Fast, Optimal, and Safe Motion Planning for Bipedal Robots

by

Pengcheng Zhao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2020

Doctoral Committee:

Assistant Professor Ram Vasudevan, Chair
Associate Professor Robert Gregg
Professor Jessy Grizzle
Assistant Professor Kenneth Shorter

Pengcheng Zhao
pczhao@umich.edu
ORCID iD: 0000-0001-6431-6926

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF APPENDICES

ABSTRACT

Bipedal robots have the potential to traverse a wide range of unstructured environments, which are otherwise inaccessible to wheeled vehicles. Though roboticists have successfully constructed controllers for bipedal robots to walk over uneven terrain such as snow, sand, or even stairs, it has remained challenging to synthesize such controllers in an online fashion while guaranteeing their satisfactory performance. This is primarily due to the lack of numerical method that can accommodate the non-smooth dynamics, high degrees of freedom, and underactuation that characterize bipedal robots. This dissertation proposes and implements a family of numerical methods that begin to address these three challenges along three dimensions: optimality, safety, and computational speed.

First, this dissertation develops a convex relaxation-based approach to solve optimal control for hybrid systems without a priori knowledge of the optimal sequence of transition. This is accomplished by formulating the problem in the space of relaxed controls, which gives rise to a linear program whose solution is proven to compute the globally optimal controller. This conceptual program is solved using a sequence of semidefinite programs whose solutions are proven to converge from below to the true optimal solution of the original optimal control problem. Moreover, a method to synthesize the optimal controller is developed. Using an array of examples, the performance of this method is validated on problems with known solutions and also compared to a commercial solver.

Second, this dissertation constructs a method to generate safety-preserving controllers for a planar bipedal robot walking on flat ground by performing reachability analysis on simplified models under the assumption that the difference between the two models can be bounded. Subsequently, this dissertation describes how this reachable set can be incorporated into a Model Predictive Control framework to select controllers that result in safe walking on the biped in an online fashion. This method is validated on a 5-link planar model.

Third, this dissertation proposes a novel parallel algorithm capable of finding guaranteed optimal solutions to polynomial optimization problems up to pre-specified tolerances. Formal proofs of bounds on the time and memory usage of such method are also given. Such algorithm is implemented in parallel on GPUs and compared against state-of-the-art solvers on a group of benchmark examples. An application of such method on a real-time trajectory-planning task of a mobile robot is also demonstrated.

Fourth, this dissertation constructs an online Model Predictive Control framework that guarantees safety of a 3D bipedal robot walking in a forest of randomly-placed obstacles. Using numerical integration and interval arithmetic techniques, approximations to trajectories of the robot are constructed along with guaranteed bounds on the approximation error. Safety constraints are derived using these error bounds and incorporated in a Model Predictive Control framework whose feasible solutions keep the robot from falling over and from running into obstacles. To ensure that the bipedal robot is able to avoid falling for all time, a finite-time terminal constraint is added to the Model Predictive Control algorithm. The performance of this method is implemented and compared against a naive Model Predictive Control method on a biped model with 20 degrees of freedom.

In summary, this dissertation presents four methods for control synthesis of bipedal robots with improvements in either optimality, safety guarantee, or computational speed. Furthermore, the performance of all proposed methods are compared with existing methods in the field.

CHAPTER 1

Introduction

1.1 Motivation

Wheeled vehicles have proven to be the most effective means of long-distance ground transportation in modern society. Despite their ability to move people and cargo, wheeled vehicles are largely restricted by the types of terrains that they are able to traverse. Most wheeled vehicles can only operate on paved surfaces or on relatively smooth natural terrains, and usually encounter difficulties when snow, mud, sand, or rocks are present. On the other hand, animals and humans, who are able to rely on legged locomotion, typically face fewer challenges in traversing through these same scenarios. This motivates the development of legged robots systems with the objective of autonomously traversing the rough terrains that are accessible to animals and humans.

In fact, researchers have illustrated the capability of bipedal robots to walk up stairs [Rob19b], run and jump over obstacles [Dyn18], and walk through grass, snow, and sand [GHD+19]. However, in contrast to wheeled or flying robots wherein algorithms have been developed to perform provably safe online control synthesis [KVJRV17, KVB+18, VKL+19, KHV19, HKZ+20], researchers have struggled to develop algorithms to guarantee the performance of bipedal robots in an online fashion. This has limited the broad deployment of bipedal robots especially with and around humans. The challenge to making such guarantees during real-time control is the lack of numerical methods that can accommodate the non-smooth dynamics, high degrees of freedom, and underactuation that characterize bipedal robots. Before describing how state of the art numerical methods address each of these challenges, we briefly summarize how each of these features that characterize bipedal robots arise.

Bipedal locomotion involves impacts between the leg ends and the ground, where the robot's velocities change drastically within a short amount of time. As a result, a *rigid* contact model is often hypothesized for control design purposes, where such impact is assumed to be instantaneous and velocities of the robot are allowed to be discontinuous at the moment of impact. Although such an assumption avoids many difficulties associated with compliant contact models, it also renders

(a)

(b)

(c)

Figure 1.1: Several bipedal robots: (a) RABBIT, a planar bipedal robot with seven degrees of freedom and four actuators, (b) ATRIAS, a 3D bipedal robot with 13 degrees of freedom and six actuators, (c) Cassie, a 3D bipedal robot with 20 degrees of freedom and 10 actuators

the dynamic model *hybrid* in nature, consisting of alternating phases of continuous dynamics (the evolution of states according to a differential equation) and discrete dynamics (the re-initialization of states due to the impacts). Such hybrid models can pose serious challenges to numerically solving optimal control problems without *a* priori knowledge of the number and type of impacts. This is because derivatives of solutions are not well defined at the moment of transition between the different continuous modes of a hybrid system. As a result, generic numerical techniques usually assume additional information about such transitions to perform simultaneous optimization across continuous and discrete dynamics. For example, GPOPS-II [PR14], a commercial optimal control solver, requires an explicit definition for each transition as a pre-defined boundary condition; FROST [HA17], on the other hand, allows for periodic solutions but the sequence of transition within a period needs to be pre-specified.

Many bipedal robots have high degrees of freedom and are underactuated. The planar robot RABBIT [CAA$^+$03] has seven degrees of freedom, but only four of these are actuated (Fig. 1.1a); the 3D robot ATRIAS [GH12] has 13 degrees of freedom when supported on one leg, but only six of these are actuated (Fig. 1.1b); the 3D robot Cassie [Rob19a] has 20 degrees of freedom, but only 10 of these are actuated (Fig. 1.1c). Real-time control synthesis for such high-dimensional systems can pose serious computational challenges. As a result, engineers typically rely on simplified

models that capture essential features of bipedal locomotion [FK99] to guide the control design for such systems[WO13, ACG+18], but it is unclear how to link a synthesized controller for such model with the true system. Note, this challenge is further aggravated by underactuation, since underactuated systems cannot be easily commanded to track arbitrary trajectories.

The above observations lead to the following open problems:

1. Can one numerically solve a hybrid system optimal control problem even in the absence of *a* priori knowledge of the optimal sequence of contacts?

2. Can one ensure the safety of a bipedal robot at run-time?

3. Can one bound the computation time to perform online control synthesis for a bipedal robot?

1.2 State of the Art

A variety of algorithms have been proposed to address each of these open problems. The following subsections describe related work to address each of these problems in order.

1.2.1 Hybrid Optimal Control

Optimal control for hybrid system (or simply put, *hybrid optimal control*) has drawn a great deal of interest from researchers. The theoretical development of both necessary and sufficient conditions for the optimal control of hybrid controlled systems has been considered using extensions of the Pontryagin Maximum Principle [PLSB11, SC07, Sus99] and Dynamic Programming [BBM98, DR05, SCEM07], respectively. Recent work has even linked these approaches [PC17]. Typically, these methods have assumed that the sequence of transitions between the systems is known *a priori*. Practitioners, as a result, have fixed the sequence of transitions and used gradient-based methods to locally optimize over the time spent and control applied within each subsystem [GG15, HCHA16, SAGVR17, WGK03].

Recent work has focused on the development of numerical optimal control techniques for mechanical systems undergoing contact without specifying the ordering of visited subsystems. One approach to address the hybrid optimal control problem has focused on the construction of a novel notion of derivative [PB17]. Though this method still requires fixing the total number of visited subsystems, assuming *a priori* knowledge of the visited subsystems, and performs optimization only over the initial condition, this gradient-based approach is able to find the locally optimal ordering of subsystems under certain regularity conditions on the nature of the state-dependent switching. Other approaches have relaxed satisfaction of the unilateral constraint directly and

instead focused on treating constraint satisfaction as a continuous decision variable that can be optimized using traditional numerical methods to find local minima [PCT14, WG15, YG05].

1.2.2 Ensuring Safety of Bipedal Robots

When performing online trajectory planning for bipedal robots, there are a variety of techniques that attempt to ensure that a biped will not fall over while walking. For instance, the Zero-Moment Point (ZMP) approach [VS72] characterizes the stability of a legged robot with planar feet by defining the notion of the Zero-Moment Point and requiring that it remains within a robot's base of support. Though this requirement can be used to design a controller that can avoid falling at run-time, the gaits designed by the ZMP approach are static and energetically expensive [Kuo07, WCC+07, Section 10.8].

In contrast, the Hybrid Zero Dynamics approach, which relies upon feedback linearization to drive the actuated degrees of freedom of a robot towards a lower dimensional manifold, is able to synthesize a controller that generates more dynamic gaits. Though this approach designs controllers for legged systems even in the presence of model uncertainty [AGSG14, HXA15, NS15, NS16, NHG+16], it is only able to prove that the gait associated with a synthesized control is locally stable. As a result, it is non-trivial to switch between multiple hybrid zero dynamics constructed controllers while guaranteeing that a biped will not fall over. Fortunately, recent work has extended the ability of the hybrid zero dynamic approach to switch between several parameterized gaits while preserving safety guarantees [MVP16, VP18, ATJ+17, SARV19]. However, these extensions either assume full-actuation [ATJ+17] or ignore the behavior of the legged system off the lower dimensional manifold while assessing stability [MVP16, VP18, SARV19].

Rather than designing controllers for legged systems, other techniques have focused on characterizing region of attraction of walking gaits by performing Sums-of-Squares (SOS) optimization [Par00]. These approaches use semi-definite programming to identify a region of attraction to a steady state locomotion pattern in the state space of a system and can even be used to design controllers that maximize the size of the region of attraction [PJ04, SVBT14a]. These safe sets can take the form of *reachable sets* [KPT16, SVBT14a] or *invariant sets* in state space [Wie02, PJ04, PKT17]. However, the representation of each of these sets in the state space restricts the size of the problem that can be tackled by these approaches. As a result, these SOS-based approaches have been primarily applied to reduced models of walking robots, ranging from spring mass models [ZMV17b], to inverted pendulum models [KPT16, TBM17], or to inverted pendulum models with an offset torso mass [PKT17]. Unfortunately, the differences between these simple models and real robots make it challenging to extend the generated safety guarantees to the real-world.

1.2.3 Bounding Computation Time During Online Control Synthesis

The difficulty of performing trajectory optimization and control synthesis increases with the dimension of the cost and constraints, with the number of constraints, and with the number of optima [NW06]. Existing methods attempt to solve such optimization problems while minimizing time and memory usage. These methods broadly fall into the following categories: derivative-based, convex relaxation, and branch-and-bound.

Derivative-based methods use derivatives (and sometimes Hessians) of the cost and constraint functions, along with first- or second-order optimality conditions [NW06, §12.3, §12.5], to attempt to find optimal, feasible solutions to nonlinear problems. These methods can find local minima rapidly despite high dimension, a large number of constraints, and high degree cost and constraints [NW06, Chapter 19.8]. However, these methods do not typically converge to global optima without requiring assumptions on the problem and constraint structure (e.g., [QWY04]).

Convex relaxation methods attempt to find global optima by approximating the original problem with a hierarchy of convex optimization problems. These methods can be scaled to high-dimensional problems (up to 10 dimensions), at the expense of limits on the degree and sparse structure of the cost function; furthermore, they typically struggle to handle large numbers of constraints, unless the problem has low-rank or sparse structure [RCBL19]. Well-known examples include the lift-and-project linear program procedure [BCC93], reformulation-linearization technique [SA90], and Semi-Definite Program (SDP) relaxations [Las01, RCBL19, MLEV19]. By assuming structure such as homogeneity of the cost function or convexity of the domain and constraints, one can approximate solutions to certain type of problems in polynomial-time, with convergence to global optima in the limit [DKLP06, LNQY09, LZ10, So11, HLZ10]. Convergence within a finite number of convex hierarchy relaxations is possible under certain assumptions [Nie13, LTY17].

Branch-and-bound methods perform an exhaustive search over the feasible region. These methods are typically limited to up to four dimensions, but can handle large numbers of constraints and high degree cost and constraints. Examples include interval analysis techniques [HW03, VEH94] and the Bernstein Algorithm (BA) [Gar93, NA11, SS15]. Traditional interval analysis requires cost and constraint function evaluations in each iteration, and therefore can be computationally slow. BA, on the other hand, does not evaluate the cost and constraint functions; instead, BA represents the coefficients of the polynomial cost and constraints in the Bernstein basis, which provides lower and upper bounds on the polynomial cost and constraints over box-shaped subsets of Euclidean space by using a subdivision procedure [Gar85, NA07]. However, parallelized implementation or bounds on the rate of convergence of either interval analysis or BA with constraints has not yet been shown in the literature. Furthermore, to the best of our knowledge, neither of such methods has been shown as a practical method for solving problems in real-time robotics applications.

1.3 Contributions and Thesis Outline

The goal of this thesis is to help bridge the gap between safety and real-time performance of legged robots. To address the aforementioned difficulties in optimization, a novel formulation of optimal control is proposed to achieve a global optima regardless of unknown sequences of transition. To establish safety guarantees at run-time, sufficient safety conditions are derived and enforced in a receding horizon trajectory planner. Then, to improve real-time performance, a parallel algorithm is developed and implemented on GPU, which solves a version of the optimization problem with provable bounds on computation time and memory usage. Below is a brief outline of this thesis:

Chapter 2 describes how to transform hybrid system optimal control problems into infinite-dimensional linear programs (LPs) over the space of measures. Subsequently, it describes how to approximate the solution to these infinite-dimensional LPs using a hierarchy of semi-definite programs (SDPs) whose solutions are proved to converge to the true optimal solution under mild assumptions. This approach is proven effective for a variety of extensions to the hybrid system optimal control problem. Its performance is compared against state-of-the-art solutions in simulation. The methods and results of this study were originally presented in [ZMV17a, ZMV19, ZV19].

Chapter 3 constructs a Model Predictive Control (MPC) framework whose feasible solutions ensure that a planar robot walking on flat ground will not fall over. To design the constraints for this MPC problem, Chapter 3 describes a set of outputs that are functions of the states of the robot, which can be used to determine whether a particular gait can be safely tracked by a legged system without falling. This enables the construction of a simple model whose behavior can predict the full model's outputs under the assumption that the discrepancy between those two models can be bounded. Using this simple model and the associated bounds, one can formulate an N-step safety condition. This is then incorporated in an online MPC framework to generate parameterized safety controllers in real-time. Lastly, the proposed approach is validated on a walking example. These results were presented in [LZG+19].

Chapter 4 develops a parallel solver capable of quickly finding global optima and polynomial optimization problems up to pre-specified tolerances. This is accomplished by utilizing a special form of polynomial expansion to find conservative bounds on the values of polynomials over boxes. By iteratively subdividing each box of interest and eliminating infeasible or sub-optimal regions, one may continuously narrow down possible locations of global optima with at least a linear rate of convergence. Moreover, bounds on the computation time and memory usage of this method are proved. The proposed method is then implemented on a GPU and compared against generic solvers on various problems.

Chapter 5 extends the MPC framework in Chapter 3 to perform online control synthesis over a 3D bipedal robot while walking on a flat ground filled with randomly placed obstacles. Safety

constraints are enforced by computing a sequence of approximation points via numerical integration, bounding the distance between these points and the exact solution using interval analysis, and ensuring safety conditions are satisfied for all points within the error bound. Such safety guarantees are further extended to infinite time by forcing trajectories to reach a pre-specified set at the end of each planning horizon, after which a controller is assumed to be able to keep the robot balanced in place on two feet. To reduced the computation time needed for the online MPC, a parallel algorithm is developed. The proposed MPC framework is validated on a variety of scenarios with different numbers of obstacles, and its performance is compared against a naive MPC.

Finally, Chapter 6 provides a review of the main contributions and results of the thesis and outline possible directions for further related work.

CHAPTER 2

Solving Optimal Control Problems with Guaranteed Performance

2.1 Introduction

[1] A variety of engineering problems require searching for optimal system trajectories while satisfying certain constraints [Ber95]. Despite the numerous applications for these optimal control problems, they remain challenging to solve. Pontryagin's maximum principle (PMP)-based indirect shooting methods can suffer from serious numerical issues; the application of Hamilton–Jacobi–Bellman (HJB) theory usually requires discretizing time and state space, therefore greatly suffering from the curse of dimensionality. Various numerical methods also exist to solve the optimal control problem, including direct shooting, collocation, and pseudospectral methods. Although rendering the optimal control problems more amenable to computation, such approaches struggle to find global optimizers that satisfy the state and input constraints with high accuracy.

For *hybrid* systems whose evolution undergoes sudden changes due to satisfaction of state-dependent conditions, such as in bipeds [WGC+07], automotive sub-systems [HSCN07], aircraft control [SOS10], and biological systems [EL00], the optimal control problem also seeks the optimal ordering of subsystems visited. Such ordering is formally called *sequence of transition*. Technical difficulties arise when the optimal sequence of transition is unknown since derivatives are not well defined in the usual sense across different subsystems (modes). Recent work has focused on the development of numerical optimal control techniques for mechanical systems undergoing impacts without specifying the sequence of transition a priori [PB17, PCT14, WG15, YG05]. However, they fail to establish convergence to a global optimal solution in all circumstances, but rather provide locally optimal results only.

On the other hand, people have found ways to relax the nonlinear optimal control problem for classical systems and obtain a infinite-dimensional linear program (LP) over the space of measures

[1] This chapter was previously published in the 2017 American Control Conference [ZMV17a], 2019 American Control Conference [ZV19], and IEEE Transactions on Automatic Control [ZMV19].

[Vin93]. Under mild conditions, the solutions of two formulations are proved to coincide. The solutions to this infinite-dimensional LP can be further approximated by solving a hierarchy of semi-definite programs (SDPs) [LHPT08]. Such an approach has proven effective in construction of a sequence of lower bounds that converges to the true optimal value with guaranteed global convergence, even in the presence of nonlinear dynamics and states constraints. However, several challenges remain.

First, the original method proposed in [LHPT08] does not provide a way to recover the optimal trajectory nor the control law, therefore its utility is strictly limited. Modifications can be made to construct approximations to the optimal trajectory [CS14] or the control law [HLS08, KHJ16], but such methods either do not provide convergence guarantees or make strong assumptions about the problem structure. A reliable, general approach to perform a control synthesis with provable convergence still needs to be developed.

Second, it has not yet been shown a measure formulation can be similarly established for hybrid systems. Being unable to model state transitions, the traditional measure for formulation cannot be directly applied to the hybrid case. Although one could fix the sequence of transition and solve multiple instances of the measure formulation at once by matching the boundary conditions across transitions, such an approach scales poorly with respect to the number of transitions and little can be shown when the sequence is not known a priori.

In this chapter, I develop an algorithm that solves a general form of optimal control of both classical and hybrid systems with guaranteed convergence to global minima and provide a means of performing control synthesis without a priori knowledge of the sequence of transition.

2.2 Problem Formulation

This section defines the controlled hybrid systems and formulates the optimal control problem of interest.

2.2.1 Controlled Hybrid Systems

Motivated by [BGV$^+$15], we define the class of *controlled hybrid systems* as

Definition 1. *A* controlled hybrid system *is a tuple $\mathcal{H} = (\mathcal{I}, \mathcal{E}, \mathcal{D}, U, \mathcal{F}, \mathcal{S}, \mathcal{R})$, where:*

- *$\mathcal{I}$ is a finite set indexing the discrete states of $\mathcal{H}$;*
- *$\mathcal{E} \subset \mathcal{I} \times \mathcal{I}$ is a set of edges, forming a directed graph structure over $\mathcal{I}$;*
- *$\mathcal{D} = \coprod_{i \in \mathcal{I}} X_i$ is a disjoint union of domains, where each X_i is a compact subset of $\mathbb{R}^{n_i}$ and $n_i \in \mathbb{N}$;*

- *U is a compact subset of $\mathbb{R}^m$ that describes the range of control inputs, where $m \in \mathbb{N}$;*
- *$\mathcal{F} = \{F_i\}_{i\in\mathcal{I}}$ is the set of vector fields, where each $F_i : \mathbb{R} \times X_i \times U \to \mathbb{R}^{n_i}$ is a Lipschitz continuous vector field defining the dynamics of the system on X_i;*
- *$\mathcal{S} = \coprod_{e\in\mathcal{E}} S_e$ is a disjoint union of guards, where each $S_{(i,i')} \subset \partial X_i$ is a compact, co-dimensional 1 guard defining a state-dependent transition from X_i to $X_{i'}$; and,*
- *$\mathcal{R} = \{R_e\}_{e\in\mathcal{E}}$ is a set of continuous reset maps, where each map $R_{(i,i')} \colon S_{(i,i')} \to X_{i'}$ defines the transition from guard $S_{(i,i')}$ to $X_{i'}$.*

For convenience, throughout this thesis we refer to these controlled hybrid systems as hybrid systems, and a vertex within the graph associated with a hybrid system is referred to as a *mode*. Though the range space of control inputs are assumed to be the same in each mode, this is not restrictive since we can always concatenate all the control inputs in different modes. The compactness of each X_i ensures the optimal control problem, defined below, is well-posed. Since the focus of this chapter is on the optimal control of deterministic hybrid systems, we avoid any ambiguity during the transition between modes by making the following assumption:

Assumption 2. *Guards do not intersect with themselves or the images of reset maps. The controlled vector fields in each mode have a nonzero normal component on the guard for all control inputs in U.*

Next, we define a *hybrid trajectory* of a hybrid system up to time $T > 0$ in Fig. 2.1. Step 1 initializes the hybrid trajectory at a given point (x_0, i) at time $t = 0$. Step 3 defines ϕ to be the maximal integral curve of F_i under the control u beginning from the initial point. Step 4 defines the hybrid trajectory on a finite interval as the curve ϕ with associated index i. As described in Steps 5 - 7, the hybrid trajectory terminates when it either reaches the terminal time T or hits $\partial X_i \setminus \bigcup_{(i,i')\in\mathcal{E}} S_{(i,i')}$ where no transition is defined. Steps 8 and 9 define a discrete transition to a new domain using a reset map where evolution continues again as a classical dynamical system by returning to Step 3. Note that this definition is a rephrasing of [BGV$^+$15, Fig. 8] and is meant to formalize what is meant by a solution to a hybrid system. This chapter applies this definition only to ensure the existence of solutions to hybrid systems. A description of how to implement this definition can be found in [BGV$^+$15]. The space of such hybrid trajectories is denoted as $\mathcal{X}$. Note that for any t at which a hybrid trajectory γ is defined, $\gamma(t) = (\gamma_{\lambda(\gamma(t))}(t), \lambda(\gamma(t)))$.

Trajectories of hybrid systems can undergo an infinite number of discrete transitions in a finite amount of time. Since the state of the trajectory after these Zeno behaviors occur may not be well defined [AZGS06] and because the focus of this chapter is on optimal control for deterministic hybrid systems, we make the following assumption:

Assumption 3. *$\mathcal{H}$ has no Zeno trajectories.*

Require: $t = 0, T > 0, i \in \mathcal{I}, (x_0, i) \in \mathcal{D}$, and $u : \mathbb{R} \to U$ Lebesgue measurable.

Set $\gamma(0) = (x_0, i)$.
loop
  Let $I \subset [t, T]$ and $\phi \in AC(I; X_i)$ such that:

  (i) $\dot{\phi}(s) = F_i(s, \phi(s), u(s))$ for almost every $s \in I$ with respect to the Lebesgue measure on I with $(\phi(t), i) = \gamma(t)$ and

  (ii) for any other $\hat{\phi} \colon \hat{I} \to X_i$ satisfying (i), $\hat{I} \subset I$.

  Let $t' = \sup I$ and $\gamma(s) = (\phi(s), i)$ for each $s \in [t, t')$.
  if $t' = T$, **or** $\nexists (i, i') \in \mathcal{E}$ such that $\phi(t') \in S_{(i,i')}$ **then**
    Stop.
  end if
  Let $(i, i') \in \mathcal{E}$ be such that $\phi(t') \in S_{(i,i')}$.
  Set $\gamma(t') = (R_{(i,i')}(\phi(t')), i')$, $t = t'$, and $i = i'$.
end loop

Figure 2.1: The procedure to define a trajectory of hybrid system $\mathcal{H}$.

2.2.2 Problem Statement

This chapter is interested in finding a (γ, u) satisfying algorithm 2.1 from a given initial condition x_0, that reaches a target set while minimizing a cost function. To formulate this problem, define the *target set*, $X_T \subset \mathcal{D}$, as $X_T = \coprod_{i \in \mathcal{I}} X_{T_i}$, where X_{T_i} is a compact subset of X_i for each $i \in \mathcal{I}$. To avoid any ambiguity, we make the following assumption:

Assumption 4. *The target set does not intersect any guards.*

Given a $T > 0$ and an initial point $(x_0, j) \in \mathcal{D}$, a pair of functions (γ, u) satisfying algorithm 2.1 is called an *admissible pair* if $\gamma(T) \in X_T$. In this instance, γ is called an *admissible trajectory* and u is called an *admissible control*. The time T at which the admissible trajectory reaches the target set is called the *terminal time*. Denote the space of admissible trajectories and controls by $\mathcal{X}_T$ and $\mathcal{U}_T$, respectively. The space of admissible pairs is denoted as $\mathcal{P}_T \subset \mathcal{X}_T \times \mathcal{U}_T$. Without loss of generality, we make the following assumption:

Assumption 5. *The initial condition is not in any guard.*

For any admissible pair (γ, u), the associated cost is defined as:

$$\begin{aligned} J(\gamma, u) = &\int_0^T h_{\lambda(\gamma(t))}\left(t, \gamma_{\lambda(\gamma(t))}(t), u(t)\right)\, dt + \\ &+ H_{\lambda(\gamma(T))}\left(\gamma_{\lambda(\gamma(T))}(T)\right) + \sum_{e \in \mathcal{E}} \sum_{t \in \mathcal{T}_e(\gamma)} c_e(\gamma_{\lambda(\gamma(t))}(t)) \end{aligned} \tag{2.1}$$

where $h_i : [0,T] \times X_i \times U \to \mathbb{R}$, $H_i : X_i \to \mathbb{R}$, and $c_e : S_e \to \mathbb{R}$ are measurable functions for each $i \in \mathcal{I}$. In this instance, h_i is a mode-dependent cost function, H_i is a mode-dependent terminal cost, and c_e is a guard dependent *switching cost*.

Our goal is to find an admissible trajectory that minimizes (2.1), which we refer to as *Hybrid Optimal Control Problem (OCP)*:

$$\inf_{(\gamma,u)\in\mathcal{P}_T} J(\gamma,u) \qquad \text{(OCP)}$$

The optimal cost of (OCP) is defined as J^*.

2.3 The Hybrid Liouville Equation

This section constructs measures whose support models the evolution of families of trajectories, an equivalent form of J, and an equivalent form of Algorithm 2.1 in the space of measures. These transformations make a convex formulation of (OCP) feasible.

Consider the projection γ_i of a hybrid trajectory γ onto mode $i \in \mathcal{I}$. Define the *occupation measure* in mode $i \in \mathcal{I}$ associated with γ, denoted by $\mu^i(\cdot \mid \gamma) \in \mathcal{M}_+([0,T] \times X_i)$, as

$$\mu^i(A \times B \mid \gamma) := \int_0^T \mathbb{1}_{A\times B}(t, \gamma_i(t))\, dt \tag{2.2}$$

for all subsets $A \times B$ in the Borel σ-algebra of $[0,T] \times X_i$. Note that $\gamma_i(t)$ may not be defined for all $t \in [0,T]$, but we use the same notation and let $\mathbb{1}_{A\times B}(t, \gamma_i(t)) = 0$ whenever $\gamma_i(t)$ is undefined. The quantity $\mu^i(A \times B \mid \gamma)$ is equal to the amount of time the graph of the trajectory, $(t, \gamma_i(t))$, spends in $A \times B$. Define the *initial measure*, $\mu_0^i(\cdot \mid \gamma) \in \mathcal{M}_+(X_i)$, as

$$\mu_0^i(B \mid \gamma) := \mathbb{1}_B(\gamma_i(0)) \tag{2.3}$$

for all subsets B in the Borel σ-algebra of X_i; define the *terminal measure*, $\mu_T^i(\cdot \mid \gamma) \in \mathcal{M}_+(X_{T_i})$, as

$$\mu_T^i(B \mid \gamma) := \mathbb{1}_B(\gamma_i(T)) \tag{2.4}$$

for all subsets B in the Borel σ-algebra of X_{T_i}.

One can show that the occupation measure, initial measure, and the terminal measure satisfy a linear equation whose solution can model the evolution of a nonlinear dynamical system [LHPT08]. This result enables one to formulate nonlinear optimal control problems as infinite dimensional linear programs [LHPT08, Theorem 2.3]. Unfortunately, the linear equation over measures is unable to describe the transitions between hybrid modes. However, these transitions can be

described using *guard measures*. Define the *guard measure*, $\mu^{S_{(i,i')}}(\cdot \mid \gamma) \in \mathcal{M}_+([0,T] \times S_{(i,i')})$, as

$$\mu^{S_{(i,i')}}(A \times B \mid \gamma) := \text{card}\{t \in A \mid \lim_{\tau \to t^-} \gamma_i(\tau) \in B\} \tag{2.5}$$

for all subsets $A \times B$ in the Borel σ-algebra of $[0,T] \times S_{(i,i')}$, given any pair $(i,i') \in \mathcal{E}$. The guard measure counts the number of times a given trajectory passes through the guard.

Next, define the occupation measure in $i \in \mathcal{I}$ associated with (γ, u), denoted $\mu^i(\cdot \mid \gamma, u) \in \mathcal{M}_+([0,T] \times X_i \times U)$, as

$$\mu^i(A \times B \times C \mid \gamma, u) := \int_0^T \mathbb{1}_{A \times B \times C}(t, \gamma_i(t), u(t))\, dt \tag{2.6}$$

for all subsets $A \times B \times C$ in the Borel σ-algebra of $[0,T] \times X_i \times U$. It is useful to collect the initial, average, terminal, and guard occupation measures in each mode. That is, define $\mu_0^{\mathcal{I}}(\cdot \mid \gamma) \in \mathcal{M}_+(\mathcal{D})$ as $\mu_0^{\mathcal{I}}(\cdot, i \mid \gamma) := \mu_0^i(\cdot \mid \gamma)$ for each $i \in \mathcal{I}$. For convenience, we refer to $\mu_0^{\mathcal{I}}$ as an initial measure and write μ_0^i when we refer to the i-th slice of $\mu_0^{\mathcal{I}}$. We define and refer to $\mu^{\mathcal{I}}(\cdot \mid \gamma, u) \in \mathcal{M}_+([0,T] \times \mathcal{D} \times U)$, $\mu_T^{\mathcal{I}}(\cdot \mid \gamma, u) \in \mathcal{M}_+(X_T)$, and $\mu^{\mathcal{S}}(\cdot \mid \gamma, u) \in \mathcal{M}_+([0,T] \times \mathcal{S})$ similarly.

Using these definitions, we can rewrite the cost function J:

Lemma 6. *Let $\mu^{\mathcal{I}}(\cdot \mid \gamma, u)$ and $\mu_T^{\mathcal{I}}(\cdot \mid \gamma)$ be the occupation measure and terminal measure associated with the pair (γ, u), respectively. Then, the cost function can be expressed as*

$$J(\gamma, u) = \sum_{i \in \mathcal{I}} \langle \mu^i(\cdot \mid \gamma, u), h_i \rangle + \sum_{i \in \mathcal{I}} \langle \mu_T^i(\cdot \mid \gamma), H_i \rangle. \tag{2.7}$$

Despite the cost function being a nonlinear function of the admissible pair in the space of functions, the analogous cost function over the space of measures is linear. A similar analogue holds true for the dynamics of the system. That is, the occupation measure associated with an admissible pair satisfies a linear equation over measures. To formulate this linear equation, let $\mathcal{L}_i : C^1([0,T] \times X_i) \to C([0,T] \times X_i \times U)$ be a linear operator that acts on a test function v, defined as

$$(\mathcal{L}_i v)(t,x,u) := \frac{\partial v(t,x)}{\partial t} + \sum_{k=1}^{n_i} \frac{\partial v(t,x)}{\partial [x]_k} [F_i(t,x,u)]_k \tag{2.8}$$

for all $i \in \mathcal{I}$. Using the dual relationship between measures and functions, we define $\mathcal{L}_i' : C([0,T] \times X_i \times U)' \to C^1([0,T] \times X_i)'$ as the adjoint operator of $\mathcal{L}_i$, satisfying $\langle \mathcal{L}_i' \mu, v \rangle = \langle \mu, \mathcal{L}_i v \rangle$ for all $\mu \in \mathcal{M}([0,T] \times X_i \times U)$ and $v \in C^1([0,T] \times X_i)$.

Each of these adjoint operators can describe the evolution of trajectories of the system within each mode [LHPT08]. However, in the case of hybrid systems, trajectories may not just begin

evolving within a mode at $t = 0$. Instead, a trajectory can enter a mode either by starting from inside it at $t = 0$ or by being reset into it. Similarly, a trajectory can terminate in a mode either by reaching the terminal time or by hitting a guard and transitioning. To formalize this, we first modify reset maps to also act on time by defining $\tilde{R}_{(i,i')} : [0,T] \times S_{(i,i')} \to [0,T] \times X_{i'}$ by $\tilde{R}_{(i,i')}(t,x) = (t, R_{(i,i')}(x))$ for all $(i,i') \in \mathcal{E}$ and $(t,x) \in [0,T] \times S_{(i,i')}$. To describe trajectories of a controlled hybrid system using measures, we rely on the following result of [SVBT14b, (16)]:

Lemma 7. *Given an admissible pair (γ, u), its initial measure, occupation measure, terminal measure, and guard measure satisfy the following linear equation over measures:*

$$\begin{aligned} \delta_0 \otimes \mu_0^i(\cdot \mid \gamma) + \mathcal{L}_i' \mu^i(\cdot \mid \gamma, u) + \sum_{(i',i) \in \mathcal{E}} \tilde{R}_{(i',i)\#} \mu^{S_{(i',i)}}(\cdot \mid \gamma) \\ = \delta_T \otimes \mu_T^i(\cdot \mid \gamma) + \sum_{(i,i') \in \mathcal{E}} \mu^{S_{(i,i')}}(\cdot \mid \gamma), \quad \forall i \in \mathcal{I}, \end{aligned} \tag{2.9}$$

where (2.9) *holds in the sense that it is true for all test functions in $C^1([0,T] \times X_i)$.*

One can ask whether the converse relationship holds: does an arbitrary set of measures, $\mu_0^{\mathcal{I}} \in \mathcal{M}_+(\mathcal{D})$, $\mu^{\mathcal{I}} \in \mathcal{M}_+([0,T] \times \mathcal{D} \times U)$, $\mu_T^{\mathcal{I}} \in \mathcal{M}_+(X_T)$, and $\mu^{\mathcal{S}} \in \mathcal{M}_+([0,T] \times \mathcal{S})$, that satisfy (2.9) correspond to an initial measure, $\mu_0^{\mathcal{I}}(\cdot \mid \gamma)$, occupation measure, $\mu^{\mathcal{I}}(\cdot \mid \gamma, u)$, terminal measure, $\mu_T^{\mathcal{I}}(\cdot \mid \gamma)$, and guard measure, $\mu^{\mathcal{S}}(\cdot \mid \gamma)$ for some admissible pair (γ, u)? To answer this question, consider a family of hybrid trajectories modeled by a non-negative probability measure $\rho \in \mathcal{M}_+(\mathcal{X})$, and define an *average occupation measure* $\zeta^i \in \mathcal{M}_+([0,T] \times X_i)$ in each mode $i \in \mathcal{I}$ for the family of trajectories as

$$\zeta^i(A \times B) := \int_{\mathcal{X}} \mu^i(A \times B \mid \gamma)\, d\rho(\gamma) \tag{2.10}$$

for any $i \in \mathcal{I}$ and $A \times B$ in the Borel σ-algebra of $[0,T] \times X_i$; Define the *average initial measure* ζ_0^i, *average terminal measure* ζ_T^i, and *average guard measure* $\zeta^{S_{(i,i')}}$ similarly.

To prove the converse of Lemma 7, we define the *Hybrid Liouville Equation*, whose solution can be disintegrated into a set of measures that we eventually prove are related to ρ in Theorem 12.

Lemma 8. *Let $\mu_0^{\mathcal{I}} \in \mathcal{M}_+(\mathcal{D})$, $\mu^{\mathcal{I}} \in \mathcal{M}_+([0,T] \times \mathcal{D} \times U)$, $\mu_T^{\mathcal{I}} \in \mathcal{M}_+(X_T)$, and $\mu^{\mathcal{S}} \in \mathcal{M}_+([0,T] \times \mathcal{S})$ satisfy the* Hybrid Liouville Equation *(HLE), which is defined as*

$$\delta_0 \otimes \mu_0^i + \mathcal{L}_i' \mu^i + \sum_{(i',i) \in \mathcal{E}} \tilde{R}_{(i',i)\#} \mu^{S_{(i',i)}} = \delta_T \otimes \mu_T^i + \sum_{(i,i') \in \mathcal{E}} \mu^{S_{(i,i')}} \tag{2.11}$$

for each $i \in \mathcal{I}$. Then, each measure μ^i can be disintegrated as

$$d\mu^i(t,x,u) = d\nu^i_{u|t,x}(u) d\mu^i_{t,x}(t,x) = d\nu^i_{u|t,x}(u) d\tilde{\mu}^i_{x|t}(x) dt \tag{2.12}$$

where $\nu^i_{u|t,x}$ is a stochastic kernel on U given $(t,x) \in [0,T] \times X_i$, $\mu^i_{t,x}$ is the (t,x)-marginal of μ^i, and $\tilde{\mu}^i_{x|t}$ is a conditional measure on X_i given $t \in [0,T]$.

For convenience, denote $\tilde{\mu}^i_{x|t}$ by $\mu^i_{x|t}$ and define:

$$\begin{aligned} \sigma^i &:= \delta_0 \otimes \mu^i_0 + \sum_{(i',i)\in\mathcal{E}} \tilde{R}_{(i',i)\#} \mu^{S_{(i',i)}}, \\ \eta^i &:= \delta_T \otimes \mu^i_T + \sum_{(i,i')\in\mathcal{E}} \mu^{S_{(i,i')}} \end{aligned} \tag{2.13}$$

Using (2.12), HLE can also be written as a non-homogeneous PDE that holds in the sense of distributions:

$$\partial_t \mu^i_{t,x} + D_x \cdot (\bar{F}_i \mu^i_{t,x}) = \sigma^i - \eta^i, \tag{2.14}$$

where

$$\bar{F}_i(t,x) := \int_U F_i(t,x,u)\, d\nu^i_{u|t,x}(u) \in \text{conv } F_i(t,x,U). \tag{2.15}$$

Note that even when F_i is Lipschitz continuous, $\bar{F}_i$ may not be Lipschitz continuous. By applying integration by parts, we can write

$$\begin{aligned} \int_0^T \int_{X_i} \left(\partial_t v(t,x) + \nabla_x v(t,x) \cdot \bar{F}_i\right)\, d\mu^i_{x|t}(x)\, dt \\ + \int_{[0,T]\times X_i} v(t,x)\, d\sigma^i(t,x) = \int_{[0,T]\times X_i} v(t,x)\, d\eta^i(t,x) \end{aligned} \tag{2.16}$$

for any test function $v \in C^1([0,T] \times X_i)$. We later show in Corollary 10 that σ^i and η^i capture the trajectories that enter and leave domain i, respectively.

