v \\ s.t. \quad & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^{*,d},u^{*,d}} + \sigma_+(j), \forall j \in \mathcal{J}^+ \\ & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^{*,d},u^{*,d}} - \sigma_-(j), \forall j \in \mathcal{J}^- \\ & 0 = \nabla_{u(j)} g(x,u)^T v\big|_{x^{*,d},u^{*,d}}, \forall j \notin \mathcal{J}^+, j \notin \mathcal{J}^- \\ & \sigma_+(j) = 0,\ \sigma_-(j) = 0, \quad \forall j \notin \mathcal{J}^+, j \notin \mathcal{J}^- \end{aligned} \tag{11.26}$$

where $\mathcal{J}^+ = \{j \in \{1,\dots,m\} \,|\, u^{*,d}(j) = u^+(j)\}$ and $\mathcal{J}^- = \{j \in \{1,\dots,m\} \,|\, u^{*,d}(j) = u^-(j)\}$. According to Algorithm 6, $\mathcal{P}_d = \{v \in \mathcal{P}_{d-1} \,|\, v^T g(x^{*,d}, u^{*,d}) \leq \bar{v}^T g(x^{*,d}, u^{*,d})\}$. Moreover, $\bar{v} \in \mathcal{P}_d$, which implies:

$$\max_{v\in\mathcal{P}_d} v^T g(x^{*,d}, u^{*,d}) = \bar{v}^T g(x^{*,d}, u^{*,d}). \tag{11.27}$$

The set of equations given by the constraints of Problem (11.26), with v, σ_+ and σ_- fixed and respectively equal to $\bar{v}$, $\bar{\sigma}_+$ and $\bar{\sigma}_-$, are exactly the optimality conditions for the solution of the following convex optimization problem:

$$\min_{u\in\mathcal{U}} \bar{v}^T g(x^{*,d}, u). \tag{11.28}$$

Since $u^{*,d}$ satisfies those optimality conditions and using (11.27), we have:

$$\begin{aligned}\max_{v\in\mathcal{P}_d} v^T g(x^{*,d},u^{*,d}) &= \bar{v}^T g(x^{*,d},u^{*,d}) \\ &= \min_{u\in\mathcal{U}} \bar{v}^T g(x^{*,d},u) \\ &\leq \min_{u\in\mathcal{U}} \max_{v\in\mathcal{P}_d} v^T g(x^{*,d},u) \\ &\leq \max_{v\in\mathcal{P}_d} v^T g(x^{*,d},u^{*,d})\end{aligned} \tag{11.29}$$

Hence, all inequalities in (11.29) are equalities. In particular,

$$\min_{u\in\mathcal{U}} \max_{v\in\mathcal{P}_d} v^T g(x^{*,d},u) = \max_{v\in\mathcal{P}_d} v^T g(x^{*,d},u^{*,d}) \tag{11.30}$$

. By uniqueness of the minimum as given by Lemma 16, it follows that $u^{*,d} = u(\mathcal{P}_d, x^{*,d})$. From Lemma 17, we have that $\lim_{d\to\infty} u\left(\mathcal{P}_d, x^{*,d}\right) = u\left(\mathcal{P}_\infty, x^*\right)$. From equation (11.25), we get that $\lim_{d\to\infty} u\left(\mathcal{P}_d, x^{*,d}\right) = u\left(\mathcal{P}, x^*\right)$. Combining the two previous observations, we get the desired result, i.e., $u\left(\mathcal{P}, x^*\right) = u\left(\mathcal{P}_\infty, x^*\right)$. □

Chapter 12

Dynamic Risk-Sensitive IRL

In this chapter, we generalize the single-step decision problem to the multi-step setting. We consider a model where the disturbance w_k is sampled every $N > 1$ time-steps and held constant in the interim. Such a model generalizes settings where disturbances are sampled i.i.d. at every time-step (corresponding to $N = 1$ in our model) and it allows us to model delays in the expert's reaction to changing disturbances. We model the expert as planning in a *receding horizon* manner by looking ahead for a finite horizon longer than N steps, executing the computed policy for N steps, and iterating. Owing to the need to account for future disturbances, the multi-step finite-horizon problem is a search over control *policies* (i.e., the executed control inputs depend on which disturbance is realized).

12.1 Prepare-React Model

In this section we outline the "prepare" – "react" model introduced in (Majumdar et al., 2017), and depicted in Figure 12.1. The expert's policy is decomposed into two phases (shown in Figure 12.1), referred to as "prepare" and "react." Intuitively, this model captures the idea that in the period preceding a disturbance (i.e., the "prepare" phase) the expert controls the system to a state from which he/she can recover well (in the "react" phase) once a disturbance is realized. Studies showing that humans have a relatively short look-ahead horizon in uncertain decision-making settings lend credence to such a model (Carton et al., 2016). As in (Majumdar et al., 2017), the delay parameter n_d would be learned directly from the demonstrations. To account for nested re-planning stages, we will leverage the theory of *dynamic* risk measures, introduced in Section 10.2.2, to assess risk over sequential realizations of uncertainty.

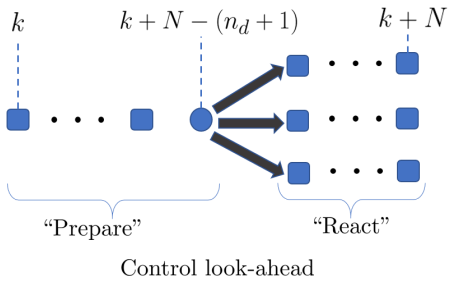


Figure 12.1: Scenario tree centered on a disturbance sampled at time $k+N-(n_d-1)$, where $N > n_d$. The "prepare" phase precedes each disturbance re-sample event by $N - n_d$ steps, while the "react" phase follows it by n_d steps; in total, an individual planning stage last N steps. Note that during the prepare and react phases the dynamics are deterministic, as we assume that the disturbances stay constant for N steps in between sampling times. As we model the expert as a receding horizon planner, the expert's planning problem at time k would account for nested re-planning stages at time $k+N, k+2N, \ldots$ up to some look-ahead horizon. The expert would then execute their policy for the first N steps and then resolve the planning problem at time $k+N$ with a receded horizon.

12.1.1 Formal Definition

We are now ready to formally define the expert's multi-step problem, from the perspective of time-step k, with look-ahead horizon $T\,N$ steps, where $T \in \mathbb{N}_{\geq 1}$ denotes the number of *branching events* (i.e., disturbance samples) within the prediction horizon. According to the prepare-react model introduced earlier, we assume that the disturbance mode for the first $N - n_d$ steps starting at time-step k corresponds to w_{k-n_d-1} (i.e., the disturbance mode realized at the last sampling event), following which the disturbance is re-sampled every N steps. An illustration of the nested prepare-react model with a look-ahead horizon $T = 2$ is provided in Figure 12.2.

Let $x_{k'|k}$ denote the predicted state for time-step $k+k'$, where $k' \in [0,\, TN-1]$, as predicted within the expert's multi-step optimization problem defined at time-step k. Similarly, let $\{w'_t\}, t \in [0, T-1]$ represent the predicted disturbance sequence, where each $w'_t \in \{1, \ldots, L\}$. Let $\hat{\pi}_t(\boldsymbol{\omega}_{t-1}, w'_t),\ t \in [0, T-1]$ denote the expert's *"prepare" – "react" control policy* for stage t (corresponding to time-steps $k' \in [tN,\, (t+1)N-1]$), where we make explicit the dependency on the partial predicted disturbance history $\boldsymbol{\omega}_{t-1} := \{w'_{-1}, w'_0, \ldots, w'_{t-1}\}$ and the next predicted disturbance mode w'_t, while we omit in the interest of brevity the dependency on x_k and partial policy history $\{\hat{\pi}_0, \ldots, \hat{\pi}_{t-1}\}$. We take $w'_{-1} =: w^*_{-1|k}$ to represent the actual disturbance mode in progress at the time of solving the multi-stage optimization problem at time-step k. Note that, by causality, only the "react" portion of $\hat{\pi}_t$ may be a function of w'_t but not the "prepare" portion. Finally, let $C_{tN:(t+1)N-1}(x_{tN|k}, \hat{\pi}_t(\cdot))$ denote the accumulated cost (i.e., a random variable adapted to the filtration generated by the sequence $\{w'_t\}$) over time-steps $k' \in [tN, (t+1)N-1]$ given the "prepare" – "react" control policy $\hat{\pi}_t$ at stage t, that is,

$$C_{tN:(t+1)N-1}(x_{tN|k}, \hat{\pi}_t(\cdot)) = \sum_{k'=tN}^{(t+1)N-1} C(x_{k'|k}, \hat{\pi}_t(x_{k'|k})).$$

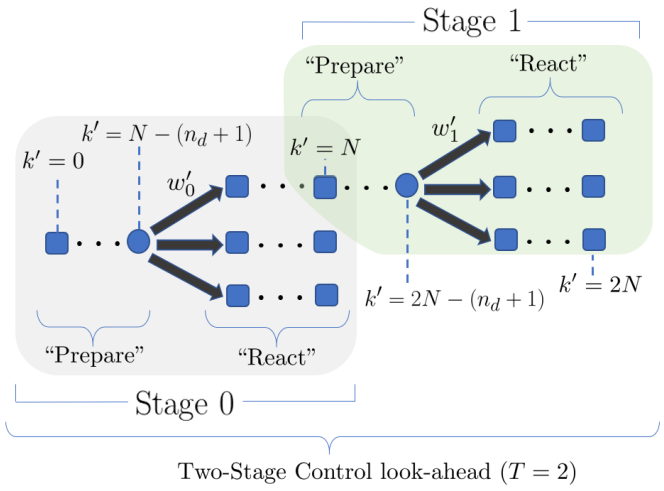


Figure 12.2: Multi-stage scenario tree for the prepare-react multi-step problem at time k (indexed internally using k' for simplicity). The disturbance is sampled every N steps. The control look-ahead consists of multiple nested branches of "prepare" and "react" sequences (indexed as "stages" in the figure above); shaded green in the figure above is one such nested branch corresponding to $w'_0 = w^{[1]}$. To evaluate costs over stage 1, it is assumed that the expert is using the conditional CRM $\rho_1(\cdot)$ where for each realization of $x_{N|k}$ (identified uniquely by the observed disturbance branch at stage 0), the expert uses the static CRM $\rho(\cdot)$ over the nested outcomes (shown in green for one possible realization of $x_{N|k}$). The observed control sequence is the beginning "prepare" – "react" sequence corresponding to the actual realized disturbance $w^*_{0|k}$.

As in Section 11.3, we assume that the time-step cost $C(x, u)$ is represented by a linear combination $c^T\phi(x, u)$ of features $\phi(x, u)$. The expert's multi-step optimization problem is then given as:

$$\min_{\substack{\hat{\pi}_t \\ t\in[0,T-1]}} \rho_0\Big(C_{0:N-1}(\cdot,\hat{\pi}_0) + \rho_1(C_{N:2N-1}(\cdot,\hat{\pi}_1) + \cdots + \rho_{T-1}(C_{(T-1)N:TN-1}(\cdot,\hat{\pi}_{T-1}))\cdots)\Big), \quad (12.1)$$

where each ρ_t, $t = 0, \ldots, T-1$ is a coherent one-step *conditional* risk measure such that each *component* of ρ_t is a CRM $\rho(\cdot)$ with respect to the probability space $(\mathcal{W}, 2^{\mathcal{W}}, p)$ and characterized by *the fixed risk envelope* $\mathcal{P} \subseteq \Delta^L$. Leveraging the translational invariance property, the objective may be equivalently re-written as

$$\begin{aligned}&\varrho(C_{0:TN-1}) := \rho_0 \circ \cdots \circ \rho_{T-1}(C_{0:TN-1})\\ =&C_{0:N-n_d-1} + \rho_0\Big(C_{N-n_d:N-1} + C_{N:2N-n_d-1} + \rho_1(C_{2N-n_d:2N-1} + \cdots + \rho_{T-1}\left(C_{TN-n_d:TN-1}\right)\cdots)\Big).\end{aligned}$$

One should notice that (1) for each $t \in \{0, \ldots, T-1\}$, the cost sequence $C_{tN:(t+1)N-1}$ is split across the risk operator ρ_t due to the "prepare"–"react" structure and the translational invariance property, (2) the risk mapping is over accumulated costs, i.e., in the notation of eq. (10.5), the stage t random cost Z_t corresponds to the accumulated cost $C_{tN:(t+1)N-1}$ since disturbances are sampled every N steps, and (3) since problem (12.1) is solved in receding horizon fashion and thus x_k is *known* at time k, $\varrho(\cdot)$ is a real valued *function*. The observed input from the expert is the first

stage optimal "prepare" – "react" control policy $\hat{\pi}_0^*(w^*_{-1|k}, w^*_{0|k})$ where $w^*_{0|k}$ represents the actual disturbance mode sampled immediately after time-step k, following which the expert re-solves the problem at time $k+N$, with a receded horizon up to time-step $k+N+TN$.

Notice that by setting $T=1$, one recovers the single-stage "prepare – react" model presented in (Majumdar et al., 2017). The strategy in (Majumdar et al., 2017) is to reduce the multi-step inference problem to a mathematically equivalent single-step problem by estimating the (un-observed) control policies of the human agent, corresponding to the *un-realized* disturbance branches. Specifically, consider the scenario tree decomposition in Figure 12.1. If disturbance $w^{[3]}$ is realized at time-step $k+N-n_d$, then we only observe the "react" control sequence corresponding to the third branch. The algorithm in (Majumdar et al., 2017) proceeded by first inferring the "react" control sequences for the un-observed branches and then constructing a bounding hyperplane using a similar version of problem (11.5). In a multiple-stage setting, however, it is exceedingly difficult to exactly infer (or approximate) the unobserved control policies as each of these policies involves an *unobserved* nested optimization over future branching events. Consequently, the optimality conditions of an observed control policy are defined by equalities that are non-linear in the unobserved variables. Therefore, extending the use of KKT conditions to infer an outer approximation of the risk envelope in the style of Theorem 12 leads to an *intractable* non-convex optimization problem. To address this fundamental observability issue, we introduce a *semi-parametric* representation of the CRM, discussed next.

12.2 Semi-Parametric CRM

Fix a set of M normal vectors $a_j \in \mathbb{R}^L, j=1,\dots,M$. Let $\mathcal{P}_r$ denote the polytope defined by the halfspace constraints

$$\mathcal{P}_r := \{v \in \Delta^L \mid a_j^T v \leq b(j) - r(j), \quad j=1,\dots,M\}, \tag{12.2}$$

where for each j, $b(j) := \max_{v\in\Delta^L} a_j^T v$, and r is a parameter vector in $\mathbb{R}^M_{\geq 0}$. The CRM with risk envelope $\mathcal{P}_r$ is denoted as $\rho^r(\cdot)$; explicitly,

$$\rho^r(Z) = \max_{v\in\mathcal{P}_r} \mathbb{E}_v[Z], \tag{12.3}$$

where $Z \in \mathbb{R}^L$ is a discrete random variable with L possible realizations (see Figure 12.3).

This induced CRM is termed *semi*-parametric since unlike methods where one seeks to find the parameters defining a *fixed* disutility function, e.g., as in (Shen et al., 2014; Ratliff and Mazumdar, 2017), here we *do not* assume a fixed chosen risk measure. Instead, by parameterizing the risk measure in the dual space (via its risk envelope characterization), we retain the generality to recover any polytopic CRM, given a sufficient number of normal vectors a_j. A potential method to choose

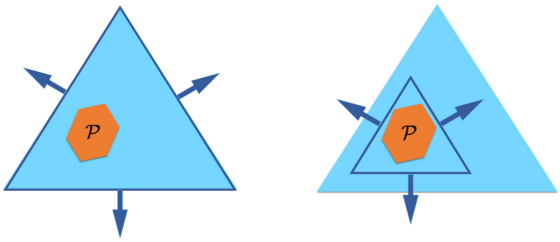


Figure 12.3: Schematic illustration of a semi-parametric CRM for a 3-scenario outcome space. The true risk envelope $\mathcal{P}$ is shown in orange; the boundary of the approximation polytope $\mathcal{P}_r$ is shown in dark blue. Left: the polytope $\mathcal{P}_r$ with $r = 0$. Right: the polytope $\mathcal{P}_r$ for some $r \in \mathbb{R}^M_{>0}$. The arrows denote the a priori fixed normal vectors $\{a_j\}_{j=1}^3$.

the normal vectors a_j is to take the halfplane normals from the multi-step KKT method described in (Majumdar et al., 2017).