Next, we prove the converse of Lemma 7 using Theorems 9 and 12. These converse theorems prove that a solution to the Hybrid Liouville Equation can be identified with a solution to the hybrid system under certain regularity conditions on the vector fields in each mode. This result enables us to formulate (OCP) as an optimization problem over measures, as described in Section 2.4. We start by showing $\mu^i_{x|t}$ is related to the solution of the ODE $\bar{F}_i$. As shown in Appendix A, (Theorem 79), when $\bar{F}_i$ satisfies certain regularity conditions (e.g., Lipschitz continuity), the relationship between $\mu^i_{x|t}$ and $\bar{F}_i$ is clear, but to deal with solutions to a non-smooth ODE, we construct the notion of evaluation maps that act on the space of absolutely continuous functions. Let $\Gamma_i :=$

$AC([0,T];\mathbb{R}^{n_i})$ be the space of absolutely continuous functions from $[0,T]$ to $\mathbb{R}^{n_i}$ endowed with the norm $\|\cdot\| : \gamma \mapsto |\gamma(0)| + \int_0^T |\dot{\gamma}(t)|\, dt$. Define an *evaluation map* $e_t : [0,t] \times [t,T] \times \Gamma_i \to \mathbb{R}^{n_i}$ as $e_t(s,\tau,\gamma) = \gamma(t)$ on $s \le t \le \tau$ for each $t \in [0,T]$. The evaluation map allows us to establish the following relationship:

Theorem 9. *Let $\mu^i_{x|t}$, σ^i, η^i satisfy the PDE* (2.16) *for some $i \in \mathcal{I}$, where $\bar{F}_i$ is defined as in* (2.15)*. Assume $\bar{F}_i$ is pointwise bounded. Then, there exists a measure $\rho^i \in \mathcal{M}_+([0,T] \times [0,T] \times \Gamma_i)$ such that*

(a) ρ^i is concentrated on the triplets (s,τ,γ), where $s \le \tau$, and $\gamma \in \Gamma_i$ are solutions of the ODE $\dot{\gamma}(t) = \bar{F}_i(t,\gamma(t))$ for a.e. $t \in [s,\tau]$.

(b) $\mu^i_{x|t} = (e_t)_{\#}\, \rho^i$ for a.e. $t \in [0,T]$.

Proof. See Appendix B. □

Theorem 9 establishes a connection between the measure $\mu^i_{x|t}$ that solves the PDE (2.16) and trajectories that satisfy the dynamics in mode i. We next show the start of those trajectories and terminate in the support of σ^i and η^i, respectively.

Corollary 10. *Let $\mu^i_{x|t}$, σ^i, and η^i satisfy the PDE* (2.16) *for some i and let $\bar{F}_i$, which is defined in* (2.15)*, be pointwise bounded. Let ρ^i be defined as in Theorem 9. Define maps $r^1, r^2 \in [0,T] \times [0,T] \times \Gamma_i \to [0,T] \times \mathbb{R}^{n_i}$ by $r^1 : (s,\tau,\gamma) \mapsto (s,\gamma(s))$ and $r^2 : (s,\tau,\gamma) \mapsto (\tau,\gamma(\tau))$. Then, $r^1_{\#}\rho^i = \sigma^i$ and $r^2_{\#}\rho^i = \eta^i$.*

Proof. Recall in the proof of Theorem 9, we mollified σ^i and η^i using a family of smooth mollifiers to obtain smooth measures σ^i_ϵ and η^i_ϵ. We also defined a *tight* family of measures $\{\rho^i_\epsilon\}_\epsilon \subset \mathcal{M}_+([0,T] \times [0,T] \times \Gamma_i)$ that converges to ρ^i in the narrow sense. The connection between each ρ^i_ϵ in that family and the mollified measures σ^i_ϵ and η^i_ϵ was established via measures $\rho^{i,+}_\epsilon$ and $\rho^{i,-}_\epsilon$.

For all $\varphi \in C_b([0,T] \times \mathbb{R}^{n_i})$, it follows from (B.16), (B.4), and (B.3) that

$$\int_{[0,T]\times[0,T]\times\Gamma_i} \varphi(s,\gamma(s))\, d\rho^i_\epsilon(s,\tau,\gamma) = \int_{[0,T]\times\mathbb{R}^{n_i}} \varphi(s,x)\, \sigma^i_\epsilon(s,x) \tag{2.17}$$

Since the families $\{\sigma^i_\epsilon\}_\epsilon$ and $\{\rho^i_\epsilon\}_\epsilon$ are tight as was shown in the proof of Theorem 9, we may let $\epsilon \downarrow 0$ to obtain $\int_{[0,T]\times[0,T]\times\Gamma_i} \varphi(r^1(s,\tau,\gamma))\, d\rho^i(s,\tau,\gamma) = \int_{[0,T]\times\mathbb{R}^{n_i}} \varphi(s,x)\, \sigma^i(s,x)$. This is also true for all measurable functions φ because $C_b(\mathbb{R}^{n_i+1})$ is dense in $L^1(\sigma^i)$ [Bog07, Corollary 4.2.2], as a result $r^1_{\#}\rho^i = \sigma^i$. The result for η^i can be proved in a similar manner. □

Theorem 9 illustrates that measures satisfying HLE in mode $i \in \mathcal{I}$ correspond to trajectories $\gamma \in \Gamma_i$ of the convexified inclusion, $\dot{\gamma}(t) \in \text{ conv } F_i(t,\gamma(t),U)$, rather than the original specified

dynamics within each mode of the system. To ensure that the there is no gap between the original dynamics and its convexified inclusion, we make the following assumption:

Assumption 11. *The set $F_i(t, x, U)$ is compact and convex for all t, x, and $i \in \mathcal{I}$.*

The above condition is sufficient to ensure that measures satisfying HLE correspond exactly to trajectories described according to Algorithm 2.1 [Vin93, p. 529]. Assumption 11 is satisfied if, for example, F_i is control affine and U is compact and convex.

As a consequence of Corollary 10 and Assumption 11, any triplet $(s, \tau, \gamma) \in \text{spt}(\rho^i)$ can be viewed as a trajectory γ in mode i that is well defined on $[s, \tau]$ and satisfies $(s, \gamma(s)) \in \text{spt}(\sigma^i)$, $(\tau, \gamma(\tau)) \in \text{spt}(\eta^i)$. Such trajectories in different modes are related by reset maps and can be combined together to be admissible trajectories for the hybrid system. To illustrate this, we define an evaluation map that acts on the trajectories of the hybrid system $e_t^i : \mathcal{X} \to X_i$ as $e_t^i(\gamma) = \gamma_i(t)$ if $\lambda(\gamma(t)) = i$ and $e_t^i(\gamma) = \emptyset$ otherwise for each $i \in \mathcal{I}$. We can establish a relationship between admissible trajectories and measures that satisfy (2.16) for each i:

Theorem 12. *Let $\mu_{x|t}^i$, σ^i, and η^i satisfy the PDE* (2.16) *for some i and let $\bar{F}_i$ which is defined in* (2.15) *be pointwise bounded. There then exists a non-negative measure $\rho \in \mathcal{M}_+(\mathcal{X})$ such that*

(a) For any hybrid trajectory $\gamma \in$ spt(ρ), γ is defined on $[0, T]$ and satisfies $\gamma(0) \in$ spt$(\mu_0^{\mathcal{I}})$, $\gamma(T) \in$ spt$(\mu_T^{\mathcal{I}})$.

(b) For a.e. $t \in [0, T]$, $\mu_{x|t}^i = (e_t^i)_{\#}\rho$.

(c) If $\sum_{i \in \mathcal{I}} \mu_0^i(X_i) = 1$, then ρ is a probability measure.

(d) If $\sum_{i \in \mathcal{I}} \mu_0^i(X_i) = 1$, then $\mu_{t,x}^i$ (resp. μ_0^i, μ_T^i, μ^{S_e}) is the average occupation measure (resp. average initial measure, average terminal measure, average guard measure) generated by the family of admissible trajectories in the support of ρ for each mode $i \in \mathcal{I}$ and $e \in \mathcal{E}$. Moreover, $\sum_{i \in \mathcal{I}} \mu_{t,x}^i([0, T] \times X_i) = T$, $\sum_{i \in \mathcal{I}} \mu_T^i(X_{T_i}) = 1$, and $\sum_{e \in \mathcal{E}} \mu^{S_e}([0, T] \times S_e) \leq C$ for some constant $C < +\infty$.

Proof. See Appendix C. □

Notice in Theorem 12 that if we define μ_0^i to be Dirac measure supported at x_0 if $x_0 \in X_i$ or zero otherwise, then spt$(\rho) \subset \mathcal{X}_T$. Finally, we establish a relationship between the solution measures and the underlying control input when the dynamics are control-affine, which enables control synthesis:

Corollary 13. *Let U be convex. For each $i \in \mathcal{I}$, suppose there exists pointwise bounded functions $f_i : \mathbb{R} \times X_i \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R} \times X_i \to \mathbb{R}^{n_i \times m}$ such that $F_i(t, x, u) = f_i(t, x) + g_i(t, x)u$ for all*

$t, x, u \in [0,T] \times X_i \times U$. *Let* $\nu^i_{u|t,x}$ *be defined as in* (2.12) *and let* ρ *be defined as in Theorem 12. Then* $t \mapsto (\gamma(t), \int_U u\, d\nu^{\lambda(\gamma(t))}_{u|t,\gamma_{\lambda(\gamma(t))}}(u))$ *is an admissible pair for all* $\gamma \in spt(\rho)$*, where*

$$\int_U u\, d\nu^i_{u|t,x}(u) := \begin{bmatrix} \int_U [u]_1\, d\nu^i_{u|t,x}(u) \\ \vdots \\ \int_U [u]_m\, d\nu^i_{u|t,x}(u) \end{bmatrix} \tag{2.18}$$

is an $m \times 1$ *real vector for each* t*,* x*, and* $i \in \mathcal{I}$*.*

Proof. For any $\gamma \in \text{spt}(\rho)$, $\dot\gamma_i(t) = f_i(t,\gamma_i(t)) + g_i(t,\gamma_i(t)) \cdot \int_U u\, d\nu^i_{u|t,\gamma_i(t)}(u)$ for a.e. $t \in [0,T]$. Since $\nu^i_{u|t,x}$ is a stochastic kernel and U is convex, $\int_U u\, d\nu^i_{u|t,\gamma_i(t)}(u) \in U$ for all $i \in \mathcal{I}$. Thus $t \mapsto (\gamma(t), \int_U u\, d\nu^{\lambda(\gamma(t))}_{u|t,\gamma_{\lambda(\gamma(t))}}(u))$ is an admissible pair. □

2.4 Infinite Dimensional Linear Program

This section formulates (OCP) as an infinite-dimensional linear program over the space of measures, proves that its solution coincides with the solution to (OCP), and illustrates how its solution can be used for control synthesis. To begin, we make the following assumption:

Assumption 14. *U is convex and for each $i \in \mathcal{I}$, there exists point-wise bounded functions $f_i : \mathbb{R} \times X_i \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R} \times X_i \to \mathbb{R}^{n_i \times m}$ such that $F_i(t,x,u) = f_i(t,x) + g_i(t,x)u$ for all $(t,x,u) \in [0,T] \times X_i \times U$.*

Define the optimization problem (P) as:

$$\begin{aligned}
\inf_{\Gamma}\ & \sum_{i\in\mathcal{I}} \langle \mu^i, h_i\rangle + \sum_{i\in\mathcal{I}} \langle \mu^i_T, H_i\rangle + \sum_{e\in\mathcal{E}} \langle \mu^{S_e}, c_e\rangle && (P)\\
\text{s.t. } & \delta_0 \otimes \mu^i_0 + \mathcal{L}_i'\mu^i + \sum_{(i',i)\in\mathcal{E}} \tilde R_{(i',i)\#}\mu^{S_{(i',i)}} = \delta_T \otimes \mu^i_T + \sum_{(i,i')\in\mathcal{E}} \mu^{S_{(i,i')}}, && \forall i \in \mathcal{I},\\
& \sum_{i\in\mathcal{I}} \langle \mathbb{1}_{X_{0_i}}, \mu^i_0\rangle = 1,\\
& \mu^i_0, \mu^i, \mu^i_T \geq 0, && \forall i \in \mathcal{I},\\
& \mu^{S_e} \geq 0, && \forall e \in \mathcal{E}
\end{aligned}$$

where the infimum is taken over a tuple of measures $\Gamma = (\mu^{\mathcal{I}}_0, \mu^{\mathcal{I}}, \mu^{\mathcal{I}}_T, \mu^{\mathcal{S}}) \in \mathcal{M}_+(X_0) \times \mathcal{M}_+([0,T] \times \mathcal{D} \times U) \times \mathcal{M}_+(X_T) \times \mathcal{M}_+([0,T] \times \mathcal{S})$.

The dual to problem (P) is given as:

$$\begin{aligned}
&\sup_{v,w} \; w && (D)\\
&\text{s.t. } \mathcal{L}_i v_i(t,x) + h_i(t,x,u) \geq 0 && \forall (t,(x,i),u) \in [0,T] \times \mathcal{D} \times U,\\
&\quad v_i(T,x) \leq H_i(x) && \forall (x,i) \in X_T,\\
&\quad v_i(t,x) \leq v_{i'}(t, R_{(i,i')}(x)) + c_{(i,i')}(x) && \\
&\quad && \forall (t,(x,(i,i'))) \in [0,T] \times \mathcal{S},\\
&\quad w \leq v_i(0,x) && \forall (x,i) \in X_0,
\end{aligned}$$

where the supremum is taken over the tuple $(v, w) \in C^1([0,T] \times \mathcal{D}) \times \mathbb{R}$ and the first and second constraint are for all $i \in \mathcal{I}$ and the third constraint is for all $i, i' \in \mathcal{E}$. Again, for notational convenience, we denote the $i \in \mathcal{I}$ slice of v using subscript i (i.e. for every $i \in \mathcal{I}$ and $(t,x) \in [0,T] \times X_i$, let $v_i(t,x) = v(t,x,i)$). Using [AN87, Theorem 3.10], we can prove the following relationship between the primal and dual:

Theorem 15. *There is no duality gap between (P) and (D).*

Next, we illustrate the (P) is well-posed by proving the existence of an optimal solution:

Lemma 16. *If (P) is feasible, the infimum value of (P), denoted by p^*, is attained.*

Now we prove that (P) solves (OCP):

Theorem 17. *Let (P) be feasible and suppose $h_i(t,x,\cdot)$ is convex for all $i \in \mathcal{I}$ and $(t,x) \in [0,T] \times X_i$. Then (P) solves (OCP), i.e., their optimal values are equal.*

By applying the same argument as shown in [ZMV17b, Theorem 21], one can perform control synthesis:

Theorem 18. *Let (P) be feasible and suppose $h(t,x,\cdot)$ is convex for all $i \in \mathcal{I}$ and $(t,x) \in [0,T] \times X_i$. Suppose the optimal trajectory γ^* is unique dt-almost everywhere. Let $\Gamma^* = (\mu_0^{\mathcal{I}*}, \mu^{\mathcal{I}*}, \mu_T^{\mathcal{I}*}, \mu^{\mathcal{S}*})$ be a vector of measures that achieves the infimum of (P), then the function $\tilde{u}_i(t,x) : [0,T] \times X_i \to U$ defined by the Radon-Nikodym derivative*

$$[\tilde{u}_i(t,x)]_j := \frac{\int_U [u]_j \, d\mu^{i*}(t,x,u)}{\int_U d\mu^{i*}(t,x,u)}, \forall j \in \{1, \cdots, m\} \tag{2.19}$$

is an optimal feedback controller for each mode $i \in \mathcal{I}$.

2.5 Numerical Implementation

We compute a solution to the infinite-dimensional problem (P) via a sequence of finite-dimensional approximations formulated as semidefinite programs (SDPs). These are generated by representing the measures in (P) using a truncated sequence of moments and restricting the functions in (D) to polynomials of finite degree. As illustrated in this section, the solutions to any of the SDPs in this sequence can be used to synthesize an approximation to the optimal initial condition. To begin, we make the following assumptions:

Assumption 19. *The functions F_i, h_i, H_i, and c_i are polynomials, that is, $[F_i]_j \in \mathbb{R}[t,x]$, $h_i \in \mathbb{R}[t,x]$, $H_i \in \mathbb{R}[x]$, and $c_i \in \mathbb{R}[x]$ for all $i \in \mathcal{I}$ and $j \in \{1,\cdots,n_i\}$. X_{0_i}, X_i, X_{T_i}, and $S_{(i,i')}$ are semi-algebraic sets, i.e.,*

$$\begin{aligned} X_{0_i} &= \{x \in \mathbb{R}^{n_i} \mid h_{0_{i_j}} \geq 0, \forall j \in \{1,\cdots,n_{0_i}\}\}, \\ X_i &= \{x \in \mathbb{R}^{n_i} \mid h_{X_{i_j}} \geq 0, \forall j \in \{1,\cdots,n_{X_i}\}\}, \\ X_{T_i} &= \{x \in \mathbb{R}^{n_i} \mid h_{T_{i_j}} \geq 0, \forall j \in \{1,\cdots,n_{T_i}\}\}, \\ U &= \{u \in \mathbb{R}^m \mid h_{U_j} \geq 0, \forall j \in \{1,\cdots,n_U\}\}, \\ S_{(i,i')} &= \{x \in \partial X_i \mid h_{(i,i')_j} \geq 0, \forall j \in \{1,\cdots,n_{(i,i')}\}\} \end{aligned} \tag{2.20}$$

for all $i \in \mathcal{I}$ and $(i,i') \in \mathcal{E}$, where $h_{0_{i_j}}$, $h_{X_{i_j}}$, $h_{T_{i_j}}$, h_{U_j}, $h_{(i,i')_j} \in \mathbb{R}[x]$.

Since X_{0_i}, X_i, X_{T_i}, and $S_{(i,i')}$ are also compact, note that Putinar's condition is satisfied by adding the redundant constraint $M - \|x\|_2^2$ for large enough M [Las09, Theorem 2.14].

To derive the SDP relaxation, we begin with a few preliminaries. Any polynomial $p \in \mathbb{R}_k[x]$ can be expressed in the monomial basis as $p(x) = \sum_{|\alpha|\leq k} p_\alpha x^\alpha = \sum_{|\alpha|\leq k} p_\alpha \cdot (x_1^{\alpha_1}\cdots x_n^{\alpha_n})$ where α ranges over vectors of non-negative integers such that $|\alpha| = \sum_{i=1}^n \alpha_i \leq k$, and we denote $\text{vec}(p) = (p_\alpha)_{|\alpha|\leq k}$ as the vector of coefficients of p. Given a vector of real numbers $y = (y_\alpha)$ indexed by α, we define the linear functional $L_y : \mathbb{R}_k[x] \to \mathbb{R}$ as:

$$L_y(p) := \sum_\alpha p_\alpha y_\alpha \tag{2.21}$$

Note that when the entries of y are moments of a measure μ, i.e. $y_\alpha = \int x^\alpha \, d\mu(x)$, then $\langle \mu, p\rangle = L_y(p)$. If $|\alpha| \leq 2k$, the *moment matrix*, $M_k(y)$, is defined as:

$$[M_k(y)]_{\alpha\beta} = y_{(\alpha+\beta)} \tag{2.22}$$

Given any polynomial $h \in \mathbb{R}_l[x]$ with $l < k$, the *localizing matrix*, $M_k(h, y)$, is defined as:

$$[M_k(h, y)]_{\alpha\beta} = \sum_{|\omega| \leq l} h_\omega y_{(\omega+\alpha+\beta)}. \tag{2.23}$$

The moment and localizing matrices are symmetric and linear in moments y.

2.5.1 LMI Relaxations and SOS Approximations

A sequence of SDPs approximating (P) can be obtained by replacing constraints on measures with constraints on moments. Since h_i, H_i, and c_i are polynomials, the objective function of (P) can be written using linear functionals as $\sum_{i\in\mathcal{I}} L_{y_{\mu^i}}(h_i) + \sum_{i\in\mathcal{I}} L_{y_{\mu_T^i}}(H_i) + \sum_{(i,i')\in\mathcal{E}} L_{y_{\mu^{S_{(i,i')}}}}(c_i)$, where y_{μ^i}, $y_{\mu_T^i}$, and $y_{\mu^{S_{(i,i')}}}$ are the sequence of moments of μ^i, μ_T^i, and $\mu^{S_{(i,i')}}$, respectively. The equality constraints in (P) can be approximated by an infinite-dimensional linear system, which is obtained by restricting to only polynomial test functions: $v_i(t, x) \in \mathbb{R}[t, x]$, for any $i \in \mathcal{I}$. The positivity constraints in (P) can be replaced with semidefinite constraints on moment and localizing matrices, which guarantees the existence of Borel measures [Las09, Theorem 3.8].

A finite-dimensional SDP is then obtained by truncating the degree of moments and polynomial test functions to $2k$. Let $\Xi_0 = \coprod_{i\in\mathcal{I}} \mu_0^i$, $\Xi_\mathcal{I} = \coprod_{i\in\mathcal{I}} \mu^i$, $\Xi_\mathcal{E} = \coprod_{e\in\mathcal{E}} \mu^{S_e}$, $\Xi_T = \coprod_{i\in\mathcal{I}} \mu_T^i$, and $\Xi = \Xi_0 \bigcup \Xi_\mathcal{I} \bigcup \Xi_\mathcal{E} \bigcup \Xi_T$. Let $(y_{k,\xi})$ be the sequence of moments truncated to degree $2k$ for each $(\xi, i) \in \Xi$, and let $\mathbf{y}_k$ be a vector of all the sequences $(y_{k,\xi})$. The equality constraints in (P) can then be approximated by a finite-dimensional linear system: $A_k(\mathbf{y}_k) = b_k$. Define the k-th relaxed SDP representation of (P) as:

$$\begin{aligned}
\inf\ & \sum_{i\in\mathcal{I}} L_{y_{k,\mu^i}}(h_i) + \sum_{i\in\mathcal{I}} L_{y_{k,\mu_T^i}}(H_i) + \sum_{e\in\mathcal{E}} L_{y_{k,\mu^{S_e}}}(c_e) && (P_k)\\
\text{s.t. } & A_k(\mathbf{y}_k) = b_k,\\
& M_k(y_{k,\xi}) \succeq 0 && \forall (\xi, i) \in \Xi,\\
& M_{k_{X_{i_j}}}(h_{X_{i_j}}, y_{k,\mu^i}) \succeq 0 && \forall (j, i) \in \coprod_{i\in\mathcal{I}} \{1, \cdots, n_{X_i}\},\\
& M_{k_{U_{i_j}}}(h_{U_j}, y_{k,\mu^i}) \succeq 0 && \forall (j, i) \in \{1, ..., n_U\} \times \mathcal{I},\\
& M_{k_{S_{e_j}}}(h_{e_j}, y_{k,\mu^{S_e}}) \succeq 0 && \forall (j, e) \in \coprod_{e\in\mathcal{E}} \{1, \cdots, n_e\},\\
& M_{k_{0_{i_j}}}(h_{0_{i_j}}, y_{k,\mu_0^i}) \succeq 0 && \forall (j, i) \in \coprod_{i\in\mathcal{I}} \{1, \cdots, n_{0_i}\},\\
& M_{k_{T_{i_j}}}(h_{T_{i_j}}, y_{k,\mu_T^i}) \succeq 0 && \forall (j, i) \in \coprod_{i\in\mathcal{I}} \{1, \cdots, n_{T_i}\},\\
& M_{k-1}(h_\tau, y_{k,\xi}) \succeq 0 && \forall (\xi, i) \in \Xi_\mathcal{I} \bigcup \Xi_\mathcal{E}
\end{aligned}$$

where the infimum is taken over $\mathbf{y}_k$; $h_\tau = t(T-t)$, $k_{X_{i_j}} = k - \lceil \deg(h_{X_{i_j}})/2 \rceil$, $k_{U_{i_j}} = k - \lceil \deg(h_{U_{i_j}})/2 \rceil$, $k_{S_{e_j}} = k - \lceil \deg(h_{e_j})/2 \rceil$, $k_{T_{i_j}} = k - \lceil \deg(h_{T_{i_j}})/2 \rceil$, $k_{0_{i_j}} = k - \lceil \deg(h_{0_{i_j}})/2 \rceil$, and $\succeq$ denotes positive semidefiniteness.

The dual of (P_k) is a Sums-of-Squares (SOS) program that is obtained by restricting the optimization space in (D) to the polynomial functions with degree truncated to $2k$ and then replacing the non-negativity constraints in (D) with SOS constraints. For notational convenience, we let x_i be the indeterminate that corresponds to X_i. Define $Q_{2k}(h_{T_{i_1}}, \cdots, h_{T_{i_{n_{T_i}}}}) \subset \mathbb{R}_{2k}[x_i]$ to be the set of polynomials $l \in \mathbb{R}_{2k}[x_i]$ expressible as $l = s_0 + \sum_{j=1}^{n_{T_i}} s_j h_{T_{i_j}}$ for polynomials $\{s_j\}_{i=0}^{n_{T_i}} \subset \mathbb{R}_{2k}[x_i]$ that are sums of squares. Every such polynomial is non-negative on X_{T_i}. Similarly, we define $Q_{2k}(h_{0_{i_1}}, \cdots, h_{0_{i_{n_{0_i}}}}) \subset \mathbb{R}_{2k}[x_i]$, $Q_{2k}(h_\tau, h_{X_{i_1}}, \cdots, h_{X_{i_{n_{X_i}}}}, h_{U_1}, \cdots, h_{U_{n_U}}) \subset \mathbb{R}_{2k}[t, x_i]$, and $Q_{2k}(h_\tau, h_{(i,i')_1}, \cdots, h_{(i,i')_{n_{(i,i')}}}) \subset \mathbb{R}_{2k}[t, x_i]$ for each $i \in \mathcal{I}$ and $(i, i') \in \mathcal{E}$. The k-th relaxed SDP representation of (D) is given as

$$
\begin{aligned}
&\sup\ w && (D_k)\\
&\text{s.t. } \mathcal{L}_i v_i + h_i \in \\
&\quad Q_{2k}(h_\tau, h_{X_{i_1}}, \cdots, h_{X_{i_{n_{X_i}}}}, h_{U_1}, \cdots, h_{U_{n_U}}) && \forall i \in \mathcal{I},\\
&\quad -v_i(T,\cdot) + H_i \in Q_{2k}(h_{T_{i_1}}, \cdots, h_{T_{i_{n_{T_i}}}}) && \forall i \in \mathcal{I},\\
&\quad v_{i'} \circ \tilde{R}_{(i,i')} + c_i - v_i \in \\
&\quad Q_{2k}(h_\tau, h_{(i,i')_1}, \cdots, h_{(i,i')_{n_{(i,i')}}}) && \forall (i,i') \in \mathcal{E},\\
&\quad v_i(0,\cdot) - w \in Q_{2k}(h_{0_{i_1}}, \cdots, h_{0_{i_{n_{0_i}}}}) && \forall i \in \mathcal{I}_0,
\end{aligned}
$$

where the supremum is over $(v_i, w) \in \mathbb{R}_{2k}[t, x_i] \times \mathbb{R}$ for all $i \in \mathcal{I}$. We prove that these pair of problems are well-posed, which follows from Slater's condition (see [BV04]), and that they converge, which follows from [ZMV17b, Theorem 25]:

Theorem 20. *For each $k \in \mathbf{N}$, if (P_k) is feasible, then there is no duality gap between (P_k) and (D_k). Moreover, if p_k^* and d_k^* denote the infimum of (P_k) and supremum of (D_k), respectively, then $\{p_k^*\}_{k=1}^\infty$ and $\{d_k^*\}_{k=1}^\infty$ converge monotonically from below to the optimal value of (OCP).*

Next, we describe how to extract a polynomial control law from the solution $\mathbf{y}_k$ of (P_k). Given moment sequences truncated to $2k$, we choose an approximate polynomial control law $\tilde{u}_{k,i}$ in each mode $i \in \mathcal{I}$ such that the analogue of (2.19) is satisfied, i.e.,

$$
\int_{[0,T]\times X_i} t^{\alpha_0} x^\alpha [\tilde{u}_{k,i}]_j(t,x) \int_U d\mu_k^{i*}(t,x,u) = \int_{[0,T]\times X_i} t^{\alpha_0} x^\alpha \int_U [u]_j \, d\mu_k^{i*}(t,x,u) \tag{2.24}
$$

for all $i \in \mathcal{I}$, $j \in \{1, \cdots, m\}$, and $(\alpha_0, \alpha) \in \mathbb{N} \times \mathbb{N}^{n_i}$ satisfying $\sum_{l=0}^n \alpha_l \le k$, $\alpha_l \ge 0$. Here μ_k^{i*}

is any measure whose truncated moments match $y^*_{\mu^i}$. In fact, these linear equations written with respect to the coefficients of $[u^*_{k,i}]_j$ are expressible in terms of the optimal solution y^*_{k,μ^i}. To see this, define (t,x)-moment matrix of y^*_{k,μ^i} as:

$$\left[M_k^{(t,x)}(y^*_{k,\mu^i})\right]_{(\alpha_0,\alpha)(\beta_0,\beta)} = L_{y^*_{k,\mu^i}}(t^{\alpha_0+\beta_0}x^{\alpha+\beta}u^{\mathbf{0}}) \tag{2.25}$$

for all $i \in \mathcal{I}$, and $(\alpha_0,\alpha,\mathbf{0}),(\beta_0,\beta,\mathbf{0}) \in \mathbb{N}\times\mathbb{N}^{n_i}\times\{0\}^m$ satisfying $\sum_{l=0}^n \alpha_l \le k$, $\alpha_l \ge 0$, $\sum_{l=0}^n \beta_l \le k$, $\beta_l \ge 0$. Also define a vector b_k^j as

$$\left[b_k^j(y^*_{k,\mu^i})\right]_\alpha = L_{y^*_{k,\mu^i}}(t^{\alpha_0}x^\alpha \cdot [u]_j) \tag{2.26}$$

for all $j \in \{1,\cdots,m\}$, and $(\alpha_0,\alpha) \in \mathbb{N}\times\mathbb{N}^{n_i}$ satisfying $\sum_{l=0}^n \alpha_l \le k$, $\alpha_l \ge 0$. Direct calculation shows Equation (2.24) is equivalent as the following linear system of equations:

$$M_k^{(t,x)}(y^*_{k,\mu^i})\,\mathrm{vec}([u^*_{k,i}]_j) = b_k^j(y^*_{k,\mu^i}) \tag{2.27}$$

Finally, we prove the convergence of the sequence of controllers generated by (2.27) by applying [MVTT14, Theorem 4.5]:

Theorem 21. *Let $\{y^*_{k,\xi}\}_{(\xi,i)\in\Xi}$ be an optimizer of (P_k), and let $\{\mu_k^{i*}\}_{i\in\mathcal{I}}$ be a set of measures such that the truncated moments of μ_k^{i*} match y^*_{k,μ^i} for each $i \in \mathcal{I}$. For each $k \in \mathbb{N}$, let $u^*_{k,i}$ denote the controller from (2.27), and $\tilde{u}_i$ be the optimal control law in mode $i \in \mathcal{I}$ from Theorem 18; then, there exists a subsequence $\{k_l\}_{l\in\mathbb{N}} \subset \mathbb{N}$ such that for all $i \in \mathcal{I}$, $v_i \in C^1([0,T]\times X_i)$, and $j \in \{1,\cdots,m\}$, as $l \to \infty$, the value $\int_{[0,T]\times X_i} v_i(t,x)[u^*_{k_l,i}]_j(t,x)\,d\mu^{i*}_{t,x;k_l}(t,x)$ converges to $\int_{[0,T]\times X_i} v_i(t,x)\,[\tilde{u}_i(t,x)]_j\,d\mu^{i*}_{t,x}(t,x)$.*

2.6 Results

This section illustrates the performance of our approach using several examples. Our algorithm is implemented using MOSEK [ApS17]. The trajectory is obtained by plugging the computed, saturated control law back into the system dynamics in each mode and simulating forward using a standard ODE solver with event detection. To provide a thorough comparison, all examples are also solved by fixing the sequence of transitions and optimizing over each mode with the method proposed in [LHPT08, ZMV16]. Since the optimal sequence is not known a priori, this method is then applied over all feasible sequences of bounded total lengths. In addition, all examples are solved either analytically or using GPOPS-II [PR14] by iterating through a finite set of possible transitions. Notice that in this latter instance we fix the sequence of transition in each GPOPS-II

call and provide an initial guess. All experiments are performed on an Intel Xeon, 144 core, 2.40 GHz, 1056 GB RAM machine. Our code and detailed description of the examples are available online at https://github.com/pczhao/hybridOCP.git.

2.6.1 Hybridized Double Integrator

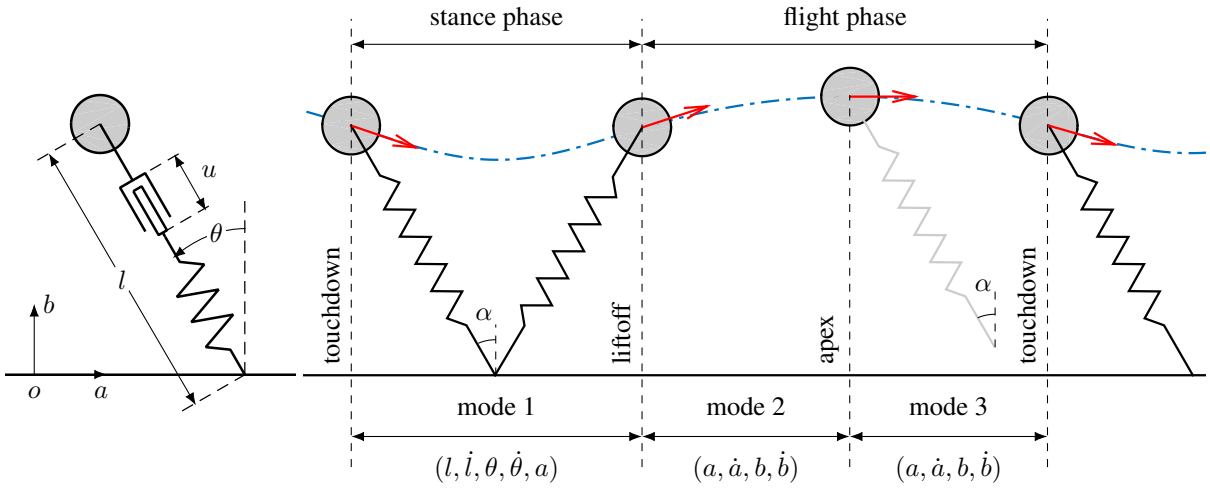
We first consider a double integrator with states $x = (x_1, x_2) \in \mathbb{R}^2$ and input $u \in [-1, 1]$. We hybridize this system by dividing the domain into two parts $X_1 = [0.5, 2] \times [-1, 1]$ and $X_2 = [-1, 0.5] \times [-1, 1]$ and with transitions only from mode 1 to mode 2 with an identity reset map between them. The guard is defined as $\{0.5\} \times \{[-1, -10^{-3}] \cup [10^{-3}, 1]\}$. We solve a Linear Quadratic Regulator (LQR) problem, where the goal is to drive the system towards $(0, 0)$ while minimizing the control action. The problem is setup according to Table 2.1. Note that Assumptions 2-5 are satisfied. Our results, which are summarized in Table 2.2, are compared to those generated by [ZMV16] with the degree of relaxation as $2k = 12$ when applied to finite mode sequences of the total length two. Table 2.2 also describes the results generated by a standard LQR solver which does not treat the problem as hybrid. This latter result is treated as ground truth. The proposed method is able to generate tight lower bounds and the optimal sequence of transitions even when degree of relaxation is low ($2k = 6$).

2.6.2 Dubins Car Model with Shortcut Path

The next example illustrates our algorithm can work with different dimensions in each mode. Consider a planar Dubins Car model with the states $x = (x_1, x_2, x_3) \in [-1, 1] \times [-1, 1] \times [-\pi/2, \pi/2]$ representing the 2D position and heading angle, and the inputs $u = (v, \omega) \in [10^{-3}, 1] \times [-3, 3]$ representing the linear and angular velocity. We hybridize this system by dividing the domain into two parts along the line $x_2 = 0$ and defining an identity reset map. Note that only transitions from the mode where x_2 is greater than or equal to zero to the mode where x_2 is less than or equal to zero are permitted. We also add to the system another one-dimensional mode with dynamics $\dot{x} = -v$, where $x \in [-1, 1]$ and $v \in [10^{-3}, 2]$. We connect this mode with the other two modes by defining $S_{(1,3)} = [-1, 1] \times \{1\} \times ([-\pi/2, -10^{-3}] \cup [10^{-3}, \pi/2])$, $R_{(1,3)}(x) = 1$, $S_{(3,2)} = \{-1\}$, and $R_{(3,2)} = (0.6, -0.8, 0)$. We are interested in solving an optimal control problem where the goal is to get to the target position as quickly as possible. To solve this free final time problem, we modify HLE by substituting $\delta_T \otimes \mu_T^i$ with μ_T^i whose support is in $[0, T] \times X_T$, for all $i \in \mathcal{I}$. (P) and (D) can be modified accordingly. Notice that by treating the measure associated with the time-varying target set as a guard measure without any associated reset map, we can extend Theorems 9 and 12 to show that (P) can solve the free final time problem [LHPT08, Remark 2.1]. The optimal control problem is defined in Table 2.1 so that Assumptions 2-5 are satisfied.

Notice the transition sequences "1-2" and "1-3-2" are both feasible in this instance according to our guard definition, but a direct calculation shows that we arrived at the target point in less time by taking the "shortcut path" in mode 3. This problem is solved using our algorithm with degrees of relaxation $2k = 6$, $2k = 8$, and $2k = 10$. As a comparison, we also solved the problem using the method presented in [ZMV16] with a $2k = 10$ degree of relaxation by applying it to each possible feasible mode sequence that has a maximum length 3, and treats the analytically computed optimal control as ground truth. The results are compared in Table 2.2. Our algorithm is able to pick the transition sequence "1-3-2" and approximate the true optimal solution even when $2k = 6$.

2.6.3 SLIP Model



(a) SLIP model

(b) SLIP locomotion phases and hybrid modes

Figure 2.2: An illustration of the SLIP model (left) and its hybrid modes (right)

The Spring-Loaded Inverted Pendulum (SLIP) is a model that describes the center-of-mass dynamics of animals and has been used to perform control synthesis for legged robots [HFKG06]. We may simulate the system numerically, but the optimal control problem is still difficult to solve if the sequence of transition is not known beforehand. We focus on the active SLIP model (Fig. 2.2a), which is an actuated mass-spring physical system, modeled as a point mass, M, a mass-less spring leg with stiffness k and length l, and a mass-less actuator u. The behavior of such a system can be fully characterized using 8 variables: leg length l, leg angle θ, horizontal displacement a, vertical displacement b, and their time derivatives (denoted as $\dot{l}$, $\dot{\theta}$, $\dot{a}$, and $\dot{b}$, respectively). The system states in each of the 3 hybrid modes are defined as shown in Fig. 2.2b. The github repo describes the physical parameters, dynamics, guards, and reset maps. To ensure that we satisfy

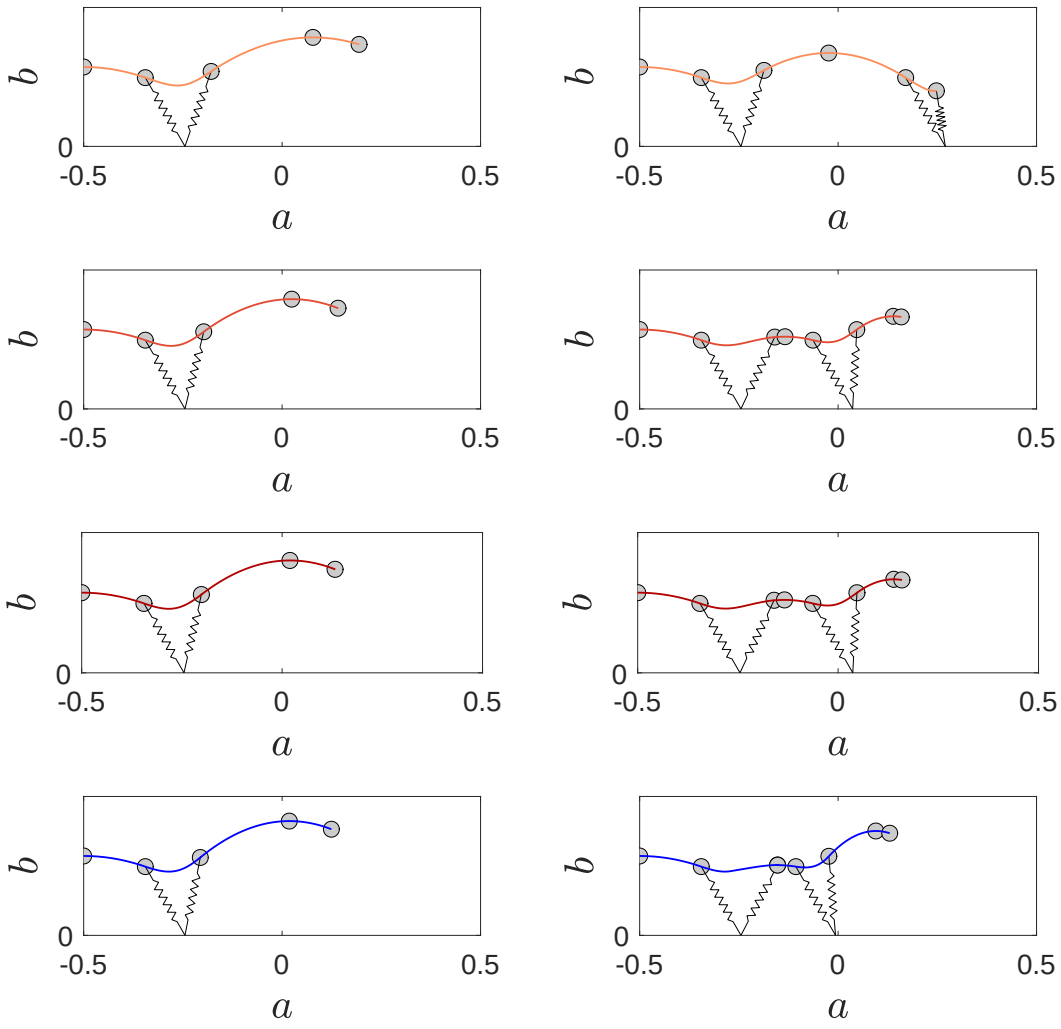
Assumptions 2-5, the guard at touch-down is satisfied when $\dot{b} \leq -10^{-3}$, and the guard at lift-off is satisfied when $\dot{l} \geq 10^{-3}$.

We fix the initial condition, and consider the following two hybrid optimal control problems for the active SLIP. In the first problem, we maximize the vertical displacement b up to time $T = 2.5$. In the stance phase, the 1st-order Taylor approximation $b = l\cos(\theta) \approx l$ is used. In the second problem, we define a constant-speed reference trajectory $a(t) = vt - 0.5$ in the horizontal coordinate, then try to follow this trajectory with active SLIP up to time $T = 3$. The optimal control problems are defined according to Table 2.1. Note that these problems are defined such that the optimal transition sequences are different in each instance, and some modes are visited multiple times.

The optimization problems are solved by our algorithm with degrees of relaxation $2k = 4$, $2k = 6$, and $2k = 8$. For the sake of comparison, the same problems are also solved using the method presented in [ZMV16] and GPOPS-II for all possible, feasible mode sequences of maximum total length 12. The results are compared in Fig. 2.3 and Table 2.2. The proposed method is able to generate the optimal sequence of transitions even at low relaxation degrees (e.g., $2k = 6$) while other methods have to search through all possible sequences. In particular, the proposed method takes an order of magnitude less time to find the optimal sequence of transitions on both examples when compared to GPOPS-II.

2.7 Conclusion

This chapter proposes a convex approach for solving hybrid optimal control problems by relating the trajectories of hybrid systems to the solutions of a system of linear equations over measures. The hybrid optimal control problem is then formulated as an infinite-dimensional LP that does not require pre-specifying the sequence of possible transitions. A sequence of provably convergent SDPs to this LP are constructed to approximate the optimal cost from below and synthesize the optimal control law. Though it does not require pre-specifying the sequence of transitions of the hybrid system, the proposed method can be difficult to apply when the state space dimension is high, since the number of decision variables in the SDP grows exponentially with the state space dimension.



(a) Maximizing vertical displacement

(b) Tracking constant speed $v = 0.1$

Figure 2.3: An illustration of the performance of our algorithm on the active SLIP model. The blue lines are the optimal control computed by GPOPS-II by iterating through all the possible transition sequences, and the red lines of various saturation are controls generated by our method. As the saturation increases, the corresponding degree of relaxation increases between $2k = 4$ to $2k = 6$ to $2k = 8$. Fig. 2.3a shows trajectories that maximize vertical displacement, where the optimal solution goes through three transitions; Fig. 2.3b shows trajectories that track $v = 0.1$, where the optimal solution goes through 6 transitions.

Table 2.1: The setup for each example problem.

	Mode	$i=1$	$i=2$	$i=3$
Double Integrator LQR	h_i	$x_1^2+x_2^2+20u^2$	$x_1^2+x_2^2+20u^2$	N/A
	H_i	0	0	
	x_0	$(1,1)$	N/A	
	X_{T_i}	$[0.5+10^{-3},2]\times[-1,1]$	X_2	
	T	5 or 15		
Dubins Car	h_i	1	1	1
	H_i	0	0	0
	x_0	(-0.8,0.8,0)	N/A	N/A
	X_{T_i}	N/A	$\{0.8\}\times\{-0.8\}\times[-\pi/2,\pi/2]$	N/A
	T	3		
SLIP Max jump	h_i	$-x_1$	$-x_3$	$-x_3$
	H_i	0	0	0
	x_0	N/A	N/A	(-0.5,0.3,0.2,0)
	X_{T_i}	$\{x\in X_1 \mid x_1\leq l_0-10^{-3}\}$	$\{x\in X_2 \mid x_4\geq 10^{-3}\}$	$\{x\in X_3 \mid x_3\geq l_0\cos(\alpha)+10^{-3}\}$
	T	2.5		
SLIP Track speed	h_i	$(vt-0.5-x_5)^2$	$(vt-0.5-x_1)^2$	$(vt-0.5-x_1)^2$
	H_i	0	0	0
	x_0	N/A	N/A	(-0.5,0.3,0.2,0)
	X_{T_i}	$\{x\in X_1 \mid x_1\leq l_0-10^{-3}\}$	$\{x\in X_2 \mid x_4\geq 10^{-3}\}$	$\{x\in X_3 \mid x_3\geq l_0\cos(\alpha)+10^{-3}\}$
	T	3		

Table 2.2: Numerical results for the proposed algorithm on each example.

		Computation time	Cost from optimization	Cost from simulation
Double Integrator LQR $T = 5$	$2k = 6$	3.2004[s]	24.9496	24.9908
	$2k = 8$	9.4318[s]	24.9496	24.9908
	$2k = 12$	252.8047[s]	24.9496	24.9914
	[ZMV16], $2k = 12$	326.1610[s]	24.9496	24.9905
	Ground truth	N/A	24.9503	N/A
Double Integrator LQR $T = 15$	$2k = 6$	3.1583[s]	26.1993	26.3557
	$2k = 8$	9.8637[s]	26.1993	26.3644
	$2k = 12$	219.8932[s]	26.1994	26.3710
	[ZMV16], $2k = 12$	295.1562[s]	26.1993	26.3694
	Ground truth	N/A	26.2033	N/A
Dubin's Car	$2k = 6$	67.6682[s]	1.5640	1.5748
	$2k = 8$	956.6177[s]	1.5646	1.5718
	$2k = 10$	1.0654×10^4[s]	1.5648	1.5708
	[ZMV16], $2k = 10$	2.6259×10^4[s]	1.5648	1.5708
	Ground truth	N/A	1.5651	N/A
SLIP Max Jump	$2k = 4$	45.1598[s]	-0.6962	-0.5525
	$2k = 6$	584.8139[s]	-0.5815	-0.5474
	$2k = 8$	7.7398×10^3[s]	-0.5776	-0.5545
	[ZMV16], $2k = 8$	2.1225×10^5[s]	-0.5737	-0.5728
	GPOPS-II	792.9885[s]	-0.5735	N/A
SLIP Track Speed	$2k = 4$	40.7036[s]	0.0534	0.2250
	$2k = 6$	565.7164[s]	0.1417	0.1813
	$2k = 8$	1.0263×10^4[s]	0.1523	0.1825
	[ZMV16], $2k = 8$	2.2373×10^5[s]	0.1592	0.1718
	GPOPS-II	673.5100[s]	0.1626	N/A

CHAPTER 3

Real-Time Safe Control for A Planar Bipedal Robot Model

3.1 Introduction

[1] Legged robots are an ideal system to perform locomotion on unstructured terrains. Unfortunately designing controllers for legged systems to operate safely in such situations has proven challenging. To robustly traverse such environments, an ideal control synthesis technique for legged robotic systems should satisfy several requirements.

First, since uncertainties and disturbances may appear during operation, any algorithm for control synthesis should run in real-time. Second, since modeling contact can be challenging, any control synthesis technique should be able to accommodate model uncertainty. Third, since the most appropriate controller may be a function of the environment and given task, a control synthesis algorithm should optimize over as rich a family of control inputs at run-time as possible. Finally, since falling can be costly both in time and expense, a control synthesis technique should be able to guarantee the satisfactory behavior of any constructed controller. As illustrated in Fig. 3.1, this chapter presents an optimization-based algorithm to design gaits for legged robotic systems while satisfying each of these requirements.

We begin by summarizing related work with an emphasis on techniques that are able to make guarantees on the safety of the designed controller. For instance, the Zero-Moment Point approach [VS72] characterizes the stability of a legged robot with planar feet by defining the notion of the Zero-Moment Point and requiring that it remains within the robot's base of support. Though this requirement can be used to design a controller that can avoid falling at run-time, the gaits designed by the ZMP approach are static and energetically expensive [Kuo07, WCC+07, Section 10.8].

In contrast, the Hybrid Zero Dynamics approach, which relies upon feedback linearization to drive the actuated degrees of freedom of a robot towards a lower dimensional manifold, is

[1] This chapter was previously published in 2020 IEEE International Conference on Robotics and Automation (ICRA) [LZG+20].

able to synthesize a controller which generates gaits that are more dynamic. Though this approach can generate safety preserving controllers for legged systems in real-time in the presence of model uncertainty [AGSG14, HXA15, NS15, NS16, NHG+16], it is only able to prove that the gait associated with a synthesized control is locally stable. As a result, it is non-trivial to switch between multiple constructed controllers while preserving any safety guarantee. Recent work has extended the ability of the hybrid zero dynamic approach beyond a single neighborhood of any synthesized gait [MVP16, VP18, ATJ+17, SARV19]. These extensions either assume full-actuation [ATJ+17] or ignore the behavior of the legged system off the lower dimensional manifold [MVP16, VP18, SARV19].

Rather than designing controllers for legged systems, other techniques have focused on characterizing the limits of safe performance by using Sums-of-Squares (SOS) optimization [Par00]. These approaches use semi-definite programming to identify the limits of safety in the state space of a system as well as associated controllers for hybrid systems [PJ04, SVBT14a]. These safe sets can take the form of *reachable sets* [KPT16, SVBT14a] or *invariant sets* in state space [Wie02, PJ04, PKT17]. However, the representation of each of these sets in state space restricts the size of the problem that can be tackled by these approaches and as a result, these SOS-based approaches have been primarily applied to reduced models of walking robots: ranging from spring mass models [ZMV17b], to inverted pendulum models [KPT16, TBM17] and to inverted pendulum models with an offset torso mass [PKT17]. Unfortunately the differences between these simple models and real robots makes it challenging to extend the safety guarantees to more realistic real-world models.

This chapter addresses the shortcomings of prior work by making the following four contributions. First, in Section 3.3.1, we describe a set of outputs that are functions of the state of the robot, which can be used to determine whether a particular gait can be safely tracked by a legged system without falling. In particular, if a particular gait's outputs satisfy a set of inequality constraints that we define, then we show that the gait can be safely tracked by the legged system without falling. To design gaits over N-steps that do not fall over, one could begin by forward propagating these outputs via the robot's dynamics for N-steps. Unfortunately performing this computation can be intractable due to the high-dimensionality of the robot's dynamics. To address this challenge, our second contribution, in Section 3.3.2, leverages the anchor and template framework to construct a simple model (template) whose outputs are sufficient to predict the behavior of the full model's (anchor's) outputs [FK99] under the assumption that the modeling error between the anchor and template can be bounded. Third, in Section 3.4.1, we develop an offline method to compute a gait parameterized forward reachable set that describes the evolution of the outputs of the simple model.

Similar to recently developed work on motion planning for ground and aerial vehicles [MT17a,

HCH+17, KVB+18, KHV19], one can then require that all possible outputs in the forward reachable set satisfy a family of conditions that we define to ensure that the robot does not fall over during the N-steps. Finally, in Section 3.4.2, we describe how to incorporate these conditions in a Model Predictive Control (MPC) framework that are sufficient to ensure N-step walking safely. Note, to simplify exposition, this chapter focuses on an example implementation on a 14-dimensional model of the robot RABBIT that is described in Section 3.2. The remainder of this chapter is organized as follows. Section 3.5 demonstrates the performance of the proposed approach on a walking example and Section 3.6 concludes the chapter.

3.2 Preliminaries

This section introduces the notation, the dynamic model of the RABBIT robot, and a Simplified Biped Model (SBM) that are used throughout the remainder of this paper. The following notation is adopted in this manuscript. All sets are denoted using calligraphic capital letters. Let $\mathbb{R}$ denote the set of real numbers, and let $\mathbb{N}_+$ denote the collection of all non-negative integers. Give a set $\mathcal{A} \subset \mathbb{R}^n$ for some $n \in \mathbb{N}_+$, let $C^1(\mathcal{A})$ denote the set of all differentiable continuous functions from $\mathcal{A}$ to $\mathbb{R}$ whose derivative is continuous and let $\lambda_{\mathcal{A}}$ denote the Lebesgue measure which is supported on $\mathcal{A}$.

3.2.1 RABBIT Model (Anchor)

This chapter considers the walking motion of a planar 5-link model of RABBIT [CAA+03]. The walking motion of the RABBIT model consists of alternating phases of *single stance* (one leg in contact with the ground) and *double stance* (both legs in contact with the ground). While in single stance, the leg in contact with the ground is called the *stance leg*, and the non-stance leg is called the *swing leg*. The double stance phase is instantaneous. The configuration of the robot at time t is $q(t) := [q_h(t), q_v(t), q_1(t), q_2(t), q_3(t), q_4(t), q_5(t)]^\top \in \mathcal{Q} \subset \mathbb{R}^7$, where $(q_h(t), q_v(t))$ are Cartesian position of the robot hip; $q_1(t)$ is the torso angle relative to the upright direction; $q_2(t)$ and $q_4(t)$ are the hip angles relative to stance and swing leg, respectively; and $q_3(t)$ and $q_5(t)$ are the knee angles. The joints (q_2, q_3, q_4, q_5) are actuated, and q_1 is an underactuated degree of freedom. Let $\theta(q) := q_1 + q_2 + q_3/2$ denote the *stance leg angle*, and let $\phi(q) := q_1 + q_4 + q_5/2$ denote the *swing leg angle*. We refer to the configuration when the robot hip is right above the stance foot, i.e. $\theta = \pi$, as *mid-stance*. We refer to the motion between the i-th and $(i+1)$-st swing leg foot touch down with the ground as the *i-th step*.

Using the method of Lagrange, we can obtain a continuous dynamic model of the robot during

swing phase:

$$\dot{a}(t) = f(a(t), u(t)) \tag{3.1}$$

where $a(t) = [q^\top(t), \dot{q}^\top(t)]^\top \in T\mathcal{Q} \subset \mathbb{R}^{14}$ denotes the tangent bundle of $\mathcal{Q}$, $u(t) \in U$, U describes the permitted inputs to the system, and t denotes time. We model the RABBIT as a hybrid system and describe the instantaneous change using the notation of a *guard* and a *reset map*. That is, suppose $(\theta(q(t)), c_{\text{foot}}(q(t)))$ denotes the stance leg angle and the vertical position of the swing foot relative to the stance foot, respectively, given a configuration $q(t)$ at time t. The guard $\mathcal{G}$ is $\{(b, b') \in T\mathcal{Q} \mid \pi/2 < \theta(b) < 3\pi/2, c_{\text{foot}}(b) = 0 \text{ and } \dot{c}_{\text{foot}}(b, b') < 0\}$. Notice the force of the ground contact imposes a holonomic constraint on stance foot position, which enables one to obtain a reset map: [WCC+07, Section 3.4.2]:

$$\dot{q}^+(t) = \Delta\big(\dot{q}^-(t)\big), \tag{3.2}$$

where Δ describes the relationship between the pre-impact and post impact velocities. More details about the definition and derivation of this hybrid model can be found in [WCC+07, Section 3.4].

To simplify exposition, this chapter at run-time optimizes over a family of reference gaits that are characterized by their average velocity and step length. These reference gaits are described by a vector of *control parameters* $P(i) = \big(p_1(i), p_2(i)\big) \in \mathcal{P}$ for all $i \in \mathbb{N}$, where $p_1(i)$ denotes the average horizontal velocity and $p_2(i)$ denotes the step length between the i-th and $(i+1)$-st mid-stance position. Note $\mathcal{P}$ is compact. These reference gaits are generated by solving a finite family of nonlinear optimization problems using FROST in which we incorporate $p_1(i)$, $p_2(i)$, and periodicity as constraints, and minimize the average torque squared over the gait period [HA17]. Each of these problems yields a reference trajectory parameterized by $P(i)$ and interpolation is applied over these generated gaits to generate a continuum of gaits. Given a control parameter, a control input into the RABBIT model is generated by tracking the corresponding reference trajectories using a classical PD controller.

Next, we define a solution to the hybrid model as a pair $(\mathcal{I}, a)$, where $\mathcal{I} = \{I_i\}_{i=0}^N$ is a *hybrid time set* with I_i being intervals in $\mathbb{R}$, and $a = \{a_i(\cdot)\}_{i=0}^N$ is a finite sequence of functions with each element $a_i(\cdot) : I_i \to T\mathcal{Q}$ satisfying the dynamics (3.1) over I_i where $N \in \mathbb{N}$ [LST12, Definitions 3.3, 3.4, 3.5]. Denote each $I_i := [\tau_i^+, \tau_{i+1}^-]$ for all $i < N$. τ_i corresponds to the time of the transition between $(i-1)$-th to i-th step. We let τ_i^- correspond to the time just before the transition and and τ_i^+ correspond to the time just after the transition. Since transitions are assumed to be instantaneous, $\tau_i = \tau_i^- = \tau_i^+$ if all values exist. When a transition never happens during the i-th step, we denote $\tau_{i-1}^- = +\infty$. Note when $\tau_{i+1} < \infty$, $a_i(\tau_{i+1}^-) \in \mathcal{G}$ and $a_{i+1}(\tau_{i+1}^+) \in \Delta(a_i(\tau_{i+1}^-))$.

3.2.2 Simplified Biped Model (Template)

As we show in Section 3.5, performing online optimization with the full RABBIT model is intractable due to the size of its state space. In contrast, performing online optimization with the *Simplified Biped Model* (SBM) adopted from [WKW08] is tractable. This model consists of a point-mass M and two mass-less legs each with a constant length l. The configuration of the SBM at time t is described by the stance leg angle, $\hat{\theta}$, and the swing leg angle, $\hat{\phi}$. The guard is the set of configurations when $\hat{\theta} + \hat{\phi} = 2\pi$. The swing leg swings immediately to a specified step length. During the swing phase, one can use the method of Lagrange to describe the evolution of the configuration as a function of the current configuration and the input. Subsequent to the instantaneous double stance phase, an impact with the ground happens with a coefficient of restitution of 0. We denote a hybrid execution of the SBM as a pair $(\hat{\mathcal{I}}, \hat{a})$ where $\hat{\mathcal{I}} = \{\hat{I}_i\}_{i=0}^{N}$ is a hybrid time set with $\hat{I}_i := [\hat{\tau}_i^+, \hat{\tau}_{i+1}^-]$ and $\hat{a} = \{\hat{a}_i(\cdot)\}_{i=0}^{N}$ is a finite sequence of solutions to the SBM's equations of motion.

3.3 Outputs to Describe Successful Walking

During online optimization, we want to optimize over the space of parameterized inputs while introducing a constraint to guarantee that the robot does not fall over. This section first formalizes what it means for the RABBIT model to walk successfully without falling over. Unfortunately due to the high-dimensionality of the RABBIT model, implementing this definition directly as a constraint during online optimization is intractable. To address this problem, in Section 3.3.1 defines a set of outputs that are functions of the state of RABBIT and proves that the value of these outputs can determine whether RABBIT is able to walk successfully. Subsequently in Section 3.3.2 we define a corresponding set of outputs that are functions of the state of the SBM and illustrate how their values can be used to determine whether RABBIT is able to walk successfully.

To define successful walking on RABBIT, we begin by defining the time during step i at which mid-stance occurs (i.e., the largest time t at which $\theta(q(t)) = \pi$ during I_i) as

$$t_i^{MS} := \begin{cases} +\infty, & \text{if } \theta(q(t)) < \pi \quad \forall t \in I_i, \\ -\infty, & \text{if } \theta(q(t)) > \pi \quad \forall t \in I_i, \\ \max\{t \in I_i \mid \theta(q(t)) = \pi\}, & \text{otherwise.} \end{cases} \tag{3.3}$$

Note if mid-stance is never reached during step i, then the mid-stance time is defined as plus or minus infinity depending upon if the hip-angle remains less than π or greater than π during step i, respectively. Using this definition, we formally define successful walking for the RABBIT model

as:

Definition 22. *The RABBIT model* walks successfully *in step* $i \in \mathbb{N}$ *if*

1. $t_i^{MS} \neq \pm\infty$,
2. $\pi/2 < \theta(q(t)) < 3\pi/2$ *for all* $t \in I_i$, *and*
3. $\tau_{i+1}^- < +\infty$.

To understand this definition, note that the first requirement ensures that mid-stance is reached, the second requirement ensures that the hip remains above the ground, and the final requirement ensures that the swing leg actually makes contact with the ground. Though satisfying this definition ensures that RABBIT takes a step, enforcing this condition directly during optimization can be cumbersome due to the high dimensionality of the RABBIT dynamics.

3.3.1 Outputs to Describe Successful RABBIT Walking

This subsection defines a set of discrete outputs that are functions of the state of RABBIT model and illustrates how they can be used to predict failure. We begin by defining another time variable t_i^0:

$$t_i^0 := \begin{cases} \tau_i^+, & \text{if } \dot{\theta}(q(t), \dot{q}(t)) < 0 \quad \forall t \in I_i, \\ \tau_{i+1}^-, & \text{if } \dot{\theta}(q(t), \dot{q}(t)) > 0 \quad \forall t \in I_i, \\ \max\{t \in I_i \mid \dot{\theta}(q(t), \dot{q}(t)) = 0\}, & \text{otherwise.} \end{cases} \tag{3.4}$$

Note t_i^0 is defined to be the last time in I_i when a sign change of $\dot{\theta}$ occurs; when a sign change does not occur, t_i^0 is defined as an endpoint of I_i associated with the sign of $\dot{\theta}$.

We first define an output, $y_1 : \mathbb{N} \to \mathbb{R}$ that can be used to ensure that $t_i^{MS} \neq +\infty$:

$$y_1(i) := \begin{cases} \dot{\theta}(q(t_i^{MS}), \dot{q}(t_i^{MS})), & \text{if } t_i^{MS} \neq \pm\infty, \\ -\sqrt{2\mathrm{g}(l_{\mathrm{st}}(t_i^0) - q_v(t_i^0))}/l_{\mathrm{st}}(t_i^0), & \text{if } t_i^{MS} = +\infty, \\ 1 & \text{if } t_i^{MS} = -\infty, \end{cases} \tag{3.5}$$

where g is gravity and $l_{\mathrm{st}}(t_i^0)$ is the stance leg length at time t_i^0. Note that $y_1(i)$ is the hip angular velocity when the mid-stance position is reached during the i-th step. When the mid-stance position is not reached, $-y_1(i)$ represents the additional hip angular velocity needed to reach the mid-stance position. In particular, notice $t_i^{MS} \neq +\infty$ whenever $y_1(i) \geq 0$.

Next, we define an output $y_2 : \mathbb{N} \to \mathbb{R}$ that can be used to ensure that $t_i^{MS} \neq -\infty$:

$$y_2(i) := \begin{cases} \phi(q(\tau_{i+1}^-)), & \text{if } \tau_{i+1}^- < +\infty, \\ 2\pi, & \text{otherwise.} \end{cases} \tag{3.6}$$

Note, $y_2(i)$ is the swing leg angle at touch-down at the end of the i-th step; if touch-down does not occur, $y_2(i)$ is defined as 2π. Recall $\phi(q(\tau_{i+1}^-)) = \theta(q(\tau_{i+1}^+))$, so if $y_2(i) \leq \pi$, it then follows from (3.3) and (3.6) that $t_{i+1}^{MS} \neq -\infty$ and $\tau_{i+1}^- < +\infty$. Fig. 3.2 illustrates the behavior of y_1 and y_2.

We now define our last two outputs $y_3, y_4 : \mathbb{N} \to \mathbb{R} \cup \{-\infty, +\infty\}$ that can be used to ensure that the hip stays above the ground:

$$y_3(i) := \begin{cases} \inf\{\theta(q(t)) \mid t \in [t_i^{MS}, t_{i+1}^{MS}]\}, & \text{if } t_{i+1}^{MS}, t_i^{MS} \in \mathbb{R}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{3.7}$$

$$y_4(i) := \begin{cases} \sup\{\theta(q(t)) \mid t \in [t_i^{MS}, t_{i+1}^{MS}]\}, & \text{if } t_{i+1}^{MS}, t_i^{MS} \in \mathbb{R}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.8}$$

Finally, we let $\mathcal{Y} := \mathbb{R} \times \mathbb{R} \times (\mathbb{R} \cup \{-\infty, +\infty\}) \times \mathbb{R}$.

The outputs are defined based on the observation that the hip usually has forward speed (e.g. moving forward, rather than falling backwards) at mid-stance and appears between thee two legs at touch-down when the RABBIT model walks safely. Specifically, y_1 represents the hip angular velocity at mid-stances and y_2 represents the swing leg angle at touch-downs. y_3 and y_4 are defined as the maximum and minimum stance leg angles between adjacent mid-stances, which are used to indicate whether the hip hits the ground.

Using these definitions, we can prove the following theorem that constructs a sufficient condition to ensure successful walking by RABBIT.

Theorem 23. *Suppose that the* 0*-th step can be successfully completed (i.e.* τ_0^+ *and* t_0^{MS} *are finite,* $\inf\{\theta(q(t)) \mid t \in [\tau_0^+, t_0^{MS}]\} > \pi/2$*, and* $\sup\{\theta(q(t)) \mid t \in [\tau_0^+, t_0^{MS}]\} < 3\pi/2$*)). Suppose* $y_1(i) \geq 0$*,* $y_2(i) \leq \pi$*,* $y_3(i) > \pi/2$ *and* $y_4(i) < 3\pi/2$ *for each* $i \in \{0, \cdots, N\}$*, then the robot walks successfully at the* i*-th step for each* $i \in \{0, \cdots, N\}$*.*

Proof. Notice $y_1(i) \geq 0 \Rightarrow t_i^{MS} \neq +\infty$ and $y_2(i) \leq \pi \Rightarrow t_{i+1}^{MS} \neq -\infty$ for each $i \in \{1, \cdots, N\}$. By induction we have t_i^{MS} is finite $\forall i \in \{1, \cdots, N\}$. $y_2(i) \leq \pi < 2\pi$ implies that $\tau_{i+1}^- < +\infty$. By using the definitions of y_3 and y_4, one has that the robot walks successfully in the i-th step based on Definition 22. □

3.3.2 Approximating Outputs Using the SBM

Finding an analytical expression describing the evolution of each of the outputs can be challenging. Instead we define corresponding outputs

$$\hat{y}(i) := \big(\hat{y}_1(i), \hat{y}_2(i), \hat{y}_3(i), \hat{y}_4(i)\big) \in \mathcal{Y} \tag{3.9}$$

for SBM. Importantly, the dynamics of each of these corresponding outputs can be succinctly described.

As we did for the RABBIT model, consider the following set of definitions for the SBM:

$$\hat{t}_i^{MS} := \begin{cases} +\infty, & \text{if } \hat{\theta}(t) < \pi \quad \forall t \in \hat{I}_i, \\ -\infty, & \text{if } \hat{\theta}(t) > \pi \quad \forall t \in \hat{I}_i, \\ \max\{t \in \hat{I}_i \mid \hat{\theta}(t) = \pi\}, & \text{otherwise.} \end{cases} \tag{3.10}$$

$$\hat{t}_i^{0} := \begin{cases} \hat{\tau}_i^{+}, & \text{if } \dot{\hat{\theta}}(t) < 0 \quad \forall t \in \hat{I}_i, \\ \hat{\tau}_{i+1}^{-}, & \text{if } \dot{\hat{\theta}}(t) > 0 \quad \forall t \in \hat{I}_i, \\ \max\{t \in \hat{I}_i \mid \dot{\hat{\theta}}(t) = 0\}, & \text{otherwise.} \end{cases} \tag{3.11}$$

$$\hat{y}_1(i) := \begin{cases} \dot{\hat{\theta}}(\hat{t}_i^{MS}), & \text{if } \hat{t}_i^{MS} \neq \pm\infty \\ -\sqrt{2\mathrm{g}(l(1+\cos(\hat{\theta}(\hat{t}_i^{0}))))}/l, & \text{if } \hat{t}_i^{MS} = +\infty \\ 1 & \text{if } \hat{t}_i^{MS} = -\infty, \end{cases} \tag{3.12}$$

$$\hat{y}_2(i) := \begin{cases} \hat{\phi}(\hat{\tau}_{i+1}^{-}), & \text{if } \hat{\tau}_{i+1}^{-} < +\infty, \\ 2\pi, & \text{otherwise.} \end{cases} \tag{3.13}$$

$$\hat{y}_3(i) := \begin{cases} \inf\{\hat{\theta}(t) \mid t \in [\hat{t}_i^{MS}, \hat{t}_{i+1}^{MS}]\}, & \text{if } \hat{t}_{i+1}^{MS}, \hat{t}_i^{MS} \in \mathbb{R}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{3.14}$$

$$\hat{y}_4(i) := \begin{cases} \sup\{\hat{\theta}(t) \mid t \in [\hat{t}_i^{MS}, \hat{t}_{i+1}^{MS}]\}, & \text{if } \hat{t}_{i+1}^{MS}, \hat{t}_i^{MS} \in \mathbb{R}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.15}$$

The discrete-time dynamics of each of these outputs of SBM can be described by the following

difference equations:

$$\begin{aligned}\hat{y}_1(i+1) &= f_{\hat{y}_1}\big(\hat{y}_1(i), P(i)\big)\\ \hat{y}_2(i) &= f_{\hat{y}_2}\big(P(i)\big)\\ \hat{y}_3(i) &= f_{\hat{y}_3}\big(\hat{y}_1(i), P(i)\big)\\ \hat{y}_4(i) &= f_{\hat{y}_4}\big(\hat{y}_1(i), P(i)\big)\end{aligned} \tag{3.16}$$

for each $i \in \mathbb{N}$, $\hat{y}(i) \in \mathcal{Y}$, and $P(i) \in \mathcal{P}$. Such functions $f_{\hat{y}_1}$, $f_{\hat{y}_2}$, $f_{\hat{y}_3}$ and $f_{\hat{y}_4}$ can be generated using elementary mechanics [2].

To describe the gap between the discrete signals y and $\hat{y}$ we make the following assumption:

Assumption 24. *For any sequence of control parameters, $\{P(i)\}_{i\in N}$, and corresponding sequences of outputs, $\{y_1(i), y_2(i), y_3(i), y_4(i)\}_{i\in\mathbb{N}}$ and $\{\hat{y}_1(i), \hat{y}_2(i), \hat{y}_3(i), \hat{y}_4(i)\}_{i\in\mathbb{N}}$, generated by the RABBIT dynamics and* (3.16)*, respectively, there exists* bounding functions $\underline{B}_1$, $\overline{B}_1 : \mathbb{R}\times\mathcal{P} \to \mathbb{R}$, $\overline{B}_2 : \mathcal{P}\times\mathbb{R}\times\mathcal{P} \to \mathbb{R}$, *and* $\underline{B}_3$, $\overline{B}_4 : \mathbb{R}\times\mathcal{P} \to \mathbb{R}$ *satisfying*

$$\underline{B}_1\big(y_1(i), P(i)\big) \le y_1(i+1) - \hat{y}_1(i+1) \le \overline{B}_1\big(y_1(i), P(i)\big) \tag{3.17}$$

$$y_2(i) - \hat{y}_2(i) \le \overline{B}_2\big(P(i-1), y_1(i), P(i)\big) \tag{3.18}$$

$$y_3(i) - \hat{y}_3(i) \ge \underline{B}_3\big(y_1(i), P(i)\big) \tag{3.19}$$

$$y_4(i) - \hat{y}_4(i) \le \overline{B}_4\big(y_1(i), P(i)\big). \tag{3.20}$$

In other words, if $y_1(i) = \hat{y}_1(i)$, then $\overline{B}_1$, $\underline{B}_1$, $\underline{B}_2$, $\underline{B}_3$, and $\overline{B}_4$ bound the maximum possible difference between $(y_1(i+1), y_2(i), y_3(i), y_4(i))$ and $(\hat{y}_1(i+1), \hat{y}_2(i), \hat{y}_3(i), \hat{y}_4(i))$. Though we do not describe how to construct these bounding functions here due to space limitations, one could apply SOS optimization to generate them [SYA19] or bound the dynamics of the system [KHV19]. Note, constructing such a bound precisely using SOS optimization can be challenging due to the high-dimensionality of the full-order model. However, several papers have proposed techniques that have begun to address these scaling challenges while applying SOS optimization [AM14, KGNSZ19, PY19, TCHL19]. To simplify further exposition, we define the following:

$$\begin{aligned}\mathcal{B}(y_1(i), P(i)) := [&f_{\hat{y}_1}\big(y_1(i), P(i)\big) + \underline{B}_1\big(y_1(i), P(i)\big),\\ &f_{\hat{y}_1}\big(y_1(i), P(i)\big) + \overline{B}_1\big(y_1(i), P(i)\big)]\end{aligned} \tag{3.21}$$

for all $(y_1(i), P(i)) \in \mathbb{R}\times\mathcal{P}$. In particular, it follows from (3.17) that for any sequence of control parameters, $\{P(i)\}_{i\in N}$, and corresponding sequences of outputs, $\{y_1(i)\}_{i\in\mathbb{N}}$ generated by the RABBIT dynamics, $y_1(i+1) \in \mathcal{B}\big(y_1(i), P(i)\big)$ for all $i \in \mathbb{N}$.

[2] A derivation can be found at: https://github.com/pczhao/TA_GaitDesign/blob/master/SBM_dynamics.pdf

3.4 Enforcing N-Step Safe Walking

This section proposes an online MPC framework to design a controller for the RABBIT model that can ensure successful walking for N-step. In fact, when $N = 1$ one can directly apply Theorem 23 and Assumption 24 to generate the following inequality constraints over $y_1(i)$, $P(i-1)$ and $P(i)$ to guarantee walking successfully from the i-th to the $(i+1)$-th mid-stance:

$$f_{\hat{y}_1}\big(y_1(i), P(i)\big) + \underline{B}_1\big(y_1(i), P(i)\big) \geq 0, \tag{3.22}$$

$$f_{\hat{y}_2}\big(P(i)\big) + \overline{B}_2\big(P(i-1), y_1(i), P(i)\big) \leq \pi, \tag{3.23}$$

$$f_{\hat{y}_3}\big(y_1(i), P(i)\big) + \underline{B}_3\big(y_1(i), P(i)\big) > \pi/2, \tag{3.24}$$

$$f_{\hat{y}_4}\big(y_1(i), P(i)\big) + \overline{B}_4\big(y_1(i), P(i)\big) < 3\pi/2. \tag{3.25}$$

Unfortunately, to construct a similar set of constraints when $N > 1$, one has to either compute $(y_1(i+M), y_2(i+M), y_3(i+M), y_4(i+M))$ for each $1 \leq M \leq N$, which can be computationally taxing, or one can apply (3.17) recursively to generate an outer approximation to $y_1(i+M)$ for each $1 \leq M \leq N$ and then apply the remainder of Assumption 24 to generate an outer approximation to $y_2(i+M), y_3(i+M)$, and $y_4(i+M)$ for each $1 \leq M \leq N$. In the latter instance, one would need the entire set of possible values for the outputs to satisfy conditions in Theorem 23 from the i-th step to the $(i+N)$-th step to ensure N-step safe walking. This requires introducing set inclusion constraints that can be cumbersome to enforce at run-time. To address these challenges, Section 3.4.1 describes how to compute in an offline fashion, an N-step *Forward Reachable Set* (FRS) that captures all possible outcomes for the output y_1 from a given initial state and set of control parameters for up to N steps. Subsequently, Section 3.4.2 illustrates how to write down N-step successful walking conditions on outputs (y_2, y_3, y_4) and set up an MPC framework to update gait parameters for RABBIT using nonlinear program with set inclusion constraints.

3.4.1 Forward Reachable Set

Letting $\mathcal{Y}_1 \subset \mathbb{R}$ be compact, we define the *N-step FRS of the output y_1*:

Definition 25. *The N-step FRS of the output beginning from $\big(y_1(i), P(i)\big) \in \mathcal{Y}_1 \times \mathcal{P}$ for $i \in \mathbb{N}$*

and for $N \in \mathbb{N}$ is defined as

$$\mathcal{W}_N\big(y_1(i), P(i)\big) := \bigcup_{n=i+1}^{i+N} \Big\{ y_1(n) \in \mathcal{Y}_1 \mid \exists P(i+1), \dots, \\ P(n-1) \in \mathcal{P} \textit{ such that } \forall j \in \{i, \dots, i+n-1\}, \\ y_1(j+1) \textit{ is generated by the RABBIT} \\ \textit{dynamics from } y_1(j) \textit{ under } P(j) \Big\} \tag{3.26}$$

In other words, given a fixed output $y_1(i)$ and the current control parameter $P(i)$, the FRS $\mathcal{W}_N$ captures all the outputs $y_1(j)$ that can be reached within N steps, provided that all subsequent control parameters are contained in a set $\mathcal{P}$. Since $\mathcal{W}_N$ is the union of all possible y_1 within the next N steps, it follows that:

Lemma 26.

$$\mathcal{W}_M\big(y_1(i), P(i)\big) \subseteq \mathcal{W}_N\big(y_1(i), P(i)\big) \quad \forall 1 \leq M \leq N \tag{3.27}$$

As a result of Lemma 26, to predict the behavior of RABBIT system over N steps, it is unnecessary to compute distinct FRS-es for each of the next N steps. Instead one only needs to compute a single FRS.

To compute an outer approximation of the FRS, inspired by [HK13], one can solve the following infinite-dimensional linear problem over the space of functions:

$$\inf_{w_N, v_1, \cdots, v_N} \int_{\mathcal{Y}_1 \times \mathcal{P} \times \mathcal{Y}_1} w_N(x_1, x_2, x_3)\, d\lambda_{\mathcal{Y}_1 \times \mathcal{P} \times \mathcal{Y}_1} \qquad \text{(FRSopt)}$$

$$\begin{aligned}
\text{s.t.} \quad & v_1(x_1, x_2, x_3) \geq 0, && \forall x_3 \in \mathcal{B}(x_1, x_2) \\
& && \forall (x_1, x_2) \in \mathcal{Y}_1 \times \mathcal{P} \\
& v_{\zeta+1}(x_1, x_2, x_4) \geq v_\zeta(x_1, x_2, x_3), && \forall \zeta \in \{1, 2, \cdots, N-1\} \\
& && \forall x_4 \in \mathcal{B}(x_3, x_5) \\
& && \forall (x_1, x_2, x_5) \in \mathcal{Y}_1 \times \mathcal{P} \times \mathcal{P} \\
& w_N(x_1, x_2, x_3) \geq 0, && \forall (x_1, x_2, x_3) \in \mathcal{Y}_1 \times \mathcal{P} \times \mathcal{Y}_1 \\
& w_N(x_1, x_2, x_3) \geq v_\zeta(x_1, x_2, x_3) + 1, && \forall \zeta = 1, 2, \cdots, N \\
& && \forall (x_1, x_2, x_3) \in \mathcal{Y}_1 \times \mathcal{P} \times \mathcal{Y}_1
\end{aligned}$$

where the sets $\mathcal{Y}_1$ and $\mathcal{P}$ are given, and the infimum is taken over an $(N+1)$-tuple of continuous functions $(w_N, v_1, \cdots, v_N) \in \left(C^1(\mathcal{Y}_1 \times \mathcal{P} \times \mathcal{Y}_1; \mathbb{R})\right)^{N+1}$. Note that only the SBM's dynamics appear in this program via $\mathcal{B}(\cdot, \cdot)$.