In order to ensure that the polytope $\mathcal{P}_r$ defined in eq. (12.2) is non-empty, we define the extended polytope

$$\tilde{\mathcal{P}}_r := \{(v, r) \in \Delta^L \times \mathbb{R}^M \mid a_j^T v + r(j) \leq b(j), \quad j = 1, \ldots, M\}.$$

Define $\mathcal{R} := \text{proj}_r \tilde{\mathcal{P}}_r$ as the projection of polytope $\tilde{\mathcal{P}}_r$ along the r variables. Then, $r \in \mathcal{R}$ ensures that the polytope $\mathcal{P}_r$ is non-empty. It is readily observed that the set $\mathcal{R}$ is also a polytope.

12.2.1 Constrained Maximum Likelihood

Given the semi-parametric representation of the CRM in (12.3), the RS-IRL problem reduces to inference over the offset vector r, and cost feature weight vector c. We will perform this inference using a constrained maximum likelihood model. Consider, first, a likelihood model inspired by the MaxEnt IRL framework (Ziebart et al., 2008) where we assume

$$\Pr(\hat{\pi}_0(w^*_{-1|k}, w^*_{0|k})) \propto \exp\left(-\tau[w^*_{-1|k}, w^*_{0|k}]\right), \tag{12.4}$$

where, $\tau[w^*_{-1|k}, w^*_{0|k}]$ is the optimal value of (12.1), computed using the semi-parametric CRM defined in (12.3) and *conditioned* on $w'_0 = w^*_{0|k}$ and $\hat{\pi}_0 = \hat{\pi}_0(w^*_{-1|k}, w^*_{0|k})$ (see Appendix 12.A at the end of this chapter for a detailed derivation of this distribution). While the original MaxEnt IRL model is motivated by finding the maximum entropy distribution subject to an expected feature matching constraint, the robust performance of MaxEnt IRL even in the absence of such a statistical motivation has been extensively observed and leveraged in the IRL literature, particularly in the context of noisy or suboptimal demonstrations.

A key limitation of the MaxEnt model, however, lies in the complexity of estimating the partition function (normalization factor for the distribution in eq. (12.4)) and its gradients. The likelihood model in eq. (12.4) represents a distribution over all possible N-length *policies*. This makes sampling-based approximations intractable, as similarly observed in (Kretzschmar et al., 2016), and Laplace

integral-based approximations as used in (Levine and Koltun, 2012) too imprecise.

In order to construct a tractable algorithm, we employ the simplification whereby at the beginning of any "prepare" stage (see Figure 12.2), the expert can only choose an *open-loop* control *trajectory* $\hat{u}$ of length N from a finite set of such trajectories $\prod$, thereby eliminating the notion of a "react" *policy* and replacing it with an open-loop sequence spanning the entire "prepare" – "react" stage. The set $\prod$ of these trajectories can be chosen for instance by running the K-Means clustering algorithm on the raw input trajectories. This simplification allows us to interpret problem (12.1) as a game between the expert with action set $\prod$ and nature with action set $\mathcal{W}$, and uniquely identify any predicted state $x_{tN|k}, t \in [1, T-1]$ using the predicted game history, i.e., disturbance history $\boldsymbol{\omega}_{t-1}$ and control history $\boldsymbol{u}_{t-1} := \{\hat{u}_0, \ldots, \hat{u}_{t-1}\}$. Leveraging such a discrete representation and dynamic programming, one can then construct the optimal solution to the expert's multi-stage optimization problem using a "risk-sensitive" Bellman recursion, defined below.

Terminal Stage: For all possible game histories up to stage $T-1$, define

$$\begin{aligned}\tau[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u}) &:= \rho\left(C_{(T-1)N:TN-1}(x_{(T-1)N|k}, \hat{u})\right)\\ \hat{\pi}^*_{T-1}[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}] &:= \operatorname*{argmin}_{\hat{u}\in\prod} \tau[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u}).\end{aligned}$$

Recursion: For all possible game histories up to stage t, for $t = T-2, \ldots, 1$:

$$\begin{aligned}\tau[\boldsymbol{u}_{t-1}, \boldsymbol{\omega}_{t-1}](\hat{u}) &:= \rho\left(C_{tN:(t+1)N-1}(x_{tN|k}, \hat{u}) + \min_{\hat{u}'\in\prod} \tau[\{\boldsymbol{u}_{t-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t-1}, w'_t\}](\hat{u}')\right)\\ \hat{\pi}^*_t[\boldsymbol{u}_{t-1}, \boldsymbol{\omega}_{t-1}] &:= \operatorname*{argmin}_{\hat{u}\in\prod} \tau[\boldsymbol{u}_{t-1}, \boldsymbol{\omega}_{t-1}](\hat{u}).\end{aligned}$$

First Stage:

$$\begin{aligned}\tau[w^*_{-1|k}](\hat{u}) &:= \rho\left(C_{0:N-1}(x_k, \hat{u}) + \min_{\hat{u}'\in\prod} \tau[\{\hat{u}\}, \{w^*_{-1|k}, w'_0\}](\hat{u}')\right)\\ \hat{\pi}^*_0[w^*_{-1|k}] &:= \operatorname*{argmin}_{\hat{u}\in\prod} \tau[w^*_{-1|k}]](\hat{u}).\end{aligned}$$

The value $\min_{\hat{u}\in\prod} \tau[w^*_{-1|k}](\hat{u})$ is the optimal value of problem (12.1) given this discrete set parameterization for each stage policy. In the equations above, it is understood that for each $t \in [0, T-1]$, the accumulated cost $C_{tN:(t+1)N-1}$ is evaluated based on the previous disturbance mode w'_{t-1} for the first $N - n_d$ steps, followed by w'_t for the remaining n_d steps.

Given the structure of the optimal solution of problem (12.1), presented in Bellman form above using the true CRM $\rho(\cdot)$, we now construct a *computationally tractable* likelihood model for the parameters r and c by defining the *soft* risk-sensitive Bellman recursion using the semi-parametric

CRM $\rho^r(\cdot)$. For the terminal stage, define

$$\tilde{\tau}[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u}) := \rho^r \left(C_{(T-1)N:TN-1}(x_{(T-1)N|k}, \hat{u}) \right); \tag{12.5}$$

for all $t = T-2, \ldots, 1$, define:

$$\tilde{\tau}[\boldsymbol{u}_{t-1}, \boldsymbol{\omega}_{t-1}](\hat{u}) := \rho^r \left(C_{tN:(t+1)N-1}(x_{tN|k}, \hat{u}) + \underset{\hat{u}' \in \prod}{\text{softmin}}\, \tilde{\tau}[\{\boldsymbol{u}_{t-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t-1}, w'_t\}](\hat{u}') \right); \tag{12.6}$$

and finally, for the first stage, define:

$$\tilde{\tau}[w^*_{-1|k}](\hat{u}) := \rho^r \left(C_{0:N-1}(x_k, \hat{u}) + \underset{\hat{u}' \in \prod}{\text{softmin}}\, \tilde{\tau}[\{\hat{u}\}, \{w^*_{-1|k}, w'_0\}](\hat{u}') \right), \tag{12.7}$$

where $\text{softmin}_x f(x) := -\log \sum_x \exp(-f(x))$.

Let $\hat{u}^*_t$ be the closest (in $\mathcal{L}_2$ norm) trajectory in $\prod$ to the observed control sequence over time-steps[1] $[tN, (t+1)N-1]$. Similar to the MaxEnt IRL approach, we allow for imperfect human demonstrations by postulating that lower risk-sensitive cost actions are exponentially preferred, i.e.,

$$\Pr(\hat{u}) \propto \exp\left(-\beta \tilde{\tau}[w^*_{-1|tN}](\hat{u}) \right), \tag{12.8}$$

where $\beta > 0$ is an inverse temperature parameter. Thus, the likelihood of parameters r, c is given by:

$$l(r, c|\hat{u}^*_t) := \frac{\exp\left(-\beta \tilde{\tau}[w^*_{-1|tN}](\hat{u}^*_t) \right)}{\sum_{\hat{u} \in \prod} \exp\left(-\beta \tilde{\tau}[w^*_{-1|tN}](\hat{u}) \right)}. \tag{12.9}$$

As the expert is assumed to solve the problem in a receding horizon fashion, we may treat each $(w^*_{-1|tN}, \hat{u}^*_t)$ tuple in the demonstrated trajectory $\mathcal{T}^*$ independently. Consequently, the log likelihood given the entire trajectory is simply

$$l(r, c|\mathcal{T}^*) = \frac{1}{|\mathcal{T}^*|} \sum_{\hat{u}^*_t \in \mathcal{T}^*} -\beta \tilde{\tau}[w^*_{-1|tN}](\hat{u}^*_t) + \underset{\hat{u}}{\text{softmin}}\, \beta \tilde{\tau}[w^*_{-1|tN}](\hat{u}), \tag{12.10}$$

where $|\mathcal{T}^*|$ is the number of N-step demonstrations in the trajectory $\mathcal{T}^*$. The inference problem is then

$$\{r^*, c^*\} := \underset{c \in \Delta^H,\ r \in \mathcal{R}}{\text{argmax}}\ l(r, c|\mathcal{T}^*), \tag{12.11}$$

where, as before, we assume that the cost weights are non-negative and sum to one (and thus lie in the simplex Δ^H). We solve the problem using projected gradient descent on r and entropic mirror descent on c. The gradient formulas are derived by propagating gradients of the $\tilde{\tau}$ variables in

[1] We use t here for notational consistency between the stage-wise decomposition of the multi-step problem and demonstrated action trajectories.

recursive fashion from the terminal to the first stage (similar to the computation of $\tilde{\tau}$ itself) and leveraging LP sensitivity results. For ease of exposition, we provide these recursive formulas at the end of the chapter.

Remark 13. *While we lose the outer-approximation of the risk envelope and convergence guarantees associated with the KKT method, in its place we obtain a tractable algorithm that enables us to accommodate a substantially larger class of dynamic decision-making inference problems. Experimental results, as discussed in Section 9.1.2, confirm that the method works well in approximating a wide range of risk profiles.*

12.3 Example: Driving Game Scenario

In this section we apply our RS-IRL framework on a simulated driving game (Figure 12.4) with ten human participants to demonstrate that our approach is able to infer individuals' varying attitudes toward risk and mimic the resulting driving styles. In particular, we note that the experimental setting here constitutes a significantly more challenging and dynamic testbed than typical benchmark examples such as grid-world or sequential investment tasks.

(a) Visualization of simulator during the interactive game experiment as seen by participant.

(b) Logitech G29 game input hardware consists of a force-feedback steering wheel and accelerator and brake pedals.

Figure 12.4: The simulated driving game considered in this chapter. The human controls the follower car using a force-feedback steering wheel and two pedals, and must follow the leader (an "erratic driver") as closely as possible without colliding. We observed a wide range of behaviors from participants reflecting varying attitudes towards risk.

12.3.1 Experimental Setting

The setting consists of a leader car and a follower car, simulated in the commercial driving simulator Vires VTD (VIRES Simulationstechnologie GmbH, 2017). Participants controlled the follower car with the Logitech G29 control suite, consisting of a steering wheel and pedals (Figure 12.4). The follower car is modeled using the simple car model with states: x_f (along-track position), y_f (lateral position), v_f (speed), θ_f (yaw angle) and δ_f (steering angle). The dynamics are given by:

$$\dot{x}_f = v_f \cos(\theta_f), \quad \dot{y}_f = v_f \sin(\theta_f), \quad \dot{v}_f = u_a, \quad \dot{\theta}_f = -\frac{v_f}{l}\tan(\delta_f), \quad \dot{\delta_f} = u_s, \tag{12.12}$$

where u_a and u_s are, respectively, the longitudinal acceleration and the steering rate inputs, and $l = 3.476$ m. The leader car plays the role of an "erratic driver" and is modeled with double integrator dynamics along-track and triple integrator dynamics in the lateral direction to mimic continuous steering inputs. The state of the leader's car is described by x_l (along-track position), y_l (lateral position), $v_{x,l}$ (forward speed), $v_{y,l}$ (lateral speed) and a_y (lateral acceleration). The dynamics are given by:

$$\dot{x} = v_x, \quad \dot{v}_x = w_x, \quad \dot{y} = v_y, \quad \dot{v}_y = a_y, \quad \dot{a}_y = w_y, \tag{12.13}$$

where $[w_x, w_y]^T$ is the leader's control input. We simulate this system in discrete time at 60 Hz and analyze the data with a time step $\Delta t = 0.1$ s.

In this setting, a disturbance $w^{[i]}$ corresponds to a sequence of control inputs executed by the leader car $w^{[i]} := \{(w_x, w_y)_k^{[i]}\}_{k=1}^N$ over N time steps. The disturbance sequence is sampled from a finite set of sequences $\mathcal{W} = \{w^{[1]}, \ldots, w^{[L]}\}$ with $L = 4$. These "disturbance" realizations correspond to different maneuvers for the leader (doing nothing, accelerating, decelerating, and swapping lanes) and are generated randomly according to the pmf $p = [0.3, 0.3, 0.3, 0.1]$.

The disturbance is sampled every $N = 15$ time steps. Thus, the leader car can be viewed as executing a random *maneuver* every 1.5 seconds. The whole system is described by the state:

$$\xi := [x_f, y_f, \theta_f, v_f, \delta_f, x_l, v_{x,l}, y_l, v_{y,l}, a_{y,l}]^T. \tag{12.14}$$

Participants in the study were informed that their goal was to follow the leader car (described as an "erratic driver"), as closely as possible in the x and y directions, while staying behind the leader and avoiding any collision. The leader car's four maneuvers were described to participants, along with the fact that these sequences of actions are generated every 1.5 s, *independent* of (as opposed to an interactive game) the participant's actions and position.

The experimental protocol for each participant consisted of three phases. The first phase (~ 1 minute) was meant for the participant to familiarize themselves with the simulator. The second and third phases (one minute each) involved the leader car acting according to the model described

above (with actions being sampled according to the pmf p). The data collected during the second phase was used to train the model and data collected during the third phase was used to test it (the second and third phase disturbance sequences were kept same for all participants).

Note that the pmf p is *not* shared with the participants. This experimental setting may thus be considered *ambiguous*. However, since participants are exposed to a training phase where they may build a mental model of disturbances, the setting may also be interpreted as one involving risk.

While the "game" setting is identical to the one introduced in our earlier work in (Majumdar et al., 2017), the use of non-linear dynamics and a realistic driving simulator as opposed to the first-order integrator MATLAB game in (Majumdar et al., 2017) lends the experiment more realism. All data, algorithm, and plotting code is made available at `https://github.com/StanfordASL/RSIRL`.

12.3.2 Modeling and Implementation

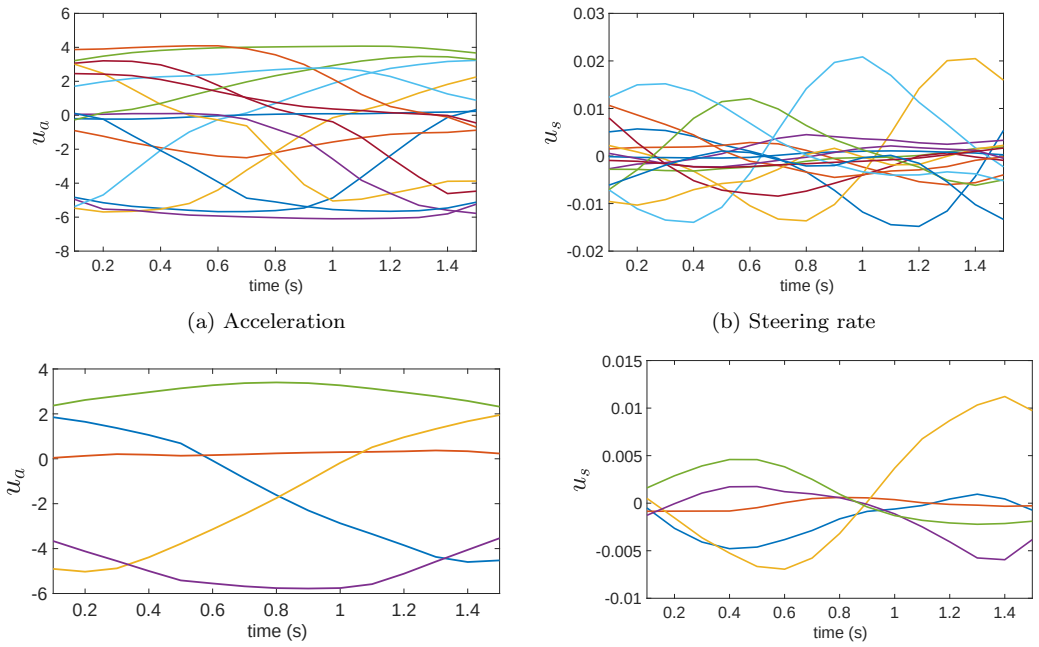
We modeled participants' behavior using the multi-stage "prepare"–"react" framework presented in Section 12.1 with the "prepare" phase starting 0.7 seconds before the leader's action is sampled. The "react" phase thus extends to 0.8 seconds after the disturbance. This parameter was chosen as being roughly reflective of observed participant behavior during the *training* phase. We use $T = 2$ decision stages to model the participants. Hence, the planning horizon is $NT = 30$, which involves planning over two disturbance branching events in a receding horizon fashion.