Next, we prove that the FRS is contained in the 1-superlevel set of all feasible w's in (FRSopt):

Lemma 27. *Let $(w_N, v_1, \cdots, v_N)$ be feasible functions to (FRSopt), then the following condition is true for all $\big(y_1(i), P(i)\big) \in \mathcal{Y}_1 \times \mathcal{P}$:*

$$\mathcal{W}_N\big(y_1(i), P(i)\big) \subseteq \big\{x_3 \in \mathcal{Y}_1 \mid w_N\big(y_1(i), P(i), x_3\big) \geq 1\big\}. \tag{3.28}$$

Proof. Let $(w_N, v_1, \cdots, v_N)$ be feasible functions to (FRSopt). Substitute an arbitrary $y_1(i) \in \mathcal{Y}_1$ and $P(i) \in \mathcal{P}$ into x_1 and x_2, respectively. Suppose $\mu \in \mathcal{W}_N\big(y(i), P(i)\big)$, then there exists a natural number $n \in [i+1, i+N]$ and a sequence of control parameters $P(i+1), \cdots, P(n-1) \in \mathcal{P}$, such that $y_1(j+1) \in \mathcal{B}\big(y_1(j), P(j)\big)$ for all $i \leq j \leq n-1$ and $\mu = y_1(n)$.

We prove the result by induction. Let $x_3 = y_1(i+1) \in \mathcal{B}\big(y_1(i), P(i)\big)$. It then follows from the first constraint of (FRSopt) that $v_1\big(y_1(i), P(i), y_1(i+1)\big) \geq 0$. Now, suppose $v_\zeta\big(y_1(i), P(i), y_1(i+\zeta)\big) \geq 0$ for some $1 < \zeta \leq n-i-1$. In the second constraint of (FRSopt), let $x_3 = y_1(i+\zeta)$, $x_4 = y_1(i+\zeta+1) \in \mathcal{B}\big(y_1(i+\zeta), P(i+\zeta)\big)$, and $x_5 = P(i+\zeta) \in \mathcal{P}'$, then $v_{\zeta+1}\big(y_1(i), P(i), y_1(i+\zeta+1)\big) \geq 0$. By induction, we know $v_N\big(y_1(i), P(i), y_1(n)\big) \geq 0$. Using the fourth constraint of (FRSopt), let $x_3 = \mu = y_1(n)$, and we get $w_N\big(y_1(i), P(i), \mu\big) \geq 1$. Therefore the result follows. □

Though we do not describe it here due to space restrictions, a feasible polynomial solution to (FRSopt) can be computed offline by making compact approximation of $\mathcal{Y}_1$ and applying Sums-of-Squares programming [ZMV17a, MLV17].

3.4.2 N-step Successful Walking and MPC

To ensure safe walking through N-steps beginning at step i, we require several set inclusions to be satisfied during online optimization based on Theorem 23. First, we require that $\mathcal{W}_N\big(y_1(i), P(i)\big) \subseteq [0, \infty)$, which sufficiently guarantee $y_1(i) \geq 0$ for each $i \leq N$. Since we cannot compute $\mathcal{W}_N\big(y_1(i), P(i)\big)$ exactly, from Lemma 27 we instead can require that the 1-superlevel set of w_N is a subset of $[0, \infty)$.

With the help of the FRS, N-step successful walking conditions on (y_2, y_3, y_4) can be guaranteed in a fashion similar to (3.23), (3.24), (3.25) if

$$f_{\hat{y}_2}\big(P(i+M)\big) + \overline{B}_2\big(P(i+M-1), y_1(i+M), P(i+M)\big) \leq \pi, \tag{3.29}$$

$$f_{\hat{y}_3}\big(y_1(i+M), P(i+M)\big) + \underline{B}_3\big(y_1(i+M), P(i+M)\big) > \pi/2, \tag{3.30}$$

$$f_{\hat{y}_4}\big(y_1(i+M), P(i+M)\big) + \overline{B}_4\big(y_1(i+M), P(i+M)\big) < 3\pi/2. \tag{3.31}$$

hold for all $y_1(i+M) \in \mathcal{W}_M(y_1(i), P(i))$, $1 \leq M \leq N$. Applying Lemma 26, one can instead enforce (3.29), (3.30), (3.31) for all $y_1(i+M) \in \mathcal{W}_N(y_1(i), P(i))$ to avoid computing

$\mathcal{W}_M(y_1(i), P(i))$ for each $1 \leq M < N$.

We then use a MPC framework to select gait parameter for RABBIT by solving the following nonlinear program:

$$
\begin{aligned}
\min_{\substack{P(i) \\ \vdots \\ P(i+N-1)}} \quad & r\left(y(i), P(i), P(i+1), \cdots, P(i+N-1)\right) && \text{(OL)} \\
\text{s.t.} \quad & P(i), P(i+1), \cdots, P(i+N-1) \in \mathcal{P} \\
& f_{\hat{y}2}\big(P(i)\big) + \overline{B}_2\big(P(i-1), y_1(i), P(i)\big) \leq \pi \\
& f_{\hat{y}3}\big(y_1(i), P(i)\big) + \underline{B}_3\big(y_1(i), P(i)\big) > \pi/2 \\
& f_{\hat{y}4}\big(y_1(i), P(i)\big) + \overline{B}_4\big(y_1(i), P(i)\big) < 3\pi/2 \\
& \mathcal{W}_N(y_1(i), P(i)) \subseteq [0, \infty) \\
& f_{\hat{y}_2}\big(P(i+M)\big) + \overline{B}_2\big(P(i+M-1), y_1(i+M), P(i+M)\big) \leq \pi, \\
& \qquad \text{if } y_1(i+M) \in \mathcal{W}_N(y_1(i), P(i)),\ 1 \leq M \leq N-1 \\
& f_{\hat{y}_3}\big(y_1(i+M), P(i+M)\big) + \underline{B}_3\big(y_1(i+M), P(i+M)\big) > \pi/2, \\
& \qquad \text{if } y_1(i+M) \in \mathcal{W}_N(y_1(i), P(i)),\ 1 \leq M \leq N-1 \\
& f_{\hat{y}_4}\big(y_1(i+M), P(i+M)\big) + \overline{B}_4\big(y_1(i+M), P(i+M)\big) < 3\pi/2, \\
& \qquad \text{if } y_1(i+M) \in \mathcal{W}_N(y_1(i), P(i)),\ 1 \leq M \leq N-1
\end{aligned}
$$

where $r \in C^1(\mathcal{Y} \times \mathcal{P}^N; \mathbb{R})$ is any user specified cost function. Note that the last four constraints in (OL) are set inclusion constraints. These can be difficult to implement these directly. Howeveer, one can conservatively represent these set inclusion constraints by the 0-super level set of a set of polynomials by using the generalized S-procedure described in Section 2.6.3 of [BEGFB94] and SOS optimization [ZMV17a, MLV17]. Note that this set of polynomial functions to conservatively represent these set inclusion constraints can be generated offline.

Notice that (OL) is solved at the i-th mid-stance and only the optimal $P(i)$ is applied to the RABBIT and the problem is then solved again for the $(i+1)$-st step. The constraints of (OL) lead to the following theorem:

Theorem 28. *Suppose that RABBIT is at the i-th mid-stance, then tracking the gait parameters associated with any feasible solution to (OL) ensures that RABBIT can walk successfully for the next N-steps.*

3.5 Results

To illustrate that the proposed method is able to guarantee safe walking performance online, we evaluate the performance of our method when it is tasked with tracking a randomly generated speed sequence. In each trial, the RABBIT model is required to track a randomly generated speed sequence that holds still for the first 4 steps and changes to a different value starting from the 5th step, i.e. a step function. The speed sequence is restricted to be in a range $[0.2, 2]$. We repeat this experiment on 300 randomly generated speed sequences. The space of control parameter is restricted to be $\mathcal{P} = [0.25, 2] \times [0.15, 0.7]$ on which the gait library is generated. The RABBIT model is initialized with the gait whose speed is closest to the initial value of the desired speed sequence in each trial. The control parameter can only be updated at the mid-stance of each step. Our MATLAB implementation of the experiments can be found at https://github.com/pczhao/TA_GaitDesign.git.

In the proposed method, the cost function of (FRSopt) is set to be the weighted Euclidean norm of the difference between the predicted speeds and the desired speeds within the next 3 steps. Note $N = 3$. We compute an outer approximation to the (FRSopt) using the commercial solver MOSEK on a machine with 144 64-bit 2.40GHz Intel Xeon CPUs and 1 Terabyte memory. To create the bounding functions that satisfy Assumption 24, we employ simulation. In particular, we initialized the RABBIT model randomly twenty thousand times and varied the control parameter randomly for 5 steps and observed the output. We repeated the process with the SBM model using the same control parameter sequence and calculated the difference of the output between the two models on each of the twenty thousand trials. Finally, we applied SOS optimization to bound the difference from above and below to generate the error bounding polynomial that satisfied the conditions in Assumption 3.

We compare our method with a naïve method and the direct method using the same speed tracking sequences. The naïve method uses the SBM model to update gaits in an MPC framework without enforcing walking successful conditions. The direct method uses the full-order dynamics of the RABBIT model to design a controller by solving an optimal control problem via FROST [HA17].

Fig. 3.3 illustrates the performance of the naïve method and the method proposed in this chapter on one of the 300 trials. Note in particular that the gait generated by the naïve method is unable to be followed by the full-order RABBIT model. On the other hand, as shown in Fig. 3.3, the method proposed in this chapter is able to generate a gait that can satisfy the safety requirements described in Theorem 23. This results in a controller which can track the synthesized gait without falling over.

Fig. 3.4 compares the speed tracking performance of another trial, where both methods gener-

ate gaits that can be followed by the RABBIT model. Notice naïve method achieves a lower speed tracking error of 0.3681 (in Euclidean norm), while the proposed method achieves a slightly higher tracking error of 0.3912. This is because the additional safety constraints in the MPC prevents rapid transition from a low-speed gait to a high-speed gait, therefore generating higher costs.

Across all 300 trials the computation time of the naïve method is 0.01 seconds, the direct method is 93.12 seconds, and the proposed method is 0.11 seconds. Moreover, the RABBIT model falls 2% of the time with the naïve method, but never falls with the proposed method or the direct method. Therefore the proposed method is able to guarantee walking performance online. The average speed tracking error (computed in Euclidean norm) of the naïve method is 0.8719, and the proposed method is 0.9808.

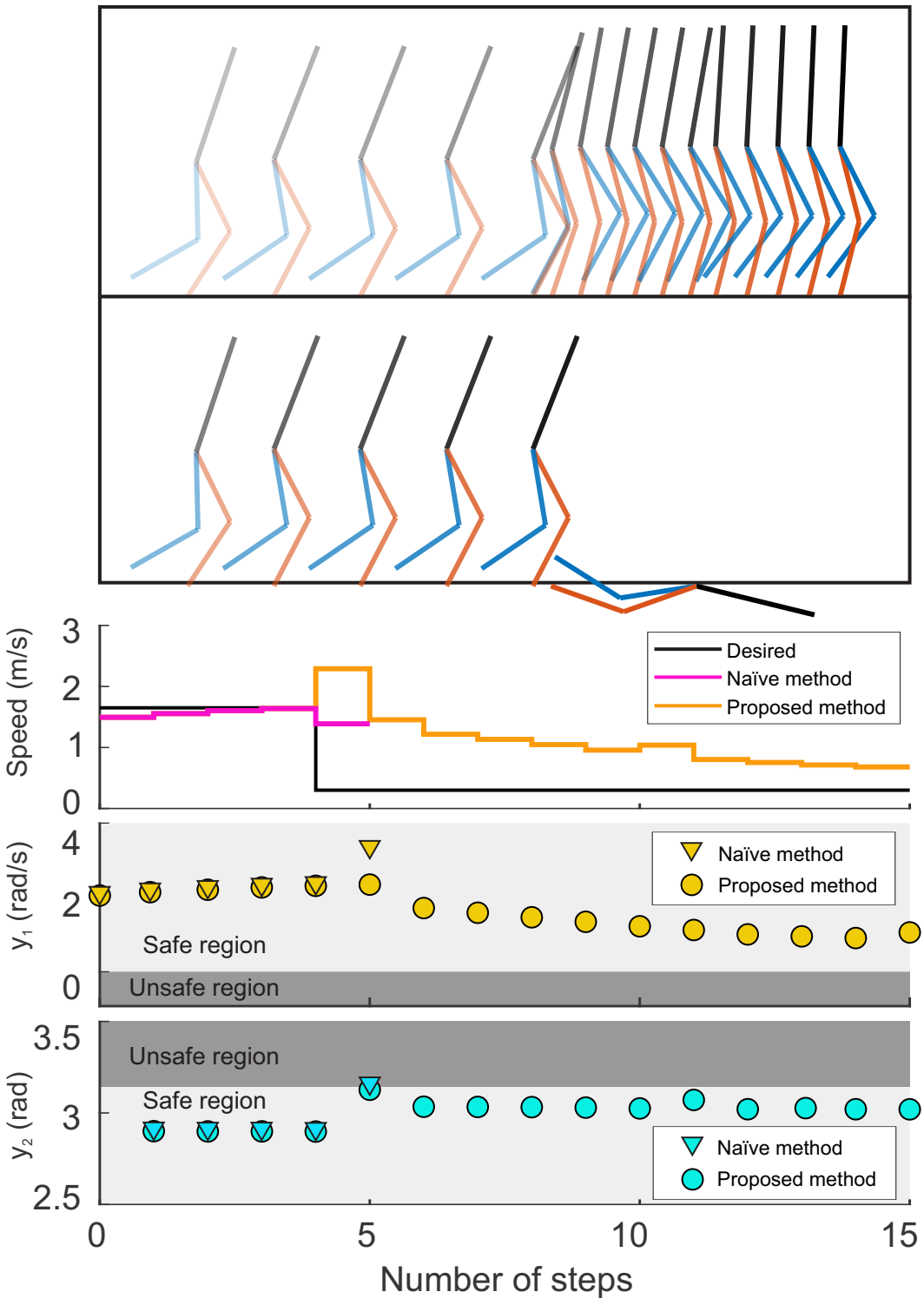


Figure 3.3: An illustration of the performance of the method proposed in this chapter (top) and the naïve method (second from top). Note that the rapid change in the desired speed (third from top) results in a gait which cannot be tracked by just considering a SBM model without successful walking constraints. By ensuring that the outputs satisfy the inequality constraints proposed in Theorem 23 (bottom two sub-figures), the proposed method is able to safely track the synthesized gaits. Note the naïve method violates the y_2 constraint proposed in Theorem 23 on Step 5.

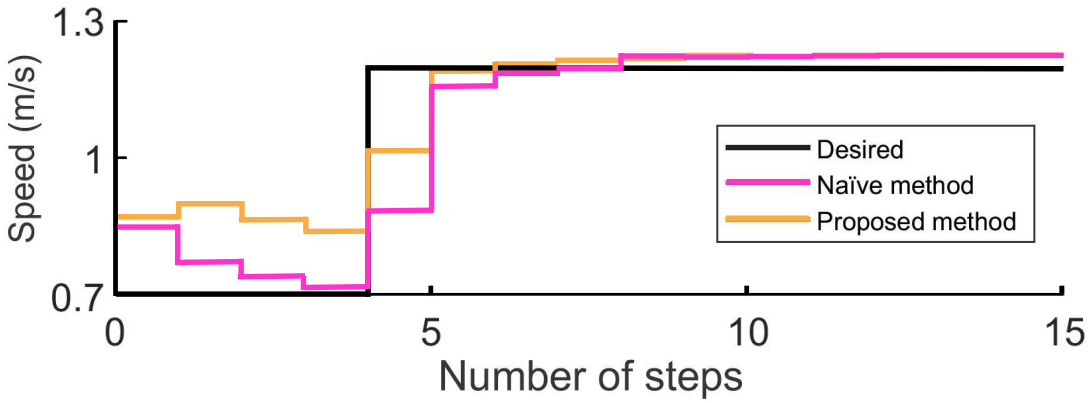


Figure 3.4: A comparison of speed tracking performance of the proposed method and the naïve method. Although both methods generate gaits that can be followed by the RABBIT model without falling over, the tracking error of naïve method is lower (0.3681) compared to the proposed method (0.3912).

3.6 Conclusion

This chapter develops a method to generate safety-preserving controllers for full-order (anchor) models by performing reachability analysis on simpler (template) models while bounding the modeling error. The method is illustrated on a 5-link, 14-dimenstional RABBIT model, and is shown to allow the robot to walk safely while utilizing controllers designed in a real-time fashion.

Though this method enables real-time motion planning, future work will consider several extensions that will enable real-world robotic control. First, a template and associated outputs need to be constructed for 3D motion. Second, no guarantee is provided that the optimization problem solved at each step in the MPC will return a feasible solution.

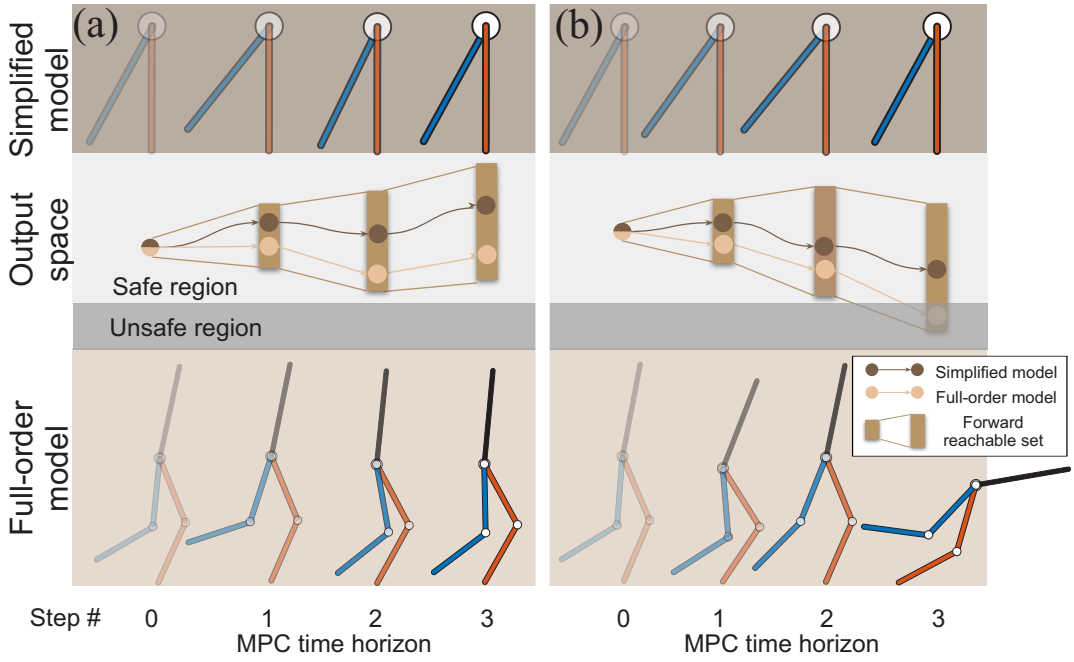


Figure 3.1: This chapter proposes a method to design gaits that are certified to be tracked by a full-order robot model (bottom row sub-figures) for N-steps without falling over. To construct this method, this chapter defines a set of outputs that are functions of the state of the robot and a chosen gait (middle row sub-figures). If the outputs associated with a particular gait satisfy a set of inequality constraints (depicted as the safe region drawn in light gray in the middle row sub-figures), then the gait is proven to be safely tracked by the legged system without falling. Due to the high-dimensionality of the robot's dynamics, forward propagating these outputs via the robot's dynamics for N-steps to design a gait that is certified to be tracked safely is intractable. To address this challenge, this chapter constructs a template model (top row sub-figures) whose outputs are sufficient to predict the behavior of the anchor's outputs. In particular, if all of the points in a bounded neighborhood of the forward reachable set of the outputs of the template model remain within the safe region, then the anchor is certified to behave safely. This chapter illustrates how this can be incorporated into a MPC framework to design safe gaits in real-time.

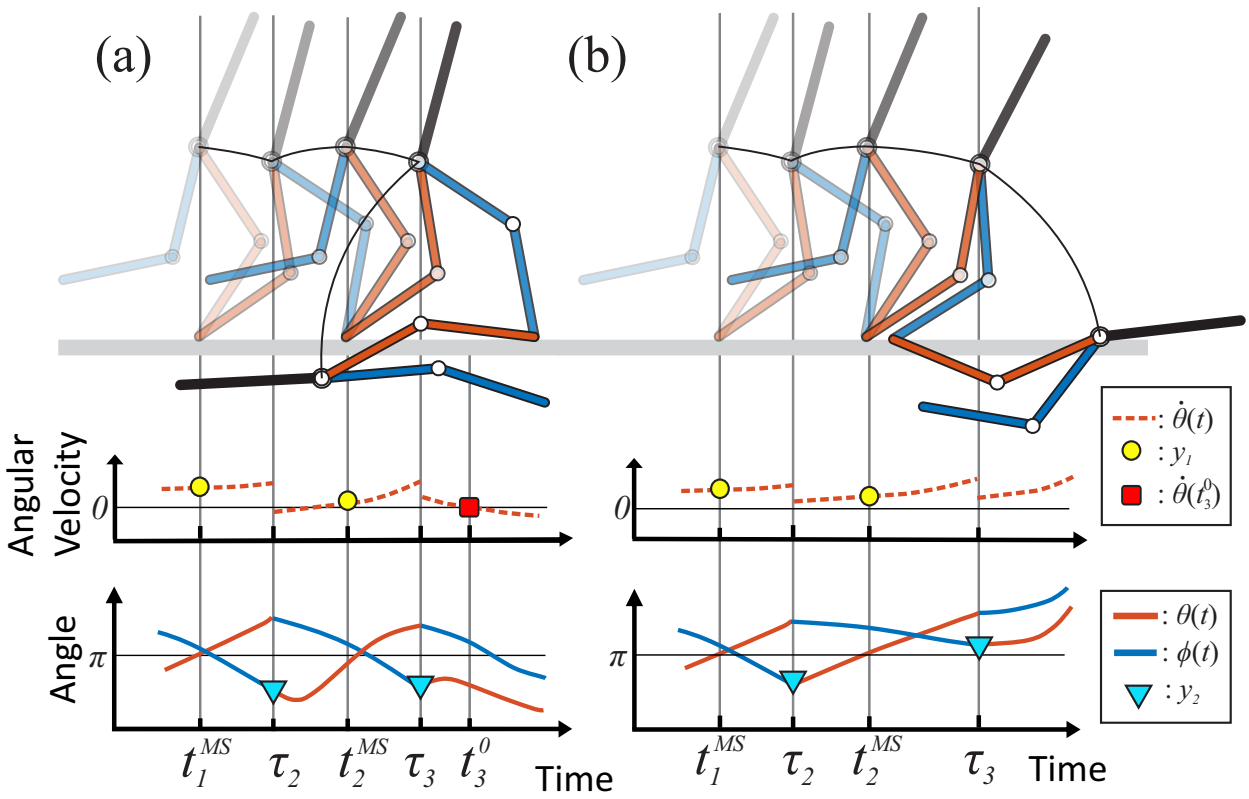


Figure 3.2: An illustration of how the values of the outputs can be used to determine whether the robot walks safely. To ensure that the robot does not fall backwards, one can require that $y_1(i) \geq 0$ (left column). In particular if $y_1(i) < 0$, then $t_i^{MS} = +\infty$ which implies that the robot is falling backwards. To ensure that the robot does not fall forward, one can require that $y_2(i) \leq \pi$ (right column).

CHAPTER 4

Efficiently Solving Polynomial Optimization Problems

4.1 Introduction

[1] For mobile robots to operate successfully in unforeseen environments, they must plan their motion as new sensor information becomes available. This *receding-horizon* strategy requires iteratively generating a plan while simultaneouly executing a previous plan. Typically, this requires solving an optimization program in each planning iteration (see, e.g., [Kuw07]).

The present work considers a mobile ground robot tasked with reaching a global goal location in an arbitrary, static environment. To assess receding-horizon planning performance, we consider the following characteristics of plans. First, a plan should be *dynamically feasible*, meaning that it obeys the dynamic description of the robot and obeys constraints such as actuator limits and obstacle avoidance. Second, a plan should maintain *liveness*, meaning that it keeps the robot moving through the world without stopping frequently to replan, which can prevent a robot from achieving a task in a user-specified amount of time. Third, a plan should be *optimal* with respect to a user-specified cost function, such as reaching a goal quickly.

Ensuring that plans have these characteristics is challenging for several reasons. First, robots typically have nonlinear dynamics; this means that creating a dynamically feasible plan often requires solving a nonlinear program (NLP) at runtime. However, it is difficult to certify that an NLP can be solved in a finite amount of time, meaning that the robot may have to sacrifice liveness. Furthermore, even if a robot has linear dynamics, the cost function may be nonlinear; in this case, it is challenging to certify optimality due to the presence of local minima.

[1] This chapter was previously submitted to IEEE Transactions on Robotics (T-RO)

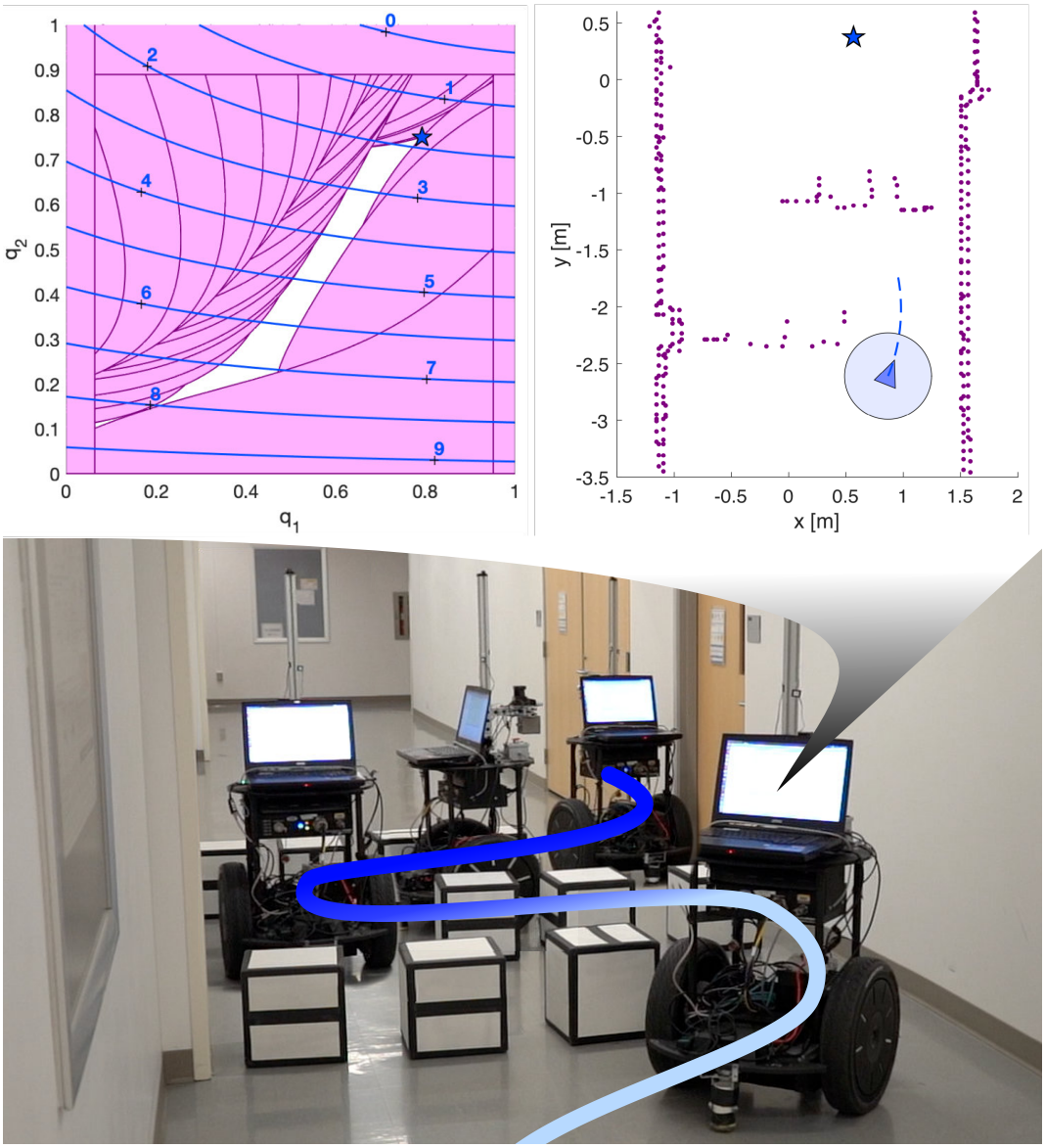


Figure 4.1: A Segway RMP mobile robot using the proposed PCBA/RTD* method to autonomously navigate a tight obstacle blockade. The executed trajectory is shown fading from light to dark blue as time passes, and the robot is shown at four different time instances. The top right plot shows the Segway's (blue circle with triangle indicating heading) view of the world at one planning iteration, with obstacles detected by a planar lidar (purple points). The top left plot shows the optimization program solved at the same planning iteration; the decision variable is (q_1, q_2), which parameterizes the velocity and yaw rate of a trajectory plan; the pink regions are infeasible with respect to constraints generated by the obstacle points in the right plot; and the blue contours with number labels depict the cost function, which is constructed to encourage the Segway to reach a waypoint (the star in the top right plot). The optimal solution found by PCBA is shown as a star on the left plot, in the non-convex feasible area (white). This optimal solution generates a provably-safe trajectory for the Segway to track, shown as a blue dashed line in the right plot.

This chapter extends prior work on Reachability-based Trajectory Design (RTD). RTD is able to provably generate *dynamically-feasible* trajectory plans in real time, but cannot guarantee optimality or liveness (the robot will generate plans in real time, but may brake to a stop often). We address this gap by proposing the **Parallel Constrained Bernstein Algorithm (PCBA)**, which provably finds globally-optimal solutions to Polynomial Optimization Problems (POPs), a special type of NLP. We apply PCBA to RTD to produce an optimal version of RTD, which we call **RTD*** in the spirit of the well-known RRT* algorithm [KF11]. We show on hardware that RTD* demonstrates *liveness* (in comparison to RTD) for trajectory optimization. For the remainder of this section, we discuss related work, then state our contributions.

4.1.1 Related Work

4.1.1.1 Receding-horizon Planning

A variety of methods exist that attempt receding-horizon planning while maintaining dynamic feasibility, liveness, and optimality. These methods can be broadly classified by whether they perform sampling or solve an optimization program at each planning iteration. Sampling-based methods typically either attempt to satisfy liveness and dynamic feasibility by choosing samples offine [PKK09, MT17b], or attempt to satisfy optimality at the potential expense of liveness and dynamic feasibility [KF11]. Optimization-based methods attempt to find a single optimal trajectory. These methods typically have to sacrifice dynamic feasibility (e.g., by linearizing the dynamics) to ensure liveness [MR93], or sacrifice liveness to attempt to satisfy dynamic feasibility [YGF+16, FFT19] (also see [KVB+18, Section 9] and [PR14]).

4.1.1.2 Reachability-based Trajectory Design

Reachability-based Trajectory Design (RTD) is an optimization-based receding horizon planner that requires solving a POP at each planning iteration [KVB+18]. RTD specifies plans as parameterized trajectories. Since these trajectories cannot necessarily be perfectly tracked by the robot, RTD begins with an offline computation of a Forward Reachable Set (FRS). The FRS contains every parameterized plan, plus the tracking error that results from the robot not tracking any plan perfectly. At runtime, in each planning iteration, the FRS is intersected with sensed obstacles to identify all parameterized plans that could cause a collision (i.e., be dynamically infeasible). This set of unsafe plans is represented as a (finite) list of polynomial constraints, and the user is allowed to specify an arbitrary (not necessarily convex) polynomial cost function, resulting in a POP. At each planning iteration, either the robot successfully solves the POP to get a new plan, or it continues executing its previously-found plan. While the decision variable is typically only two- or three-dimensional, each POP often has hundreds of constraints, making it challenging to find a

feasible solution in real-time [KVB+18]. Each plan includes a braking maneuver, ensuring that the robot can always come safely to a stop if the POP cannot be solved quickly enough in any planning iteration.

Note, for RTD, *optimality* means finding the optimal solution to a POP at each planning iteration. The cost function in RTD's POPs typically encode behavior such as reaching a waypoint between the robot's current position and the global goal (e.g., [KVB+18, Section 9.2.1]; RTD attempts to find the best dynamically feasible trajectory to the waypoint. RTD does not attempt to find the *best waypoints* themselves (best, e.g., with respect to finding the shortest path to the global goal). Such waypoints can be generated quickly by algorithms such as A* or RRT* by ignoring dynamic feasibility [KF11, KVB+18].

4.1.1.3 Polynomial Optimization Problems

POPs require minimizing (or maximizing) a polynomial objective function, subject to polynomial equality or inequality constraints. As a fundamental class of problems in non-convex optimization, POPs arise in various applications, including signal processing [QT03, TCL93, MLD03], quantum mechanics [DLMO07, Gur03], control theory [KP14, KVB+18, BSS15], and robotics [RCBL19, MLEV19]. This chapter presents a novel parallelized constrained Bernstein Algorithm for solving POPs.

The difficulty of solving a POP increases with the dimension of the cost and constraints, with the number of constraints, and with the number of optima [NW06]. Existing methods attempt to solve POPs while minimizing time and memory usage (i.e., complexity). Doing so typically requires placing limitations on one of these axes of difficulty to make solving a POP tractable. These methods broadly fall into the following categories: derivative-based, convex relaxation, and branch-and-bound.

Derivative-based methods use derivatives (and sometimes Hessians) of the cost and constraint functions, along with first- or second-order optimality conditions [NW06, Sections 12.3, 12.5], to attempt to find optimal, feasible solutions to nonlinear problems such as POPs. These methods can find local minima of POPs rapidly despite high dimension, a large number of constraints, and high degree cost and constraints [NW06, Chapter 19.8]. However, these methods do not typically converge to global optima without requiring assumptions on the problem and constraint structure (e.g., [QWY04]).

Convex relaxation methods attempt to find global optima by approximating the original problem with a hierarchy of convex optimization problems. These methods can be scaled to high-dimensional problems (up to 10 dimensions), at the expense of limits on the degree and sparse structure of the cost function; furthermore, they typically struggle to handle large numbers of constraints (e.g., the hundreds of constraints that arise in RTD's POPs), unless the problem has

low-rank or sparse structure [RCBL19]. Well-known examples include the lift-and-project linear program procedure [BCC93], reformulation-linearization technique [SA90], and Semi-Definite Program (SDP) relaxations [Las01, RCBL19]. By assuming structure such as homogeneity of the cost function or convexity of the domain and constraints, one can approximate solutions to a POP in polynomial-time, with convergence to global optima in the limit [DKLP06, LNQY09, LZ10, So11, HLZ10]. Convergence within a finite number of convex hierarchy relaxations is possible under certain assumptions (e.g., a limited number of equality constraints [Nie13, LTY17]).

Branch-and-bound methods perform an exhaustive search over the feasible region. These methods are typically limited to up to four dimensions, but can handle large numbers of constraints and high degree cost and constraints. Examples include interval analysis techniques [HW03, VEH94] and the Bernstein Algorithm (BA) [Gar93, NA11, SS15]. Interval analysis requires cost and constraint function evaluations in each iteration, and therefore can be computationally slow. BA, on the other hand, does not evaluate the cost and constraint functions; instead, BA represents the coefficients of the cost and constraints in the Bernstein basis, as opposed to the monomial basis. The coefficients in the Bernstein basis provide lower and upper bounds on the polynomial cost and constraints over box-shaped subsets of Euclidean space by using a subdivision procedure [Gar85, NA07]. Note, one can use the Bernstein basis to transform a POP into a linear program (LP) on each subdivided portion of the problem domain, which allows one to find tighter solution bounds that given by the Bernstein coefficients alone [SS15]. Since subdivision can be parallelized [DN17], the time required to solve a POP can be greatly reduced by implementing BA on a Graphics Processing Unit (GPU). However, a parallelized implementation or bounds on the rate of convergence of BA with constraints has not yet been shown in the literature. Furthermore, to the best of our knowledge, BA has not been shown as a practical method for solving problems in real-time robotics applications.

4.1.2 Contributions and chapter Organization

In this chapter, we make the following contributions. First, we propose a **Parallel Constrained Bernstein Algorithm (PCBA)** (Section 4.3). Second, we prove that PCBA always finds an *optimal solution* (if one exists), and prove bounds on PCBA's time and memory usage (Section 4.4). Third, we evaluate PCBA on a suite of well-known POPs in comparison to the Bounded Sums-of-Squares (BSOS) [LTY17] solver and a generic nonlinear solver (MATLAB's `fmincon`) (Section 4.5). Fourth, we apply PCBA to RTD to make **RTD***, a provably-safe, optimal, and real-time trajectory planning algorithm for mobile robots (Section 4.6), thereby demonstrating *dynamic feasibility* and *liveness*. The remainder of the chapter is organized as follows. Section 4.2 defines notation for RTD and POPs. Section 4.7 draws conclusions. Appendix E lists benchmark problems for PCBA.

4.2 Preliminaries

This section introduces notation, RTD, POPs, the Bernstein form, and subdivision.

4.2.1 Notation

4.2.1.1 Polynomial Notation

We follow the notation in [NA11]. Let $x := (x_1, x_2, \cdots, x_l) \in \mathbb{R}^l$ be real variable of dimension $l \in \mathbb{N}$. A multi-index J is defined as $J := (j_1, j_2, \cdots, j_l) \in \mathbb{N}^l$ and the corresponding multi-power x^J is defined as $x^J := (x_1^{j_1}, x_2^{j_2}, \cdots, x_l^{j_l}) \in \mathbb{R}^l$. Given another multi-index $N := (n_1, n_2, \cdots, n_l) \in \mathbb{N}^l$ of the same dimension, an inequality $J \leq N$ should be understood component-wise. An l-variate polynomial p in canonical (monomial) form can be written as

$$p(x) = \sum_{J \leq N} a_J x^J, \quad x \in \mathbb{R}^l, \tag{4.1}$$

with coefficients $a_J \in \mathbb{R}$ and some multi-index $N \in \mathbb{N}^l$. The space of polynomials of degree $d \in \mathbb{N}$, with variable $x \in \mathbb{R}^l$, is $\mathbb{R}_d[x]$.

Definition 29. *We call $N \in \mathbb{N}^l$ the* multi-degree *of a polynomial p; each i^{th} element of N is the maximum degree of the variable x_i out of all of the monomials of p. We call $d \in \mathbb{N}$ the* degree *of p; d is the maximum sum, over all monomials of p, of the powers of the variable x. That is, $d = ||N||_1$, where $|| \cdot ||_1$ is the sum of the elements of a multi-index.*

4.2.1.2 Point and Set Notation

Let $\mathbf{x} := [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_l, \overline{x}_l] \subset \mathbb{R}^l$ denote a general l-dimensional box in $\mathbb{R}^n$, with $-\infty < \underline{x}_\mu < \overline{x}_\mu < +\infty$ for each $\mu = 1, \cdots, l$. Let $\mathbf{u} := [0, 1]^l \subset \mathbb{R}^l$ be the l-dimensional unit box. Denote by $|\mathbf{x}|$ the maximum width of a box $\mathbf{x}$, i.e. $|\mathbf{x}| = \max\{\overline{x}_\mu - \underline{x}_\mu : \mu = 1, \cdots, l\}$. For any point $y \in \mathbb{R}^l$, denote by $\|y\|$ the Euclidean norm of y, and denote by $\mathcal{B}_R(y)$ the closed Euclidean ball centered at y with radius $R > 0$.

4.2.1.3 RTD Notation

Let $T = [0, t_\mathrm{f}]$ denote the time interval of a single plan (in a single receding-horizon iteration). Let $X \subset \mathbb{R}^{n_X}$ denote the robot's state space, and $U \subset \mathbb{R}^{n_U}$ denote the robot's control inputs where $n_X, n_U \in \mathbb{N}$. The robot is described by dynamics $f_\mathrm{hi} : T \times X \times U \to \mathbb{R}^{n_X}$, which we call a *high-fidelity model*. The parameterized trajectories are described by $f : T \times X \times Q \to \mathbb{R}^{n_X}$, where $Q \subset \mathbb{R}^{n_Q}$ is the space of trajectory parameters. A point $q \in Q$ parameterizes a trajectory

$x_{\text{des}} : T \to X$. For any q, the robot uses a feedback controller $u_q : T \times X \to U$ to track the trajectory parameterized by q (we say it tracks q for short).

4.2.2 Polynomial Optimization Problems

We denote a POP as follows:

$$\begin{aligned} \min_{x \in D \subset \mathbb{R}^l} \quad & p(x) \\ \text{s.t} \quad & g_i(x) \leq 0 \quad i = 1, \cdots, \alpha \\ & h_j(x) = 0 \quad j = 1, \cdots, \beta. \end{aligned} \tag{P}$$

The decision variable is $x \in D \subset \mathbb{R}^n$, where D is a compact, box-shaped domain and $l \in \mathbb{N}$ is the *dimension* of the program. The cost function is $p \in \mathbb{R}_d[x]$, and the constraints are $g_i, h_j \in \mathbb{R}_d[x]$ ($\alpha, \beta \in \mathbb{N}$). We assume for convenience that d is the greatest degree amongst the cost and constraint polynomials; we call d the *degree of the problem*.

4.2.3 Reachability-based Trajectory Design

Recall that RTD begins by computing an FRS offline for a robot tracking a parameterized set of trajectories. At runtime, in each receding-horizon planning iteration, the FRS is used to construct a POP as in (P); solving this POP is equivalent to generating a new trajectory plan. We now briefly describe how this POP is constructed (see [KVB+18] for details).

We define the FRS $F \subset X \times Q$ as the set of all states reachable by the robot when tracking any parameterized trajectory:

$$\begin{aligned} F = \Big\{ (x, q) \in X \times Q \mid \exists\, t \in T \text{ s.t. } x = \hat{x}(t),\ \hat{x}(0) \in X_0 \text{ and} \\ \dot{\hat{x}}(\tau) = f_{\text{hi}}(\tau, \hat{x}(\tau), u_q(\tau, \hat{x}(\tau)))\, \forall\, \tau \in T \Big\} \end{aligned} \tag{4.2}$$

where $X_0 \subset X$ is the set of valid initial conditions for the robot in any planning iteration. To implement RTD, we conservatively approximate the FRS using sums-of-squares programming. In particular, we compute a polynomial $w \in \mathbb{R}_d[q]$ ($d = 10$ or 12) for which the 0-superlevel set contains the FRS:

$$(x, q) \in F \implies w(x, q) \geq 0. \tag{4.3}$$

See [KVB+18, Section 3.2] for details on how to compute w.

At runtime, we use w to identify unsafe trajectory plans. If $\{x_i\}_{i=1}^n \subset X$ is a collection of

points on obstacles, then we solve the following program in each planning iteration:

$$\begin{aligned} \underset{q \in Q}{\text{argmin}} \quad & p(q) \\ \text{s.t} \quad & w(x_i, q) < 0 \; \forall \; i = 1, \cdots, n, \end{aligned} \tag{4.4}$$

where p is a user-constructed polynomial cost function (see (4.29) in Section 4.6 as an example). Note that, by (4.3), the set of safe trajectory plans is open, so we implement the constraints in (4.4) as $w(x_i, q) + \epsilon_q \leq 0$, $\epsilon_q \approx 10^{-4}$. Critically, any feasible solution to (4.4) is provably dynamically feasible (and collision-free) [KVB+18, Theorem 68]. To understand RTD's online planning in more detail, see [KVB+18, Algorithm 2].

In this work, instead of using a derivative-based method to solve (4.4), we use our proposed PCBA method, which takes advantage of the polynomial structure of p and w. Next, we discuss Bernstein polynomials.

4.2.4 Bernstein Form

A polynomial p in monomial form (4.1) can be expanded into *Bernstein form* over an arbitrary l-dimensional box $\mathbf{x}$ as

$$p(x) = \sum_{J \leq N} B_J^{(N)}(\mathbf{x}) \, b_J^{(N)}(\mathbf{x}, x), \tag{4.5}$$

where $b_J^{(N)}(\mathbf{x}, \cdot)$ is the J^{th} multivariate *Bernstein polynomial* of multi-degree N over $\mathbf{x}$, and $B_J^{(N)}(\mathbf{x})$ are the corresponding *Bernstein coefficients* of p over $\mathbf{x}$. A detailed definition of Bernstein form is available in [Ham18]. Note that the Bernstein form of a polynomial can be determined quickly [TG17], by using a matrix multiplication on a polynomial's monomial coefficients, with the matrix determined by the polynomial degree and dimension. This matrix can be precomputed, and the conversion from monomial to Bernstein basis only needs to happen once for the proposed method (see Algorithm 1 in Section 4.3).

For convenience, we collect all such Bernstein coefficients in a multi-dimensional array $B(\mathbf{x}) := \left(B_J^{(N)}(\mathbf{x})\right)_{J \leq N}$, which is called a *patch*. We denote by $\min B(\mathbf{x})$ (resp. $\max B(\mathbf{x})$) the minimum (resp. maximum) element in the patch $B(\mathbf{x})$. The range of polynomial p over $\mathbf{x}$ is contained within the interval spanned by the extrema of $B(\mathbf{x})$, formally stated as the following theorem:

Theorem 30 ([NA11, Lemma 2.2])**.** *Let p be a polynomial defined as in* (4.5) *over a box $\mathbf{x}$. Then, the following property holds for a patch $B(\mathbf{x})$ of Bernstein coefficients*

$$\min B(\mathbf{x}) \leq p(x) \leq \max B(\mathbf{x}), \quad \forall x \in \mathbf{x}. \tag{4.6}$$

This theorem provides a means to obtain enclosure bounds of a multivariate polynomial over a box

by transforming the polynomial to Bernstein form. This range enclosure can be further improved either by degree elevation or by subdivision. This work uses subdivision, discussed next, to refine the bounds.

4.2.5 Subdivision Procedure

Consider an arbitrary box $\mathbf{x} \subset \mathbb{R}^l$. The range enclosure in Theorem 30 is improved by subdividing $\mathbf{x}$ into subboxes and computing the Bernstein patches over these subboxes. A *subdivision* in the r^{th} direction ($1 \leq r \leq l$) is a bisection of $\mathbf{x}$ perpendicular to this direction. That is, let

$$\mathbf{x} := [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_r, \overline{x}_r] \times \cdots \times [\underline{x}_l, \overline{x}_l] \tag{4.7}$$

be an arbitrary box over which the Bernstein patch $B(\mathbf{x})$ is already computed. By subdividing $\mathbf{x}$ in the r^{th} direction we obtain two subboxes $\mathbf{x}_L$ and $\mathbf{x}_R$, defined as

$$\begin{aligned} \mathbf{x}_L &= [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_r, (\underline{x}_r + \overline{x}_r)/2] \times \cdots \times [\underline{x}_l, \overline{x}_l], \\ \mathbf{x}_R &= [\underline{x}_1, \overline{x}_1] \times \cdots \times [(\underline{x}_r + \overline{x}_r)/2, \overline{x}_r] \times \cdots \times [\underline{x}_l, \overline{x}_l]. \end{aligned} \tag{4.8}$$

Note that we have subdivided $\mathbf{x}$ by halving its width in the r^{th} direction; we choose $1/2$ as the subdivision parameter in this work, but one can choose a different value in $[0, 1]$ (see [NA11, Equation (10)]).

The new Bernstein patches, $B(\mathbf{x}_L)$ and $B(\mathbf{x}_R)$, can be computed by a finite number of linear transformations [NA11, Section 2.2]:

$$\begin{aligned} B(\mathbf{x}_L) &= M_{r,L} B(\mathbf{x}), \\ B(\mathbf{x}_R) &= M_{r,R} B(\mathbf{x}). \end{aligned} \tag{4.9}$$

where $M_{r,L}$ and $M_{r,R}$ are constant matrices, which can be precomputed, for each r (notice that [NA11, Equation (10)] obtains $B(\mathbf{x}_L)$ with linear operations on $B(\mathbf{x})$). The patches and one iteration of the subdivision procedure are shown in Figure 4.2.

Remark 31. *To reduce wordiness, we say that we* subdivide a patch *to mean the subdivision of a single subbox into two subboxes* and *the computation of the corresponding Bernstein patches for the POP cost and constraints.*

By repeatedly applying the subdivision procedure and Theorem 30, the bounds on the range of polynomial in a subbox can be improved. In fact, such bounds can be exact in the limiting sense if the subdivision is applied evenly in all directions:

Theorem 32 ([MMTG92, Theorem 2]). *Let $\mathbf{x}^{(n)}$ be a box of maximum width 2^{-n} ($n \in \mathbb{N}$) and let $B(\mathbf{x}^{(n)})$ be the corresponding Bernstein patch of a given polynomial p, then*

$$\min B(\mathbf{x}^{(n)}) \leq \min_{x \in \mathbf{x}^{(n)}} p(x) \leq \min B(\mathbf{x}^{(n)}) + \zeta \cdot 2^{-2n} \tag{4.10}$$

where ζ is a non-negative constant that can be given explicitly independent of n.

Notice from the proof of Theorem 32 that changing the sign of p does not change the value of ζ. By substituting p with $-p$ in Theorem 32, one can easily show a similar result holds for the maximum of p over $\mathbf{x}^{(n)}$:

Corollary 33. *Let $\mathbf{x}^{(n)}$ and $B(\mathbf{x}^{(n)})$ be as in Theorem 32. Then,*

$$\max B(\mathbf{x}^{(n)}) - \zeta \cdot 2^{-2n} \leq \max_{x \in \mathbf{x}^{(n)}} p(x) \leq \max B(\mathbf{x}^{(n)}), \tag{4.11}$$

where ζ is the same non-negative constant as in Theorem 32.

Theorems 32 and Corollary 33 provide shrinking bounds for values of a polynomial over subboxes as the subdivision process continues. By comparing the bounds over all subboxes, one can argue that the minimizers of a polynomial may appear in only a subset of the subboxes. This idea underlies the *Bernstein Algorithm* for solving POPs [NA07, NA11], discussed next.

4.3 Parallel Constrained Bernstein Algorithm

This section proposes the Parallel Constrained Bernstein Algorithm (PCBA, Algorithm 1) for solving a general POP. We extend the approach in [NA11]. This approach utilizes Bernstein form to obtain upper and lower bounds of both objective and constraint polynomials (Theorem 30), iteratively improves such bounds using subdivision (Theorem 32 and Corollary 33), and removes patches that are cannot contain a solution (Theorem 40). We discuss the algorithm, the list used to store patches, tolerances and stopping criteria, subdivision, a cut-off test for eliminating patches, and the advantages and disadvantages of PCBA. The next section, 4.4, proves PCBA finds globally optimal solutions to POPs, up to user-specified tolerances.

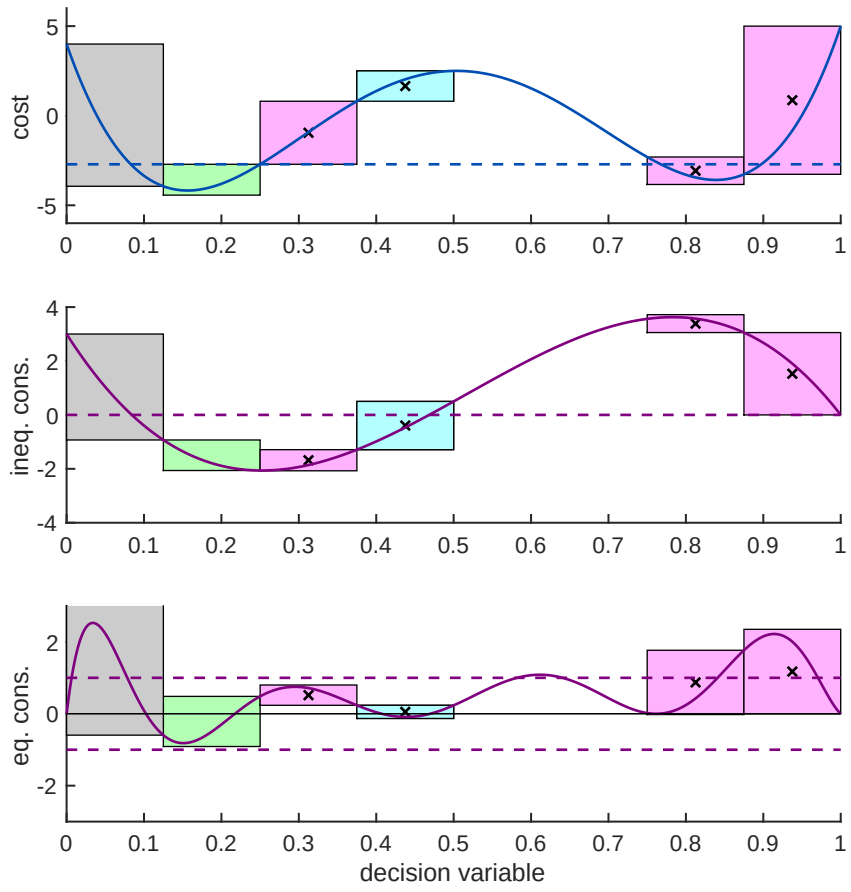


Figure 4.2: The 3rd iteration of PCBA (Algorithm 1) on a one-dimensional polynomial cost (top) with one inequality constraint (middle) and one equality constraint (bottom). The rectangles represent Bernstein patches (as in Section 4.2.5), where the horizontal extent of each patch corresponds to an interval of the decision variable over which Bernstein coefficients are computed. The top and bottom of each patch represent the maximum and minimum Bernstein coefficients, which bound the cost and constraint polynomials on the corresponding interval. As per Definition 37, the green patch is feasible, the pink patches are infeasible, and the grey patches are undecided; the purple dashed lines show the inequality constraint cut-off (zero) and the equality constraint tolerance $\epsilon_{\text{eq}} = 1$ (note that ϵ_{eq} is chosen to be this large only for illustration purposes). Per Definition 39, the light blue patch is suboptimal; the blue dashed line in the top plot is the current solution estimate (Definition 38). The infeasible and suboptimal patches are each marked with $\times$ for elimination (Algorithm 5), since they cannot contain the global optimum (Theorem 40); the feasible and undecided patches are kept for the next iteration.

Algorithm 1 Parallel Constrained Bernstein Algorithm (PCBA)

Inputs: Polynomials p, $\{g_i\}_{i=1}^{\alpha}$, $\{h_j\}_{j=1}^{\beta}$ as in (P), of dimension l; optimality tolerance $\epsilon > 0$, step tolerance $\delta > 0$, and equality constraint tolerance $\epsilon_{\text{eq}} > 0$; and maximum number of patches $M \in \mathbb{N}$ and of iterations $N \in \mathbb{N}$.

Outputs: Estimate $p^* \in \mathbb{R}$ of optimal solution, and subbox $\mathbf{x}^* \subset \mathbf{u}$ containing optimal solution.

Algorithm:

1: Initialize patches of p, g_i, and h_j over l-dimensional initial domain box $\mathbf{u}$ as in [TG17]
$[B_p(\mathbf{u}), B_{gi}(\mathbf{u}), B_{hj}(\mathbf{u})] \leftarrow \text{InitPatches}(p, g_i, h_j)$

2: Initialize lists of undecided patches and patch extrema on the GPU
$\mathcal{L} \leftarrow \{(\mathbf{u}, B_p(\mathbf{u}), B_{gi}(\mathbf{u}), B_{hj}(\mathbf{u}))\}$
$\mathcal{L}_{\text{bounds}} \leftarrow \{\}$

3: Initialize iteration count and subdivision direction
$n \leftarrow 1, r \leftarrow 1$

4: Test for sufficient memory (iteration begins here)
if $2 \times \text{length}(\mathcal{L}) > M$ **then** go to 12
else continue
end if

5: (Parallel) Subdivide each patch in $\mathcal{L}$ in the r^{th} direction to create two new patches using Algorithm 2
$\mathcal{L} \leftarrow \text{Subdivide}(\mathcal{L}, r)$

6: (Parallel) Find bounds of p, g_i, and h_j on each new patch using Algorithm 3
$\mathcal{L}_{\text{bounds}} \leftarrow \text{FindBounds}(\mathcal{L})$

7: Estimate upper bound p^*_{up} of the global optimum as the least upper bound of all feasible patches, and determine which patches to eliminate using Algorithm 4
$[p^*_{\text{up}}, p^*_{\text{lo}}, \mathcal{L}_{\text{save}}, \mathcal{L}_{\text{elim}}] \leftarrow \text{CutOffTest}(\mathcal{L}_{\text{bounds}})$;

8: Test if problem is feasible
if $\text{length}(\mathcal{L}_{\text{save}}) = 0$ **then** go to 14

9: Test stopping criteria for all $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$
if $p^*_{\text{up}} - p^*_{\text{lo}} \leq \epsilon$
and $|\mathbf{x}| \leq \delta$
and $-\epsilon_{eq} \leq \min B_{hj}(\mathbf{x}) \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{eq}$
then go to 12
end if

10: (Parallel) Eliminate infeasible, suboptimal patches using Algorithm 5
$\mathcal{L} \leftarrow \text{Eliminate}(\mathcal{L}, \mathcal{L}_{\text{save}}, \mathcal{L}_{\text{elim}})$;

11: Prepare for next iteration
$r \leftarrow (\text{mod}(r + 1, l)) + 1$
if $r = 1$ **then** $n \leftarrow n + 1$
end if
if $n = N$ **then** go to Step 12
else go to Step 4
end if

12: Return current best approximate solution
$p^* \leftarrow p^*_{\text{up}}$

Before proceeding, we make an assumption for notational convenience, and to make the initial computation of Bernstein patches easier.

Assumption 34. *Without loss of generality, the domain of the decision variable is the unit box (i.e., $D = \mathbf{u}$), since any nonempty box in $\mathbb{R}^l$ can be mapped affinely onto $\mathbf{u}$ [TG17].*

4.3.1 Algorithm Summary

We now summarize PCBA, implemented in Algorithm 1. The algorithm is initialized by computing the Bernstein patches of the cost and constraints on the domain $\mathbf{u}$ (Line 1). Subsequently, PCBA subdivides each patch as in Remark 36 (Line 5 and Algorithm 2). Then, PCBA finds the upper and lower bounds of each new patch (Line 6 and Algorithm 3). These bounds are used to determine which patches are feasible, infeasible, and undecided as in Definition 37 (Line 7; see Algorithm 4 and Theorem 40). Algorithm 4 also determines the current solution estimate (the smallest upper bound over all feasible patches), and marks any patches that are suboptimal as in Definition 39. If every patch is infeasible (Line 8), PCBA returns that the problem is infeasible (Line 14); otherwise, PCBA checks if the current solution estimate meets user-specified tolerances (Line 9). If the tolerances are met, PCBA returns the solution estimate (Line 12). Otherwise, PCBA eliminates all infeasible and suboptimal patches (Line 10 and Algorithm 5), then moves to the next iteration (Line 11). Note, algorithms 2, 3, and 5 are parallelized.

4.3.2 Items and The List

Denote an *item* as the tuple $\ell = (\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x})), B_{hj}(\mathbf{x}))$, where $B_{gi}(\mathbf{x})$ (resp. $B_{hj}(\mathbf{x})$) is shorthand for the set of patches $\{B_{g_i}(\mathbf{x})\}_{i=1}^{\alpha}$ (resp. $\{B_{h_j}(\mathbf{x})\}_{j=1}^{\beta}$). We use the following notation for items. If $\ell = (\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$, then $\ell_1 = \mathbf{x}$, $\ell_2 = B_p(\mathbf{x})$, $\ell_3 = B_{gi}(\mathbf{x})$, and $\ell_4 = B_{hj}(\mathbf{x})$.

We denote the *list* $\mathcal{L} = \{\ell_\mu : \mu = 1, \cdots, N_\mathcal{L}\}$, $N_\mathcal{L} \in \mathbb{N}$, indexed by $\mu \in \mathbb{N}$. PCBA adds and removes items from $\mathcal{L}$ by assessing the feasibility and optimality of each item.

4.3.3 Tolerances and Stopping Criteria

Recall that, by Theorem 30, Bernstein patches provide upper and lower bounds for polynomials over a box. From Theorem 32 and Corollary 33, as we subdivide $\mathbf{u}$ into smaller subboxes, the bounds of the Bernstein patches on each subbox more closely approximate the actual bounds of the polynomial. While the bounds will converge to the value of the polynomial in the limit as the maximum width of subboxes goes to zero, to ensure the algorithm terminates, we must set tolerances on optimality and equality constraint satisfaction (the equality constraints $h_j(x) = 0$

may not be satisfied for *all* points in certain boxes). During optimization one is also usually interested in finding the optimizer of the cost function up to some resolution. In our case, this resolution corresponds to the maximum allowable subbox width which we refer to as the *step tolerance*.

Definition 35. *We denote the* optimality tolerance *as* $\epsilon > 0$*, the* equality constraint tolerance *as* $\epsilon_{\text{eq}} > 0$*, and the* step tolerance *as* $\delta > 0$*. We terminate Algorithm 1 either when* $\mathcal{L}$ *is empty (the problem is infeasible) or when there exists an item* $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x})), B_{hj}(\mathbf{x})) \in \mathcal{L}$ *that satisfies all of the following conditions:*

(a) $|\mathbf{x}| \leq \delta$*,*

(b) $\max B_{gi}(\mathbf{x}) \leq 0$ *for all* $i = 1, \cdots, \alpha$*,*

(c) $-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}}$ *for all* $j = 1, \cdots, \beta$*, and*

(d) $\max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \epsilon$ *for all* $\mathbf{y} \in \mathcal{L}$*.*

We discuss feasibility in more detail in section 4.3.5 Note that, to implement the step tolerance δ, since we subdivide by halving the width of each subbox, we need only ensure that sufficiently many iterations have passed.

Note that we do not set a tolerance on inequality constraints, since these are "one-sided" constraints; for any inequality constraint g_i and subbox $\mathbf{x}$, we satisfy the constraint if $\max B_{gi}(\mathbf{x}) \leq 0$ (see Definition 37 and Theorem 45).

4.3.4 Subdivision

Recall that subdivision is presented in Section 4.2.5. We implement subdivision with Algorithm 2. Since the subdivision of one Bernstein patch is computationally independent of another, each subdivision task is assigned to an individual GPU thread, making Algorithm 2 parallel.

Note that the subdivision of Bernstein patches can be done in any direction, leading to the question of how to select the direction in practice. Example rules are available in the literature, such as maximum width [RC95, Section 3], derivative-based [ZG98, Section 3], or a combination of the two [RC95, Section 3]. In the context of constrained optimization, the maximum width rule is usually favored over derivative-based rules for two reasons: first, computing the partial derivatives of all constraint polynomials can introduce significant computational burden, especially when the number of constraints is large (see Section 4.5.3); second, the precision of Bernstein patches as bounds to the polynomials depends on the *maximum* width of each subbox (Theorem 32 and Corollary 33), so it is beneficial to subdivide along the direction of maximum width for better convergence results.

In each n^{th} iteration of PCBA, we subdivide in each direction r, in the order $1, 2, \cdots, l$. We halve the width of each subbox each time we subdivide, leading to the following remark.

Remark 36. *In the n^{th} iteration, the maximum width of any subbox in $\mathcal{L}$ is 2^{-n}.*

Algorithm 2 $\mathcal{L} = \text{Subdivision}(\mathcal{L}, r)$ (Parallel)

$K \leftarrow \text{length}(\mathcal{L})$
for $k \in \{1, \ldots, K\}$ **do** *in parallel*
  $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \leftarrow \mathcal{L}[k]$;
  Subdivide $\mathbf{x}$ along the r^{th} direction into $\mathbf{x}_L$ and $\mathbf{x}_R$
  Compute patches $B_p(\mathbf{x}_L)$ and $B_p(\mathbf{x}_R)$
  Compute patches $B_{gi}(\mathbf{x}_L)$ and $B_{gi}(\mathbf{x}_R)$
  Compute patches $B_{hj}(\mathbf{x}_L)$ and $B_{hj}(\mathbf{x}_R)$
  $\mathcal{L}[k] \leftarrow (\mathbf{x}_L, B_p(\mathbf{x}_L), B_{gi}(\mathbf{x}_L), B_{hj}(\mathbf{x}_L))$
  $\mathcal{L}[k + K] \leftarrow (\mathbf{x}_R, B_p(\mathbf{x}_R), B_{gi}(\mathbf{x}_R), B_{hj}(\mathbf{x}_R))$
end for
return $\mathcal{L}$

4.3.5 Cut-Off Test

Subdivision would normally occur for every patch in every iteration, leading to exponential memory usage (2^n patches in iteration n). However, by using a cut-off test, some patches can be deleted, reducing both the time and memory usage of PCBA (see Section 4.4 for complexity analysis). To decide which patches are to be eliminated, we require the following definitions.

Definition 37. *An item $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$ is* feasible *if both of the following hold:*

(a) $\max B_{gi}(\mathbf{x}) \leq 0$ *for all* $i = 1, \cdots, \alpha$, *and*

(b) $-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}}$ *for all* $j = 1, \cdots, \beta$.

An item is infeasible *if any of the following hold:*

(c) $\min B_{gi}(\mathbf{x}) > 0$ *for at least one* $i = 1, \cdots, \alpha$, *or*

(d) $\min B_{hj}(\mathbf{x}) > 0$ *for at least one* $j = 1, \cdots, \beta$, *or*

(e) $\max B_{hj}(\mathbf{x}) < 0$ *for at least one* $j = 1, \cdots, \beta$.

An item is undecided *if it is neither feasible nor infeasible.*

Notice in particular, a feasible item must not be infeasible.

Definition 38. *The* solution estimate p^*_{up} *is the smallest upper bound of the cost over all feasible items in* $\mathcal{L}$*:*

$$p^*_{\text{up}} = \min\Big\{ \max\{\ell_2 \mid \ell \in \mathcal{L},\ \ell \text{ feasible}\}\Big\}, \tag{4.12}$$

where $\ell_2 = B_p(\mathbf{x})$ *if* $\ell = (\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$.

Definition 39. *An item* $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$ *is* suboptimal *if*

$$\min B_p(\mathbf{x}) > p^*_{\text{up}}. \tag{4.13}$$

Note that Definitions 38 and 39 are dependent on $\mathcal{L}$; that is, for the purposes of PCBA, optimality is defined in terms of the elements of $\mathcal{L}$. We show in Corollary 41 below how this notion of optimality coincides with optimality of the POP itself.

Feasible, infeasible, undecided, and suboptimal patches are illustrated in Figure 4.2. Any item that is infeasible or suboptimal can be eliminated from $\mathcal{L}$, because the corresponding subboxes cannot contain the solution to the POP (formalized in the following Theorem). We call checking for infeasible and suboptimal items the *cut-off test*.

Theorem 40 (Cut-Off Test)**.** *Let* $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$ *be an item. If the item is infeasible (as in Definition 37) or suboptimal (as in Definition 39), then* $\mathbf{x}$ *does not contain a global minimizer of* (P)*. Such item can be removed from the list* $\mathcal{L}$*.*

Proof. Let $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$ be an item in $\mathcal{L}$. We only need to show:

(a) if $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$ is feasible, then all points in $\mathbf{x}$ are feasible (up to the tolerance ϵ_{eq}), and

(b) if $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$ is infeasible, then all points in $\mathbf{x}$ are infeasible (up to the tolerance ϵ_{eq}), and

(c) if $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x}))$ is suboptimal, then all points in $\mathbf{x}$ are not optimal.

Note that (a) and (b) follow directly from Theorem 30. To prove (c), let $\mathbf{y} \subset \mathbf{u}$ be a subbox on which the solution estimate p^*_{up} is achieved, i.e., $(\mathbf{y}, B_p(\mathbf{y}), B_{gi}(\mathbf{y}), B_{hj}(\mathbf{y}))$ is feasible and

$$\max B_p(\mathbf{y}) = p^*_{\text{up}}. \tag{4.14}$$

Let $y \in \mathbf{y}$ be arbitrary, then it follows from Theorem 30 and the definition of suboptimality that

$$p(x) \geq \min B_p(\mathbf{x}) > \max B_p(\mathbf{y}) \geq p(y) \tag{4.15}$$

for all $x \in \mathbf{x}$. Since such point y is necessarily feasible (obtained from condition (b)), x cannot be global minimum to the POP. □

Corollary 41. *Suppose there exists a (feasible) global minimizer x^* of the POP* (P). *Then, there always exists an item $(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$ such that $x^* \in \mathbf{x}$ while executing Algorithm 1.*

Proof. This result is the contrapositive of Theorem 40. □

We implement the cut-off tests as follows. Algorithm 3 (FindBounds) computes the maximum and minimum element of each Bernstein patch; Algorithm 4 (CutOffTest) implements the cut-off tests and marks all subboxes to be eliminated with a list $\mathcal{L}_{\text{elim}}$; and Algorithm 5 (Eliminate) eliminates the marked subboxes from the list $\mathcal{L}$. Algorithms 3 and 5 are parallelizable, whereas Algorithm 4 must be computed serially.

Algorithm 3 $\mathcal{L}_{\text{bounds}} = \text{FindBounds}(\mathcal{L})$ (Parallel)

1: $K \leftarrow \text{length}(\mathcal{L})$
2: **for** $k \in \{1, \ldots, K\}$ **do** *in parallel*
3: $\quad (\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \leftarrow \mathcal{L}[k]$
4: $\quad$ Find $\min B_p(\mathbf{x})$ and $\max B_p(\mathbf{x})$ by parallel reduction
5: $\quad$ Find $\min B_{gi}(\mathbf{x})$ and $\max B_{gi}(\mathbf{x})$ similarly
6: $\quad$ Find $\min B_{hj}(\mathbf{x})$ and $\max B_{hj}(\mathbf{x})$ similarly
7: $\quad \mathcal{L}_{\text{bounds}}[k] \leftarrow \begin{pmatrix} & \{\min B_p(\mathbf{x}), & \max B_p(\mathbf{x})\} \\ \mathbf{x}, & \{\min B_{gi}(\mathbf{x}), & \max B_{gi}(\mathbf{x})\} \\ & \{\min B_{hj}(\mathbf{x}), & \max B_{hj}(\mathbf{x})\} \end{pmatrix}$
8: **end for**
9: **return** $\mathcal{L}_{\text{bounds}}$

Algorithm 4 $[p^*_{\text{up}}, p^*_{\text{lo}}, \mathcal{L}_{\text{save}}, \mathcal{L}_{\text{elim}}]$ = CutOffTest$(\mathcal{L}_{\text{bounds}})$

$p^*_{\text{up}} \leftarrow +\infty,\ p^*_{\text{lo}} \leftarrow +\infty$
$K \leftarrow$ length$(\mathcal{L}_{\text{bounds}})$
for $k \in \{1, \ldots, K\}$ **do**
$$\begin{pmatrix} & \{\min B_p(\mathbf{x}), & \max B_p(\mathbf{x})\} \\ \mathbf{x}, & \{\min B_{gi}(\mathbf{x}), & \max B_{gi}(\mathbf{x})\} \\ & \{\min B_{hj}(\mathbf{x}), & \max B_{hj}(\mathbf{x})\} \end{pmatrix} \leftarrow \mathcal{L}_{\text{bounds}}[k]$$
  if $-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}}$ **then**
    if $\max B_{gi}(\mathbf{x}) \leq 0$ **then**
      $p^*_{\text{up}} \leftarrow \min(p^*_{\text{up}}, \max B_p(\mathbf{x}))$
    end if
    if $\min B_{gi}(\mathbf{x}) \leq 0$ **then**
      $p^*_{\text{lo}} \leftarrow \min(p^*_{\text{lo}}, \min B_p(\mathbf{x}))$
    end if
  end if
end for
Initialize lists for indices of patches to save or eliminate $\mathcal{L}_{\text{save}} \leftarrow \{\}$, $\mathcal{L}_{\text{elim}} \leftarrow \{\}$
for $k \in \{1, \cdots, K\}$ **do**
  if $-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}}$
  and $\min B_{gi}(\mathbf{x}) \leq 0$
  and $\min B_p(\mathbf{x}) \leq p^*_{\text{up}}$ **then**
    Append k to $\mathcal{L}_{\text{save}}$
  else
    Append k to $\mathcal{L}_{\text{elim}}$
  end if
end for
return $p^*_{\text{up}}, p^*_{\text{lo}}, \mathcal{L}_{\text{save}}, \mathcal{L}_{\text{elim}}$

Algorithm 5 $\mathcal{L} = \text{Eliminate}(\mathcal{L}, \mathcal{L}_{\text{save}}, \mathcal{L}_{\text{elim}})$ (Parallel)

$K_{\text{save}} \leftarrow \text{length}(\mathcal{L}_{\text{save}})$
$K_{\text{elim}} \leftarrow \text{length}(\mathcal{L}_{\text{elim}})$
$K_{\text{replace}} \leftarrow K_{\text{elim}} - 1$
if $K_{\text{elim}} = 0$ **or** $\mathcal{L}_{\text{elim}}[1] > K_{\text{elim}}$ **then**

  return $\mathcal{L}$
end if
for $k \in \{1, \cdots, K_{\text{elim}}\}$ **do**
  if $\mathcal{L}_{\text{elim}}[k] \geq K_{\text{save}}$ **then**
    $K_{\text{replace}} \leftarrow k - 1$
    break
  end if
end for
for $k \in \{1, \cdots, K_{\text{replace}}\}$ **do** *in parallel*
  $\mathcal{L}[\mathcal{L}_{\text{elim}}[k]] \leftarrow \mathcal{L}[\mathcal{L}_{\text{save}}[K_{\text{save}} + 1 - k]]$
end for

return $\mathcal{L}$

4.3.6 Advantages and Disadvantages of PCBA

PCBA has several advantages. First, it always finds a global optimum (if one exists), subject to tolerances. PCBA does not require an initial guess, and does not converge to local minima, unlike generic nonlinear solvers (e.g., `fmincon` [Mat19]). It also does not require tuning hyperparameters. As we show in Section 4.4, PCBA has bounded time and memory complexity under certain assumptions. Finally, due to parallelization, PCBA is fast enought to enable RTD* for real-time, safe, optimal trajectory planning, which we demonstrate in Section 4.6.

However, PCBA also has several limitations in comparison to traditional approaches to solving POPs. First, to prove the bounds on time and memory usage, at any global minimum, we require that active constraints are linearly independent, and that the Hessian of the cost function is positive definite (see Theorems 42 and 45). Furthermore, due to the number of Bernstein patches growing exponentially with the decision variable dimension, we have not yet applied PCBA to problems larger than four-dimensional.

4.4 Complexity Analysis

In this section, we prove that Algorithm 1 terminates by bounding the number of iterations of PCBA for both unconstrained and constrained POPs. We also prove the number of Bernstein patches (i.e., the length of the list $\mathcal{L}$ in Algorithm 1) is bounded after sufficiently many iterations, under certain assumptions. For convenience, in the remainder of this section we use $\mathbf{x} \in \mathcal{L}$ as a shorthand notation for $\mathbf{x} = \ell_1$ where $\ell = (\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}$. The proofs of Theorems 44, 45, and 46 are in the Appendix.

4.4.1 Unconstrained Case

We first consider unconstrained POPs whose optimal solutions are not on the boundary of $\mathbf{u}$. Note, we can treat optimal solutions on the boundary of $\mathbf{u}$ as having active linear constraints; see Section 4.4.2 for the corresponding complexity analysis. In the unconstrained case, all points in $\mathbf{u}$ are feasible, and we are interested in solving

$$\min_{x \in \mathbf{u} \subset \mathbb{R}} \quad p(x) \tag{4.16}$$

where p is an l-dimensional multivariate polynomial. Given an optimality tolerance ϵ and step tolerance δ, we bound the number of iterations to solve (4.16) with PCBA as follows:

Theorem 42. *Let p in (4.16) be a multivariate polynomial of dimension l with Lipschitz constant L_p. Then the maximum number of iterations needed to solve (4.16) up to accuracy ϵ and δ is*

$$N = \left\lceil \max \left\{ -\log_2 \delta, -\frac{1}{2} \log_2 \left(\frac{\epsilon}{4\zeta_p} \right), -\log_2 \left(\frac{\epsilon}{2L_p\sqrt{l}} \right) \right\} \right\rceil, \tag{4.17}$$

where ζ_p is the constant in Theorem 32 corresponding to polynomial p, and $\lceil \cdot \rceil$ rounds up to the nearest integer.

Proof. Let n be the current iteration number. It is sufficient to show that, for all $n > N$, there exists a subbox $\mathbf{x} \in \mathcal{L}$ of $\mathbf{u}$ such that the following conditions hold:

(a) $|\mathbf{x}| \leq \delta$; and

(b) $\max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \epsilon$ for all $\mathbf{y} \in \mathcal{L}$.

Let $x^* \in \mathbf{u}$ be a minimizer of (4.16). According to Corollary 41, there exists a subbox $\mathbf{x} \in \mathcal{L}$ such that $x^* \in \mathbf{x}$. To prove such $\mathbf{x}$ satisfies Condition (a), notice from Remark 36 that

$$|\mathbf{x}| \leq 2^{-n} \leq 2^{-N} \leq \delta. \tag{4.18}$$

To prove Condition (b), first notice for any $\mathbf{y} \in \mathcal{L}$,

$$\min B_p(\mathbf{y}) \geq \min_{y \in \mathbf{y}} p(y) - \zeta_p \cdot 2^{-2n} \tag{4.19}$$

$$\geq p(x^*) - \zeta_p \cdot 2^{-2n}, \tag{4.20}$$

where (4.19) follows from Theorem 32; and (4.20) follows from the definition of x^*. Therefore

$$\max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \max_{x \in \mathbf{x}} p(x) - p(x^*) + 2\zeta_p \cdot 2^{-2n} \tag{4.21}$$

$$\leq L_p \cdot \left(\sqrt{l} \cdot |\mathbf{x}|\right) + 2\zeta_p \cdot 2^{-2n} \tag{4.22}$$

$$\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \tag{4.23}$$

where (4.21) follows from Corollary 33 and (4.20); (4.22) is true because $\|x^* - x\| \leq \sqrt{l} \cdot |\mathbf{x}|$ for all $x \in \mathbf{x}$; and (4.23) follows from (4.17) and the assumption that $n > N$. □

According to (4.17), the rate of convergence with respect to (WRT) the decision variables is quadratic (1st term); the rate of convergence WRT the objective function is either quadratic (2nd term) or linear (3rd term), depending on which term dominates. However, a tighter bound exists if one of the global minimizers satisfies the second-order sufficient condition for optimality [NW06, Theorem 2.4], which is shown in the following theorem:

Theorem 43. *Let p in (4.16) be a multivariate polynomial of dimension l. Let the Hessian $\nabla^2 p$ be positive definite at some global minimizer x^* of (4.16), where x^* is not on the boundary of $\mathbf{u}$. Then the maximum number of iterations needed to solve (4.16) up to accuracy ϵ and δ is*

$$N = \left\lceil \max \left\{ -\log_2 \delta,\, C_1,\, -\frac{1}{2}\log_2 \epsilon + C_2 \right\} \right\rceil, \tag{4.24}$$

where C_1 and C_2 are constants that only depend on p.

Theorem 43 proves a quadratic rate of convergence WRT the objective function once the number of iterations exceeds a threshold C_1, given that the second-order optimality condition is satisfied at some global minimizer. We now discuss the number of patches remaining after sufficiently many iterations, which gives an estimate of memory usage when Algorithm 1 is applied to solve (4.16).

Theorem 44. *Suppose there are $m < \infty$ global minimizers $x_1^*, \cdots, x_m^*$ of (4.16), and none of them are on the boundary of the unit box $\mathbf{u}$. Let the Hessian $\nabla^2 p$ be positive definite at these minimizers. Then after sufficiently many iterations of Algorithm 1, the number of Bernstein patches remaining (i.e., length of the list $\mathcal{L}$ in Algorithm 1) is bounded by a constant.*

The constant bound in Theorem 44 scales exponentially with the problem dimension, and is a function of the condition number of the cost function's Hessian at the minimizers.

4.4.2 Constrained Case

Theorem 45. *Suppose that the linear independence constraint qualification (LICQ) [NW06, Definition 12.4] is satisfied at all global minimizers $x_1^*, \cdots x_m^*$ of the constrained POP* (P)*, and at least one constraint is active (i.e., the active set $\mathcal{A}(x^*)$ [NW06, Definition 12.1] is nonempty) at some minimizer $x^* \in \{x_1^*, \cdots, x_m^*\}$. Then the maximum number of iterations needed to solve* (P) *up to accuracy ϵ, δ, and equality constraint tolerance ϵ_{eq} is*

$$N := \left\lceil \max \left\{ C_7,\ -\log_2 \delta,\ -\log_2 \epsilon_{\text{eq}} + C_8,\ -\log_2 \epsilon + C_9 \right\} \right\rceil, \tag{4.25}$$

where C_7, C_8, C_9 are constants.

Theorem 45 gives a bound on the number of PCBA iterations needed to solve a POP up to specified tolerances. In particular, (4.25) shows the rate of convergence is *linear* in step tolerance (2nd term), equality constraint tolerance (3rd term), and objective function (4th term), once the number of iterations is larger than a constant (1st term). We next prove a bound on the number of items in the list $\mathcal{L}$ after sufficiently many iterations.