We represent the cost function as a linear combination of the following features (with unknown weights):

- $\phi_1 = \mathbf{1}_{x_{\text{rel}}<2.5}[\log(1 + e^{-r_1(x_{\text{rel}}-2.5)}) - \log(2)]$,
- $\phi_2 = \mathbf{1}_{x_{\text{rel}}>2.5}[\log(1 + e^{r_2(x_{\text{rel}}-2.5)}) - \log(2)]$,
- $\phi_3 = \log(1 + e^{r_3|v_{x,\text{rel}}|}) - \log(2)$,
- $\phi_4 = r_4 \sum_{k=2}^{N} (u_{a,k} - u_{a,k-1})^2$,
- $\phi_5 = \log(1 + e^{r_5|y_{\text{rel}}|}) - \log(2)$,
- $\phi_6 = \mathbf{1}_{y_f>2}[\log(1 + e^{r_6(y_f-2)}) - \log(2)] + \mathbf{1}_{y_f<-2}[\log(1 + e^{-r_6(y_f+2)}) - \log(2)]$,

where x_{rel}, y_{rel}, and $v_{x,\text{rel}}$ are respectively the relative along-track position, lateral position, and along-track velocity between the leader and the follower, and $\mathbf{1}_{(\cdot)}$ is the indicator function. Hence, the first feature translates the instruction of staying behind the leader; the second, third, and fifth features penalize the relative distance and velocity between the leader and follower; the fourth feature penalizes change in longitudinal acceleration (effectively jerk of the trajectory); the sixth feature penalizes crossing the road boundaries. We use $r_1 = 1$, $r_2 = 0.05$, $r_3 = .1$, $r_4 = 1.0$, $r_5 = 0.1$, and $r_6 = 0.5$. These values were chosen to ensure that the costs were well conditioned over the usual range of relative states observed during the experiments.

In order to optimize the model by maximum likelihood estimation as described in Section 12.1, we discretized the control space $[u_a, u_s]$ to generate the participant action space. For each participant, we ran the K-Means clustering algorithm on the training control inputs, and chose 15 control trajectories for the first decision stage and 5 trajectories for the second stage. Since we model each participant as planning over a receding horizon with two decision stages, it is reasonable to assume that the plan for the second stage is not as fine-grained as over the first one. In addition to reducing the computational burden, this concept of mixing coarse-fine planning is a feature also described in (Carton et al., 2016) to model human locomotion. We observe that using 15 control trajectories for the first stage and 5 trajectories for the second stage was sufficient to generate a diverse expert action set in terms of accelerations (Figure 12.5a, 12.5c) and steering rates (Figure 12.5b, 12.5d); the span of all x/y traces resulting from the combination of these first and second stage control trajectories is shown in Figure 12.6.



(a) Acceleration

(b) Steering rate

(c) Acceleration

(d) Steering rate

Figure 12.5: Example of control trajectories computed by the K-Means algorithm using 15 centroids ((a), (b)) and 5 centroids ((c), (d)). Acceleration in m/s^2 and steering rate in rad/s.

The polytope $\mathcal{P}_r$ (alternatively, the semi-parameterized CRM $\rho^r(\cdot)$) was parametrized with 8 normal vectors $\{a_j\}_{j=1}^{8}$ corresponding to the positive and negative standard basis vectors in $\mathbb{R}^4$ (i.e., $\{(e_i, -e_i)\}_{i=1}^{4}$). Our MATLAB implementation uses the parser YALMIP (Löfberg, 2004) and the solver Mosek (ApS, 2017).

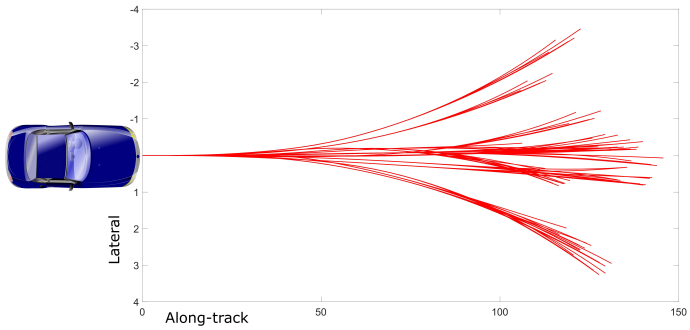


Figure 12.6: Span of all possible along-track/lateral (x/y) 3 second trajectories encapsulated within a single 2-stage optimization problem with the 15/5 discrete control trajectory space. All lengths in meters.

12.3.3 Results

Interestingly, our simulated driving scenario was rich enough to elicit a wide variety of qualitative behaviors from the ten participants. In particular, we observed two extreme policies. One extreme involved the driver following the leader very closely with a small separation (Figure 12.14). Another extreme was to follow the leader with a distance large enough to avoid any collision, often decelerating preemptively to avoid such an event (Figure 12.8). These two extremes can be interpreted as reflecting varying attitudes towards risk. The first policy corresponds to risk-neutral behavior, where the perceived (as captured by the inferred risk measure) probability of collision is lower than for highly risk-averse participants who were more sensitive to the worst-case eventuality (leader slowing down). We also observed a range of behaviors that lie between these two extremes.

We compare the RS-IRL approach with one where the expert is modeled as minimizing the expected value of his/her cost function computed with respect to the pmf p, in a receding horizon fashion with two decision stages. Similar to eq. (12.8), we assume that the risk-neutral stochastic policy is given by the Boltzmann distribution induced by the risk-neutral costs, thereby coinciding with the standard, MaxEnt IRL model and representing an important benchmark for comparison. We refer to this approach as risk-neutral IRL (RN-IRL). The analog of the recursion equations (12.5)–(12.7) for the risk-neutral model are obtained by simply replacing the conditional risk measures with the expected value with respect to the pmf p.

Since the expert is assumed to plan his/her decisions every 1.5 s in a receding horizon fashion, we evaluate RS-IRL and RN-IRL predictions based on their errors with respect to each 1.5 s observed demonstration in the test trajectory. In particular, we define the following error metric for predictions in $x_{\text{rel}} = x_l - x_f$:

$$\Delta x_{\text{rel},t} := \mathbb{E}\left[\sqrt{\sum_k (x^{\text{predicted}}_{\text{rel},k|t} - x^{\text{human}}_{\text{rel},k|t})^2}\right], \tag{12.15}$$

where $x^{\text{predicted}}_{\text{rel},k|t}$ and $x^{\text{human}}_{\text{rel},k|t}$ are, respectively, the predicted and actual x_{rel} trajectories at time $k \in [tN, (t+1)N]$ corresponding to the t^{th} 1.5 s segment in the demonstrated trajectory. The expectation is taken with respect to the stochastic policy (i.e., Boltzmann distribution) induced by the RS or RN costs (see eq. (12.8) and (12.16)). The errors Δy_{rel}, $\Delta v_{x,\text{rel}}$ and $\Delta v_{y,\text{rel}}$ are computed similarly. As RN-IRL consistently performed better with $T = 2$ decision stages, we only present comparison results between RS-IRL and RN-IRL for $T = 2$. To get a scale for the values reported in this section, the figure below illustrates the two cars almost colliding ($x_{\text{rel}} \approx 2.5$ m) and when they are 5 m apart.

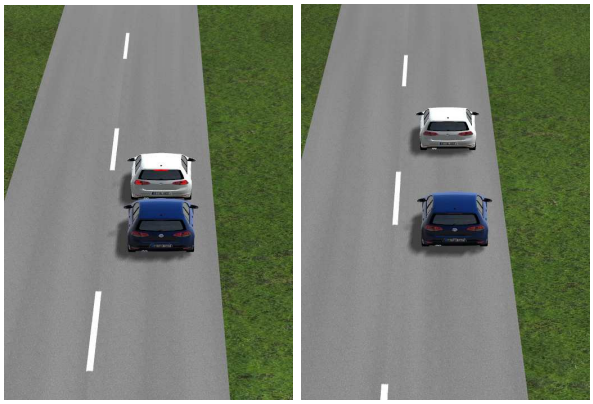


Figure 12.7: Left: Simulator visual when cars are almost at collision distance ($x_{\text{rel}} \approx 2.5$ m); Right: Simulator visual of a participant driving 5 meters behind the leader. Lane-width: 3 m.

Case Study # 1: Risk-Averse Participant

Figure 12.8 plots the x_{rel} trajectory (normalized by car length ≈ 4.2 m) during the third (test) phase for a participant inferred to be highly risk-averse. On average, the along-track relative distance is quite large (≈ 6 car-lengths). The expected prediction errors $\Delta x_{\text{rel},t}$ (normalized by car-length) from RS-IRL and RN-IRL for each of the 51 1.5 s demonstrations comprising the test phase are plotted in Figure 12.9a as absolute errors, and in Figure 12.9b as percentage differences with positive values indicating an improvement of RS-IRL over RN-IRL. A similar error plot is shown in Figure 12.9c and 12.9d for along-track velocity error $\Delta v_{x,\text{rel},t}$.

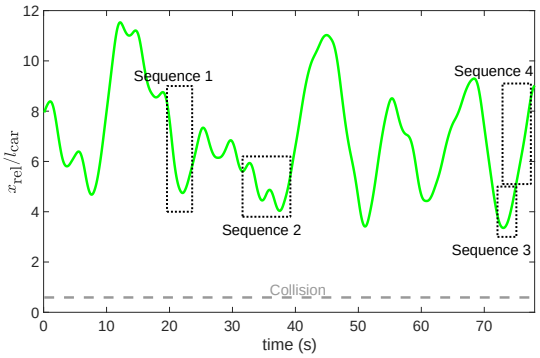
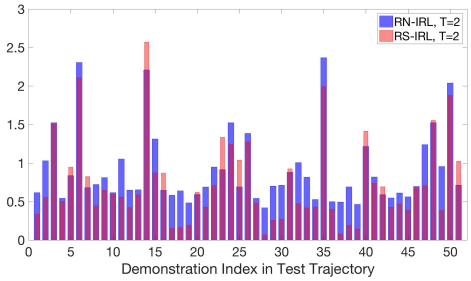
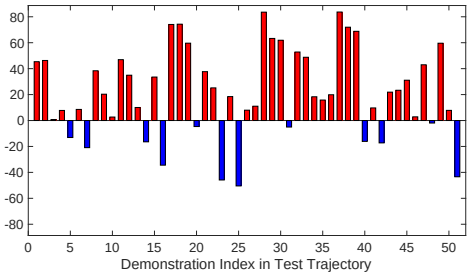


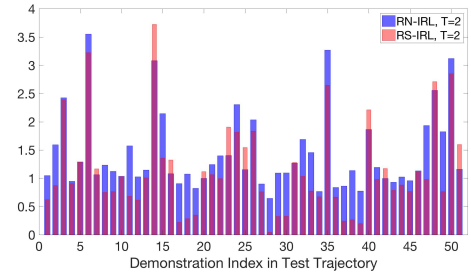
Figure 12.8: Full x_{rel} (longitudinal distance) trajectory (normalized by car length) for a highly risk-averse participant. On average, the relative distance is quite large ($\approx$ 6 car-lengths). The boxed sections are discussed in further detail below.



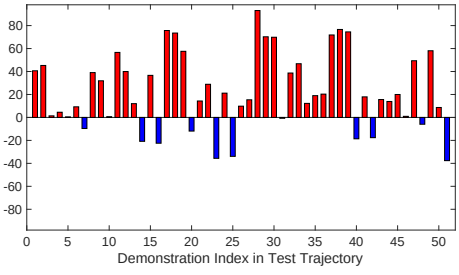
(a) Expected (w.r.t. stochastic policy) prediction errors $\Delta x_{\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.



(b) Percentage improvement in $\Delta x_{\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.

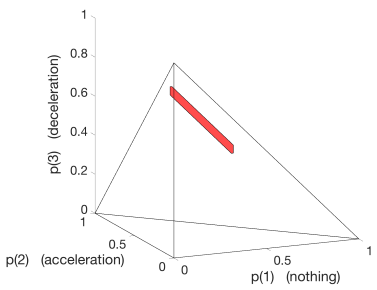


(c) Expected (w.r.t. stochastic policy) prediction errors $\Delta v_{x,\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.

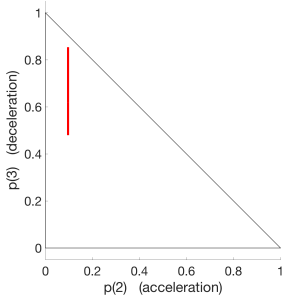


(d) Percentage improvement in $\Delta v_{x,\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.

Figure 12.9: Comparison of the $\Delta x_{\text{rel},t}$ and $\Delta v_{x,\text{rel},t}$ prediction errors (normalized by car length) for the RS-IRL and RN-IRL models for a highly risk-averse participant. The RS-IRL model almost always outperforms RN-IRL, on average providing about 22% improvement.



(a) Projection of the risk-averse participant's risk envelope along the first three dimensions, corresponding to the {nothing, accelerate, decelerate} maneuvers by the leader.



(b) Projection of the risk-averse participant's risk envelope along the second two dimensions, corresponding to the {accelerate, decelerate} maneuvers by the leader.

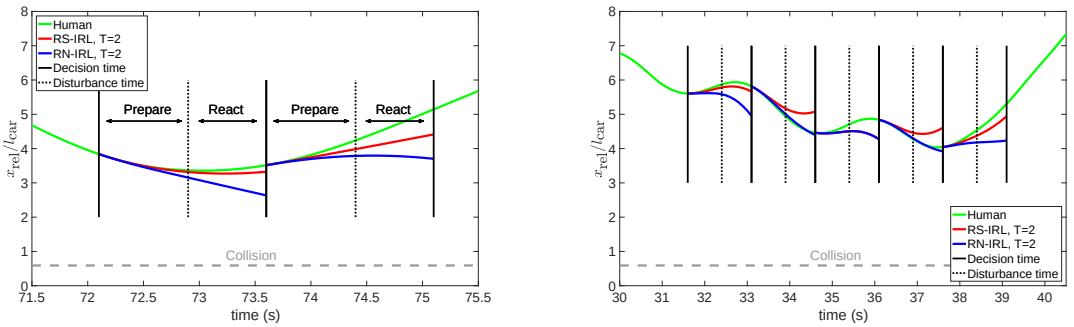
Figure 12.10: Inferred risk envelope for risk-averse participant. Notice the overweighting (and ambiguity therein) of probabilities associated with the leader's deceleration maneuver and the underweighting of the leader's acceleration. The wire-frame pyramid is the projection of the probability simplex Δ^4 onto the first three dimensions.

It can be observed that the RS-IRL model almost always outperforms RN-IRL, on average providing an improvement of about 22%. Notice however, the four prominent negative peaks in Figure 12.9b corresponding to the RN-IRL model outperforming RS-IRL on these instances. All four peaks can be explained by considering the inferred risk envelope $\mathcal{P}_r$, whose projections are plotted in Figures 12.10a and 12.10b. Notice how the participant's polytope is significantly biased towards high probabilities in the third dimension (corresponding to the decelerate maneuver for the leader), and furthermore, spans a wide range in probabilities ($\approx [0.5, 0.85]$) for this event. Thus, not only does the participant overweight the true probability of this outcome (0.3), he/she is also *ambiguous* about the true probability. On the other hand, the envelope suggests a very low and narrow range of probabilities for the leader's accelerate maneuver. These two properties are what result in the four negative peaks in Figure 12.9b.

Consider the first negative peak (corresponding to Sequence 1 in Figure 12.8). At this decision stage, the true distance decreases due to the participant accelerating. As the learned RS-IRL model assumes significant ambiguity in the leader's deceleration yet small probability in acceleration, the most probable trajectories from the RS-IRL model prefer weaker acceleration. Thus, this particular artifact may be attributed to a slight overfitting of the parameterized polytope offsets r. This overfitting manifests itself again at the very end (Sequence 4) where the cars are far apart (6 car-lengths) and the RS-IRL model predicts a weaker deceleration than that observed. Essentially the high bias associated with the leader's deceleration as encoded in the inferred polytope along with the already large separation between the cars (recall that the second feature penalizes large separations) leads to a less aggressive deceleration prediction. The remaining peaks (Sequence 2) are shown in detail in Figure 12.11b. At both these decision stages, the RS-IRL model *preempts*, by about one decision stage, a deceleration maneuver for the participant. Again this is a result of the deceleration

ambiguity implied by the RS-IRL model and suggests that additional training data is needed to further refine (decrease) this ambiguity. This would naturally also alleviate the prediction errors in Sequences 1 and 4.