Theorem 46. *Suppose there are m ($m < \infty$) global minimizers $x_1^*, \cdots, x_m^*$ of the constrained problem* (P)*, and none of them are on the boundary of the unit box* $\mathbf{u}$*. Let the critical cone (see [NW06, (12.53)]) be nonempty for* (P_n) *as in the proof of Theorem 45. Then after sufficiently many iterations of Algorithm 1, the number of Bernstein patches remaining (i.e., length of the list $\mathcal{L}$) is bounded by a constant.*

The constant proved in Theorem 46 scales exponentially with respect to the dimension of the problem.

4.4.3 Memory Usage Implementation

We now state the amount of GPU memory required to store a single item, namely

$$(\mathbf{x}, B_p(\mathbf{x}), B_{gi}(\mathbf{x}), B_{hj}(\mathbf{x})) \in \mathcal{L}, \tag{4.26}$$

given the degree and dimension of the cost and constraint polynomials. Note that, for our implementation, all numbers in an item are represented using 4B of space, as either floats or unsigned integers.

For a multi-index $J = (j_1, \cdots, j_l) \in \mathbb{N}^l$, let $\Pi J = j_1 \times \cdots \times j_l$, and let $J+n = (j_1+n, \cdots, j_l+n)$ for $n \in \mathbb{N}$. Let P be the multi-degree of the cost p. Let G be a multi-degree large enough for all inequality constraints g_i, and H a multi-degree large enough for all equality constraints h_j. By "large enough" we mean that, if G_i is the multi-degree of any g_i then $G_i \leq G$ (and similarly for H). Then, as per [TG17, §4.1], an item can be stored in memory as an array with the following number of entries:

$$2l + (\Pi(P+1)) + (\alpha \cdot \Pi(G+1)) + (\beta \cdot \Pi(H+1)), \tag{4.27}$$

where the first $2l$ entries store the upper and lower bounds (in each dimension) of the subbox $\mathbf{x}$.

4.4.4 Summary

We have shown that PCBA will find a solution to (P), if one exists, in bounded time. We have also shown that the memory usage of PCBA is bounded after a finite number of iterations, which implies that the memory usage is bounded; and we have provided a way to compute how much memory is required to store the list $\mathcal{L}$. Next, we benchmark PCBA on a variety of problems and compare it to two other solvers.

4.5 PCBA Evaluation

In this section, we compare PCBA against a derivative-based solver (`fmincon` [Mat19]) and a convex relaxation method (BSOS [LTY17]). First, we test all three solvers on eight **Benchmark Evaluation** problems with dimension less than or equal to 4. Second, we compare all three solvers on several **Increasing Number of Constraints** problems, to assess how each solver scales on a variety of difficult objective functions [Gav19].

All of the solvers/problems in this section are run on a computer with a 3.7GHz processor, 16 GB of RAM, and an Nvidia GTX 1080 Ti GPU. PCBA is implemented with MATLAB R2017b executables and CUDA 10.2. Our code is available on GitHub: `https://github.com/ramvasudevan/GlobOptBernstein`.

4.5.1 Parameter Selection

To set up a fair comparison, we scale each problem to the $\mathbf{u} = [0, 1]^l$ box, where l is the problem dimension. For PCBA, we use the stopping criteria in Section 4.3. To choose ϵ, we first compute

the patch $B(\mathbf{u})$, then set

$$\epsilon = (10^{-7}) \cdot (\max B(\mathbf{u}) - \min B(\mathbf{u})) \,. \tag{4.28}$$

We set the maximum number of iterations to $N = 28$. We do not set δ, which determines the minimum number of iterations; δ is only needed to prove complexity bounds in Section 4.4.

BSOS [LTY17, Section 4] requires the user to specify the size of the semidefinite matrix associated with the convex relaxation of the POP. This is done by by selecting a pair of parameters, d and k (note these are different from our use of d and k). Though one has to increase d and k gradually to ensure convergence, larger values of d and k correspond to larger semidefinite programs, which can be difficult to solve. We chose d and k separately for the Benchmark Evaluation. We used $d = k = 2$ for the Increasing Number of Constraints.

For `fmincon` [Mat19], we set the `OptimalityTolerance` option to ϵ in (4.28). We set `MaxFunctionEvaluations` $= 10^5$ and `MaxIterations` $= 10^4$. We also provide `fmincon` with the analytic gradients of the cost and constraints.

4.5.2 Benchmark Evaluation

4.5.2.1 Setup

We tested PCBA, BSOS, and `fmincon` on eight benchmark POPs [NA11], listed as P1 through P8; the problems are reported in the Appendix. We ran each solver 50 times on each problem, and report the median solution error and time required to find a solution. Since `fmincon` may or may not converge to the global optimum depending on its initial guess, we used random initial guesses for each of the 50 attempts.

4.5.2.2 Results

The results are summarized in Table 4.1. For additional results (e.g., to plot the results of any of the problems), see the GitHub repository.

In terms of solution quality, PCBA always found the solution to within the desired optimality tolerance ϵ, except for on P1, where PCBA stopped at the maximum allowed number of iterations (28); PCBA always used between 22 and 28 iterations. BSOS always found a lower bound to the solution, as expected. While `fmincon` converged to the global optimum at least once on every problem, it often converged to local minima, hence the large error values on some problems. In terms of solve time, `fmincon` solves the fastest (in 10–20 ms), PCBA is about twice as slow as `fmincon`, and BSOS is one to two orders of magnitude slower than PCBA.

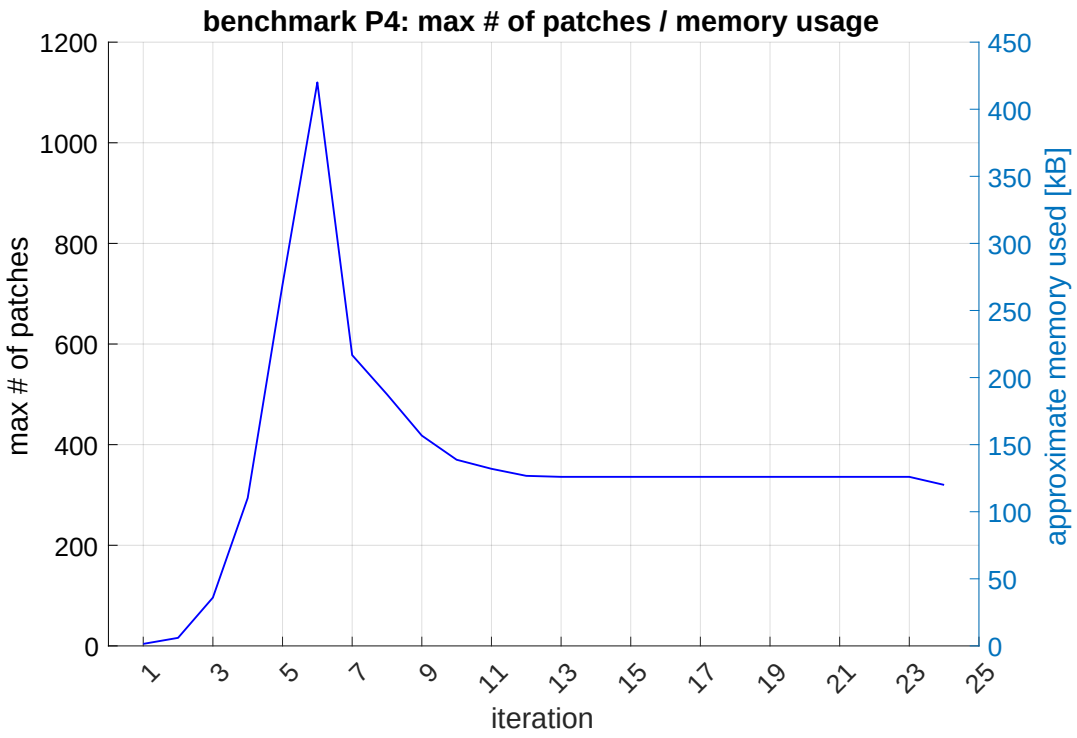


Figure 4.3: The maximum number of patches (left axis) and corresponding GPU memory used (right axis) at each iteration of PCBA, for P4 of the benchmark problems (see Section 4.5). This problem took 24 iterations to solve. Notice that the number of patches peaks in iteration 5, then stays under 400 patches at every iteration from iteration 9 onwards; this visualizes Theorem 46.

For PCBA, the memory usage (computed with (4.27)) increases roughly by one order of magnitude for each additional dimension of the decision variable increases (Table 4.1 reports the peak GPU memory used by PCBA on each benchmark problem). Notice that PCBA never uses more than several MB of GPU memory, which is much less than the 11 GB available on the Nvidia GTX 1080 Ti GPU. Figure 4.3 shows the number of patches and the amount of GPU memory used on P4. We see that the memory usage peaks, then stays below a constant, as predicted by Theorem 46.

4.5.3 Increasing Constraint Problems

Next, we tested each solver on problems with an increasing number of constraints, to each solver for use with RTD; we find in practice that a robot running RTD must handle between 30 and 300 constraints at each planning iteration.

4.5.3.1 Setup

We first choose an objective function with either many local minima or a nearly flat gradient near the global optimum (the global optimizer is known for each function). In particular, we tested on the ElAttar-Vidyasagar-Dutta, Powell, Wood, Dixon-Price (with $l = 2, 3, 4$), Beale, Bukin02, and Deckkers-Aarts problems (see the Appendix and [Gav19]).

For each objective function, we generate 200 random constraints in total, while ensuring at least one global optimizer stays feasible (if there are multiple global optimizers, we choose one at random that will be feasible for all constraints). To generate a single constraint $g : \mathbf{u} \to \mathbb{R}$, we first create a polynomial g_{temp} as a sum of the monomials of the decision variable with maximum degree 2, with random coefficients in the range $[-5, 5]$. To ensure x^* is feasible, we evaluate g_{temp} on x^*, then subtract the resulting value from g_{temp} to produce g (i.e., $g \leftarrow g_{\text{temp}} - g_{\text{temp}}(x^*)$).

We run PCBA, BSOS, and `fmincon` on each objective function for 20 trials, with 10 random constraints in the first trial, and adding 10 constraints in each trial. As before, we run `fmincon` 50 times for each trial with random initial guesses, since its performance is dependent upon the initial guess.

4.5.3.2 Results

To illustrate the results, data for the Powell objective function are shown in Figure 4.4. The data (and plots) for the other objective functions are available in the GitHub repository

In terms of solution quality, all three algorithms converge to the global optimum often when the number of constraints is low, but `fmincon` converges to suboptimal solutions more frequently as the number of constraints increases. PCBA and BSOS are always able to find the optimal solution. PCBA is always able to find the global optimum regardless of the number of constraints, unlike BSOS (which runs out of memory) or `fmincon` (which converges to local minima).

All three solvers require an increasing amount of solve time as the number of constraints increases. PCBA is comparable in speed to `fmincon` on 2-D problems, but is typically slower on higher-dimensional problems. Regardless of the number of constraints, BSOS takes three to four orders of magnitude more time to solve than PCBA or `fmincon`.

More details on PCBA are presented in Table 4.2. PCBA's time to find a solution increases roughly by an order of magnitude when the decision variable dimension increases by 1; however, PCBA solves all of the increasing constraint POPs within 0.5 s. The memory usage increases by 1–3 orders of magnitude with each additional dimension; however, PCBA never uses more than 650 MB of GPU memory, well below the 11 GB available. Figure 4.5 shows PCBA's GPU memory usage versus the number of constraints for the Powell objective function.

4.5.4 Summary

As expected from the complexity bounds in Section 4.4.2, the results in this section indicate that PCBA is practical for quickly solving 2-D POPs with hundreds of constraints. We leverage this next by applying PCBA to solve RTD's POP (4.4) for real-time receding-horizon trajectory optimization.

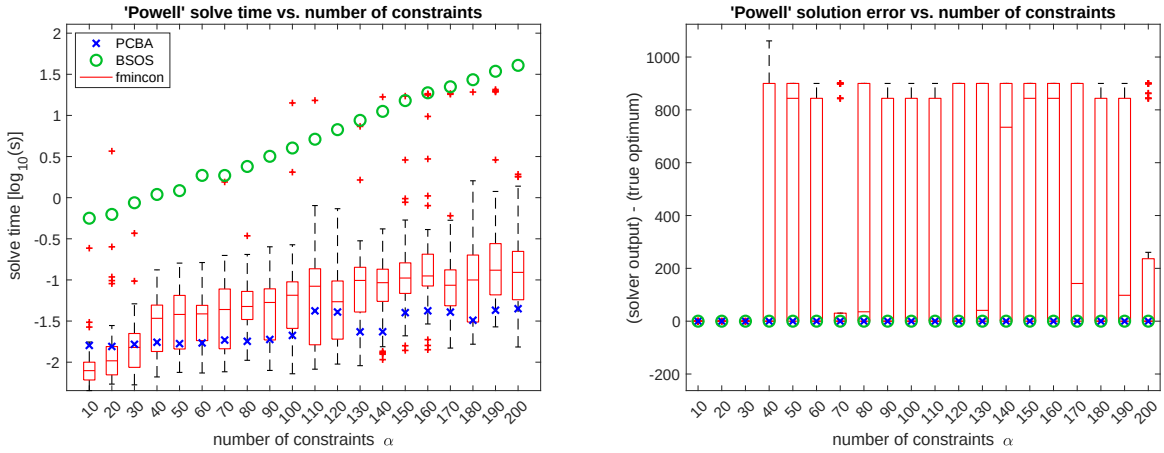


Figure 4.4: Results for an increasing number of constraints on the Powell objective function (see the Appendix) for the PCBA, BSOS, and `fmincon`. The top plot shows the time required to solve the problem as the number of constraints increases. The bottom plot shows the error between each solver's solution and the true global optimum. For both time and error, `fmincon` is shown as a box plot over 50 trials with random initial guesses; the central red line indicates the median, the top and bottom of the red box indicate the 25^{th} and 75^{th} percentiles, the black whiskers are the most extreme values not considered outliers, and the outliers are red plus signs. PCBA solves the fastest in general; `fmincon` typically solves slightly slower than PCBA for more than 40 constraints; and BSOS is the slowest solver. PCBA and BSOS always find the global optimum, as does `fmincon` when there are not many constraints, because the Powell objective function is convex. Above 30 constraints, `fmincon` frequently has large error due to convergence to local minima.

	l	max time [s]	max items	max memory
E-V-D	2	0.0360	66	171 kB
Powell	2	0.0447	1200	6.24 MB
Wood	2	0.0532	54	220 kB
D-P 2-D	2	0.0259	90	433 kB
D-P 3-D	3	0.0675	356	3.47 MB
D-P 4-D	4	0.402	4994	193 MB
Beale	2	0.0302	106	259 kB
Bukin02	4	0.393	10550	110 MB
D-A	4	0.389	51886	647 MB

Table 4.2: Results for the increasing constraints PCBA evaluation. Abbreviated problem names (as in the Appendix) are on the left, along with each problem's decision variable dimension l. Over all 20 trials (with between 10 and 200 constraints), we report the maximum time spent find a solution, the maximum number of items in the list $\mathcal{L}$, and the maximum amount of GPU memory used. Note that the problems all solved under 0.5 s regardless of the number of constraints, and no problem requested more than 650 MB of memory.

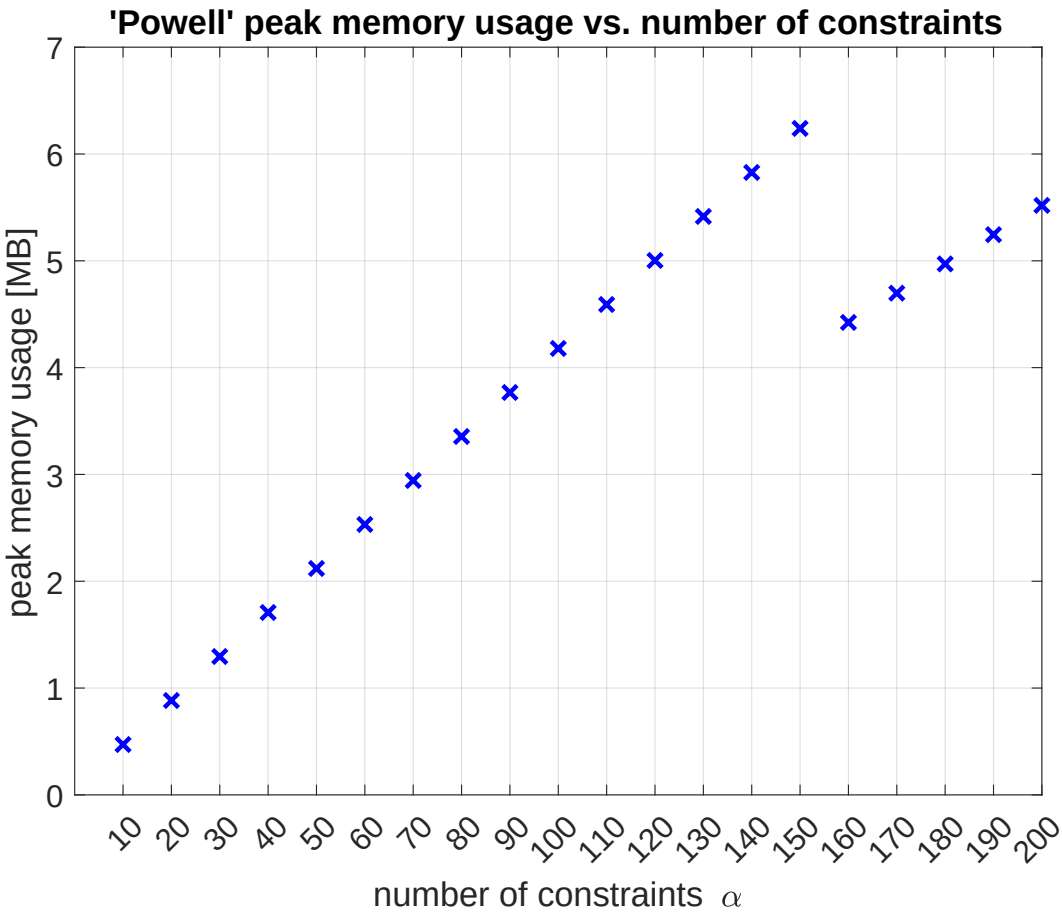


Figure 4.5: The approximate peak GPU memory used by PCBA for the Powell problem, as a function of the number of constraints. Since the amount of memory required per item in the list $\mathcal{L}$ grows linearly with the number of constraints, the overall memory usage also grows linearly. However, at 160 constraints, we see a drop in the memory usage; this is because the additional constraints render more parts of the problem domain infeasible, resulting in more items being eliminated per PCBA iteration. Note that the maximum memory usage is well under the several GB of memory available on a typical GPU.

4.6 Hardware Demonstrations

Recall from Section 4.2.3 that RTD enables real-time, provably collision-free trajectory planning via solving a POP every planning iteration. RTD is provably collision-free regardless of the POP solver used, meaning that we are able to test PCBA safely on hardware; when PCBA is applied to RTD, we call the resulting trajectory planning algorithm **RTD***. See Figure 4.1 and the video (click for link).

In this section, we apply RTD* to a Segway robot navigating a hallway with static obstacles.

Recall that RTD always produces *dynamically feasible* trajectory plans [KVB+18]. As proven in Corollary 41 and demonstrated in Section 4.5, PCBA always finds *optimal* solutions. This section shows that PCBA/RTD* improves the *liveness* of a robot by successfully navigating a variety of scenarios, and outperforming `fmincon`/RTD.

4.6.1 Overview

4.6.1.1 Demonstrations

We ran two demonstrations in a 20×3 m^2 hallway. In the first demonstration, we filled the hallway with random static obstacles and ran RTD*. In the second demonstration, we constructed two difficult scenarios and ran bot PCBA/RTD* and `fmincon`/RTD on each.

4.6.1.2 Hardware

We use a Segway differential-drive robot with a planar Hokuyo UTM-30LX LIDAR for mapping and obstacle detection (see Figure 4.1). Mapping, localization, and trajectory optimization run onboard, on a 4.0 GHz laptop with an Nvidia GeForce GTX 1080 GPU.

4.6.1.3 POPs

As in Section 4.2.3 (also see Figure 4.1), the robot plans by optimizing over a set $Q \subset \mathbb{R}^2$ of parameterized trajectories as in [KVB+18, (16)]. The parameters are speed $q_1 \in [0, 1.2]$ m/s and yaw rate $q_2 \in [-1, 1]$ rad/s, so $Q = [0, 1] \times [-1, 1]$. The trajectories are between 1 and 2 s long, depending on the robot's initial speed (since each trajectory includes a braking maneuver). The robot must find a new plan (i.e., PCBA must solve a new POP) every 0.5 s, or else it begins executing the braking maneuver associated with its previously-computed plan [KVB+18, Remark 70]. In other words, we require PCBA to return a feasible solution or that the problem is infeasible. PCBA is given a time limit of 0.4 s to find a solution, because the robot requires 0.1 s for other onboard processes.

At each planning iteration, obstacles are converted into constraints as in [KVB+18, Sections 6 and 7]; this produces a list of $\alpha \in \mathbb{N}$ inequality constraints, which we denote $w(x_1, \cdot), \ldots, w(x_\alpha, \cdot) : Q \to \mathbb{R}$. In practice, $30 \leq \alpha \leq 300$ (see Figure 4.7).

The objective function is constructed at each planning iteration as follows. Offline, we represent the endpoint of any parameterized trajectory as a degree 10 polynomial $x : Q \to X$. At runtime, we generate $N_{\text{wp}} \in \mathbb{N}$ waypoints (i.e., desired locations for the robot to reach), denoted

$\{x_n\}_{n=1}^{N_{\text{wp}}}$. We then create the following POP:

$$\begin{array}{ll} \underset{q \in Q}{\text{argmin}} & \prod_{n=1}^{N_{\text{wp}}} \left(\|x(q) - x_n\|_2^2 \right) \\ \text{s.t} & w(x_i, q) \leq 0 \; \forall \; i = 1, \cdots, \alpha. \end{array} \tag{4.29}$$

The objective function is degree $10 \cdot N_{\text{wp}}$, and the constraints are each degree 12. Furthermore, ignoring the constraints, the objective function requires the POP solver to choose between as many global minima as there are waypoints.

4.6.2 Demo 1

The first demo shows the ability of the RTD* to plan safe trajectories in randomly-generated scenarios in real time, demonstrating *dynamic feasibility*, *optimality*, and *liveness*.

4.6.2.1 Setup

The robot was required to move autonomously back and forth ten times between two global goal locations spaced 12 m apart, while $(30 \text{ cm})^3$ box-shaped obstacles were randomly placed in its path. At each planning iteration, we generated $N_{\text{wp}} = 2$ waypoints, x_{L} and x_{R}, both 1.5 m ahead of the robot in the direction of the global goal; x_{L} is on the left side of the hallway relative to the robot, and x_{R} is on the right.

After running the robot with RTD*, we ran `fmincon` on the 528 saved POPs generated during these 10 trials (we do not run BSOS due to its slow solve time). Each POP has between 49 and 245 constraints (see Figure 4.7). For each POP, we initialized `fmincon` with 25 random initial guesses, and did not require `fmincon` to solve within 0.4 s (i.e., we did not enforce the real time planning constraint). To understand the timing of PCBA and for fair comparison with `fmincon`, we re-ran PCBA 25 times on each trial and did not require it to solve in real time.

4.6.2.2 Results

The robot running RTD* successfully completed every trial (meaning that it reached the global desired goal location without collisions, and without human assistance).

When re-running on the saved POPs, `fmincon` performs nearly as well as PCBA in terms of finding solutions. Out of all 25×528 attempts in which `fmincon` converged to a feasible solution, `fmincon` converged to a greater cost than PCBA 93.9% of the time (recall that PCBA provably upper-bounds the optimal solution); however, the `fmincon` solution was only 0.77% greater in cost than the PCBA solution on average, indicating that `fmincon` was often able to find a global optimum when given enough attempts. In terms of feasibility, PCBA and `fmincon`

also show similar results. PCBA reports that 7.01% of the POPs are infeasible, whereas `fmincon` converged to an infeasible result on 8.08% of the 25×528 total attempts. Note that, on 14.2% of the 528 POPs, `fmincon` converged to an infeasible result least once out of 25 attempts.

Where `fmincon` suffers with respect to PCBA is in its consistency of finding an answer within the time limit (see Figure 4.6). While `fmincon` is often able to solve in 10^{-2} s (an order of magnitude faster than PCBA), it has a standard deviation of up to 10 s. On the other hand, PCBA always finds a solution or returns infeasible within 0.4 s, and has a standard deviation of 2.4 ms on average over all 25×528 POPs. To summarize: as we expect from the theory in Section 4.4, PCBA's solve time in practice appears constant *on real trajectory optimization problems*.

4.6.3 Demo 2

The second demo shows that RTD* can navigate difficult scenarios because PCBA is able to rapidly solve POPs with hundreds of constraints. Recall that, in any planning iteration, RTD and RTD* command the robot to begin braking if they cannot find a new trajectory plan (i.e., solve (P)). By *difficult* scenarios, we mean that the obstacles are arranged to cause the robot to have to brake often. Therefore, by RTD*'s successful navigation of these scenarios, we demonstrate *liveness*.

4.6.3.1 Setup

The robot was required to navigate two difficult scenarios autonomously. In the first scenario, static obstacles were arranged to force the robot to turn frequently, and to decide to go left or right around each obstacle. In the second scenario, the robot was required to navigate a tight obstacle blockade. For each scenario, we ran PCBA/RTD* and `fmincon`/RTD once each. At each planning iteration, we generate $N_{\text{wp}} = 1$ waypoint positioned 1.5 m away from the robot along a straight line to the global goal; this produces a convex cost function for (4.29), but the constraints make the problem nonconvex.

4.6.3.2 Results

In the first scenario, both RTD* and RTD are able to successfully navigate the scenario. Recall that the robot begins emergency braking when it does not find a feasible trajectory in a planning iteration. RTD* brakes 6 times, whereas RTD brakes 13 times; furthermore, RTD* only takes 27 s to navigate to the goal, whereas RTD takes 43 s. In other words, PCBA/RTD* is half as conservative as `fmincon`/RTD, because it finds feasible solutions more frequently.

The results of the second scenario confirm that RTD* is less conservative than RTD. In this scenario, RTD* is able to navigate the entire scenario autonomously without human assistance,

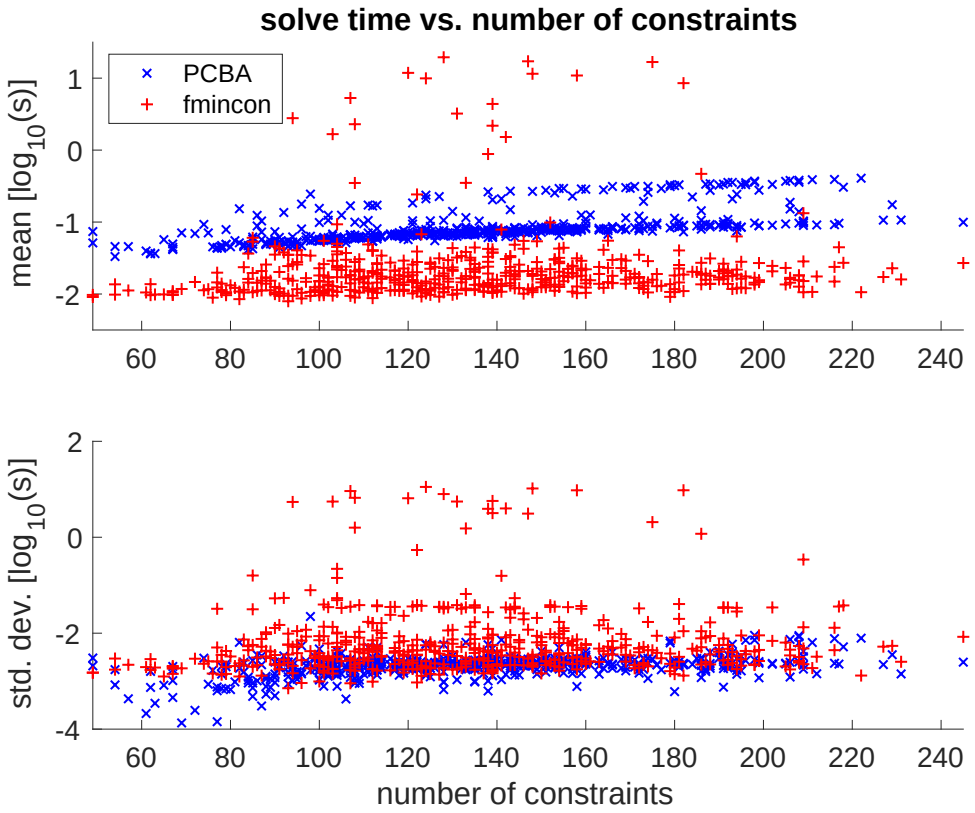


Figure 4.6: Solve times of PCBA and `fmincon` on 528 POPs generated by the Segway robot navigating random scenarios in Demo 1. Each POP was solved 25 times by each solver. While `fmincon` can often find a solution an order of magnitude faster than PCBA, it also has a much higher standard deviation, meaning that it is less consistent at obeying the real-time limit required by mobile robot trajectory planning.

whereas RTD causes the robot to become stuck (see Fig. 4.8), and a human operator must drive the robot for a short time to enable to it to continue moving autonomously.

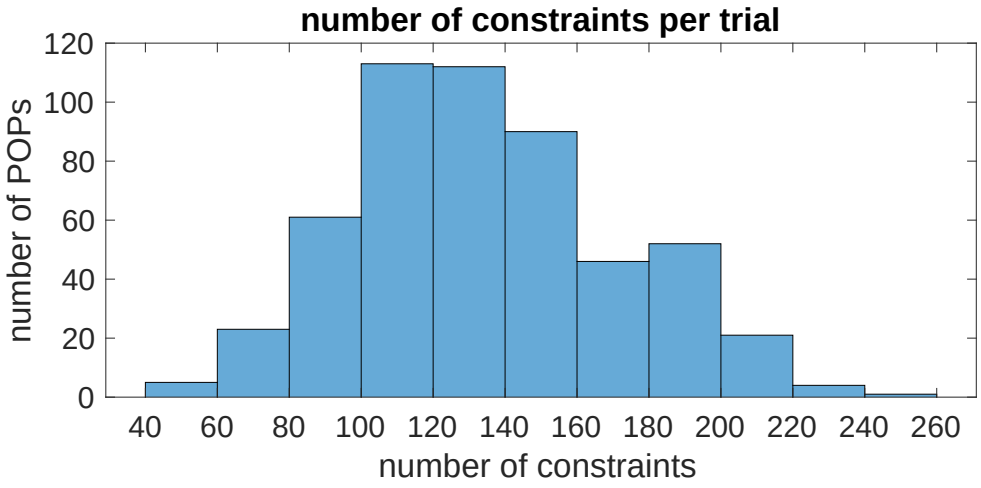


Figure 4.7: The number of POPs from Demo 1, out of 528, that fall into the given bins of number of constraints; we see that most of the POPs had 100 – 140 constraints. This number of constraints can makes it challenging to solve a POP while constrained by a real-time planning limit.

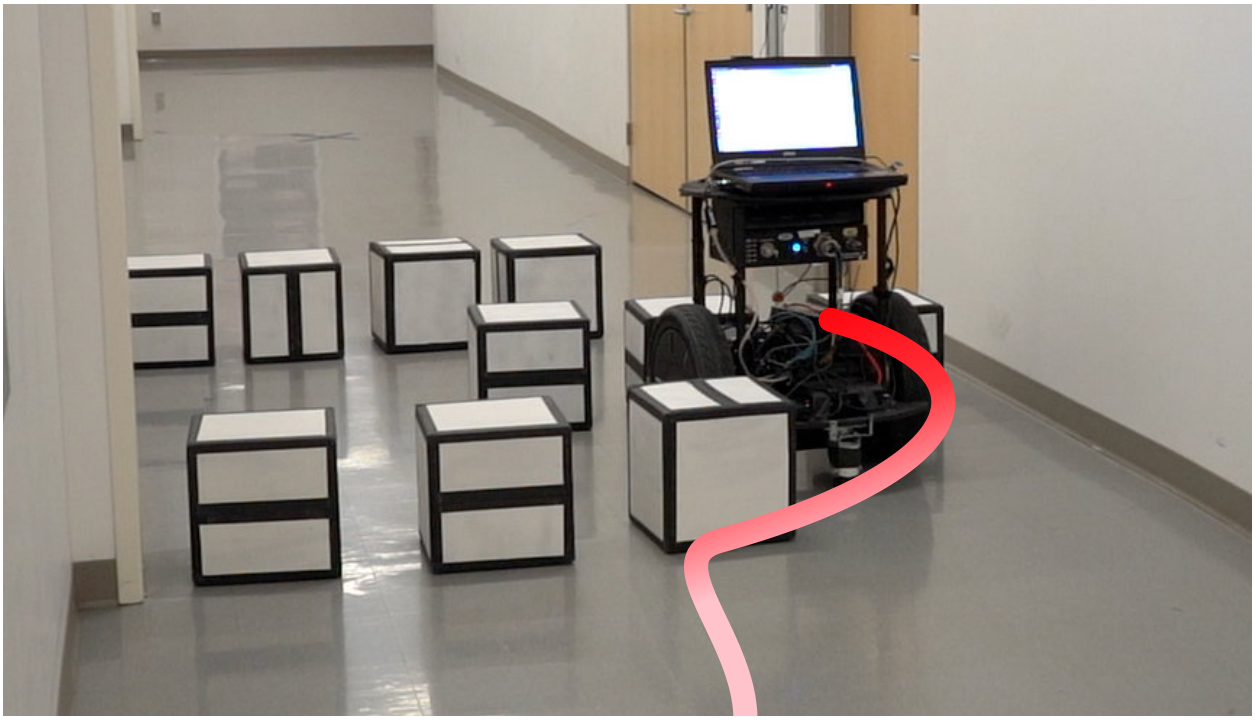


Figure 4.8: The robot becomes stuck when planning with RTD/`fmincon` in the second scene of the second hardware demo, because `fmincon` cannot find an optimal solution quickly enough given the high number of constraints produced by the surrounding obstacles. The robot requires human assistance to proceed, whereas it is able to navigate the entire scene autonomously when planning with RTD*/PCBA (Figure 4.1). See the video.

4.6.4 Discussion

We have demonstrated that RTD* is capable of *dynamic feasibility*, *optimality*, and *liveness* for online trajectory optimization on robot hardware. RTD* is able to find an optimal solution, if it exists, at every receding-horizon planning iteration, leading to it consistently navigating random scenarios without collisions. Furthermore, RTD* outperforms RTD at the same navigation and obstacle-avoidance tasks. This performance increase is due to PCBA's ability to find solutions more quickly than `fmincon` on problems with hundreds of constraints. To the best of our knowledge, this is the first time any Bernstein algorithm has been demonstrated as practical for a real-time mobile robotics application.

4.7 Conclusion

Mobile robots typically use receding-horizon planning to move through the world. Plans should be dynamically feasible and optimal, but also need to be generated quickly, otherwise the robot may stop frequently and never complete a user-specified task. We propose and implement a Parallel Constrained Bernstein Algorithm (PCBA) to rapidly generate provably dynamically-feasible and optimal plans. PCBA takes advantage of the polynomial optimization problem (POP) structure used by the existing Reachability-based Trajectory Design (RTD) method; but, where RTD uses a gradient-based nonlinear solver that cannot guarantee optimality or real-time performance, we use PCBA. The resulting algorithm, RTD*, is shown to outperform RTD on a variety of hardware demonstrations. To the best of our knowledge, this is the first time a Bernstein Algorithm has been used for real-time mobile robotics. Furthermore, PCBA outperforms the Bounded Sums-of-Squares (BSOS) and `fmincon` solvers on a variety of benchmark POPs. For future work, we plan to explore non-polynomial optimization problems and to improve the proposed time and space complexity bounds of PCBA; our goal is to apply RTD* and PCBA to more types of robots.

		PCBA					BSOS [LTY17]			`fmincon` [Mat19]	
	l	ϵ	error	time [s]	iterations	memory	(d,k)	error	time [s]	error	time [s]
P1	2	7.0000e-07	7.9002e-07	0.0141	28	23 kB	(3,3)	-0.0068	1.2264	1.0880	0.0083
P2	2	0.1369	0.0731	0.0131	26	60 kB	(2,2)	-3.4994e-04	0.7671	-0.4447	0.0180
P3	2	2.0000e-06	1.9879e-06	0.0128	26	57 kB	(2,2)	-3.2747	0.5065	-3.0000	0.0220
P4	3	1.7000e-06	6.5565e-07	0.0204	24	416 kB	(4,4)	-0.9455	6.6546	3.2000e-05	0.0130
P5	3	1.0000e-06	0	0.0312	28	3 MB	(2,2)	-4.4858e-07	0.8462	7.9985e-06	0.0062
P6	4	4.2677e-04	1.8685e-04	0.0315	23	2 MB	(3,3)	-36.6179	6.2434	0.0040	0.0065
P7	4	5.0000e-07	2.5280e-07	0.0374	26	282 kB	(1,1)	-1.0899	0.1839	2.4002e-07	0.0139
P8	4	1.3445e-04	6.7803e-05	0.0440	22	6 MB	(2,2)	-3.2521	1.8295	9.9989e-04	0.0169

Table 4.1: Results for PCBA, BSOS, and `fmincon` on eight benchmark problems with 2, 3, and 4 dimensional (column l) decision variables (see the Appendix for more details). The error columns report each solver's result minus the true global minimum. For all three solvers, the reported error and time to find a solution are the median over 50 trials (with random initial guesses for `fmincon`). For PCBA, we also report the optimality tolerance ϵ (as in (4.28)), number of iterations to convergence, and peak GPU memory used. Note that, on P1 and P5, PCBA stopped at the maximum number of iterations (28).

CHAPTER 5

Fast, Safe Control Synthesis for a 3D Bipedal Robot

5.1 Introduction

In the prior two chapters, an MPC framework is constructed to perform control synthesis on bipedal robots in an online fashion, and a parallel computing scheme is developed to solve polynomial optimization problems with provable guarantees on optimality, computation time, and memory usage. However, the method developed in each chapter has shortcomings. The MPC framework proposed in Chapter 3 is only applicable to planar robots, and is unable to guarantee safety when a feasible solution to the optimization is not found. The application of PCBA as proposed in Chapter 4 requires expressing the objective and constraint functions in polynomial form. This chapter proposes to combine and extend the work from the prior two chapters to construct an online MPC framework capable of keeping a 3D bipedal robot safe for all time even when the optimization problem is infeasible.

In each planning iteration of the proposed MPC framework, we generate a plan made up of two phases. The first phase of the plan moves the robot forward while minimizing a user-specified cost; the second phase brings the robot to a fixed configuration where it balances on two feet. However, the second phase is only ever applied if a feasible new control input is not found. By forcing the robot to reach a pre-specified set of "safe" states at the end of the second phase, any safety guarantees that are previously established over finite time horizon can be extended to all future times.

The contributions of this chapter are three-fold: first, Section 5.3 develops a numerical simulation scheme for hybrid systems that generates numerical approximations to hybrid trajectories, and proposes to bound the approximation error by using interval-arithmetic techniques; second, Section 5.4 formulates sufficient conditions for safety using the constructed error bounds; third, Section 5.5 describes a parallel algorithm that solves each optimization up to a user-specified tolerance, and proves persistent feasibility of the method.

The remainder of this chapter is organized as follows: Section 5.2 describes a 3D bipedal robot

and its environment, and gives a formal definition of safety; Section 5.6 provides implementation details; Section 5.7 demonstrates the performance of the proposed approach on a walking example;

5.2 Dynamic Models and Environments

We start by introducing a bipedal robot and its environment. Specifically, Section 5.2.1 introduces necessary notations to define a dynamical model of Cassie; Section 5.2.2 defines a two-phase parameterized control input that is applied in each planning horizon; and Section 5.2.3 defines the environment and provides a formal definition of safety.