Notice however that all of the spurious predictions by the RS-IRL model as discussed above occur when the cars are quite far apart, on the order of 5–6 car-lengths. In contrast, when the two cars are quite close (e.g., less than 4 car-lengths during Sequence 3), there is now a significant collision risk in the event that the leader decelerates. Figure 12.11a illustrates the RS-IRL model correctly predicting the participant braking to increase the relative separation, and with high probability (≈ 0.89). In contrast, the RN-IRL induced stochastic policy is not only of high entropy but also suggests quite absurd trajectories that lead to a further decrease in the relative separation. Thus, these results suggest that when RS-IRL underperforms RN-IRL, it does so at non-critical stages characterized by low risk.



(a) Close-up of Sequence 3 in Figure 12.8 demonstrating the accuracy of RS-IRL over RN-IRL at a critical decision stage when cars are quite close.

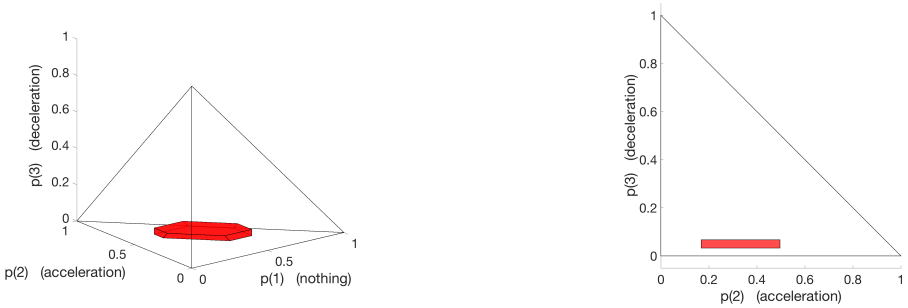
(b) Close-up of Sequence 2 in Figure 12.8 demonstrating the preemptive braking maneuvers predicted by RS-IRL. Notice that the error occurs at a non-critical decision stage when cars are far apart.

Figure 12.11: Comparisons of the *most-probable* (under the stochastic Boltzmann policy) RS-IRL and RN-IRL trajectory predictions in x_{rel} as compared with the true data.

Case Study # 2: Risk/Ambiguity-Averse Participant

We next discuss a participant for whom RS-IRL again significantly outperformed RN-IRL, however, due to a different manifestation of risk. Consider the inferred risk envelopes in Figure 12.12.

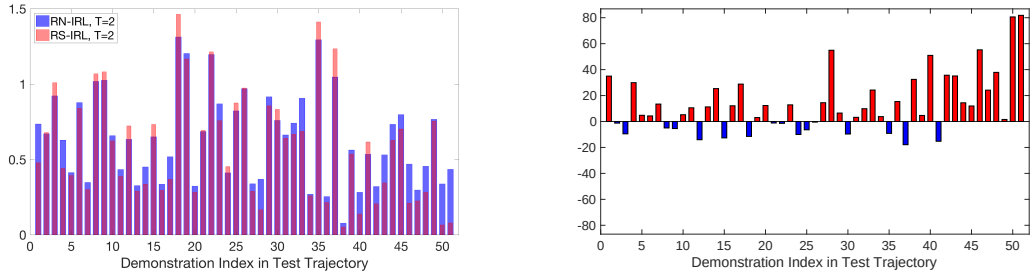
Notice that the participant places very low probability on deceleration yet is ambiguous regarding the leader's acceleration (i.e., a polar opposite of the first participant). This ambiguity led to several preemptive braking maneuvers in an attempt to maintain a "safe" distance to the leader car. Figures 12.13a–12.13d illustrate the consistency of RS-IRL's improvement over RN-IRL in predicting these maneuvers over the entire test trajectory.



(a) Projection of polytope along the {nothing, accelerate, decelerate} "dimensions".

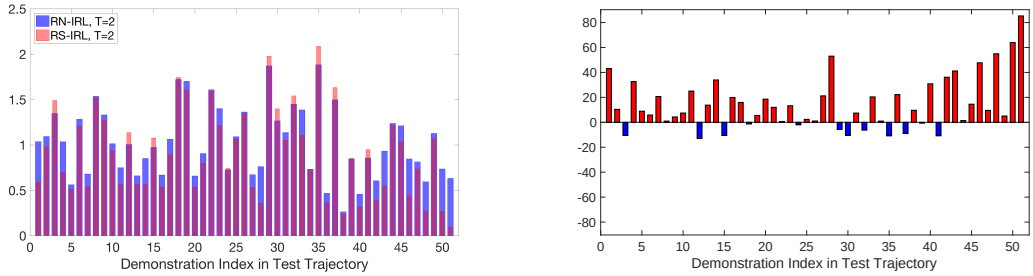
(b) Projection of polytope along the {accelerate, decelerate} "dimensions".

Figure 12.12: Inferred risk envelope for ambiguity-averse participant. Notice the drastic underweighting of probabilities associated with leader's deceleration and significant ambiguity regarding leader's acceleration.



(a) Expected (w.r.t. stochastic policy) prediction errors $\Delta x_{\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.

(b) Percentage improvement in $\Delta x_{\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.



(c) Expected (w.r.t. stochastic policy) prediction errors $\Delta v_{x,\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.

(d) Percentage improvement in $\Delta v_{x,\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.

Figure 12.13: Comparison of the $\Delta x_{\text{rel},t}$ and $\Delta v_{x,\text{rel},t}$ prediction errors (normalized by car length) for the RS-IRL and RN-IRL models for an ambiguity-averse participant. The RS-IRL model almost always outperforms RN-IRL, on average providing about 13–14% improvement.

Case Study # 3: Risk-Neutral Participant

Finally, we consider a participant for whom both RS-IRL and RN-IRL performed on par with each other. Figure 12.14 plots the x_{rel} trajectory while Figure 12.15 illustrates the inferred risk envelope.

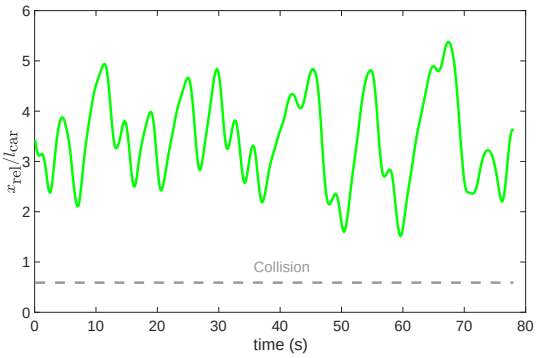
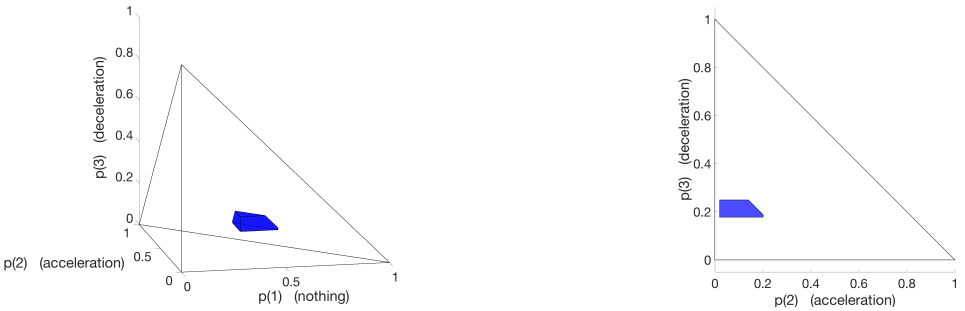


Figure 12.14: Full x_{rel} trajectory (normalized by car length) for a risk-neutral participant. On average, the relative distance is noticeably smaller, on the order of 3 car-lengths.



(a) Projection of polytope along the {nothing, accelerate, decelerate} "dimensions".

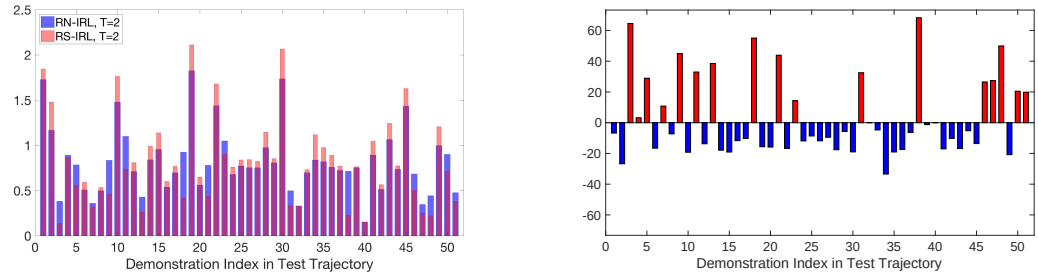
(b) Projection of polytope along the {accelerate, decelerate} "dimensions".

Figure 12.15: Inferred risk envelope for risk-neutral participant. Notice how there is no appreciable level of ambiguity nor is the polytope biased along any dimension; hence suggesting the risk-neutral categorization.

The inferred risk envelope features no bias along any dimension nor any appreciable level of ambiguity; thereby suggesting a risk-neutral profile for this participant. Furthermore, notice in Figure 12.14 that the participant stays quite a bit closer to the leader car than the first participant, a clear indicator of his/her risk-neutral stance. Figures 12.16 and 12.17 confirm the two models performing on par with each other. This, however, is to be expected for a strongly risk-neutral participant since the RS-IRL model subsumes the RN-IRL model.

Limitations of Cost Shaping

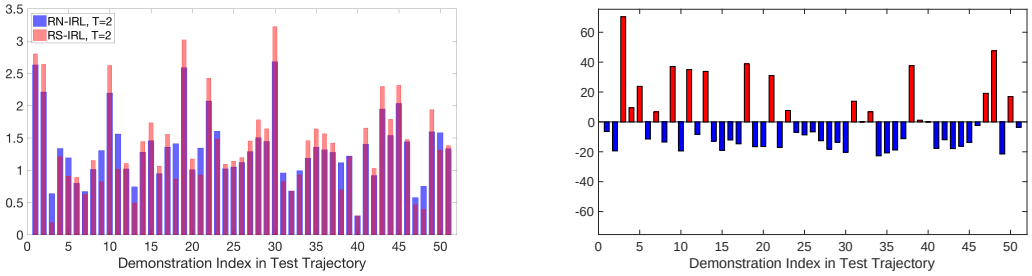
In this section we argue that *both* the cost weights c and the risk measure $\rho(\cdot)$ are *necessary* to reasonably approximate diverse risk-sensitive behaviors. In Table 12.1, we provide the learned feature weights from RS-IRL and RN-IRL for features $\{\phi_1, \phi_2, \phi_3, \phi_4\}$ for the participants in case studies #1 and #2. These are the four features that dominate the along-track behavior of the participants.



(a) Expected (w.r.t. stochastic policy) prediction errors $\Delta x_{\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.

(b) Percentage improvement in $\Delta x_{\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.

Figure 12.16: Comparison of the $\Delta x_{\text{rel},t}$ prediction errors (normalized by car length) for the RS-IRL and RN-IRL models for a risk-neutral participant. The two models perform on par with each other.



(a) Expected (w.r.t. stochastic policy) prediction errors $\Delta v_{x,\text{rel},t}$ from RS-IRL and RN-IRL for each 1.5 s trajectory segment.

(b) Percentage improvement in $\Delta v_{x,\text{rel},t}$ for the RS-IRL model over RN-IRL for each 1.5 s trajectory segment.

Figure 12.17: Comparison of the $\Delta v_{x,\text{rel},t}$ prediction errors (normalized by car length) for the RS-IRL and RN-IRL models for a risk-neutral participant. The two models perform on par with each other.

	Case Study #1		Case Study #2	
Feature weight	RS-IRL	RN-IRL	RS-IRL	RN-IRL
$c(1)$	0.0918	0.0466	0.0748	0.1884
$c(2)$	0.5174	0.5589	0.6354	0.2313
$c(3)$	0.0993	0.1052	0.1864	0.3170
$c(4)$	0.2352	0.2330	0.0562	0.0624

Table 12.1: Comparison of the inferred cost weights from RS-IRL and RN-IRL for the participants in case studies #1 and 2.

Notice that in spite of the extremely similar cost weights for the participant in case study #1, the improvement in performance using RS-IRL is substantial. For the participant in case study #2, a difference in the cost weights *and* the use of a risk measure were needed to obtain the observed performance boost with RS-IRL. This clearly establishes the benefits of risk sensitive inference, particularly highlighting the deficiency of only using cost shaping (in general, risk-neutral algorithms)

due to its inability to cope with more nuanced manifestations of uncertainty in decision-making.

Summary

Table 12.2 presents the average (over the 51 1.5 s segments in the test trajectory) percentage improvement (RN-IRL to RS-IRL) in the prediction errors, i.e., Δx_{rel}, Δy_{rel}, $\Delta v_{x,\text{rel}}$ and $\Delta v_{y,\text{rel}}$. As expected, the RS-IRL predictions are almost always better than those provided by RN-IRL, with as much as 22.0% improvement in x_{rel} and 23.1% improvement in $v_{x,\text{rel}}$. Regarding errors in y_{rel} and $v_{y,\text{rel}}$, RS-IRL and RN-IRL perform comparably (absolute errors were in the range 0.19 – 0.42 m) since the primary source of risk-aversion stems from the leader's acceleration and deceleration rather than its lateral motion. In the cases with noticeable improvement, either in position or velocity (i.e., participants #2, 5, 6, 9), the better predictions were a result of the RS-IRL model more accurately representing participants with higher levels of risk- and/or ambiguity-aversion. Indeed the first two detailed case studies presented earlier correspond to participants #2 and #5. In contrast, for more risk-neutral participants (e.g., the last presented case study corresponding to participant #4), the performance improvements were either less pronounced or the two models performed comparably.

Participant #	1	2	3	4	5	6	7	8	9	10
Δx_{rel} (T=2)	7.5	22.0	3.3	2.6	13.3	14.9	3.9	9.7	18.6	10.1
Δy_{rel} (T=2)	10.7	22.2	-2.7	12.0	4.8	9.5	8.4	7.3	16.3	21.4
$\Delta v_{x,\text{rel}}$ (T=2)	12.7	23.1	5.0	-0.3	14.3	13.0	8.1	10.5	17.1	10.4
$\Delta v_{y,\text{rel}}$ (T=2)	8.7	22.9	-3.0	13.7	0.2	6.0	9.1	3.0	14.9	22.5

Table 12.2: Average percentage improvement (over 51 segments in the test trajectory) in prediction errors for RS-IRL over RN-IRL. The RS-IRL predictions with $T = 2$ for x_{rel} and $v_{x,\text{rel}}$ are more accurate than those for RN-IRL for all but one participant, with as much as 23.1% average improvement. The most pronounced improvements (highlighted in red) corresponded to participants with higher levels of risk- and/or ambiguity-aversion, some of whom are studied in detail in the case studies. The improvements in the lateral direction are less significant since the primary source of risk and ambiguity was in the longitudinal dynamics.

12.4 Summary

We have presented an approach for IRL that explicitly accounts for risk *and* ambiguity sensitivity in experts. We proposed a flexible modeling framework based on coherent risk measures that allows us to capture an entire spectrum of risk assessments from risk-neutral to worst-case for a rich class of static and dynamic decision-making settings. We developed efficient LP based non-parametric algorithms for static (Chapter 11), and likelihood-based semi-parametric algorithms for dynamic decision-making settings. Notably, we significantly improved the modeling framework in (Majumdar et al., 2017) for dynamic decision-making, and verified the performance improvements from

RS-IRL, despite transitioning from exact LP iteration to the likelihood-based algorithm. The proposed inference framework was rigorously evaluated on a realistic simulated driving game with ten participants and shown to be able to infer and mimic qualitatively different driving styles ranging from risk-neutral to highly risk-averse in a data-efficient manner, while more accurately capturing participant behavior than with a risk-neutral model. Most importantly, by performing inference using the dual representation of coherent risk measures, we retain the generality to be able to recover any risk measure within such a class of risk measures, *without* assuming any a priori knowledge (e.g., fixed disutility function and/or risk measure).

Appendix

In this appendix, we provide derivations of the full (computationally intractable) likelihood model, and the gradient formulas for the semi-parametric maximum likelihood model introduced within the chapter.

Appendix 12.A Derivation of Prepare-React Distribution

Recall the multi-stage optimization problem objective, repeated here for convenience:

$$C_{0:N-n_d-1} + \rho_0\Big(C_{N-n_d:N-1} + C_{N:2N-n_d-1} + \rho_1(C_{2N-n_d:2N-1} + \cdots + \rho_{T-1}\left(C_{TN-n_d:TN-1}\right)\cdots)\Big).$$

For a "prepare" – "react" policy $\hat{\pi}_t$ at stage t, let $\hat{\pi}_{t|\mathfrak{p}}$ denote the "prepare" portion and $\hat{\pi}_{t|\mathfrak{r}}$ denote the "react" portion. Then, we can re-write the objective above to show explicit dependence as follows:

$$\begin{aligned} C_{0:N-n_d-1}(\cdot,\hat{\pi}_{0|\mathfrak{p}}) + \rho_0\Big(&C_{N-n_d:N-1}(\cdot,\hat{\pi}_{0|\mathfrak{r}}) + C_{N:2N-n_d-1}(\cdot,\hat{\pi}_{1|\mathfrak{p}}) + \\ &\rho_1(C_{2N-n_d:2N-1}(\cdot,\hat{\pi}_{1|\mathfrak{r}}) + \cdots + \rho_{T-1}\left(C_{TN-n_d:TN-1}(\cdot,\hat{\pi}_{T-1|\mathfrak{r}})\right)\cdots)\Big). \end{aligned}$$

Note that the expression within the large brackets is a random variable in $\mathbb{R}^L$, indexed by all possible realizations of w_0', and a function of the first "prepare" sequence $\hat{\pi}_{0|\mathfrak{p}}$. Thus, for a given "prepare" sequence $\hat{\pi}_{0|\mathfrak{p}}$ and first disturbance mode w_0', define the optimal tail cost as a function of $\hat{\pi}_{0|\mathfrak{r}}$:

$$\begin{aligned} \tau[\hat{\pi}_{0|\mathfrak{p}}, w_0'](\hat{\pi}_{0|\mathfrak{r}}) := C_{N-n_d:N-1}(\cdot,\hat{\pi}_{0|\mathfrak{r}}) + \\ \min_{\substack{\hat{\pi}_t \\ t\in[1,T-1]}} \quad \rho_1(C_{N:2N-1}(\cdot,\hat{\pi}_1) + \cdots + \rho_{T-1}(C_{(T-1)N:TN-1}(\cdot,\hat{\pi}_{T-1}))). \end{aligned}$$

Then, we define the *conditional* distribution for the first "react" sequence corresponding to disturbance mode w'_0, given the first "prepare" sequence, as

$$\Pr(\hat{\pi}_{0|\mathfrak{r}} \mid \hat{\pi}_{0|\mathfrak{p}} \; ; w'_0) \propto \exp\left(-\tau[\hat{\pi}_{0|\mathfrak{p}}, w'_0](\hat{\pi}_{0|\mathfrak{r}})\right).$$

The distribution for the first "prepare" sequence is then given by

$$\Pr(\hat{\pi}_{0|\mathfrak{p}}) \propto \exp\left(-\left[C_{0:N-n_d-1}(\cdot, \hat{\pi}_{0|\mathfrak{p}}) + \rho^r\left(\operatorname*{softmin}_{\hat{\pi}_{0|\mathfrak{r}}} \tau[\hat{\pi}_{0|\mathfrak{p}}, w'_0](\hat{\pi}_{0|\mathfrak{r}})\right)\right]\right),$$

where we use softmin in place of min to ensure differentiability. Thus, for an observed "prepare" – "react" sequence $\hat{\pi}^*_0$ associated with the observed disturbance mode w^*_0, we obtain

$$\Pr(\hat{\pi}^*_0) = \Pr(\hat{\pi}^*_{0|\mathfrak{p}}) \cdot \Pr(\hat{\pi}^*_{0|\mathfrak{r}} \mid \hat{\pi}^*_{0|\mathfrak{p}} \; ; w^*_0).$$

Appendix 12.B Likelihood Gradient Computations

Define

$$\sigma[w^*_{-1|tN}](\hat{u}) := \frac{\exp\left(-\beta\tilde{\tau}[w^*_{-1|tN}](\hat{u})\right)}{\sum_{\hat{u}' \in \prod} \exp\left(-\beta\tilde{\tau}[w^*_{-1|tN}](\hat{u}')\right)} \tag{12.16}$$

to be the probability of choosing action trajectory $\hat{u}$ at time-step tN as assumed by the Boltzmann likelihood model in (12.8). Then, the gradient of the log-likelihood in (12.10) with respect to parameter $s \in \{r, c\}$ is given by

$$\frac{\beta}{|\mathcal{T}^*|} \sum_{\hat{u}^*_t \in \mathcal{T}^*} \left[\sum_{\hat{u} \in \prod \setminus \hat{u}^*_t} \sigma[w^*_{-1|tN}](\hat{u}) \nabla_s \tilde{\tau}[w^*_{-1|tN}](\hat{u}) + \left(\sigma[w^*_{-1|tN}](\hat{u}^*_t) - 1\right) \nabla_s \tilde{\tau}[w^*_{-1|tN}](\hat{u}^*_t)\right]. \tag{12.17}$$

For notational clarity, we use t to denote the t^{th} N-step segment in the demonstrated trajectory $\mathcal{T}^*$ and t' as the stage-wise index within the multi-step planning problem. From equations (12.5), (12.6), and (12.7), we see that the derivative of $\tilde{\tau}[w^*_{-1|tN}](\hat{u})$ can be computed through a recursive implementation of the chain rule, starting from the terminal stage. In the event that all nested LPs are non-degenerate, we obtain the following recursive set of equations for computing the gradients.

Terminal Stage: From eq. (12.5), $\tilde{\tau}[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u})$ is the optimal value of the LP:

$$\max_{v \in \mathcal{P}_r} \quad v^T g(w'_{T-1}, \hat{u}; c), \tag{12.18}$$

where $g \in \mathbb{R}^L$ is the accumulated cost vector over the terminal stage, $C_{(T-1)N:TN-1}$, indexed by the

terminal disturbance mode w'_{T-1}. Let v^* denote the optimal primal solution and λ^* the optimal dual variables for the constraints defined in (12.2). Then,

$$\nabla_r \tilde{\tau}[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u}) = -\lambda^*, \tag{12.19}$$

$$\nabla_c \tilde{\tau}[\boldsymbol{u}_{T-2}, \boldsymbol{\omega}_{T-2}](\hat{u}) = \sum_{j=1}^{L} v^*(j) \left[\Phi^{[j]}(\hat{u})\right], \tag{12.20}$$

where $\Phi^{[j]}(\hat{u})$ is the feature vector sum over N time steps corresponding to $C_{(T-1)N:TN-1}$, given[2] action trajectory $\hat{u}$ and disturbance $w^{[j]}$.

Recursion: For $t' \in \{0, \ldots, T-2\}$, $\tilde{\tau}[\boldsymbol{u}_{t'-1}, \boldsymbol{\omega}_{t'-1}](\hat{u})$ is the optimal value of the LP[3]:

$$\max_{v \in \mathcal{P}_r} \quad v^T \left(g(w'_t, \hat{u}; c) + \tilde{g}(w'_t, \hat{u}; c, r)\right),$$

where $g \in \mathbb{R}^L$ is the accumulated cost vector $C_{t'N:(t'+1)N-1}$, and $\tilde{g} \in \mathbb{R}^L$ is the vector of $\text{softmin}_{\hat{u}'}$ over the tail risk-sensitive costs, i.e., $\tilde{\tau}[\{\boldsymbol{u}_{t'-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t'-1}, w'_t\}](\hat{u}')$, indexed by the next disturbance mode w'_t, and parameterized with respect to r, c. Recall that $w'_{-1} = w^*_{-1|tN}$. Let v^* denote the optimal primal solution and λ^* the optimal dual variables for the constraints in eq. (12.2). Then,

$$\nabla_r \tilde{\tau}[\boldsymbol{u}_{t'-1}, \boldsymbol{\omega}_{t'-1}](\hat{u}) = \sum_{j=1}^{L} v^*(j)\, \nabla_r \tilde{g}(w^{[j]}, \hat{u}; c, r) - \lambda^*, \tag{12.21}$$

$$\nabla_c\, \tilde{\tau}[\boldsymbol{u}_{t'-1}, \boldsymbol{\omega}_{t'-1}](\hat{u}) = \sum_{j=1}^{L} v^*(j) \left[\Phi^{[j]}(\hat{u}) + \nabla_c\, \tilde{g}(w^{[j]}, \hat{u}; c, r)\right], \tag{12.22}$$

where $\Phi^{[j]}$ is the feature vector sum corresponding to $C_{t'N:(t'+1)N-1}$. The gradients of the components of the softmin vector are given by:

$$\nabla_r \tilde{g}(w^{[j]}, \hat{u}; c, r) = \mathbb{E}_{\hat{u}' \sim \sigma\left[\tilde{\tau}[\{\boldsymbol{u}_{t'-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t'-1}, w^{[j]}\}]\right]} \left[\nabla_r \tilde{\tau}[\{\boldsymbol{u}_{t'-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t'-1}, w^{[j]}\}](\hat{u}')\right], \tag{12.23}$$

$$\nabla_c \tilde{g}(w^{[j]}, \hat{u}; c, r) = \mathbb{E}_{\hat{u}' \sim \sigma\left[\tilde{\tau}[\{\boldsymbol{u}_{t'-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t'-1}, w^{[j]}\}]\right]} \left[\nabla_c \tilde{\tau}[\{\boldsymbol{u}_{t'-1}, \hat{u}\}, \{\boldsymbol{\omega}_{t'-1}, w^{[j]}\}](\hat{u}')\right], \tag{12.24}$$

where $\sigma\left[\tilde{\tau}[\boldsymbol{u}_{t'}, \boldsymbol{\omega}_{t'}]\right]$ is the discrete (cost-based) Boltzmann distribution, i.e.,

$$\sigma\left[\tilde{\tau}[\boldsymbol{u}_{t'}, \boldsymbol{\omega}_{t'}]\right](\hat{u}') \propto \exp\left(-\tilde{\tau}[\boldsymbol{u}_{t'}, \boldsymbol{\omega}_{t'}](\hat{u}')\right).$$

For our experiments, we found a different form of the softmin function to be more numerically stable.

[2] For notational convenience, we omit the obvious dependence on state.
[3] For notational simplicity, take $\boldsymbol{u}_{-1} = \{\}$.

In particular, we used

$$\underset{\beta}{\text{softmin}}\, f(x) = \mathbb{E}_{x\sim\sigma_\beta[f]} f(x),$$

where $\sigma_\beta[f]$ is the Boltzmann distribution defined with inverse temperature β as:

$$\sigma_\beta[f](x) \propto \exp\left(-\beta f(x)\right).$$

The gradients take the exact same form as (12.23) and (12.24) with σ replaced by σ_β.

Thus, to compute the gradient in (12.17) for the t^{th} demonstration, one would start with the terminal stage derivatives in (12.19) and (12.20) for the planning problem defined at time step tN, and proceed backwards inductively using (12.21)–(12.24) to arrive at $\nabla\tilde{\tau}[w^*_{-1|tN}](\hat{u})$.

Note that the derivation above assumes non-degeneracy of the LPs. It is readily observed that by the piecewise linearity of LPs with respect to the objective coefficients and constraint right-hand-sides, and the Lipschitz property of the softmin function, $\tilde{\tau}$ is locally Lipschitz. Consequently, the log-likelihood too is locally Lipschitz. Thus, by the Rademacher theorem, the log-likelihood is non-differentiable only over a Lebesgue set of measure zero. If, however, during the updates, any (nested) LP is primal degenerate (multiple dual optimal solutions) or dual degenerate (multiple primal optimal solutions), the log-likelihood is non-differentiable (despite directional-derivatives existing in all directions). Thus, in its full generality, the max likelihood problem corresponds to a non-convex, non-smooth optimization and at points of non-differentiability, one must do extra work to compute a suitable descent direction. Some proposed approaches in the literature include penalized smoothing (Chen, 2012), sampling-based estimation (Burke et al., 2005) to approximate the Clarke generalized subdifferential and compute a descent direction, and majorization-minimization (Lanza et al., 2017) to iteratively optimize an upper-bound on a minimization problem. Arguably, the field is an active area of research.

In our implementation, we implemented the following two heuristics to avoid non-degeneracy (and consequent non-differentiability of the log-likelihood):

- During the projection step of the projected gradient method for r, if a constraint $a_j^T v \leq b(j) - r(j)$ is found to be redundant, the parameter $r(j)$ is re-adjusted as $r(j) \leftarrow r(j) + 0.01$, provided that the resulting polytope is not empty. This has the effect of eliminating the possibility of redundant constraints at primal optimal vertices (thereby eliminating primal degeneracy).

- In the event that the objective vector is parallel to one of the bounding hyperplanes of the region $\mathcal{P}_r$ (i.e., dual degeneracy), we added a small distortion to the objective vector.

Chapter 13

Conclusions and Future Directions

In this chapter we conclude this thesis, review some of the limitations of the methods proposed in Parts I - III, and suggest avenues for future work.

13.1 Concluding Remarks

With the growing prevalence of autonomous robots in our daily lives, it is imperative that these robots are able to systematically reason about the inherent uncertainty that underpins their surroundings. This necessarily entails the ability to characterize the effect of such uncertainties at various levels of the autonomy stack and formally incorporate this information into the robot's planning and decision-making algorithms. This thesis presented algorithms for imbuing robustness within two hierarchically distinct components of the autonomy stack: motion planning and decision-making. The contributions within the planning component of this work were on (i) enabling real-time computation of motion plans using simplified dynamic models and accounting for the resulting model mismatch, (ii) developing a systematic robust control synthesis technique using contraction theory for trajectory tracking with strong disturbance rejection guarantees, and (iii) leveraging the control-theoretic advancements as a means of context-driven hypothesis pruning within model-based reinforcement learning. We validated the algorithms both in simulation and on hardware testbeds to illustrate the clear connections between theory and practice. Collectively, these results provide the practitioner with a suite of analysis and computational tools to enable a tighter integration of robust nonlinear control theory within robot motion planning. The contribution within the decision-making component of this thesis was to introduce a general notion of risk theory for sequential decision-making, for the purpose of risk- and/or ambiguity-sensitive inference for humans in safety-critical scenarios. Notably, we motivated the need for more nuanced decision models that account for subjectivity of the decision-maker with respect to both an underlying cost function and the interpretation of probabilities (i.e., uncertainty). The algorithms were validated

on a realistic simulator with humans in-the-loop, where we illustrated advantages from both numerical (in terms of prediction accuracy) and user (in terms of interpretability of the inferred decision models) standpoints.

13.2 Future Directions

We now provide suggestions for future work pertaining to all parts of the thesis.

13.2.1 Parts I and II

We begin by first discussing themes common to both Parts I and II.

Common Themes

Exponential Stabilizability: The form of stabilization leveraged within the construction of the robust feedback controllers was exponential convergence. Similarly, the control-theoretic regularization employed within dynamics learning was founded upon conditions for exponential stabilizability of smooth systems via state feedback. The two key reasons for this are that exponential stabilizability leads to (i) convex algebraic conditions, and (ii) strong disturbance rejection properties. In order to broaden the applicability of *systematic robust control synthesis* and incorporating more diverse control-theoretic notions within learning, it is of interest to investigate not only other, more general forms of stabilizability (e.g., boundedness), but also extend the analysis to hybrid mechanical systems. This would enable tackling a larger class of problems, such as those within the manipulation and locomotion domains, where disturbance rejection is fundamentally more challenging and prior models can be severely inaccurate due to the complexity of modeling contact physics.

Computational: By far, the primary computational challenge within robust planning and learning stabilizable models involved solving semi-infinite optimization problems. As long as we continue to deal with nonlinear systems, this will always be a recurring theme. Within robust planning, SOS programming allowed us to obtain sufficient certificates for the infinite-dimensional constraints but at a cost of scalability. The size of the equivalent SDP re-formulation of a SOS constraint scales as $O(n^d)$ where n is the state-space dimension and $2d$ is the degree of the polynomial. While there are some alternative relaxations available based on LPs and SOCPs (Ahmadi and Majumdar, 2014), these are more conservative than the SDP re-formulation. For the model-mismatch work, where optimal solutions (as recovered by HJ methods) are non-smooth, this is not a feasible solution strategy.