5.2.1 Dynamical Model of a Biped

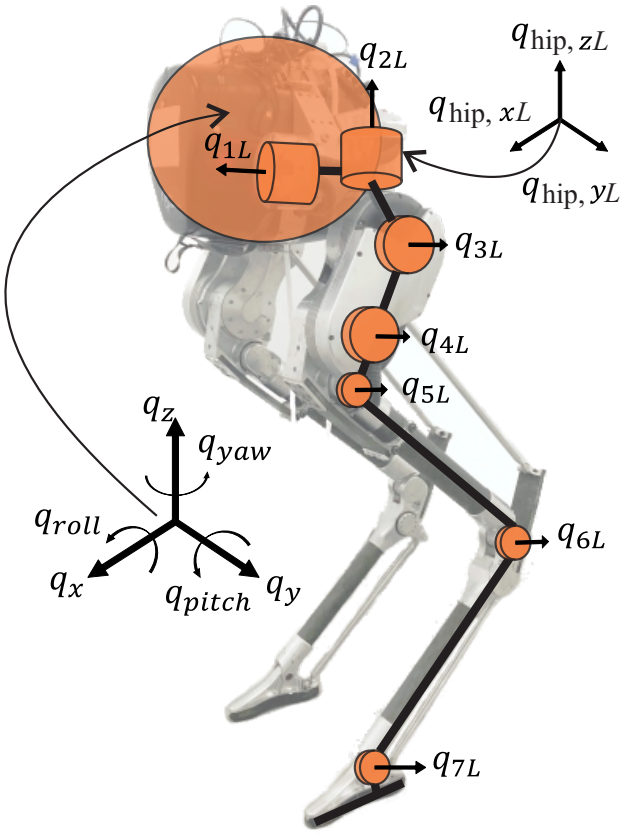


Figure 5.1: A kinematic model of Cassie where the joints on the right limb are omitted for ease of understanding. The joints $q_{1L}, \ldots, q_{4L}$, $q_{1R}, \ldots, q_{4R}$, q_{7L}, q_{7R} are actuated and the corresponding control inputs are labeled $u_{1L}, \ldots, u_{4L}$, $u_{1R}, \ldots, u_{4R}$, u_{5L}, u_{5R}, respectively.

We focus on designing trajectories online for Cassie while it walks on a flat ground plane that we denote by $\mathcal{G} \subset \mathbb{R}^2$. We define the *flow map* of Cassie as $\phi : [t_0, \infty) \times \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ where

$\phi(t, y_0, u(\cdot))$ denotes the solution to Cassie's dynamics under the control input $u(t) \in \mathcal{U}$ at time $t \in [t_0, \infty)$ starting from an initial condition $y_0 \in \mathcal{X}$. We refer to $\mathcal{T} = [t_0, \tau_f]$ as the *planning time horizon* where $\tau_f < \infty$. When the control input and initial condition are clear in context, we refer to the solution at time t by $\phi(t)$. Note that the dynamics of Cassie are typically described using a hybrid system [GHD+19] and $\phi(\cdot, y_0, u(\cdot))$ must be understood as a hybrid system trajectory (c.f. Fig. 2.1). Detailed discussion about hybrid trajectory of Cassie and its discrete approximation is postponed until Section 5.3.2 . Since planning is done in a receding-horizon fashion, without loss of generality (WLOG), let each planned trajectory (i.e., each planning iteration) begin at $t_0 = 0$.

The state space of $\mathcal{X}$ is composed of variables corresponding to the configuration variables of Cassie and their velocities. The configuration variables of Cassie are depicted in Figure 5.1. In particular, the variables q_x, q_y, and q_z are the Cartesian position of the pelvis, and q_{yaw}, q_{pitch}, and q_{roll} are the intrinsic Euler Angles of the pelvis. The coordinates (q_{1L}, q_{1R}), (q_{2L}, q_{2R}), (q_{3L}, q_{3R}) are the hip roll, hip yaw, and hip pitch angles for left and right limbs, respectively; (q_{4L}, q_{4R}) are the left and right knee pitch angles; (q_{5L}, q_{5R}) are the left and right shin pitch angles; (q_{6L}, q_{6R}) are the tarsus pitch angles; (q_{7L}, q_{7R}) are the left and right toe pitch angles. In addition the variables $(q_{\text{hip},xL}, q_{\text{hip},yL}, q_{\text{hip},zL})$ and $(q_{\text{hip},xR}, q_{\text{hip},yR}, q_{\text{hip},zR})$ are the Cartesian positions of the left and right hips, respectively. From now on, we refer to Cassie's toe as its "foot" since it behaves like the foot of other bipedal robots. We also refer to a particular configuration coordinate of the flow map by using that coordinate as a subscript (e.g., ϕ_{q_x} denotes the q_x-coordinate of the flow map ϕ, whereas $\phi_{\dot{q}_x}$ denotes the velocity coordinate of q_x in ϕ).

Typically one refers to Cassie's *footprint* as the area occupied by Cassie when projected onto the ground, $\mathcal{G}$. However, this definition prohibits real-time computation because the shape of such projection can evolve as Cassie moves its limbs. To conservatively represent such a projection, we make the following assumption:

Assumption 47. *Cassie's footprint in $\mathcal{G}$ can be bounded by a circle centered at (q_x, q_y) with constant radius $R \geq 0$, where q_x and q_y are the 2D Cartesian position of Cassie's pelvis on $\mathcal{G}$.*

5.2.2 Control Input

Rather than try to optimize over arbitrary control inputs during online optimization, we instead focus on optimizing over a recently developed family of parameterized controllers for Cassie [GHD+19] . These control inputs are parameterized by a pair of *control parameters*, $P := (P_1, P_2) \in \mathcal{P}$ with $\mathcal{P} := [0, 1] \times [-0.1, 0.1]$ where P_1 corresponds to a forward speed in meters per second and P_2 corresponds to a yaw rate in radians per second. In particular, each parameter is associated with a control input, $(u_{1L,P}, \ldots, u_{5L,P}, u_{1R,P}, \ldots, u_{5R,P}) : [0, \tau_1) \to \mathcal{U}$, whose components corresponds to motor torques at the joints $q_{1L}, \ldots, q_{4L}, q_{7L}, q_{1R}, \ldots, q_{4R}, q_{7R}$, respectively,

as a function in time for some pre-specified $\tau_1 \in (0, \tau_f)$. Note that we refer to the concatenation of the controller at each joint by u_P for each $P \in \mathcal{P}$. These control inputs try to get the pelvis to track a forward speed P_1 and yaw rate P_2. In this chapter, we append each control input with a second phase $u_0 : [\tau_1, +\infty) \to \mathcal{U}$ where the robot decelerates to zero pelvis velocity and then balances on two feet. This second phase of the control input *attempts* to bring Cassie to a fixed double-stance configuration with zero joint velocity. The specific details of this controller u_0 are included in Appendix G. We denote the concatenated controller as $\hat{u}_P : [0, +\infty) \to \mathcal{U}$, where

$$\hat{u}_P(t) := \begin{cases} u_P(t) & \text{if } 0 \leq t < \tau_1 \\ u_0(t) & \text{if } t \geq \tau_1 \end{cases} \tag{5.1}$$

Note, for the remainder of this chapter, when it is clear in context, we abuse notation and denote this two phase controller by the control parameter applied during the first phase.

5.2.3 Environment and Safety Criterion

This paper assumes that Cassie is tasked with reaching a user-specified goal location while walking without falling over on the ground and avoiding randomly placed obstacles. To walk safely, first Cassie must not fall, which is defined formally as follows:

Definition 48. *Cassie* falls *if any portion of it other than its feet touch the ground.*

However, checking the above condition directly requires bounding the position of all joints over the planning horizon $\mathcal{T}$. Instead, we rely on the following sufficient condition to ensure that Cassie does not fall:

Theorem 49. *If the left and right hip height states of Cassie is above 0.75 [m], then Cassie has not yet fallen.*

Proof. See Appendix F. □

As a result of Theorem 49, we only need to make sure Cassie's left and right hip heights do not fall below the threshold 0.75 [m] to prevent Cassie from falling before τ_f.

Next, we formalize the notion of obstacles in the environment, which are treated as subsets of $\mathcal{G}$, and as:

Definition 50. *At any time $t \geq 0$, an* obstacle *is a static closed polygon in $\mathcal{G}$ that Cassie's footprint cannot intersect without being in* collision. *Denote the l^{th} obstacle by $O_l \subset \mathcal{G}$ for each $l \in \{1, \cdots, N_{obs}\}$.*

Note that this definition treats obstacles as if they have an infinite height (i.e. Cassie cannot step over an obstacle without being in collision). Once again, checking the above condition directly requires bounding the position of all joints over the planning horizon $\mathcal{T}$. To simplify enforcement of this requirement, we buffer each obstacle by the footprint radius R, and require Cassie's pelvis position to stay outside such buffered obstacles. To do this, define a function $\texttt{buffer} : O_l \mapsto \hat{O}_l$ that maps each obstacle O_l to its corresponding obstacle, $\hat{O}_l$, that is buffered by the footprint radius R. A detailed definition of such function can be found in [KVB+18, Definition 47]. We then obtain a sufficient condition that ensures Cassie is not in collision with any obstacles:

Theorem 51. *Let (q_x, q_y) be the 2D Cartesian position of Cassie's pelvis. If $\forall l \in \{1, \dots, N_{obs}\}$, $(q_x, q_y) \notin \texttt{buffer}(O_l)$, then Cassie is not in collision with any obstacles.*

Using these pair of definitions, we can define safety as avoiding falling and collisions, which is defined formally as follows:

Definition 52. *Cassie is* safe *if it does not fall or collide with any obstacles for all $t \geq 0$.*

5.2.4 Planning, Sensing, and Persistent Feasibility

To ensure that Cassie is able to walk safely at run-time, we begin with an assumption about the amount of time that planning is allotted.

Assumption 53. *During each planning iteration, Cassie has $\tau_{plan} > 0$ amount of time to pick a new input. If the robot cannot find a new input in a planning iteration, it continues applying the parameterized control input it computed during the previous iteration.*

At the beginning of each planning iteration, time is reset to $t = 0$. During each planning iteration, we create a new desired trajectory for the next planning iteration by choosing $P \in P$ while applying the previously-computed parameterized input.

Next, we describe how the planning loop constructs controllers for Cassie that are ensured to keep it persistently safe. During each online trajectory design iteration, we would like to find a control parameter $P \in \mathcal{P}$ such that Cassie is safe for all future times if u_P is applied over $[0, \tau_1)$ and u_0 is applied over $[\tau_1, +\infty)$. In particular, each iteration is given τ_{plan} amount of time to construct new control input. During this time, typically the safe control input constructed in the previous planning iteration is applied. If a new control input that can be applied safely can be found in τ_{plan}, then it is applied to Cassie beginning at $t = \tau_1$ and the online trajectory design process is repeated. If no new control input can be found, then the u_0 whose safe behavior was verified in the previous iteration is applied.

Unfortunately, verifying that u_0 can be applied safely over an infinite time horizon is computationally taxing. Instead, we verify during online optimization that u_0 over some time horizon, $[\tau_1, \tau_f]$ is able to drive the states of Cassie to a pre-specified set $\mathcal{X}_f$. This set which we formally define in the implementation section (c.f. Section 5.6) corresponds to a set of states in double-stance where the velocity of each joint is small and Cassie's center of mass is within its base of support. Once Cassie arrives in this set, we then make the following assumption about u_0 and $\mathcal{X}_f$ to ensure that it can remain in a safe, balanced state indefinitely:

Assumption 54. *There exists a compact set $\mathcal{X}_f \subset \mathcal{X}$ of double stance configurations, such that if u_0 is applied to Cassie with current state contained in $\mathcal{X}_f$, Cassie will keep balanced in place indefinitely.*

To make sure the planning framework is able to accommodate newly observed obstacles as Cassie moves through the environment, we make several assumptions about the sensing model. Using these assumptions we can prove persistent feasibility of the proposed planning framework. We begin by making an assumption about how the obstacles are perceived:

Assumption 55. *The robot has a finite sensor horizon D_{sense}, which is a radius around the pelvis position in $\mathcal{G}$ within which obstacles are perceived. During operation, new obstacles appear from outside the sensor horizon and are sensed as soon as they appear. In particular, occluded regions are treated as obstacles.*

We may now formally prove safety and persistent feasibility of the proposed planning framework using similar arguments from [KVB+18, Theorem 35]:

Theorem 56. *Let τ_{plan} be the overall computation time for online planning. Let Assumption 55 be satisfied, and suppose $\tau_{plan} \leq \tau_1$. Furthermore, suppose the sensor horizon D_{sense} satisfies*

$$D_{sense} \geq (\tau_{plan} + \tau_1) \cdot v_{\max} \tag{5.2}$$

where $v_{\max}$ Cassie's maximum speed. Suppose that their exists a safe controller u_P which can be applied for the next τ_1 seconds, then the planning framework is persistently feasible. That is, in the following planning iterations the robot can either find a safe plan every τ_1 seconds, or can follow the controller u_0 to come to a fixed position where it balances on two feet indefinitely.

In practice, to ensure real-time operation, the condition $\tau_{\text{plan}} \leq \tau_1$ is enforced by placing a hard time limit of τ_1 on the planning time, after which any ongoing computation is terminated.

5.3 Error Bound for Hybrid System Simulations

To establish safety guarantees of Cassie, one needs to verify Theorem 49 and 51 are satisfied along a hybrid execution $\phi(\cdot, y_0, u(\cdot))$. Although such ϕ is not directly accessible to us, we may find approximations to it through numerical simulation. This section defines hybrid systems and presents a numerical technique to simulate hybrid executions with provable error bounds. We start by introducing necessary notations and preliminaries for interval arithmetic. A longer introduction to interval arithmetic can be found in [Moo66].

5.3.1 Preliminaries

Let $\mathbb{R}^d_+$ be the set of vectors in d-dimensional real Euclidean space whose components are positive, and let $\mathbb{R}^d_-$ be the set of d-dimensional real vectors with negative components. Define $\mathrm{BV}([0,+\infty), \mathcal{U})$ to be the set of all functions of bounded variation from $[0,+\infty)$ to $\mathcal{U}$.

Given two intervals $X = [\underline{X}, \overline{X}]$ and $Y = [\underline{Y}, \overline{Y}]$ where $\underline{X} \leq \overline{X}$ and $\underline{Y} \leq \overline{Y}$, define their product by

$$X \cdot Y := [\min\{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}\}, \max\{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{XY}\}] \tag{5.3}$$

Notice in particular for any $x \in X$ and $y \in Y$, $xy \in X \cdot Y$. Given any positive real number $a > 0$, let $[a] := [-a, a] \subset \mathbb{R}$ be a closed real interval that contains 0. For any vector $b \in \mathbb{R}^d_+$, let $[b] := [b_1] \times \cdots \times [b_d] \subset \mathbb{R}^d$ be a closed box in d-dimensional Euclidean space, where b_j stands for the j^{th} component of b. Given any $c \in \mathbb{R}^d$, let $c + [b]$ be defined as

$$c + [b] = [c_1 - b_1, c_1 + b_1] \times \cdots \times [c_d - b_d, c_d + b_d] \subset \mathbb{R}^d \tag{5.4}$$

Such objects $[b]$ and $c+[b]$ are essentially vectors with interval components, therefore often referred to as *interval vectors*. In a similar fashion, given any matrix $A \in \mathbb{R}^{l\times d}_+$ with positive components, let $[A]$ be a matrix with interval components $[A_{ij}]$ where A_{ij} stands for the $(i,j)^{\text{th}}$ component of A. Such a matrix $[A]$ is referred to as *interval matrix*. The interval-arithmetic operations involving interval vectors and matrices are defined by using the same formulas as in the scalar case, except that scalars are replaced by intervals [Moo66]. For example, if a $d \times d$ interval matrix $[A]$ has components $[A_{ij}]$ for all $1 \leq i, j \leq d$, and a $d \times 1$ interval vector $[b]$ has components $[b_k]$ for all $1 \leq k \leq d$, then the i^{th} components of their product $[A] \cdot [b]$ is given by $\sum_{k=1}^d [A_{ik}] \cdot [b_k] = [\sum_{k=1}^d A_{ik} b_k]$.

Let $f : \mathbb{R}^d \to \mathbb{R}^l$ be a continuous function whose analytical representation contains only a finite number of constants, variables, arithmetic operations, and standard functions (sin, cos, log,

exp, etc.). The *interval-arithmetic evaluation* of f on $[b]$, which we denote by $f([b])$, is obtained by replacing each occurrence of a real variable with the corresponding interval and performing interval-arithmetic operations. In particular, such an interval-arithmetic evaluation provides an *outer* approximation of f over the interval vector [Moo66, Theorem 5.1], that is, $\{f(v) \mid v \in [b]\} \subseteq f([b])$.

One useful result that we use extensively in the subsequent subsections is an extension of the mean value theorem to the interval-arithmetic case formally stated as follows:

Theorem 57. *Let $a, b \in \mathbb{R}^d$, $\epsilon \in \mathbb{R}^d_+$ be such that $a \in b + [\epsilon]$. Suppose $f : b + [\epsilon] \to \mathbb{R}^l$ is differentiable and let $J_f(c)$ be its Jacobian matrix at $c \in b + [\epsilon]$. Then, there exists $\theta \in (0, 1)$ such that*

$$f(a) - f(b) = J_f((1-\theta)a + \theta b) \cdot (a - b) \tag{5.5}$$

Furthermore, suppose there exists a matrix $B \in \mathbb{R}^{l \times d}_+$ such that $|J_f(c)| \leq B$ for each $c \in b + [\epsilon]$, where $|\cdot|$ is meant component-wise. Then,

$$f(a) - f(b) \in [B\epsilon]. \tag{5.6}$$

Proof. (5.5) follows from the mean value theorem of multivariate functions by noticing $a, b \in b + [\epsilon]$, where the box $b + [\epsilon]$ is closed and convex. To prove (5.6), notice $J_f((1-\theta)a + \theta b) \in [B]$ and $a - b \in [\epsilon]$. □

For any $v \in \mathbb{R}^d$, let $|v| := (|v_1|, \ldots, |v_d|)$ be the component-wise absolute value of v. Similarly, let the inequalities $\leq, <, >, \geq$ between vectors be defined component-wise.

5.3.2 Hybrid System and Its Numerical Simulation

Recall from Definition 1 that a hybrid system is a tuple $\mathcal{H} = (\mathcal{I}, \mathcal{E}, \mathcal{X}, \mathcal{U}, \mathcal{F}, \mathcal{S}, \mathcal{R})$, where:

- $\mathcal{I}$ is a finite set indexing the discrete states of $\mathcal{H}$;
- $\mathcal{E} \subset \mathcal{I} \times \mathcal{I}$ is a set of edges, forming a directed graph structure over $\mathcal{I}$;
- $\mathcal{X} = \coprod_{i \in \mathcal{I}} X_i$ is a disjoint union of domains, where each X_i is a compact subset of $\mathbb{R}^{n_i}$ and $n_i \in \mathbb{N}$;
- $\mathcal{U}$ is a compact subset of $\mathbb{R}^m$ that describes the range of control inputs, where $m \in \mathbb{N}$;
- $\mathcal{F} = \{F_i\}_{i \in \mathcal{I}}$ is the set of vector fields, where each $F_i : \mathbb{R} \times X_i \times \mathcal{U} \to \mathbb{R}^{n_i}$ is a Lipschitz continuous vector field defining the dynamics of the system on X_i;

- $\mathcal{S} = \coprod_{e\in\mathcal{E}} S_e$ is a disjoint union of guards, where each $S_{(i,i')} \subset \partial X_i$ is a compact, co-dimensional 1 guard defining a state-dependent transition from X_i to $X_{i'}$; and,
- $\mathcal{R} = \{R_e\}_{e\in\mathcal{E}}$ is a set of continuous reset maps, where each map $R_{(i,i')} \colon S_{(i,i')} \to X_{i'}$ defines the transition from guard $S_{(i,i')}$ to $X_{i'}$.

To avoid any ambiguity during transitions, we make the same set of assumptions as in Section 2.2:

Assumption 58. *Guards do not intersect with themselves or the images of reset maps. The controlled vector fields in each mode have a nonzero normal component on the guard for all control inputs in $\mathcal{U}$.*

To guarantee the existence and uniqueness of solutions to ordinary differential equations in individual domains, we assume the following regularity of the controller:

Assumption 59. *The controller u that is applied to a hybrid system is of bounded variation. That is, $u \in BV([0,+\infty),\mathcal{U})$.*

Notice the vector field F_i for each $i \in \mathcal{I}$ is Lipschitz continuous by definition. One may show the ordinary differential equation associated with F_i in each domain has a unique solution:

Theorem 60 ([BGV$^+$15, Lemma 10])**.** *For each $i \in \mathcal{I}$, $x_0 \in X_i$, and $u \in BV([0,+\infty),\mathcal{U})$, there exists an interval $I \subset [0,+\infty)$ such that the following differential equation has a unique solution:*

$$\dot{\gamma}(t) = F_i(t,\gamma(t),u(t)), \quad t \in I, \quad \gamma(0) = x_0. \tag{5.7}$$

Moreover, γ is absolutely continuous.

With the above assumptions, given a controller $u \in \mathrm{BV}([0,+\infty),\mathcal{U})$ and an initial condition $y_0 \in \mathcal{X}$, we define a hybrid trajectory, $\phi(\cdot, y_0, u(\cdot))$, of the hybrid system $\mathcal{H}$ as in Algorithm 2.1. WLOG, we assume such trajectory does not start from inside a guard:

Assumption 61. *The initial condition of a hybrid trajectory is not in any guard.*

Such hybrid trajectories, however, are usually not directly accessible to us due to lack of numerical method that finds *exact* solution to ordinary differential equation, and the inability to perform accurate event detection. Instead, we seek numerical approximations to them and prove bounds on the approximation error.

To make sure such hybrid trajectories are numerically approximable, we impose a few regularity assumptions on the vector fields as well as hybrid trajectories themselves:

Assumption 62. *For each $i \in \mathcal{I}$, the vector field $F_i : \mathbb{R} \times X_i \times \mathcal{U} \to \mathbb{R}^{n_i}$ is differentiable with respect to the state variable (i.e., second argument of F_i).*

Assumption 63. *For each $y_0 \in \mathcal{X}$ and $u \in BV([0, +\infty), \mathcal{U})$, the hybrid trajectory $\phi(\cdot, y_0, u(\cdot))$ of $\mathcal{H}$ is orbitally stable. The hybrid trajectory has a continuous second derivative at each instance in time. Such $\phi(\cdot, y_0, u(\cdot))$ either has a finite number of discrete transitions or is a Zeno trajectory that accumulates.*

One may refer to [BGV+15, Definition 23-25] for definitions of orbital stability and accumulation points of Zeno trajectories.

Next, to construct a numerical simulation scheme of hybrid system that does not rely on exact computation of the time instant at which a hybrid trajectory intersects a guard, we define a *relaxation* of a hybrid system by introducing slackness in the guards and reset maps. This requires additional regularity of the guards and reset maps:

Assumption 64. *Each guard of the hybrid system $\mathcal{H}$ is a 0-sublevel set of a differentiable function. That is, for each $(i, j) \in \mathcal{E}$, there exists a differentiable function $c_{(i,j)} : \mathbb{R}^{n_i} \to \mathbb{R}$ such that*

$$S_{(i,j)} = \{x \in X_i \mid c_{(i,j)}(x) = 0\}. \tag{5.8}$$

Moreover, let $c_{(i,j)}(x) \geq 0$ for each $x \in X_i$. For each $(i, j) \in \mathcal{E}$, the reset map $R_{(i,j)}$ is differentiable and admits a differentiable extension on $\mathbb{R}^{n_i}$. That is, there exists a differentiable function $\hat{R}_{(i,j)} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_j}$ such that

$$\hat{R}_{(i,j)}(x) = R_{(i,j)}(x) \quad \forall x \in S_{(i,j)} \tag{5.9}$$

We now formally define relaxed guards, relaxed reset maps, and relaxed domains as follows:

Definition 65. *Let $\mathcal{H} = (\mathcal{I}, \mathcal{E}, \mathcal{X}, \mathcal{U}, \mathcal{F}, \mathcal{S}, \mathcal{R})$ be a hybrid system. Let $\delta > 0$. For each $(i, j) \in \mathcal{E}$, define a δ-relaxation of the guard $S_{(i,j)}$ as*

$$S^{\delta}_{(i,j)} := \{x \in \mathbb{R}^{n_i} \mid -\delta \leq c_{(i,j)}(x) \leq 0\} \subset S_{(i,j)} \tag{5.10}$$

The δ-relaxation of the reset map $R_{(i,j)}$, denoted as $R^{\delta}_{(i,j)} : S^{\delta}_{(i,j)} \to \mathbb{R}^{n_j}$, is a differentiable extension of $R_{(i,j)}$ onto $S^{\delta}_{(i,j)}$ such that

$$R^{\delta}_{(i,j)}(x) = R_{(i,j)}(x) \quad \forall x \in S_{(i,j)}. \tag{5.11}$$

For each $i \in \mathcal{I}$, define the δ-relaxation of the domain X_i, denoted as X^{δ}_i, to also contain the

relaxed guards:

$$X_i^\delta := X_i \cup_{(i,j)\in\mathcal{E}} S_{(i,j)}^\delta \tag{5.12}$$

The definitions of relaxed guards, relaxed reset maps, and relaxed domains allow us to construct a relaxation of the hybrid system that is closely related to the evolution of numerical simulations:

Definition 66. *Let $\mathcal{H} = (\mathcal{I}, \mathcal{E}, \mathcal{X}, \mathcal{U}, \mathcal{F}, \mathcal{S}, \mathcal{R})$ be a hybrid system. A δ-relaxation of $\mathcal{H}$ is a tuple*

$$\mathcal{H}^\delta = (\mathcal{I}, \mathcal{E}, \mathcal{X}^\delta, \mathcal{U}, \mathcal{F}, \mathcal{S}^\delta, \mathcal{R}^\delta) \tag{5.13}$$

where:

- $\mathcal{X}^\delta = \coprod_{i\in\mathcal{I}} X_i^\delta$ *is a disjoint union of δ-relaxations of domains;*
- $\mathcal{S}^\delta = \coprod_{e\in\mathcal{E}} S_e^\delta$ *is a disjoint union of δ-relaxations of the guards;*
- $\mathcal{R}^\delta = \{R_e^\delta\}_{e\in\mathcal{E}}$ *is a set of δ-relaxations of the reset maps.*

Notice the set of vector fields $\mathcal{F}$ is not defined on all points in the relaxed guards $\mathcal{S}^\delta$. Such definition is intentional because the numerical simulations, as we define next, does not require evaluation of the vector fields on S_e^δ.

Finally, we define the numerical simulation of a relaxed hybrid system, which is constructed as an extension of any existing ODE numerical integration algorithm. Let $\Psi_i^h : \mathbb{R} \times X_i \times \mathcal{U} \to \mathbb{R}^{n_i}$ be a numerical integrator in domain X_i with step size $h > 0$. Let $[0, T]$ be the time domain over which the numerical simulation is computed, and let $N := \lceil T/h \rceil$ be the total number of steps in the numerical simulation. For a sufficiently large $\delta > 0$ such that the range of Φ_i^h is contained in X_i^δ for each $i \in \mathcal{I}$, a *numerical simulation* of a relaxed hybrid system $\mathcal{H}^\delta$ under a set of numerical simulators $\{\Psi_i^h\}_{i\in\mathcal{I}}$ consists of a time sequence $\{t_k\}_{1\le k\le N}$ and a state sequence $\{y_k\}_{1\le k\le N}$ constructed via Algorithm. 6.

In the remainder of this section, we let the numerical integrator Ψ_i^h be obtained from the Euler method with a *fixed* step size $h > 0$. That is,

$$\Psi_i^h(t, x, u(t)) := x + h \cdot F_i(t, x, u(t)) \tag{5.14}$$

for all $t \in [0, Nh]$, $i \in \mathcal{I}$, and $x \in X_i$. For convenience, given any $y_k = (x_k, i) \in \mathcal{X}$, let

$$\Phi_k^i(x_k) := \Psi_i^h(t_k, x_k, u(t_k)). \tag{5.15}$$

Algorithm 6 The procedure to define a numerical simulation of the δ-relaxation of a hybrid system $\mathcal{H}^\delta$.

Require: $h > 0$, $N > 0$, $i \in \mathcal{I}$, $(x_0, i) \in \mathcal{X}$, and $u \in \mathrm{BV}([0, +\infty), \mathcal{U})$

1: Set $t_0 \leftarrow 0$, $y_0 \leftarrow (x_0, i)$
2: **loop**
3: Set $t_{k+1} \leftarrow t_k + h$.
4: Set $x_{k+1} \leftarrow \Psi_i^h(t_k, x_k, u(t_k))$.
5: Set $y_{k+1} \leftarrow (x_{k+1}, i)$.
6: **if** $\exists (i,j) \in \mathcal{E}$ such that $x_{k+1} \in S^\delta_{(i,j)}$ **then**
7: Set $t_{k+2} \leftarrow t_{k+1}$.
8: Set $x_{k+2} \leftarrow R^\delta_{(i,j)}(x_{k+1})$.
9: Set $y_{k+2} \leftarrow (x_{k+2}, i)$
10: Set $k \leftarrow k + 2$.
11: **else**
12: Set $k \leftarrow k + 1$.
13: **end if**
14: **end loop**

5.3.3 Bounding Errors in Numerical Simulation

Given a hybrid system $\mathcal{H}$, an initial condition $y_0 = (x_0, i) \in \mathcal{X}$, and a control $u \in \mathrm{BV}([0, +\infty), \mathcal{U})$, it is our interest to understand how a hybrid trajectory $\phi(\cdot, y_0, u(\cdot))$ of $\mathcal{H}$ differs from a numerical simulation $\{t_n, y_n\}_{1 \le n \le N}$ of $\mathcal{H}^\delta$. To simplify notation, let $y^i(\cdot)$ denote the projection of a hybrid trajectory $\phi(\cdot, y_0, u(\cdot))$ onto X_i for each $i \in \mathcal{I}$, and let x_k be the projection of y_k onto the continuous state space such that $y_k = (x_k, i)$ for some $i \in \mathcal{I}$. Notice in Lines 7-14 of Algorithm. 6 that the pair (t_{k+1}, y_{k+2}) is updated differently depending on whether y_{k+1} reaches the relaxed guard. In a similar vein we shall prove an error bound in two separate cases.

5.3.3.1 Error bound when $y_k \notin \mathcal{S}^\delta$

Let the discrete state component of y_k be $i \in \mathcal{I}$. It then follows from Line 4 in Algorithm. 6 that $x_{k+1} = \Phi_k^i(x_k)$. We refer to the difference between $y^i(t_k)$ and x_k as the *global error* at time t_k. In particular, suppose we are given a bound $\epsilon_k \in \mathbb{R}_+^{n_i}$ of the global error at t_k such that $\epsilon_k \ge |y^i(t_k) - x_k|$, we are interested in finding an $\epsilon_{k+1} \in \mathbb{R}_+^{n_i}$ such that $\epsilon_{k+1} \ge |y^i(t_{k+1}) - x_{k+1}|$, where the inequality is meant component-wise.

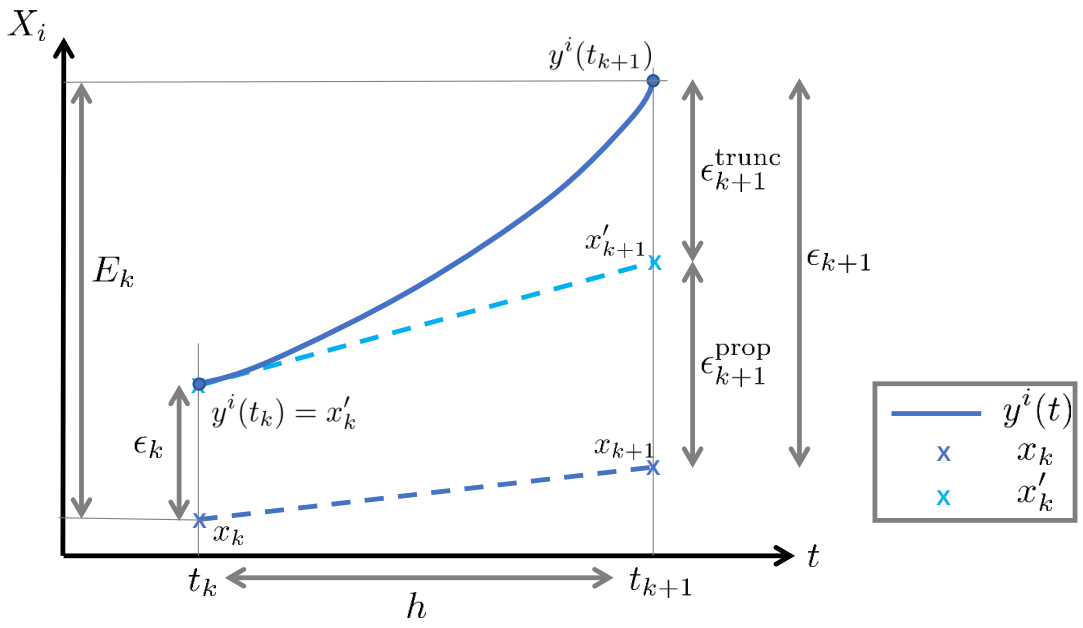


Figure 5.2: An illustration of global errors when $y_k \notin \mathcal{S}^\delta$

To understand how ϵ_{k+1} can be obtained from ϵ_k, consider the scenario shown in Fig. 5.2. Let x'_{k+1} be the point we would have computed using Euler's method if the global error at time t_k was zero, then the distance between x'_{k+1} and $y^i(t_{k+1})$ represents the amount of error introduced by one step of Euler method. In addition, the error at t_k is propagated through the dynamical system, which is represented as the difference between x'_{k+1} and x_{k+1}. So, the global error at time t_k can be decomposed into 2 parts: *truncation error* due to describing the solution to the differential equation by using the Euler method, and *propagation error* due to the inexactness of the initial condition at each subsequent discrete time step since it is generated by using the Euler method. We now explain how each error is computed.

Let $x'_k := y^i(t_k)$ and $x'_{k+1} := \Phi_k^i(x'_k)$, then the *truncation error* $\epsilon_{k+1}^{\text{trunc}} := y^i(t_{k+1}) - x'_{k+1}$ characterizes the difference between $y^i(t_{k+1})$ and x'_{k+1}. It can be shown [Atk08, Theorem 6.3] that

Theorem 67. *There exists some $\xi \in [t_k, t_{k+1}]$ such that*

$$\epsilon_{k+1}^{trunc} = \frac{h^2}{2}\frac{d^2(y^i)}{dt^2}(\xi) \tag{5.16}$$

where y^i is the projection of y onto X_i.

It can be seen from Theorem 67 that bounding $\epsilon_{n+1}^{\text{trunc}}$ requires a priori bounds of second derivative of the trajectory in each domain. In Section 5.6 we describe how we construct this bound; however, for now we assume such a bound is given:

Assumption 68. *There exists functions $B_1^i : [0,T] \to \mathbb{R}_+^{n_i}$ such that for all $i \in \mathcal{I}$, $1 \le l \le N$, and $\xi \in [t_l, t_{l+1}]$,*

$$\left| \frac{d^2(y^i)}{dt^2}(\xi) \right| \le B_1^i(t_l) \tag{5.17}$$

whenever the projection of a trajectory y onto the continuous state space, denoted as y^i, is well defined. The absolute value $|\cdot|$ and inequality are meant component-wise.

It then follows directly from Theorem 67 and Assumption 68 that

$$\left|\epsilon_{k+1}^{\text{trunc}}\right| \le \frac{h^2}{2} B_1^i(t_k) \tag{5.18}$$

Next, notice x_k and x_k' may not be identical due to the error from previous numerical simulation steps, thus the propagation of such separation from the k^{th} step into the $(k+1)^{\text{th}}$ step through the numerical integrator (5.14) must be considered. The *propagation error*, $\epsilon_{k+1}^{\text{prop}}$, is defined as the difference between x_{k+1}' and x_{k+1}, namely

$$\epsilon_{k+1}^{\text{prop}} := x_{k+1}' - x_{k+1} = \Phi_k^i(x_k') - \Phi_k^i(x_k). \tag{5.19}$$

Notice the function $\Phi_k^i : X_i \to X_i$ is differentiable because F_i is assumed to be differentiable with respect to the state variable. Let $J_{\Phi_k^i}(a)$ be the Jacobian matrix of Φ_k^i at $a \in X_i$, which we assume is bounded in a neighborhood of x_k:

Assumption 69. *For each $i \in \mathcal{I}$, there exists a function $B_2^i : X_i \times \mathbb{R}_+^{n_i} \to \mathbb{R}_+^{n_i \times n_i}$, such that for all $1 \le k \le N$ satisfying $y_k = (x_k, i)$,*

$$\left| J_{\Phi_k^i}(a) \right| \le B_2^i(x_k, \epsilon_k) \tag{5.20}$$

for all $a \in x_k + [\epsilon_k]$, where the absolute value $|\cdot|$ and inequality are meant component-wise.

In Section 5.6 we describe how we construct this bound.

It follows from (5.19) and Theorem 57 that

$$\left|\epsilon_{k+1}^{\text{prop}}\right| \le B_2^i(x_k, \epsilon_k)\epsilon_k. \tag{5.21}$$

Combining (5.21) and (5.18), we may obtain a bound for the total error at time t_{n+1}, denoted as ϵ_{n+1}:

$$\epsilon_{k+1} := \frac{h^2}{2} B_1^i(t_k) + B_2^i(x_k, \epsilon_k)\epsilon_k \ge \left|\epsilon_{k+1}^{\text{trunc}}\right| + \left|\epsilon_{k+1}^{\text{prop}}\right| \tag{5.22}$$

However, such ϵ_{n+1} provides global error bound only at a discrete time instant t_{n+1}, rather than for all of $[t_n, t_{n+1}]$. This issue can be resolved by taking the convex hull of two adjacent error sets:

Theorem 70. *Let $y_0 \in \mathcal{X}$ be an initial condition, and let $u \in BV([0,+\infty),\mathcal{U})$ be a control law. Let y_k, y_{k+1} be obtained from the algorithm in Algorithm. 6 for some $1 \leq k \leq N-1$ such that $y_k \notin \mathcal{S}^\delta$. Let x_k (resp., x_{k+1}) be the projection of y_k (resp., y_{k+1}) onto the continuous state space such that $y_k = (x_k, i)$ and $y_{k+1} = (x_{k+1}, i)$. Let $\phi(\cdot, y_0, u(\cdot))$ be a hybrid trajectory of $\mathcal{H}$, whose projection onto X_i is denoted as $y^i(\cdot)$. Let ϵ_k be such that $\epsilon_k \geq |y^i(t_k) - x_k|$. Let ϵ_{k+1} be defined as in* (5.22)*. Let a set $E_k \subset \mathcal{Y}$ be defined as*

$$E_k := \mathit{conv}\left((x_k + [\epsilon_k]) \cup (x_{k+1} + [\epsilon_{k+1}])\right) \tag{5.23}$$

where conv stands for the convex hull. Then, $y^i(t) \in E_k$ for all $t \in [t_k, t_{k+1}]$.

Proof. Let $x'_k = y^i(t_k)$ and $x'_{k+1} = \Phi^i_k(x'_k)$. Define a continuous function $\theta : [t_k, t_{k+1}] \to [0,1]$ as

$$\theta(t) = \frac{t - t_k}{t_{k+1} - t_k} \tag{5.24}$$

for each $t \in [t_k, t_{k+1}]$. Let $z : [t_k, t_{k+1}] \to X_i$ be a linear function defined as

$$z(t) = (1 - \theta(t))x_k + \theta(t)x_{k+1} \tag{5.25}$$

for all $t \in [t_k, t_{k+1}]$. In particular, $z(t_k) = x_k$ and $z(t_{k+1}) = x_{k+1}$. Similarly, let $z' : [t_n, t_{n+1}] \to X_i$ be defined as

$$z'(t) = (1 - \theta(t))x'_k + \theta(t)x'_{k+1} \tag{5.26}$$

for all $t \in [t_k, t_{k+1}]$. An illustration of z and z' is shown in Fig. 5.2.

Let $t \in [t_k, t_{k+1}]$ be arbitrary. We then have

$$\begin{aligned}
|z'(t) - z(t)| =& \left|(1 - \theta(t))x'_k + \theta(t)x'_{k+1} - (1 - \theta(t))x_k - \theta(t)x_{k+1}\right| && (5.27)\\
\leq& (1 - \theta(t))\,|x'_k - x_k| + \theta(t)|x'_{k+1} - x_{k+1}| && (5.28)\\
\leq& (1 - \theta(t))\epsilon_k + \theta(t)B^i_2(x_k, \epsilon_k)\epsilon_k && (5.29)
\end{aligned}$$

where (5.27) follows from definitions of z and z'; (5.28) follows from triangle inequality; (5.29) follows from (5.21).