There are two potential future directions for alleviating this limitation. The first involves leveraging non-polynomial SOS decompositions using Pfaffian function chain theory. Formally, a Pfaffian chain is defined as follows:

Definition 5 (Pfaffian Chains and Functions)**.** *Let $\mathcal{F} = (f_1, \ldots, f_k)$ be a sequence of real analytic functions defined over some subset $\mathcal{X}$ of $\mathbb{R}^n$. Then, $\mathcal{F}$ constitutes a Pfaffian Chain if there exists polynomials $p_{i,j}$, $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, n\}$, such that*

$$\frac{\partial f_i}{\partial x^j}(x) = p_{i,j}(x, f_1(x), \ldots, f_i(x)) \quad \forall x \in \mathcal{X}.$$

Let $\mathcal{F}$ be a fixed Pfaffian chain, and q an analytic function on $\mathcal{X}$. Then, q is a Pfaffian function in the chain $\mathcal{F}$ if there exists a polynomial Q, such that:

$$q(x) = Q(x, f_1(x), \ldots, f_k(x)) \quad \forall x \in \mathcal{X}.$$

Examples of Pfaffian functions include of course, polynomials, but also a much larger set of real analytic functions such as exponentials and their compositions, algebraic functions, elementary functions, hyperbolic functions, and trigonometric functions on bounded domains. Essentially, Pfaffian functions allow us to leverage a significantly larger class of functions, for instance, as basis expansions, while still retaining polynomial relationships between its members when taking derivatives. This opens the door for expanding SOS programming techniques by utilizing *non-polynomial* bases, thereby eliminating the need for high-degree polynomial approximations of non-polynomial functions, potentially leading to much smaller SDPs. An extensive overview on Pfaffian functions and associated theory may be found in (Zell, 2003).

The second approach for addressing infinite-dimensional constraints involves sampling-based methods, as was the strategy adopted within the learning dynamics work. Here, there are three key methods that merit future study. The first involves leveraging distributed optimization techniques such as ADMM (Boyd et al., 2011) to distribute and parallelize the constraints. A second approach is to collapse the pointwise LMI constraints into a single LMI that is the weighted average of all the $\mathcal{F}_\lambda$ matrices, where the weighting scales with the maximum eigenvalue of each of the $\mathcal{F}_\lambda$ matrices. This is a non-convex constraint that can be solved using alternation by fixing the weights from the previous iteration. Indeed, this is equivalent to re-writing all the LMI constraints in Lagrangian form and performing primal-dual descent on the complementarity condition. While the above methods can allow scaling in terms of *number of constraint points*, in order to scale with respect to problem dimensionality, one must resort to first-order unconstrained gradient descent methods, where non-degeneracy in the contraction metric can be achieved by parameterizing the dual metric W in Cholesky form LL^T. This opens the possibility of using more expressive function approximators such as deep-neural-networks, albeit, at the expense of a fully non-convex formulation.

Robust Motion Planning

Conservatism: Our analysis is conservative due to the fact that we derive a *globally valid* invariant tube, as opposed to the funnel library approach in which one computes bounds valid locally around a trajectory. This issue was slightly tempered by the introduction of less conservative *time-varying* tubes in Chapter 4, used within a local receding-horizon re-planner as for the planar quadrotor example. A more promising method for reducing the conservatism is to partition the state space into regions in which CCMs are computed *locally*, while ensuring sufficient smoothness at the region boundaries. Such a computational approach may also allow the use of lower-order polynomial expansions, thereby reducing the size of the SOS programs.

A second limitation of the approach is that the invariant tubes are computed based on a worst-case disturbance bound assumption. A less conservative solution may be achieved by decomposing the disturbance term into an unknown constant (or slowly-varying) mean, estimated online, plus zero-mean stochastic noise. In this way, one may leverage modern adaptive and disturbance estimation techniques to counteract the constant disturbance term, and a stochastic modification of CCM theory to obtain *probabilistic* invariance guarantees, e.g., based on super-martingale analysis (Steinhardt and Tedrake, 2012).

Perception and Estimation Uncertainty: In all of the planning work, we assumed full state knowledge and the ability to perfectly sense obstacles in the neighborhood of the robot. In reality, noisy sensors inject uncertainty into both these terms. In this case, providing *almost sure* guarantees is practically infeasible and one must additionally incorporate perception/sensing uncertainty, for instance, in a Bayesian sense, to obtain probabilistic guarantees. This is however, a well-known open problem.

Control-Theoretic Model-Based RL

Multi-Step Error: An immediate theoretically sound extension of the work to incorporate *multi-step* regression error would be to leverage the linearity in parameters of the estimated model and integrate the nonlinear features and the control signal in time. This would result in an inner-product between time-varying features and the parameters, thereby retaining linearity in parameters, with the supervisory signal now being state samples along the trajectory, as opposed to the time-derivative of the state. A similar transformation is central also to composite adaptive control for robotic systems (Slotine and Li, 1989; Wensing and Slotine, 2018). While such a transformation would additionally require the full control time signal for each sampled trajectory (as opposed to state-control samples), one eliminates the need for numerical estimation of the state time-derivative along the trajectory, thereby improving the signal-to-noise ratio.

Leveraging Model Priors: The problem studied in this work concerned learning the full dynamics model from scratch. However, one can often leverage physics-based modeling to form a baseline, reducing the role of learning to estimate the residual dynamics. Instead of adopting an additive

error formulation as is common in the literature, a variation of contraction theory – partial contraction (Wang and Slotine, 2005) – may be utilized to *smoothly interpolate* between the prior model and the model learned from data using an auxiliary dynamical system, termed the "virtual dynamics." This system has the special property in that the prior model and the learned model are both particular solutions to the virtual dynamics. Using similar stabilizability constraints, one can construct a model and feedback controller that forces the solution of the learned dynamics model (representing the true dynamics) to converge exponentially to the solution of the prior model, which can be designed to satisfy desirable performance criteria.

Learning Latent Dynamics: Finally, aside from the computational challenges associated with learning in high-dimensional output spaces, e.g., RGB images for visuo-servomotor control or manipulation via visual-tactile feedback, there is a fundamental theoretical challenge associated with representation. Recent works advocate for the use of planning in structured lower-dimensional latent spaces, e.g., (Lenz et al., 2015; Watter et al., 2015; Levine et al., 2016; Banijamali et al., 2018). Typically these latent spaces encodings are learned by minimizing the re-construction loss between the observed state transitions and the transitions decoded from predictions within the latent space. This is another example of model-fitting that is viewed purely from a statistical dynamics prediction point of view as opposed to within context of the downstream control task. An exciting future research area therefore is to formulate a set of appropriate control-theoretic regularizers for a more principled recovery of such latent spaces and the embedded dynamics.

13.2.2 Part III

We begin by first justifying our use of discrete models of uncertainty.

Justifying Discrete Model of Uncertainty: Throughout this part of the thesis, we assumed a *discrete* model of uncertainty for both the static and dynamic decision-making settings. While one would like to be able to address large or continuous sets of disturbances, we believe that a hierarchical representation of uncertainty is a more tractable approach. For instance, at the higher-level, one reasons about various uncertain *modes* of operation (e.g., the random erratic car maneuvers). At the lower-level, conditional on a given mode (e.g., deceleration), one may consider continuous models of uncertainty (e.g., the set of all robot deceleration profiles). The overall framework thus constitutes a mixture model. At the continuous lower-level of uncertainty (e.g., due to the natural variance in demonstrations), aspects such as risk-sensitivity are less relevant. Thus, this work studies risk-sensitivity at the *higher-level* hierarchy of decision making where discrete/modal models of uncertainty induce more nuanced behavior.

Feature Selection: As in the majority of IRL literature, we hand-picked features for the driving game. While performance on the test trajectory given ~ 1 minute of training data supported our choice of features, incorporating risk-sensitivity in large-scale IRL algorithms requires automatic feature

extraction. There has been some recent work on using deep neural nets within the MaxEnt IRL framework (Wulfmeier et al., 2015) as well as using general non-linear cost representations (Finn et al., 2016). A promising area of future research then is to embed the semi-parametric approach proposed in the previous chapter within deep cost networks to yield scalable RS-IRL algorithms.

Interaction: The inference framework assumes that the human expert is subject to an independent (non-interactive) source of disturbance. The natural extension therefore is to modularize the entire risk-sensitive IRL algorithm within a game-theoretic *interactive* setting involving multiple human agents and robotic systems. The key challenge here is to efficiently balance offline and online learning (see e.g., (Sadigh et al., 2016b; Waugh et al., 2011)), to enable the autonomous robot to actively infer intent and risk-sensitive preferences for the human agents, and use the resulting information to consequently *influence* the human agents.

Fusion with Generative Modeling: Recent work, e.g., (Schmerling et al., 2018), advocates for the use of generative time-series predictive models for human agents using recurrent neural networks. These generative models are given a hierarchical mixture model structure through the use of an intermediate discrete latent state. Due to the black-box nature of the modeling itself, the learned latent states are difficult to interpret in an interactive sense. An interesting area of work therefore would be to fuse such black-box models with the RS-IRL approach, particularly at the latent-state level of inference. That is, by associating each latent state realization with a particular instantiation of a CRM, one may be able to infuse such black-box models with the level of insight that accompanies RS-IRL.

We believe that the methods and results presented within this thesis along with the indicated future directions represent an important step towards endowing future robotic systems with the ability to *hierarchically* reason about planning and decision-making under uncertainty, which is crucial for safety-critical applications where humans and robots interact.

Bibliography

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *Int. Conf. on Machine Learning*.

Abbeel P and Ng AY (2005) Exploration and apprenticeship learning in reinforcement learning. In: *Int. Conf. on Machine Learning*.

Acerbi C (2002) Spectral measures of risk: A coherent representation of subjective risk aversion. *Journal of Banking & Finance* 26(7): 1505–1518.

Acerbi C and Tasche D (2002) On the coherence of expected shortfall. *Journal of Banking & Finance* 26(7): 1487–1503.

Adelstein IM and Epstein J (2017) Morse theory for the uniform energy. *Journal of Geometry* 108(3): 1193–1205.

Agha-mohammadi A, Chakravorty S and Amato NM (2014) FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *Int. Journal of Robotics Research* 33(2): 268–304.

Ahmadi AA and Majumdar A (2014) DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In: *IEEE Annual Conf. on Information Sciences and Systems*.

Ahmadi AA and Majumdar A (2016) Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters* 20(4): 709–729.

Allais M (1953) Le comportement de l'homme rationnel devant le risque: critique des postulats et axiomes de lécole américaine. *Econometrica* 21(4): 503–546.

Althoff D, Althoff M and Scherer S (2015) Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Althoff M and Dolan J (2014) Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics* 30(4): 903–918.

Aminzarey Z and Sontag ED (2014) Contraction methods for nonlinear systems: A brief introduction and some open problems. In: *Proc. IEEE Conf. on Decision and Control*.

Amos B, Rodriguez IDJ, Sacks J, Boots B and Kolter JZ (2018) Differentiable MPC for end-to-end planning and control. In: *Conf. on Neural Information Processing Systems*.

Angeli D (2002) A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control* 47(3): 410–422.

Angeli D (2009) Further results on incremental input-to-state stability. *IEEE Transactions on Automatic Control* 54(6): 1386–1391.

ApS M (2017) MOSEK optimization software. Available at `https://mosek.com/`.

Artzner P, Delbaen F, Eber JM and Heath D (1999) Coherent measures of risk. *Mathematical Finance* 9(3): 203–228.

Auslender A, Goberna MA and López MA (2009) Penalty and smoothing methods for convex semi-infinite programming. *Mathematics of Operations Research* 34(2): 303–319.

Avron H, Sindhwani V, Yang J and Mahoney MW (2016) Quasi-Monte Carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research* 17(120): 1–38.

Axelrod A, Carlone L, Chowdhary G and Karaman S (2016) Data-driven prediction of EVAR with confidence in time-varying datasets. In: *Proc. IEEE Conf. on Decision and Control*.

Banijamali E, Shu R, Ghavamzadeh M, Bui HH and Ghodsi A (2018) Robust locally-linear controllable embedding. In: *AI & Statistics*.

Bansal S, Akametalu AK, Jiang FJ, Laine F and Tomlin CJ (2016) Learning quadrotor dynamics using neural network for flight control. In: *Proc. IEEE Conf. on Decision and Control*.

Bansal S, Calandra R, Xiao T, Levine S and Tomlin CJ (2017a) Goal-driven dynamics learning via Bayesian optimization. In: *Proc. IEEE Conf. on Decision and Control*.

Bansal S, Chen M, Herbert S and Tomlin CJ (2017b) Hamilton-Jacobi reachability: A brief overview and recent advances. In: *Proc. IEEE Conf. on Decision and Control*.

Barberis NC (2013) Thirty years of prospect theory in economics: A review and assessment. *Journal of Economic Perspectives* 27(1): 173–195.

Barry AJ (2016) *High-speed autonomous obstacle avoidance with pushbroom stereo*. PhD Thesis, Massachusetts Inst. of Technology.

Barry AJ, Majumdar A and Tedrake R (2012) Safety verification of reactive controllers for uav flight in cluttered environments using barrier certificates. In: *Proc. IEEE Conf. on Robotics and Automation*.

Battilotti S and De Santis A (2003) Stabilization in probability of nonlinear stochastic systems with guaranteed region of attraction and target set. *IEEE Transactions on Automatic Control* 48(9): 1585–1599.

Bäuerle N and Ott J (2011) Markov decision processes with average-value-at-risk criteria. *Mathematics of Operations Research* 74(3): 361–379.

Bayer F, Bürger M and Allgöwer F (2013) Discrete-time incremental ISS: A framework for robust NMPC. In: *European Control Conference*.

Ben-Tal A and Nemirovski A (2001) *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM.

Berenson D, Diankov R, Nishikawi K, Kagami S and Kuffner J (2007) Grasp planning in complex scenes. In: *IEEE RAS Int. Conf. on Humanoid Robots*.

Berkenkamp F, Turchetta M, Schoellig A and Krause A (2017) Safe model-based reinforcement learning with stability guarantees. In: *Conf. on Neural Information Processing Systems*.

Betts JT (2010) *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*. Second edition. SIAM.

Blackmore L, Li H and Williams B (2006) A probabilistic approach to optimal robust path planning with obstacles. In: *American Control Conference*.

Blackmore L, Ono M and Williams BC (2011) Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics* 27(6): 1080–1094.

Bonalli R, Cauligi A, Bylard A and Pavone M (2019) GuSTO: guaranteed sequential trajectory optimization via sequential convex programming. In: *Proc. IEEE Conf. on Robotics and Automation*.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J et al. (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3(1): 1–122.

Brault R, Heinonen M and d'Alché Buc F (2016) Random Fourier features for operator-valued kernels. In: *Asian Conf. on Machine Learning*. pp. 110–125.

Bravo JM, Alamo T and Camacho EF (2006) Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica* 42(10): 1745–1751.

Buehler EA, Paulson JA and Mesbah A (2016) Lyapunov-based stochastic nonlinear model predictive control: Shaping the state probability distribution functions. In: *American Control Conference*.

Burke JV, Lewis AS and Overton ML (2005) A robust gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization* 15(3): 751–779.

Burridge RR, Rizzi AA and Koditschek DE (1999) Sequential composition of dynamically dexterous robot behaviors. *Int. Journal of Robotics Research* 18(6): 534–555.

Cannon M, Buerger J, Kouvaritakis B and Rakovic S (2011) Robust tubes in nonlinear model predictive control. *IEEE Transactions on Automatic Control* 56(8): 1942–1947.

Carson III JM, Açikmeşe B, Murray RM and MacMartin DG (2013) A robust model predictive control algorithm augmented with a reactive safety mode. *Automatica* 49(5): 1251–1260.

Carton D, Nitsch V, Meinzer D and Wollherr D (2016) Towards assessing the human trajectory planning horizon. *PLoS ONE* 11(12): e0167021.

Charnes A and Cooper WW (1959) Chance-constrained programming. *Management Science* 6(1): 73–79.

Chebotar Y, Hausman K, Zhang M, Sukhatme G, Schaal S and Levine S (2017) Combining model-based and model-free updates for trajectory-centric reinforcement learning. In: *Int. Conf. on Machine Learning*.

Chen M, Bansal S, Fisac JF and Tomlin CJ (2018a) Robust sequential path planning under disturbances and adversarial intruder. *IEEE Transactions on Control Systems Technology* In press.

Chen M, Herbert S and Tomlin CJ (2016a) Fast reachable set approximations via state decoupling disturbances. In: *Proc. IEEE Conf. on Decision and Control*.

Chen M, Herbert SL, Vashishtha MS, Bansal S and Tomlin CJ (2016b) Decomposition of reachable sets and tubes for a class of nonlinear systems. Available at `https://arxiv.org/abs/1611.00122`.

Chen M, Herbert SL, Vashishtha MS, Bansal S and Tomlin CJ (2018b) Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control* In press.

Chen X (2012) Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical Programming* 134(1): 71–99.

Chitsaz H and LaValle SM (2007) Time-optimal paths for a Dubins airplane. In: *Proc. IEEE Conf. on Decision and Control*.

Chow Y (2017) *Risk-Sensitive and Data-Driven Sequential Decision Making*. PhD Thesis, Stanford University, Dept. of Aeronautics and Astronautics.

Chow Y, Ghavamzadeh M, Janson L and Pavone M (2018) Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research* .

Chow Y and Pavone M (2014) A framework for time-consistent, risk-averse model predictive control: Theory and algorithms. In: *American Control Conference*.

Chow Y, Tamar A, Mannor S and Pavone M (2015) Risk-sensitive and robust decision-making: a CVaR optimization approach. In: *Conf. on Neural Information Processing Systems*.

Chua K, Calandra R, McAllister R and Levine S (2018) Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114* .

Coddington EA and Levinson N (1955) *Theory of Ordinary Differential Equations*. McGraw-Hill.

Crouch PE and van der Schaft AJ (1987) *Variational and Hamiltonian Control Systems*. Springer.

Cui R, Yang C, Li Y and Sharma S (2017) Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. *IEEE Transactions on Systems, Man, & Cybernetics: Systems* 47(6): 1019–1029.

D'Aspremont A and Karoui NE (2014) A stochastic smoothing algorithm for semidefinite programming. *SIAM Journal on Optimization* 24(3): 1138–1177.

Dean S, Tu S, Matni N and Recht B (2019) Safely learning to control the constrained linear quadratic regulator. In: *American Control Conference*. In press.

Deisenroth MP and Rasmussen CE (2011) PILCO: A model-based and data-efficient approach to policy search. In: *Int. Conf. on Machine Learning*. pp. 465–472.

Eichhorn A and Römisch W (2005) Polyhedral risk measures in stochastic programming. *SIAM Journal on Optimization* 16(1): 69–95.

Ellsberg D (1961) Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics* 75(4): 643–669.

Englert P and Toussaint M (2015) Inverse KKT learning cost functions of manipulation tasks from demonstrations. In: *Int. Symp. on Robotics Research*.

Faessler M, Franchi A and Scaramuzza D (2018) Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters* 3(2): 620–626.

Fahroo F and Ross IM (2002) Direct trajectory optimization by a Chebyshev pseudospectral method. *AIAA Journal of Guidance, Control, and Dynamics* 25(1): 160–166.

Farina M and Scattolini R (2012) Tube-based robust sampled-data MPC for linear continuous-time systems. *Automatica* 48(7): 1473–1476.

Filar JA, Kallenberg LCM and Lee HM (1989) Variance-penalized Markov decision processes. *Mathematics of Operations Research* 14(1): 147–161.

Finn C, Levine S and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: *Int. Conf. on Machine Learning*.

Fisac JF, Akametalu AK, Zeilinger MN, Kaynama S, Gillula J and Tomlin CJ (2017) A general safety framework for learning-based control in uncertain robotic systems. Available at `https://arxiv.org/abs/1705.01292`.

Fleming J, Kouvaritakis B and Cannon M (2015) Robust tube MPC for linear systems with multiplicative uncertainty. *IEEE Transactions on Automatic Control* 60(4): 1087–1092.

Forni F and Sepulchre R (2014) A differential Lyapunov framework for contraction analysis. *IEEE Transactions on Automatic Control* 3(59): 614–628.

Frazzoli E, Dahleh MA and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21(6): 1077–1091.

Fridovich-Keil D, Herbert SL, Fisac JF, Deglurkar S and Tomlin CJ (2018) Planning, fast and slow: A framework for adaptive real-time safe trajectory planning. In: *Proc. IEEE Conf. on Robotics and Automation*.

Geibel P and Wysotzki F (2005) Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research* 24(1): 81–108.

Gilboa I and Marinacci M (2016) Ambiguity and the Bayesian paradigm. In: *Readings in Formal Epistemology*, first edition, chapter 21.

Gillula JH, Huang H, Vitus MP and Tomlin CJ (2010) Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In: *Proc. IEEE Conf. on Robotics and Automation*.

Glimcher PW and Fehr E (2014) *Neuroeconomics*. Second edition. Elsevier.

Goberna MA and López MA (2018) Recent contributions to linear semi-infinite optimization: an update. *Annals of Operations Research* 271(1): 237–278.

Gul F and Pesendorfer W (2014) Expected uncertain utility theory. *Econometrica* 82(1): 1–39.

Hauser K (2018) Semi-infinite programming for trajectory optimization with nonconvex obstacles. In: *Workshop on Algorithmic Foundations of Robotics*.

Hearst MA, Dumais ST, Osuna E, Platt J and Scholkopf B (1998) Support vector machines. *IEEE Intelligent Systems and their Applications* 13(4): 18–28.

Helwa MK, Heins A and Schoellig AP (2019) Provably robust learning-based approach for high-accuracy tracking control of lagrangian systems. *IEEE Robotics and Automation Letters* 4(2): 1587–1594.

Heng L, Gotovos A, Krause A and Pollefeys M (2015) Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In: *Proc. IEEE Conf. on Robotics and Automation*.

Herbert SL, Chen M, Han S, Bansal S, Fisac JF and Tomlin CJ (2017) FaSTrack: a modular framework for fast and guaranteed safe motion planning. In: *Proc. IEEE Conf. on Decision and Control*.

Hernandéz JD, Moll M, Vidal E, Carreras M and Kavraki LE (2016) Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Hettich R and Kortanek KO (1993) Semi-infinite programming: Theory, methods, and applications. *SIAM Review* 35(3): 380–429.

Howard R and Matheson J (1972) Risk-sensitive Markov decision processes. *Management Science* 8(7): 356–369.

Hsu M, Bhatt M, Adolphs R, Tranel D and Camerer CF (2005) Neural systems responding to degrees of uncertainty in human decision-making. *Science* 310(5754): 1680–1683.

Huang PS, Avron H, Sainath TN, Sindhwani V and Ramabhadran B (2014) Kernel methods match deep neural networks on TIMIT. In: *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, pp. 205–209.

Ichter B, Harrison J and Pavone M (2018) Learning sampling distributions for robot motion planning. In: *Proc. IEEE Conf. on Robotics and Automation*.

Iwasaki T and Hara S (2005) Generalized KYP lemma: unified frequency domain inequalities with design applications. *IEEE Transactions on Automatic Control* 50(1): 41–59.

Janson L, Schmerling E, Clark A and Pavone M (2015a) Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions. *Int. Journal of Robotics Research* 34(7): 883–921.

Janson L, Schmerling E and Pavone M (2015b) Monte Carlo motion planning for robot trajectory optimization under uncertainty. In: *Int. Symp. on Robotics Research*.

Jouffroy J (2003) A simple extension of contraction theory to study incremental stability properties. In: *European Control Conference*.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2): 99–134.

Kahneman D and Tversky A (1979) Prospect theory: An analysis of decision under risk. *Econometrica* : 263–291.

Kai JM, Allibert G, Hua MD and Hamel T (2017) Nonlinear feedback control of quadrotors exploiting first-order drag effects. In: *IFAC World Congress*.

Kamthe S and Deisenroth MP (2018) Data-efficient reinforcement learning with probabilistic model predictive control. In: *AI & Statistics*.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research* 30(7): 846–894.

Khalil HK (2002) *Nonlinear Systems*. Third edition. Prentice Hall.

Khansari-Zadeh SM and Billard A (2011) Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics* 27(5): 943–957.

Khansari-Zadeh SM and Khatib O (2017) Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots* 41(1): 45–69.

Kocijan J, Murray-Smith R, Rasmussen CE and Girard A (2004) Gaussian process model based predictive control. In: *American Control Conference*.

Kögel M and Findeisen R (2015) Discrete-time robust model predictive control for continuous-time nonlinear systems. In: *American Control Conference*.

Köhler J, Soloperto R, A MM and Algöwer F (2019) A computationally efficient robust model predictive control framework for uncertain nonlinear systems. Submitted to IEEE Transactions on Automatic Control; Available at: `https://www.ist.uni-stuttgart.de/de/institut/team/PDFs_MA-Seiten/JK/Robust_Nonlin.pdf` .

Kolter JZ, Abbeel P and Ng AY (2007) Hierarchical apprenticeship learning with application to quadruped locomotion. In: *Conf. on Neural Information Processing Systems*.

Kong S, Gao S, Chen W and Clarke E (2015) dreach: δ-Reachability analysis for hybrid systems. In: *Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems* .

Kortanek KO and No H (1992) A central cutting plane algorithm for convex semi-infinite programming problems. *SIAM Journal on Optimization* 3(4): 901–918.

Kousik S, Vaskov S, Johnson-Roberson M and Vasudevan R (2017) Safe trajectory synthesis for autonomous driving in unforeseen environments. In: *Proc. ASME Dynamic Systems and Control Conference*.

Kretzschmar H, Spies M, Sprunk C and Burgard W (2016) Socially compliant mobile robot navigation via inverse reinforcement learning. *Int. Journal of Robotics Research* 35(11): 1289–1307.

Kuderer M, Gulati S and Burgard W (2015) Learning driving styles for autonomous vehicles from demonstration. In: *Proc. IEEE Conf. on Robotics and Automation*.

Kurdila AJ and Zabarankin M (2006) *Convex functional analysis*. Birkhäuser Basel edition. Springer Science & Business Media.

Kurniawati H, Hsu D and Lee WS (2008) SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Robotics: Science and Systems*.

Langson W, Chryssochoos I, Raković SV and Mayne DQ (2004) Robust model predictive control using tubes. *Automatica* 40(1): 125–133.

Lanza A, Morigi S, Selesnick I and Sgallari F (2017) Nonconvex nonsmooth optimization via convexnonconvex majorizationminimization. *Numerische Mathematik* 136(2): 343–381.

LaValle SM (2011) Motion planning: Wild frontiers. *IEEE Robotics and Automation Magazine* 18(2): 108–118.

LaValle SM and Kuffner JJ (2001) Randomized kinodynamic planning. *Int. Journal of Robotics Research* 20(5): 378–400.

Lax P (2007) *Linear Algebra and its Applications*. 2 edition. John Wiley & Sons.

Lemme A, Neumann K, Reinhart RF and Steil JJ (2014) Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing* 141(1): 3–14.

Lenz I, Knepper R and Saxena A (2015) DeepMPC: Learning deep latent features for model predictive control. In: *Robotics: Science and Systems*.

Leung K and Manchester IR (2017) Chebyshev pseudospectral method for nonlinear stabilization using control contraction metrics. In: *American Control Conference*. To Appear.

Levine S, Finn C, Darrell T and Abbeel P (2016) End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(1): 1–40.

Levine S and Koltun V (2012) Continuous inverse optimal control with locally optimal examples. In: *Int. Conf. on Machine Learning*.

Li Y, Littlefield Z and Bekris KE (2016) Asymptotically optimal sampling-based kinodynamic planning. *Int. Journal of Robotics Research* 35(5): 528564.

Liang T and Rakhlin A (2018) Just interpolate: Kernel ridgeless regression can generalize. *arXiv preprint arXiv:1808.00387v1* .

Likhachev M and Ferguson D (2009) Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. Journal of Robotics Research* 28(8): 933–945.

Limon D, Alvarado I, Alamo T and Camacho EF (2010) Robust tube-based MPC for tracking of constrained linear systems with additive disturbances. *Journal of Process Control* 20(3): 248–260.

Lin Y and Saripalli S (2014) Path planning using 3D Dubins curve for unmanned aerial vehicles. In: *IEEE Int. Conf. on Unmanned Aircraft Systems*.

Lin Y and Sontag ED (1991) A universal formula for stabilization with bounded controls. *System and Control Letters* 16(6): 393–397.

Liu C and Atkeson CG (2009) Standing balance control using a trajectory library. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Liu GX (2007) A homotopy interior point method for semi-infinite programming problems. *Journal of Global Optimization* 37(4): 631–646.

Löfberg J (2004) YALMIP : A toolbox for modeling and optimization in MATLAB. In: *IEEE Int. Symp. on Computer Aided Control Systems Design*.

Lohmiller W and Slotine JJE (1998) On contraction analysis for non-linear systems. *Automatica* 34(6): 683–696.

Lopez BT, Slotine JJE and How JP (2018) Robust collision avoidance via sliding control. In: *Proc. IEEE Conf. on Robotics and Automation*.

López M and Still G (2007) Semi-infinite programming. *European Journal of Operational Research* 180(2): 491–518.

Lorenzetti J, Landry B, Singh S and Pavone M (2019) Reduced order model predictive control for setpoint tracking. In: *European Control Conference*.

Lu Z and Pong TK (2011) Minimizing condition number via convex programming. *SIAM Journal on Matrix Analysis and Applications* 32(4): 1193–1211.

Luders B, Ellertson A, How JP and Sugel I (2016) Wind uncertainty modeling and robust trajectory planning for autonomous parafoils. *AIAA Journal of Guidance, Control, and Dynamics* 39(7): 1614–1630.

Majumdar A, Ahmadi AA and Tedrake R (2013) Control design along trajectories with sums of squares programming. In: *Proc. IEEE Conf. on Robotics and Automation*.

Majumdar A and Pavone M (2017) How should a robot assess risk? Towards an axiomatic theory of risk in robotics. In: *Int. Symp. on Robotics Research*.

Majumdar A, Singh S, Mandlekar A and Pavone M (2017) Risk-sensitive inverse reinforcement learning via coherent risk models. In: *Robotics: Science and Systems*.

Majumdar A and Tedrake R (2012) Robust online motion planning with regions of finite time invariance. In: *Workshop on Algorithmic Foundations of Robotics*.

Majumdar A and Tedrake R (2013) Robust online motion planning with regions of finite time invariance. In: *Algorithmic Foundations of Robotics X*. Springer.

Majumdar A and Tedrake R (2017) Funnel libraries for real-time robust feedback motion planning. *Int. Journal of Robotics Research* 36(8): 947–982.

Majumdar A, Tobenkin MM and Tedrake R (2012) Algebraic verification for parameterized motion planning libraries. In: *American Control Conference*.

Majumdar A, Vasudevan R, Tobenkin MM and Tedrake R (2014) Convex optimization of nonlinear feedback controllers via occupation measures. *Int. Journal of Robotics Research* 33(9): 1209–1230.

Manchester I, Tang JZ and Slotine JJE (2015) Unifying classical and optimization-based methods for robot tracking control with control contraction metrics. In: *Int. Symp. on Robotics Research*.

Manchester IR and Slotine JJE (2014) Output-feedback control of nonlinear systems using control contraction metrics and convex optimization. In: *Australian Control Conference*.

Manchester IR and Slotine JJE (2017) Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control* In Press.

Manchester Z and Kuindersma S (2017) DIRTREL: Robust trajectory optimization with ellipsoidal disturbances and LQR feedback. In: *Robotics: Science and Systems*.

Manchester Z and Kuindersma S (2019) Robust direct trajectory optimization using approximate invariant funnels. *Autonomous Robots* 43(2): 375–387.

Mayne DQ (2014) Model predictive control: Recent developments and future promise. *Automatica* 50(12): 2967–2986.

Mayne DQ, Kerrigan EC, van Wyk EJ and Falugi P (2011) Tube-based robust nonlinear model predictive control. *Int. Journal of Robust and Nonlinear Control* 21(11): 1341–1353.

Mayne DQ, Seron MM and Raković SV (2005) Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica* 41(2): 219–224.

Medina JR and Billard A (2017) Learning stable task sequences from demonstration with linear parameter varying systems and hidden Markov models. In: *Conf. on Robot Learning*. pp. 175–184.

Mehrotra S and Papp D (2014) A cutting surface algorithm for semi-infinite convex programming with an application to moment robust optimization. *SIAM Journal on Optimization* 24(4): 1670–1697.

Micheli M and Glaunés JA (2014) Matrix-valued kernels for shape deformation analysis. *Geometry, Imaging and Computing* 1(1): 57–139.

Mihatsch O and Neuneier R (2002) Risk-sensitive reinforcement learning. *Machine Learning* 49(2): 267–290.

Minh HQ (2016) Operator-valued Bochner theorem, Fourier feature maps for operator-valued kernels, and vector-valued learning. *arXiv preprint arXiv:1608.05639* .

Mohajerin N, Mozifian M and Waslander S (2019) Deep learning a quadrotor dynamic model for multi-step prediction. In: *Proc. IEEE Conf. on Robotics and Automation*.

Mombaur K, Truong A and Laumond JP (2010) From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots* 28(3): 369–383.

Nagabandi A, Kahn G, Fearing RS and Levine S (2017) Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596* .

Ng A and Russell S (2000) Algorithms for inverse reinforcement learning. In: *Int. Conf. on Machine Learning*.

Nguyen VB, Shew RL and Xia Y (2016) Maximizing the sum of a generalized Rayleigh quotient and another Rayleigh quotient on the unit sphere via semidefinite programming. *Journal of Global Optimization* 64(2): 399–416.

Nilim A and El Ghaoui L (2005) Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53(5): 780–798.

Ohnishi M, Wang L, Notomista G and Egerstedt M (2019) Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on Robotics* Early access.

Olfati-Saber R (2001) *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. PhD Thesis, Massachusetts Inst. of Technology.

Ono M, Williams BC and Blackmore L (2013) Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research* 46(1): 511–577.

Osogami T (2012) Robustness and risk-sensitivity in Markov decision processes. In: *Conf. on Neural Information Processing Systems*.

Ostafew C, Schoellig AP and Barfoot TD (2016) Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *Int. Journal of Robotics Research* 35(13): 1547–1563.

Overton ML and Womersley RS (1995) Second derivatives for optimizing eigenvalues of symmetric matrices. *SIAM Journal on Matrix Analysis and Applications* 16(3): 697–718.

Owen M, Beard RW and McLain TW (2015) Implementing Dubins airplane paths on fixed-wing UAVs. In: *Handbook of Unmanned Aerial Vehicles*. Springer Dordrecht.

Park T and Levine S (2013) Inverse optimal control for humanoid locomotion. In: *Robotics: Science and Systems Workshop on Inverse Optimal Control and Robotic Learning from Demonstration*.

Parrilo PA (2000) *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD Thesis, Massachusetts Inst. of Technology.

Petrik M and Subramanian D (2012) An approximate solution method for large risk-averse Markov decision processes. In: *Proc. Conf. on Uncertainty in Artificial Intelligence*.

Pin G, Raimondo DM, Magni L and Parisini T (2009) Robust model predictive control of nonlinear systems with bounded and state-dependent uncertainties. *IEEE Transactions on Automatic Control* 54(7): 1681–1687.

Platt R, Kaelbling L, Tomas LP and Tedrake R (2012) Non-Gaussian belief space planning: Correctness and complexity. In: *Proc. IEEE Conf. on Robotics and Automation*.

Polydoros AS and Nalpantidis L (2017) Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems* 86(2): 153–173.

Posa M, Koolen T and Tedrake R (2017) Balancing and step recovery capturability via sums-of-squares optimization. In: *Robotics: Science and Systems*.

Prajna S, Jadbabaie A and Pappas GJ (2007) A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control* 52(8): 1415–1428.

Prashanth LA, Jie C, Fu M, Marcus S and Szepesvári C (2016) Cumulative prospect theory meets reinforcement learning: Prediction and control. In: *Int. Conf. on Machine Learning*.

Prentice S and Roy N (2009) The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. *Int. Journal of Robotics Research* 28(11-12): 1448–1465.

Primbs JA, Nevistić V and Doyle JC (1999) Nonlinear optimal control: A control Lyapunov function and receding horizon perspective. *Asian Journal of Control* 1(1): 14–24.

Punjani A and Abbeel P (2015) Deep learning helicopter dynamics models. In: *Proc. IEEE Conf. on Robotics and Automation*.

Putinar M (1993) Positive polynomials on compact semi-algebraic sets. *Indiana Univ. Mathematics Journal* 42(3): 969–984.

Quiggin J (1982) A theory of anticipated utility. *Journal of Economic Behavior & Organization* 3(4): 323–343.

Quinlan S and Khatib O (1993) Elastic bands: Connecting path planning and control. In: *Proc. IEEE Conf. on Robotics and Automation*.

Rabin M (2000) Risk aversion and expected-utility theory: A calibration theorem. *Econometrica* 68(5): 1281–1292.

Rahimi A and Recht B (2007) Random features for large-scale kernel machines. In: *Conf. on Neural Information Processing Systems*. pp. 1177–1184.

Rahimi A and Recht B (2008) Uniform approximation of functions with random bases. In: *Allerton Conf. on Communications, Control and Computing*. IEEE.

Rajamani R (2012) *Vehicle Dynamics and Control*. 2 edition. Boston, MA: Springer. DOI:10.1007/978-1-4614-1433-9.

Raković SV (2009) Set theoretic methods in model predictive control. In: *Nonlinear Model Predictive Control*. Springer Berlin Heidelberg.

Raković SV, Kerrigan EC, Mayne DQ and Lygeros J (2006a) Reachability analysis of discrete-time systems with disturbances. *IEEE Transactions on Automatic Control* 51(4): 546–561.

Rakovic SV, Kouvaritakis B, Cannon M, Panos C and Findeisen R (2012) Parameterized tube model predictive control. *IEEE Transactions on Automatic Control* 57(11): 2746–2761.

Raković SV, Teel AR, Mayne DQ and Astolfi A (2006b) Simple robust control invariant tubes for some classes of nonlinear discrete time systems. In: *Proc. IEEE Conf. on Decision and Control*.

Ramachandran D and Amir E (2007) Bayesian inverse reinforcement learning. In: *Int. Joint Conf. on Artificial Intelligence*.

Ratliff LJ and Mazumdar E (2017) Risk-sensitive inverse reinforcement learning via gradient methods. Available at `https://arxiv.org/abs/1703.09842`.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) CHOMP: Gradient optimization techniques for efficient motion planning. In: *Proc. IEEE Conf. on Robotics and Automation*.

Ravichandar H, Salehi I and Dani A (2017) Learning partially contracting dynamical systems from demonstrations. In: *Conf. on Robot Learning*.

Rayguru MM and Kar IN (2015) Contraction based stabilization of approximate feedback linearizable systems. In: *European Control Conference*.

Reyes-Báez R, van der Schaft A and Jayawardhana B (2017) Tracking control of fully-actuated port-hamiltonian mechanical systems via sliding manifolds and contraction analysis. In: *IFAC World Congress*.

Reyhanoglu M, van der Schaft A, McClamroch NH and Kolmanovsky I (1999) Dynamics and control of a class of underactuated mechanical systems. *IEEE Transactions on Automatic Control* 44(9): 1663–1671.

Richter C, Bry A and Roy N (2016) Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: *Robotics Research*. Springer.

Rockafellar RT (2007) Coherent approaches to risk in optimization under uncertainty. In: *OR Tools and Applications: Glimpses of Future Technologies*, chapter 3. INFORMS.

Rockafellar RT and Uryasev S (2000) Optimization of conditional value-at-risk. *Journal of Risk* 2: 21–41.

Rockafellar RT and Uryasev S (2002) Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance* 26(7): 1443–1471.

Rubagotti M, Raimondo DM, Ferrara A and Magni L (2011) Robust model predictive control with integral sliding mode in continuous-time sampled-data nonlinear systems. *IEEE Transactions on Automatic Control* 56(3): 556–570.

Russell S (1998) Learning agents for uncertain environments. In: *Proc. Computational Learning Theory*.

Ruszczyński A (2010) Risk-averse dynamic programming for Markov decision process. *Mathematical Programming* 125(2): 235–261.

Sadigh D, Sastry S, Seshia SA and Dragan AD (2016a) Planning for autonomous cars that leverage effects on human actions. In: *Robotics: Science and Systems*.

Sadigh D, Sastry SS, Seshia SA and Dragan A (2016b) Information gathering actions over human internal state. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Sanner RM and Slotine JJE (1992) Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks* 3(6): 837–863.

Schmerling E, Janson L and Pavone M (2015a) Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In: *Proc. IEEE Conf. on Decision and Control*.

Schmerling E, Janson L and Pavone M (2015b) Optimal sampling-based motion planning under differential constraints: the driftless case. In: *Proc. IEEE Conf. on Robotics and Automation*. Extended version available at `http://arxiv.org/abs/1403.2483/`.

Schmerling E, Leung K, Vollprecht W and Pavone M (2018) Multimodal probabilistic model-based planning for human-robot interaction. In: *Proc. IEEE Conf. on Robotics and Automation*.

Schmerling E and Pavone M (2017) Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. In: *Robotics: Science and Systems*.

Scholköpf B and Smola AJ (2001) *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press.

Schulman J, Ho J, Lee A, Awwal I, Bradlow H and Abbeel P (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: *Robotics: Science and Systems*.

Schwartz L (1964) Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *Analyse Mathématique* 13(1): 115–256.

Scott JK and Barton PI (2013) Bounds on the reachable sets of nonlinear control systems. *Automatica* 49(1): 93–100.

Shapiro A (2009) On a time consistency concept in risk averse multi-stage stochastic programming. *Operations Research Letters* 37(3): 143–147.

Shapiro A, Dentcheva D and Ruszczyński A (2014) *Lectures on stochastic programming: Modeling and theory*. Second edition. SIAM.

Sharma BB and Kar IN (2009) Contraction based adaptive control of a class of nonlinear systems. In: *American Control Conference*.

Shen Y, Tobia MJ, Sommer T and Obermayer K (2014) Risk-sensitive reinforcement learning. *Neural Computation* 26(7): 1298–1328.

Shi G, Shi X, O'Connell M, Yu R, Azizzadenesheli K, Anandkumar A, Yue Y and Chung S (2019) Neural lander: Stable drone landing control using learned dynamics. In: *Proc. IEEE Conf. on Robotics and Automation*.

Simpson-Porco JW and Bullo F (2014) Contraction theory on Riemannian manifolds. *System and Control Letters* 65(1): 74–80.

Sindhwani V, Tu S and Khansari M (2018) Learning contracting vector fields for stable imitation learning. *arXiv preprint arXiv:1804.04878* .

Singh S, Chow YL, Majumdar A and Pavone M (2018) A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms. *IEEE Transactions on Automatic Control* 64(7): 2905–2912. Extended version available at: `http://arxiv.org/abs/1703.01029`.

Singh S, D'Amico S and Pavone M (2015a) High-fidelity modeling and control system synthesis for a drag-free microsatellite. In: *Int. Symp. on Space Flight Dynamics*.

Singh S, Majumdar A, Slotine JJE and Pavone M (2017) Robust online motion planning via contraction theory and convex optimization. In: *Proc. IEEE Conf. on Robotics and Automation*. Extended Version, Available at `http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf`.

Singh S, Schmerling E and Pavone M (2015b) Decentralized algorithms for 3D symmetric formations in robotic networks - a contraction theory approach. In: *Proc. IEEE Conf. on Robotics and Automation*.

Slotine JJE (2007) Sliding controller design for non-linear systems. *Int. Journal of Control* 38(2): 421–434.

Slotine JJE and Li W (1987) On the adaptive control of robot manipulators. *Int. Journal of Robotics Research* 6(3): 49–59.

Slotine JJE and Li W (1989) Composite adaptive control of robot manipulators. *Automatica* 25(4): 509–519.

Sontag ED (2010) Contractive systems with inputs. In: *Perspectives in Mathematical System Theory, Control, and Signal Processing*. Springer Berlin Heidelberg.

Sontag ED, Margaliot M and Tuller T (2014) On three generalizations of contraction. In: *Proc. IEEE Conf. on Decision and Control*.

Spivak M (1999) *A Comprehensive Introduction to Differential Geometry, Vol. I*. Third edition. Publish or Perish, Inc.

Spong MW (1998) Underactuated mechanical systems. In: *Control Problems in Robotics and Automation*. Springer Berlin Heidelberg.

Stein O and Steuermann P (2012) The adaptive convexification algorithm for semi-infinite programming with arbitrary index sets. *Mathematical Programming* 136(1): 183–207.

Steinhardt J and Tedrake R (2012) Finite-time regional verification of stochastic non-linear systems. *Int. Journal of Robotics Research* 31(7): 901–923.

Sun W, Patil S and Alterovitz R (2015) High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics* 31(1): 104–116.

Tal E and Karaman S (2018) Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. In: *Proc. IEEE Conf. on Decision and Control*.

Tamar A, Chow Y, Ghavamzadeh M and Mannor S (2016) Sequential decision making with coherent risk. *IEEE Transactions on Automatic Control* 62(7): 3323–3338.

Tamar A, Di Castro D and Mannor S (2012) Policy gradients with variance related risk criteria. In: *Int. Conf. on Machine Learning*.

Tanabe K and Chen M (2018) BEACLS: Berkeley Efficient API in C++ for Level Set methods. Available at `https://github.com/HJReachability/beacls`.

Taylor AJ, Dorobantu VD, Le HM, Yue Y and Ames AD (2019) Episodic learning with control Lyapunov functions for uncertain robotic systems. Available at `https://arxiv.org/abs/1903.01577`.

Tedrake R, Manchester IR, Tobenkin M and Roberts JW (2010) LQR-trees: Feedback motion planning via sums-of-squares verification. *Int. Journal of Robotics Research* 29(8): 1038–1052.

Thuruthel TG, Falotico E, Renda F and Laschi C (2019) Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics* 35(1): 124–134.

Tobenkin MM, Permenter F and Megretski A (2013) SPOTLESS polynomial and conic optimization. Software available from `https://github.com/spottoolbox/spotless`.

Tomlin CJ, Mitchell I, Bayen AM and Oishi M (2003) Computational techniques for the verification of hybrid systems. *Proc. of the IEEE* 91(7): 986–1001.

Venkatraman A, Capobianco R, Pinto L, Hebert M, Nardi D and Bagnell A (2016) Improved learning of dynamics models for control. In: *Int. Symp. on Experimental Robotics*. Springer, pp. 703–713.

Venkatraman A, Hebert M and Bagnell JA (2015) Improving multi-step prediction of learned time series models. In: *Proc. AAAI Conf. on Artificial Intelligence*.

Vidal E, Moll M, Palomeras N, Hernández JD, Carreras M and Kavraki LE (2019) Online multi-layered motion planning with dynamic constraints for autonomous underwater vehicles. In: *Proc. IEEE Conf. on Robotics and Automation*.

Villanueva ME, Quirynen R, Diehl M, Chachuat B and Houska B (2017) Robust MPC via minmax differential inequalities. *Automatica* 77(1): 311–321.

VIRES Simulationstechnologie GmbH (2017) VTD - Virtual Test Drive. Available at `https://vires.com/vtd-vires-virtual-test-drive/`.

von Neumann J and Morgenstern O (1944) *Theory of Games and Economic Behavior*. Princeton Univ. Press.

Wang W and Slotine JJE (2005) On partial contraction analysis for coupled nonlinear oscillators. *Biological Cybernetics* 92(1): 38–53.

Wang YS, Matni N and Doyle JC (2019) A system level approach to controller synthesis. *IEEE Transactions on Automatic Control* Early Access.

Wang Z, Singh S, Pavone M and Schwager M (2018) Cooperative object transport in 3D with multiple quadrotors using no peer communication. In: *Proc. IEEE Conf. on Robotics and Automation*.

Watter M, Springenberg JT, Boedecker J and Riedmiller MA (2015) Embed to Control: A locally linear latent dynamics model for control from raw images. In: *Conf. on Neural Information Processing Systems*.

Waugh K, Ziebart BD and Bagnell JA (2011) Computational rationalization: The inverse equilibrium problem. In: *Int. Conf. on Machine Learning*.

Wensing PW and Slotine JJ (2018) Cooperative adaptive control for cloud-based robotics. In: *Proc. IEEE Conf. on Robotics and Automation*.

Wu C and Yuanlie L (1999) Minimizing risk models in Markov decision process with policies depending on target values. *Journal of Mathematical Analysis and Applications* 231(1): 47–67.

Wulfmeier M, Ondruska P and Posner I (2015) Maximum entropy deep inverse reinforcement learning. Available at `https://arxiv.org/abs/1507.04888`.

Xu H and Mannor S (2010) Distributionally robust Markov decision processes. In: *Conf. on Neural Information Processing Systems*.

Yaari ME (1987) The dual theory of choice under risk. *Econometrica* 55(1): 95–115.

Yu S, Böhm C, Chen H and Allgöwer F (2010) Robust model predictive control with disturbance invariant sets. In: *American Control Conference*.

Yu S, Maier C, Chen H and Allgöwer F (2013) Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems. *System and Control Letters* 62(2): 194–200.

Zamani M, van de Wouw N and Majumdar R (2013) Backstepping controller synthesis and characterizations of incremental stability. *System and Control Letters* 62(10): 949–962.

Zell T (2003) *Quantitative study of semi-Pfaffian sets*. PhD Thesis, Georgia Institute of Technology.

Zhang L, Wu SY and López MA (2010) A new exchange method for convex semi-infinite programming. *SIAM Journal on Optimization* 20(6): 2959–2977.

Zhou DX (2008) Derivative reproducing properties for kernel methods in learning theory. *Journal of Computational and Applied Mathematics* 220(1-2): 456–463.

Zhou S, Helwa MK and Schoellig AP (2017) Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking. In: *Proc. IEEE Conf. on Decision and Control*.

Zhu Z, Wang S, Leung H and Ding Z (2000) Matrix filter design using semi-infinite programming with application to doa estimation. *IEEE Transactions on Signal Processing* 48(1): 267–271.

Ziebart BD, Maas A, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *Proc. AAAI Conf. on Artificial Intelligence*.

Ziebart BD, Ratliff N, Gallagher G, Mertz C, Peterson K, Bagnell JA, Hebert M, Key AK and Srinivasa S (2009) Planning-based prediction for pedestrians. In: *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*.

Zucker M, Bagnell JA, Atkeson CG and Kuffner J (2010) An optimization approach to rough terrain locomotion. In: *Proc. IEEE Conf. on Robotics and Automation*.