Notice the value $z(t)$ can be obtained by applying Euler method to x_k with step size $t - t_k$, it

follows from Theorem 67 and Assumption 68 that

$$|y^i(t) - z'(t)| \leq \frac{(t-t_k)^2}{2} B_1^i(t_k) = \theta(t)^2 \frac{h^2}{2} B_1^i(t_k) \leq \theta(t) \frac{h^2}{2} B_1^i(t_k) \tag{5.30}$$

By combining (5.29), (5.30), (5.22), and applying triangle inequality we obtain

$$|y^i(t) - z(t)| \leq (1-\theta(t))\epsilon_k + \theta(t) \left(\frac{h^2}{2} B_1^i(t_k) + B_2^i(x_k, \epsilon_k) \cdot \epsilon_k \right) = (1-\theta(t))\epsilon_k + \theta(t)\epsilon_{k+1} \tag{5.31}$$

Since $z(t) = (1-\theta(t))x_k + \theta(t)x_{k+1}$, we may rewrite (5.31) as

$$(1-\theta(t))(x_k - \epsilon_k) + \theta(x_{k+1} - \epsilon_{k+1}) \leq y^i(t) \leq (1-\theta(t))(x_k + \epsilon_k) + \theta(x_{k+1} + \epsilon_{k+1}) \tag{5.32}$$

Or equivalently,

$$y^i(t) \in \text{conv}(x_k + [\epsilon_k], x_{k+1} + [\epsilon_{k+1}]) = E_k \tag{5.33}$$

□

5.3.3.2 Error bound when $y_k \in \mathcal{S}^\delta$

Without loss of generality, let $y_k \in S^\delta_{(i,j)}$, where $(i,j) \in \mathcal{E}$. Let σ be in a neighborhood of t_k at which a hybrid trajectory $\phi(\cdot, y_0, u(\cdot))$ intersects the guard $S_{(i,j)}$, as shown in Fig. 5.3. Note in the remainder of this chapter we assume $t_k \leq \sigma$. Note that the argument presented generalizes to the case when $t_k > \sigma$. Let ϵ_k be a bound for global error at time t_k. For notational convenience, we denote by $y(\sigma^-)$ the limit of $y(t)$ as t approaches σ from the left, and denote by $y(\sigma^+)$ the limit of $y(t)$ as t approaches σ from the right. That is,

$$y(\sigma^-) := \lim_{t \to \sigma^-} y(t), \tag{5.34}$$

$$y(\sigma^+) := \lim_{t \to \sigma^+} y(t). \tag{5.35}$$

In particular, $y(\sigma^-)$ denotes the state of a hybrid trajectory y right before the transition happens, and $y(\sigma^+)$ denotes the state of y right after a transition.

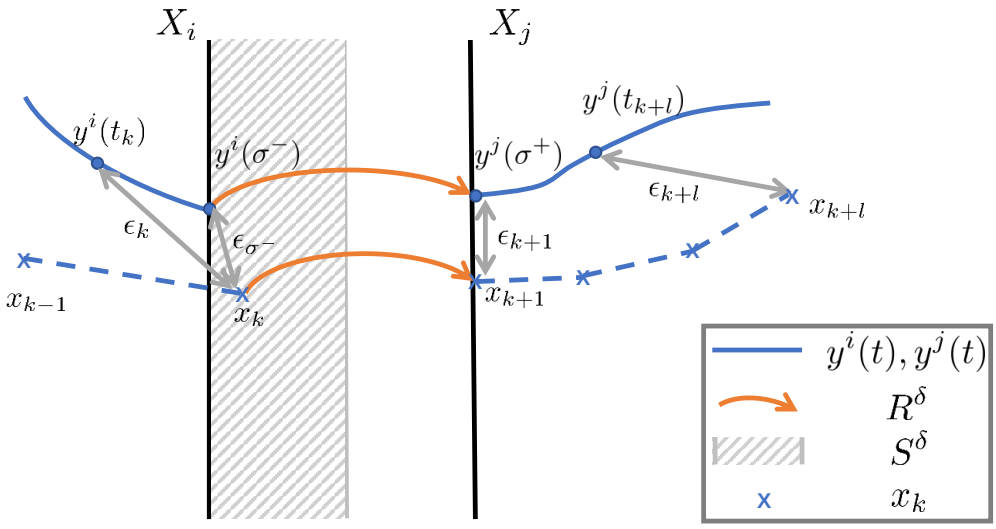


Figure 5.3: An illustration of global errors when $y_k \in \mathcal{S}^\delta$

We now make an assumption on the transition time, σ:

Assumption 71. *For each $(i,j) \in \mathcal{E}$, there exist a function $B_3^{(i,j)} : S_{(i,j)}^\delta \times \mathbb{R}_+^{n_i} \to \mathbb{R}_+$ such that for all hybrid trajectory $\tilde{y}$ satisfying $\tilde{y}^i(t) \in (x_k + [\epsilon_k]) \cup X_i$ for some $t \in [0,T]$, there exists $\tilde{\sigma} \in [t, t + B_3^{(i,j)}(x_k, \epsilon_k)]$ such that*

$$c_{(i,j)}(\tilde{y}^i(\tilde{\sigma}^-)) = 0 \tag{5.36}$$

In other words, if the projection of a hybrid trajectory $\tilde{y}$ onto domain X_i, denoted as $\tilde{y}^i$, reaches a neighborhood $x_k + [\epsilon_k]$ of x_k at time $t \in [0,T]$ but has not yet reached the guard $S_{(i,j)}$, it will reach the guard before time $t + B_3^{(i,j)}(x_k, \epsilon_k)$. In Section 5.6 we describe how we construct this bound.

Notice $y^i(t_k) \in x_k + [\epsilon_k]$ by definition, it then follows from Assumption 71 that

$$\sigma - t_k \leq B_3^{(i,j)}(x_k, \epsilon_k) \tag{5.37}$$

We next assume the vector field in each domain is bounded:

Assumption 72. *For each $i \in \mathcal{I}$, there exists a vector $B_4^i \in \mathbb{R}_+^{n_i}$ such that*

$$|F_i(t, x, u(t))| \leq B_4^i \tag{5.38}$$

for all $t \in [0,T]$, $x \in X_i$, and $u \in BV([0,+\infty), \mathcal{U})$.

In Section 5.6 we describe how we construct this bound.

Using fundamental theorem of calculus, we have

$$|y^i(t) - y^i(t_k)| = \left| \int_{t_k}^{t} F_i(s, y^i(s), u(s))ds \right| \leq B_3^{(i,j)}(x_k, \epsilon_k)B_4^i \tag{5.39}$$

for all $t \in [t_k, \sigma)$, where the inequality follows from (5.37) and Assumption 72. Using triangle inequality, the value of $y^i(t)$ for $t \in [t_k, \sigma)$ can be bounded by

$$|y^i(t) - y_k| \leq |y^i(t_k) - y_k| + |y^i(t) - y^i(t_k)| \leq \epsilon_n + \Delta t B_5 =: \epsilon_{\sigma^-} \tag{5.40}$$

To under stand how the global error changes when reset map is applied, we assume the Jacobian matrix of $R^{\delta}_{(i,j)}$, denoted as $J_{R^{\delta}_{(i,j)}}$, is bounded in a neighborhood of x_k:

Assumption 73. *For each $(i, j) \in \mathcal{E}$, there exists a function $B_5^{(i,j)} : S^{\delta}_{(i,j)} \times \mathbb{R}^d_+ \to \mathbb{R}_+^{n_j \times n_i}$ such that*

$$|J_{\hat{R}}(a)| \leq B_5^{(i,j)}(x_k, \epsilon_k) \tag{5.41}$$

for all $1 \leq k \leq N$ satisfying $y_k \in S^{\delta}_{(i,j)}$ and $a \in x_k + [\epsilon_k]$. The absolute value $|\cdot|$ and inequality are meant component-wise.

In Section 5.6 we describe how we construct this bound. Since $R^{\delta}_{(i,j)}$ is differentiable, it follows from Theorem 57 that

$$|y^j(\sigma^+) - x_{k+1}| \leq B_5^{(i,j)}(x_k, \epsilon_{\sigma^-})\epsilon_{\sigma^-} =: \epsilon_{\sigma^+} \tag{5.42}$$

Let $l \in \mathbb{N}$ be such that $(l-1)h < B_3^{(i,j)}(x_k, \epsilon_k) \leq lh$. For all $t \in [\sigma, t_{k+l}]$, we have

$$|y^j(t) - x_{k+1}| \leq |y^j(\sigma^+) - x_{k+1}| + |y^j(t) - y^j(\sigma^+)| \leq \epsilon_{\sigma^+} + hB_5^j =: \epsilon_{k+1} \tag{5.43}$$

Finally, we can bound the difference between $y^j(t_{k+l})$ and x_{k+l} using triangle inequality:

$$|y^j(t_{k+l}) - x_{k+l}| \leq |y^j(t_{k+l}) - x_{k+1}| + |x_{k+l} - x_{k+1}| \leq \epsilon_{\sigma^+} + lhB_5^j =: \epsilon_{k+l} \tag{5.44}$$

Using the above arguments, we may bound the evolution of y over $[t_k, t_{k+l}]$ as follows:

Theorem 74. *Let $y_0 \in \mathcal{X}$ be an initial condition, and let $u \in BV([0, +\infty), \mathcal{U})$ be a control law. Let y_k, y_{k+1} be obtained from the algorithm in Algorithm. 6 for some $1 \leq k \leq N-1$ such that $y_k \in \mathcal{S}^{\delta}$ for some $(i, j) \in \mathcal{E}$. Let x_k (resp., x_{k+1}) be the projection of y_k (resp., y_{k+1}) onto the*

continuous state space such that $y_k = (x_k, i)$ and $y_{k+1} = (x_{k+1}, j)$. Let $\phi(\cdot, y_0, u(\cdot))$ be a hybrid trajectory of $\mathcal{H}$, whose projection onto X_i is denoted as $y^i(\cdot)$. Let $\sigma \in [0, T]$ be in a neighborhood of t_k such that $y^i(\sigma^-) \in S_{(i,j)}$. Let ϵ_{σ^-} be defined as in (5.40)*. Let ϵ_{k+1} be defined as in* (5.43)*. Let $l \in \mathbb{N}$ be such that $(l-1)h < B_3^{(i,j)}(x_k, \epsilon_k) \leq lh$. Define sets E_k and E_{k+1} as*

$$E_k := x_k + [\epsilon_{\sigma^-}] \tag{5.45}$$

$$E_{k+1} := x_{k+1} + [\epsilon_{k+1}] \tag{5.46}$$

Then, $y^i(t) \in E_k$ for all $t \in [t_k, \sigma)$, and $y^j(t) \in E_{k+1}$ for all $t \in [\sigma, t_{k+l}]$.

Proof. Theorem 74 is a restatement of (5.40) and (5.43). □

5.4 Representing Safety for Online Optimization

Although a bound on the difference between a hybrid trajectory y and its discrete approximation $\{y_n\}$ is established through Theorems 70 and 74, such error bounds must be projected onto subspaces of $\mathcal{X}$ to produce sufficient conditions for safety that can be enforced during online optimization. To do this, let $\pi_{q_{\text{hip},zL}}$, $\pi_{q_{\text{hip},zR}}$, $\pi_{q_{xy}}$ be projection operators that map subsets of $\mathcal{X}$ to its $q_{\text{hip},zL}$-, $q_{\text{hip},zR}$-, and (q_x, q_y)-coordinates, respectively. Using the safety criteria established in Section 5.2.3, we obtain a set of sufficient conditions for safety:

Theorem 75. *Let E_k be defined as in Theorem 70 and Theorem 74 for each $1 \leq k \leq N$. Let $\mathcal{A} := \cup_{l \in \{1,\ldots,N_{obs}\}} \hat{O}_l$ be the set of buffered obstacles. Suppose for each $1 \leq k \leq N$ the following conditions are satisfied:*

$$\pi_{q_{hip,zL}}(E_k) - 0.75 \subseteq \mathbb{R}_+, \tag{5.47}$$

$$\pi_{q_{hip,zR}}(E_k) - 0.75 \subseteq \mathbb{R}_+, \tag{5.48}$$

$$\pi_{q_{xy}}(E_k) \cap \mathcal{A} = \emptyset, . \tag{5.49}$$

Furthermore assume:

$$E_N \subset \mathcal{X}_f. \tag{5.50}$$

Then Cassie is safe.

Proof. (5.47) and (5.48) follows from Theorems 49, 70, and 74. (5.49) follows from Theorems 51, 70, and 74. (5.50) follows from Assumption 54 and Theorem 70. □

Theorem 75 extends safety conditions over points to corresponding sufficient conditions over sets. In particular, (5.47) and (5.48) imply the left and right hip height states of Cassie are above 0.75

[m], which according to Theorem 49 guarantees Cassie does not fall before τ_f; (5.49) implies Cassie's pelvis trajectory does not intersect any buffered obstacles, which according to Theorem 51 guarantees Cassie does not run into obstacles before τ_f; (5.50) ensures that Cassie's trajectory reaches a set $\mathcal{X}_f$ at the end of planning horizon, which according to Assumption 54 keeps Cassie safe indefinitely.

5.5 Online Trajectory Optimization

Using the safety conditions established in the previous section, we may formulate an optimization problem over control parameters and solve such programs online in a receding horizon fashion in a provably safe manner. Since the E_k's as defined in Theorem 70 and 74 are dependent on Cassie's initial state y_0, the control law $\hat{u}_P$, and the global error ϵ_0 at $t_0 = 0$, we abuse notation and denote as $E_k(y_0, P, \epsilon_0)$ the set E_k as in Theorems 70 and 74 obtained by numerically bounding Cassie's trajectory starting from $y_0 \in \mathcal{X}$ with control $P \in \mathcal{P}$ under the assumption that there was global error ϵ_0 at $t_0 = 0$. Let $\mathcal{A} := \cup_{l \in \{1,\dots,N_{\text{obs}}\}} \hat{O}_l$ be the set of buffered obstacles. Let Γ be the space of hybrid trajectories of Cassie. We now formally define the optimization problem to be solved online:

$$
\begin{aligned}
\min_{P \in \mathcal{P}} \quad & J(\phi(\cdot, y_0, \hat{u}_P(\cdot))) && (\texttt{TrajOpt}) \\
\text{s.t.} \quad & \pi_{q_{\text{hip},zL}}(E_k(y_0, P, \epsilon_0)) - 0.75 \subseteq \mathbb{R}_+ && \forall k \in \{1, \dots, N\}, \\
& \pi_{q_{\text{hip},zR}}(E_k(y_0, P, \epsilon_0)) - 0.75 \subseteq \mathbb{R}_+ && \forall k \in \{1, \dots, N\}, \\
& \pi_{q_{xy}}(E_k(y_0, P, \epsilon_0)) \cap \mathcal{A} = \emptyset && \forall k \in \{1, \dots, N\}, \\
& E_N(y_0, P, \epsilon_0) \subset \mathcal{X}_f.
\end{aligned}
$$

where the optimization is taken over the control parameter space $\mathcal{P}$; $y_0 \in \mathcal{X}$ represents the current state of Cassie; $\phi(\cdot, y_0, \hat{u}_P(\cdot))$ is a hybrid trajectory of Cassie starting from y_0 at $t_0 = 0$ under control law $\hat{u}_P$; ϵ_0 is a vector of zeros representing the global error at $t_0 = 0$; $\mathcal{A} \subset \mathcal{G}$ is the set of buffered obstacles obtained from the function `buffer`; $J \in \Gamma \to \mathbb{R}$ is a user specified cost function that maps a hybrid trajectory to a real number. Notice the constraints are adopted from Theorem 75. It then follows from Theorem 75 that any *feasible* solution to (`TrajOpt`) keeps Cassie safe:

Theorem 76. *Let y_0 be the current state of Cassie, and let $P \in \mathcal{P}$ be any feasible point to (`TrajOpt`). If P is applied to Cassie, then Cassie is safe.*

Although (`TrajOpt`) provides a means to find safe control inputs, solving such an optimization problem in practice can be challenging due to a lack of gradient information about the con-

straints. As such, gradient-descent methods that rely on numerical construction of the derivative often struggle to find feasible solutions. Exhaustive search methods, on the other hand, require evaluation of the constraints at many points which can be slow. In this chapter, we borrow the idea from Chapter 4 and develop a parallel algorithm that finds solutions to (`TrajOpt`) with safety guarantees.

To start, notice from Theorem 70 and 74 that $E_k(y_0, P, \epsilon_0)$ for all $1 \leq k \leq N$ are set enclosures of the hybrid trajectory $\phi(\cdot, y_0, \hat{u}_P(\cdot))$ over the time $[0, T]$. We abuse notation and denote by $J(\{E_k(y_0, P, \epsilon_0)\}_{1\leq k\leq N})$ the interval-arithmetic evaluation of J obtained by replacing the hybrid trajectory $\phi(\cdot, y_0, \hat{u}_P(\cdot))$ with the corresponding set enclosures $\{E_k(y_0, P, \epsilon_0)\}_{1\leq k\leq N}$. Let tol be a pre-specified optimality tolerance . We now formally define our approach in Algorithm 7: first, in Lines 1-2, the search space $\mathcal{P}$ is represented as a box $P + [\epsilon_P]$, where $P \in \mathcal{P}$ is the center and $\epsilon_P := (\epsilon_{P_1}, \epsilon_{P_2}) \in \mathbb{R}^2_+$ the width in each dimension divided by two; second, in Line 5, for each box $P + [\epsilon_P]$, we append the control parameters to the states and global errors, that is, let

$$\hat{y}_0 := \begin{bmatrix} y_0 \\ P \end{bmatrix}, \quad \hat{\epsilon}_0 := \begin{bmatrix} \epsilon_0 \\ \epsilon_P \end{bmatrix} \tag{5.51}$$

be the aggregated states and the aggregated global error at t_0, respectively. Here $\epsilon_0 \in \mathbb{R}^{n_i}$ is a vector of zeros, where i is the discrete state of y_0; third, in Lines 6-7, the Euler method is applied from initial state $\hat{y}_0$ with initial global error $\hat{\epsilon}_0$ where the dynamics in $\mathcal{P}$-coordinates are assumed to be zero, through which the set enclosures $E_k(\hat{y}(0), P, \hat{\epsilon}_0)$, $1 \leq k \leq N$ are computed; fourth, in Line 8, the constraints in (`TrajOpt`) are evaluated over each $E_k(\hat{y}(0), P, \hat{\epsilon}_0)$ where $1 \leq k \leq N$, and any infeasible box $P + [\epsilon_P]$ is removed from the search space; fifth, in Line 9, an interval-arithmetic evaluation $J(\{E_k(\hat{y}_0, P, \hat{\epsilon}_0)\}_{1\leq k\leq N})$ is computed; sixth, in Line 11, any box $P + [\epsilon_P]$ is removed from the search space if it gives a strictly worse value of cost function than another box; seventh, in Line 12, a solution P^* is chosen such that it gives a lower value of cost function when compared to other $(P, \epsilon_P) \in \mathcal{L}$; eighth, in Line 13, the algorithm returns P^* if the value of the cost function over the remaining search space is within a range of tol; ninth, in Lines 16-17, each box $P + [\epsilon_P]$ is subdivided along the r^{th} direction to obtain pairs (P', ϵ'_P) and (P'', ϵ''_P), such that

$$\begin{aligned}
&\epsilon'_P = \epsilon''_P = \begin{bmatrix} \frac{1}{2}(\epsilon_P)_1 \\ (\epsilon_P)_2 \end{bmatrix}, \ P' = \begin{bmatrix} (P)_1 - (\epsilon'_P)_1 \\ (P)_2 \end{bmatrix}, \ P' = \begin{bmatrix} (P)_1 - (\epsilon'_P)_1 \\ (P)_2 \end{bmatrix}, \text{ if } r = 1; \\
&\epsilon'_P = \epsilon''_P = \begin{bmatrix} (\epsilon_P)_1 \\ \frac{1}{2}(\epsilon_P)_2 \end{bmatrix}, \ P' = \begin{bmatrix} (P)_1 \\ (P)_2 - (\epsilon'_P)_2 \end{bmatrix}, \ P' = \begin{bmatrix} (P)_1 \\ (P)_2 - (\epsilon'_P)_2 \end{bmatrix}, \text{ if } r = 2,
\end{aligned} \tag{5.52}$$

where the subscripts $(\cdot)_1$ and $(\cdot)_2$ denote the first and second component of a vector, respectively; Finally, in Line 20, the subdivision direction r is changed at the end of each iteration.

Algorithm 7 $P^* = \texttt{SolveTrajOpt}(y_0, \mathcal{A}, J)$

Require: $y_0 \in \mathcal{X}$, $\mathcal{A} \subset \mathcal{G}$, $J : \Gamma \to \mathbb{R}$, $tol \in \mathbb{R}_+$.

1: Choose $P \in \mathcal{P}$ and $\epsilon_P \in \mathbb{R}^2$ such that $P + [\epsilon_P] = \mathcal{P}$.
2: Set $\mathcal{L} \leftarrow \{(P, \epsilon_P)\}$, $r \leftarrow 1$, $\epsilon_0 \leftarrow 0^{n_i}$, where i is the discrete state of y_0.
3: **loop**
4: **for** $(P, \epsilon_P) \in \mathcal{L}$ **do**
5: Define $\hat{y}_0$, $\hat{\epsilon}_0$ as in (5.51).
6: Construct a numerical simulation $\{t_k, y_k\}_{1 \le k \le N}$ with initial condition $\hat{y}_0$ and control u_P via Algorithm 6.
7: Using $\hat{\epsilon}_0$ as a global error bound at $t = 0$, construct set enclosures $E_k(\hat{y}_0, P, \hat{\epsilon}_0)$ as defined in Theorems 70 and 74.
8: If any of the constraints in (`TrajOpt`) is not satisfied for such $E_k(\hat{y}_0, P, \hat{\epsilon}_0)$, remove the pair (P, ϵ_P) from $\mathcal{L}$.
9: Set $Cost_{(P,\epsilon_P)} \leftarrow J(\{E_k(\hat{y}_0, P, \hat{\epsilon}_0)\}_{1 \le k \le N})$
10: **end for**
11: If there exists some $(P, \epsilon_P), (P', \epsilon'_P) \in \mathcal{L}$ such that $\max Cost_{(P,\epsilon_P)} < \min Cost_{(P',\epsilon'_P)}$, remove the pair (P', ϵ'_P) from $\mathcal{L}$;
12: Choose$(P^*, \epsilon_{P^*}) \in \mathcal{L}$ so that $\min Cost_{(P^*,\epsilon_{P^*})} < \min Cost_{(P,\epsilon_P)}$ for all $(P, \epsilon_P) \in \mathcal{L}$, $P \neq P^*$.
13: If $|\max Cost_{(P,\epsilon_P)} - \min Cost_{(P^*,\epsilon_{P^*})}| < tol$ for all $(P, \epsilon_P) \in \mathcal{L}$, then **return** P^*.
14: Set $\mathcal{L}' \leftarrow \{\}$.
15: **for** $(P, \epsilon_P) \in \mathcal{L}$ **do**
16: subdivide (P, ϵ_P) along r^{th} dimension to obtain (P', ϵ'_P) and (P'', ϵ''_P) as in (5.52).
17: Set $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{(P', \epsilon'_P), (P'', \epsilon''_P)\}$.
18: **end for**
19: Set $\mathcal{L} \leftarrow \mathcal{L}'$.
20: Set $r \leftarrow mod(r + 1, 2) + 1$.
21: **end loop**

Similar to Theorem 76, it can be shown that any feasible solution generated by Algorithm 7 keeps Cassie safe:

Theorem 77. *Let y_0 be the current state of Cassie, and let $P^* \in \mathcal{P}$ be any feasible solution to Algorithm 7 which is applied to Cassie. Then, Cassie is safe.*

Algorithm 8 describes how the online planning scheme works. `SenseObstacles` senses all obstacles within the sensor horizon in Line 5; The function `buffer` generates buffered region around each sensed obstacle in Line 6; `ComputeFutureState` takes the current state and control parameter and applies the flow map to compute the future state of Cassie at time τ_1 in line 7; Finally, Algorithm 7 solves an optimization to generate safe control parameters that also minimize the cost function J in Line 8. Recall that we use Theorem 56 to check Algorithm 7 is persistently feasible.

Algorithm 8 Online Planning Scheme

Require: $\phi(0) \in \mathcal{X}$, $P_0 \in \mathcal{P}$, and $J : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$.

1: Set $j = 0$, $t_j = 0$, $P^* = P_0$
2: **loop**
3: // Line 4 executes at the same time as Lines 5–11
4: Apply P^* for $[t_j, t_j + \tau_1)$ to reach state $\phi(t_j + \tau_1)$
5: $\{O_n\}_{n=1,\dots,N_{\text{obs}}} \leftarrow \texttt{SenseObstacles}()$
6: $\mathcal{A} \leftarrow \emptyset$
7: $\mathcal{A} \leftarrow \mathcal{A} \cup (\texttt{buffer}(O_k))$ for each $n = 1, \dots, N_{\text{obs}}$
8: $y_0 \leftarrow \texttt{ComputeFutureState}(\tau_1, \phi(t_j), P_j)$
9: **Try** $P^* \leftarrow \texttt{SolveTrajOpt}(y_0, \mathcal{A}, J)$ until $t_j + \tau_1$
10: **Catch** continue // P^* is unchanged
11: $t_{j+1} \leftarrow t_j + \tau_1$, $P_{j+1} \leftarrow P^*$, $j \leftarrow j + 1$
12: **end loop**

5.6 Implementation

This section provides some implementation details of the proposed approach.

5.6.1 Hybrid Model of Cassie

Recall that the hybrid model of Cassie is a tuple $\mathcal{H} = (\mathcal{I}, \mathcal{E}, \mathcal{X}, \mathcal{U}, \mathcal{F}, \mathcal{S}, \mathcal{R})$. In this work, let:

- $\mathcal{I} := \{1, 2, 3\}$, where the indices 1, 2, and 3 correspond to left leg stance phase, right leg stance phase, and double stance phase, respectively;
- $\mathcal{E} := \{(1, 2), (2, 1), (1, 3), (2, 3)\}$;
- $\mathcal{X} = \coprod_{i \in \mathcal{I}} X_i$, where each $X_i \subset \mathbb{R}^{26}$ is subject to joint limits;
- $\mathcal{U} \in \mathbb{R}^{10}$, where $\mathcal{U}$ is subject to torque limits;
- $\mathcal{F} = \{F_i\}_{i \in \mathcal{I}}$, where $F_i : \mathbb{R} \times X_i \times \mathcal{U} \to \mathbb{R}^{n_i}$ is obtained using the method of Lagrange [WGC+18, Appendix D];
- $\mathcal{S} = \coprod_{e \in \mathcal{E}} S_e$ such that $S_{(i,j)} := \{x \in X_i \mid c_{(i,j)}(x) = 0\}$ for all $(i, j) \in \mathcal{E}$, where $c_{(i,j)}$ maps Cassie's states to the height of non-stance foot; and,
- $\mathcal{R} = \{R_e\}_{e \in \mathcal{E}}$, where each reset map R_e is obtained as in [GHD+19, (9)-(10)].

Note that this hybrid system definition satisfies Assumptions 58, 62, 63, and 64.

5.6.2 Definition of $\mathcal{X}_f$

In this work, $\mathcal{X}_f$ is defined to be the set of states where the velocity of each joint is small and the joint angles are sufficiently close to a set of pre-specified reference values. That is, let a reference configuration, $q^{\text{ref}} \in X_3$, be defined as

$$\begin{aligned} q^{\text{ref}} := (&q_x^{\text{ref}}, q_y^{\text{ref}}, q_z^{\text{ref}}, q_{yaw}^{\text{ref}}, -0.357, 0, 0.0813, 0.0007, 0.4555, \\ &- 1.1979, -0.0379, 1.4989, -1.6044, 0.0813, 0.0007, \\ &0.4555, -1.1979, -0.0379, 1.4989, -1.6044, 0.0813, \\ &q_{\text{hip},xL}^{\text{ref}}, q_{\text{hip},yL}^{\text{ref}}, q_{\text{hip},zL}^{\text{ref}}, q_{\text{hip},xR}^{\text{ref}}, q_{\text{hip},yR}^{\text{ref}}, q_{\text{hip},zR}^{\text{ref}}) \end{aligned} \tag{5.53}$$

where $q_x^{\text{ref}}, q_y^{\text{ref}}, q_z^{\text{ref}}, q_{yaw}^{\text{ref}} \in \mathbb{R}$ are set to be the current value of q_x, q_y, q_z and q_{yaw}. The reference hip positions $q_{\text{hip},xL}^{\text{ref}}, q_{\text{hip},yL}^{\text{ref}}, q_{\text{hip},zL}^{\text{ref}}, q_{\text{hip},xR}^{\text{ref}}, q_{\text{hip},yR}^{\text{ref}}, q_{\text{hip},zR}^{\text{ref}}$ are computed based on the the first 6 elements of q^{ref} and the forward kinematics of Cassie. Then, define

$$\mathcal{X}_f := \{(q, \dot{q}) \in X_3 \mid |q - q^{\text{ref}}| \leq 0.1, |\dot{q}| \leq 0.1, \} \tag{5.54}$$

where q, $\dot{q}$ are the configuration and velocity coordinates of Cassie's state, respectively; the absolute value $|\cdot|$ and inequality are meant component-wise.

5.6.3 Bounding Functions

We now describe how to construct the bounding functions in Assumptions 68, 69, 71, 72, and 73. To generate a function $B_2^i : X_i \times \mathbb{R}^{n_i} \to \mathbb{R}_+^{n_i \times n_i}$ as in Assumption 69, we apply Algorithm 6 to a group of randomly selected initial conditions and control parameters to obtain a set of time and state sequences. Each of the state sequence, denoted as $\{y_k\}_{1 \leq k \leq N}$, is then projected to the continuous state space to obtain the sequence $\{x_k\}_{1 \leq k \leq N}$. For each point x_k in that sequence, we randomly sample a and ϵ such that $a \in x_k + [\epsilon]$, and compute the Jacobian matrix, $J_{\Phi_k^i(a)}$. We then collect all the data and fit a polynomial function, f, to it. To make sure the bounding function is conservative, an offset C is added to f to generate B_2^i. This procedure is described in Algorithm 9. The bounding functions in Assumptions 68, 72, and 73 are obtained in a similar fashion.

We next describe how a bounding function $B_3^{(i,j)}$ in Assumption 71 is constructed. We start by defining $J_{c_{(i,j)}}(a)$ as the Jacobian of $c_{(i,j)}$ at $a \in X$, which we assume is bounded by a function $B_{Jc}^{(i,j)} : X_i \times \mathbb{R}_+^{n_i} \to \mathbb{R}_+^{1 \times n_i}$ in a neighborhood of x_k, i.e.,

$$\left| J_{c_{(i,j)}}(a) \right| \leq B_{Jc}^{(i,j)}(x_k, \epsilon_k) \tag{5.55}$$

Algorithm 9 Algorithm that generates B_2^i

Set $\mathcal{L}_i = \{\}$ for each $i \in \mathcal{I}$.
for $itr1 \leftarrow 1$ to 100 **do**
  Sample $y_0 \in \mathcal{X}$, $P \in \mathcal{P}$.
  Apply Algorithm 6 with initial condition y_0 and control law $\hat{u}_P$ to obtain $\{y_k\}_{1 \leq k \leq N}$.
  for $k = 1$ to N **do**
    Let $(x_k, i) = y_k$.
    for $itr2 \leftarrow 1$ to 100 **do**
      Sample $a \in X_i$ and $\epsilon \in \mathbb{R}_+^{n_i}$ such that $a \in x_k + [\epsilon]$.
      Let $J_{\Phi_k^i}(a)$ be computed numerically.
      Set $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \left(x_k, \epsilon, \left|J_{\Phi_k^i}(a)\right|\right)$
    end for
  end for
end for
for $i \in \mathcal{I}$ **do**
  Fit a polynomial $f : X_i \times \mathbb{R}_+^{n_i} \to \mathbb{R}_+^{n_i \times n_i}$ to the data in $\mathcal{L}_i$ such that for each triplet $(x, \epsilon, M) \in \mathcal{L}_i$, the value $f(x, \epsilon)$ approximates M.
  Choose an offset $C \in \mathbb{R}_+^{n_i \times n_i}$ such that $f(x, \epsilon) + C \geq 1.2M$ for each triplet $(x, \epsilon, M) \in \mathcal{L}_i$.
  Set $B_2^i \leftarrow f + C$.
end for

for all $1 \leq k \leq N$ satisfying $y_k \in S_{(i,j)}^{\delta}$ and $a \in x_k + [\epsilon_k]$. The absolute value $|\cdot|$ and inequality are meant component-wise. Such function $B_{Jc}^{(i,j)}$ can be generated in a similar way as in Algorithm 9. It then follows from Theorem 57 that

$$c_{(i,j)}(y(t_k)) \leq c_{(i,j)}(x_k) + B_{Jc}^{(i,j)}(x_k, \epsilon_k)\epsilon_k \leq B_{Jc}^{(i,j)}(x_k, \epsilon_k)\epsilon_k \tag{5.56}$$

where the last inequality is true because $c_{(i,j)}(x_k) \leq 0$. Then $B_3^{(i,j)}$ can be generated by bounding the rate of change of $c_{(i,j)}$ in a neighborhood of the guard $S_{(i,j)}$. Specifically, we want to find $\beta \in \mathbb{R}_+$ such that

$$\frac{d}{dt} c_{(i,j)}(y^i(t)) \leq -\beta < 0 \tag{5.57}$$

and

$$B_{Jc}^{(i,j)}(x_k, \epsilon_k)\epsilon_k \leq \beta \cdot B_3^{(i,j)}(x_k, \epsilon_k) \tag{5.58}$$

where $y^i(t) \in X_i$ and for all $t \in [\sigma - B_3(x_k, \epsilon_k), \sigma]$. β and $B_3^{(i,j)}$ are constructed using the procedure described in Algorithm 10.

Algorithm 10 Algorithm that generates β and B_3

Require: $B^e_{J_c}$ for all $e \in \mathcal{E}$.

Set $\mathcal{L}_e = \{\}$ for each $e \in \mathcal{E}$.
Set $\Delta t \leftarrow 0.005$.
for $itr \leftarrow 1$ to 100 **do**
  Sample $y_0 \in \mathcal{X}$, $P \in \mathcal{P}$.
  Apply Algorithm 6 with initial condition y_0 and control law $\hat{u}_P$ to obtain $\{t_k\}_{1\leq k\leq N}$ and $\{y_k\}_{1\leq k\leq N}$.
  Let $\epsilon_0 \in \mathbb{R}^{26}$ be a vector of zeros. Compute the global error bounds $\{\epsilon_k\}_{1\leq k\leq N}$ using (5.22),(5.43), and (5.44).
  for each k such that $y_k \in S^\delta$ **do**
    WLOG let $y_k \in S^\delta_{(i,j)}$ for some $(i,j) \in \mathcal{E}$. Let $(x_k, i) = y_k$. Set $\Delta t \leftarrow 0.005$.
    while true **do**
      Let $k' \in \{1, \ldots, k\}$ be such that $t_k - t'_k - h \leq \Delta t < t_k - t'_k$.
      Set $\beta \leftarrow -\max_{\kappa\in\{k',\ldots,k\}}\{\text{Swing foot vertical velocity at } t_\kappa\}$.
      Set $\alpha \leftarrow 1/\beta \cdot \max_{\kappa\in\{k',\ldots,k\}}\{B^{(i,j)}_{J_c}(x_\kappa, \epsilon_\kappa)\epsilon_\kappa\}$.
      if $\alpha \leq \Delta t$ **then**
        $\mathcal{L}_{(i,j)} \leftarrow \mathcal{L}_{(i,j)} \cup (x_k, \epsilon_k, \alpha)$.
        break.
      else
        Set $\Delta t \leftarrow 1.5\Delta t$.
      end if
    end while
  end for
end for
for $(i,j) \in \mathcal{E}$ **do**
  Fit a polynomial $f : X^\delta_i \times \mathbb{R}^{n_i} \to \mathbb{R}_+$ to the data in $\mathcal{L}_{(i,j)}$ such that for each triplet $(x, \epsilon, \alpha) \in \mathcal{L}_{(i,j)}$, the value $f(x, \epsilon)$ approximates α.
  Choose an offset $C \in \mathbb{R}_+$ such that $f(x, \epsilon) + C \geq 1.2\alpha$ for each triplet $(x, \epsilon, \alpha) \in \mathcal{L}_{(i,j)}$.
  Set $B^{(i,j)}_3 \leftarrow f + C$.
end for

The idea behind Algorithm 10 is as follows: We start by setting Δt to be some small value and search for the largest value of β that satisfies (5.57), then check if (5.58) holds. If (5.58) is violated, then we increase the value of Δt and repeat the procedure. A group of data points is generated through 100 numerical simulations of Cassie over 30 seconds each with randomly generated initial conditions and control parameters. A function $B^{(i,j)}_3$ is then obtained by fitting a polynomial to the data and adding a constant offset to it.

5.6.4 GPU Implementation of `SolveTrajOpt`

In this work, the function `SolveTrajOpt` as described in Algorithm 7 is implemented on GPUs where double-precision floating-point representations are used. The optimality tolerance tol in `SolveTrajOpt` is set to be 1.

5.7 Results

The proposed method is evaluated by requiring Cassie to reach a pre-defined goal and avoid randomly generated obstacles along the way in 100 simulation trials. In each simulation trial, let $\mathcal{G} = [-1, 7] \times [-3, 3]$, and Cassie is required to reach the goal $(5, 0) \in \mathcal{G}$ from the origin of the $\mathcal{G}$ in which the units of x-axis and y-axis are both meter. Obstacles are randomly located in $\mathcal{G}$ with the number of obstacles varying from 4 to 8. Each obstacle has 0 orientation, length and width both as 0.2[m]. The foot print radius R of Cassie is set to be 0.3[m]. In the MPC framework, let $\tau_1 = 2$[sec] and $\tau_f = 4$[sec]. We also assume the sensing horizon D_{sense} to be 3 meters.

An RRT is used as our high-level planner to generate waypoints that Cassie attempts to track. In particular, in the proposed method, the cost function J is set to be the summation of two parts: the Euclidean norm of the difference between the predicted robot xy-position 2 seconds later and the waypoint to be traveled to next, and the 2-norm of the difference between the predicted robot heading 2 seconds later and the desired heading. The trials are tested on a machine with 32 64-bit 2.60GHz Intel Core i9 CPUs, 128 GB of RAM, and two Nvidia GTX TITAN Xp GPUs.

The proposed method is compared with a direct method using the same obstacle locations, sizes, and orientations. The direct method also updates control parameter K via optimization, which has the same cost function as the proposed method does, but ensures the safety of the robot by only checking whether the discretization of the predicted trajectory of Cassie intersects with obstacles. In particular, no error propagation, hip height, or parallel computing is considered in the direct method.

Fig. 5.4 illustrates one of the 100 trials in which Cassie runs into obstacles with the direct method, but is able to successfully reach the target with the method developed in this chapter. Fig. 5.5 depicts the proposed method, at 3 arbitrarily chosen time instances. Note the global error bounds along the trajectory never intersect the obstacles in the xy-plane. In addition, note that the global error bounds along the trajectories of hip heights remain above 0.75[m] at the same time instances as in Fig. 5.6. Fig. 5.7 illustrates another trail in which Cassie crashes the boundary using the direct method, but successfully reaches the target again with the proposed method.

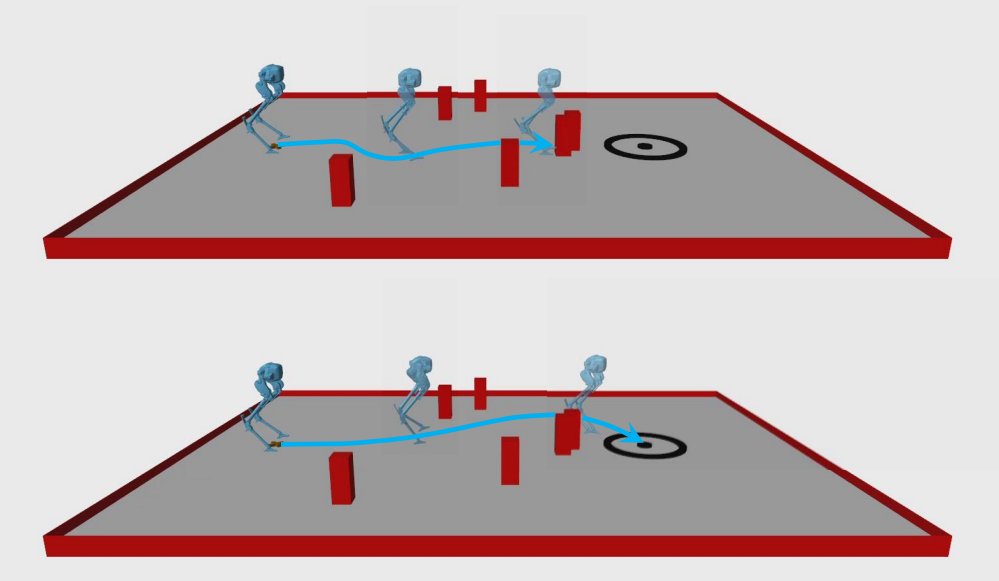


Figure 5.4: An illustration of the performance of the direct method (top) and the proposed method (bottom). Cassie is colored in blue, its trajectory is colored in light blue. Obstacles and the boundaries are colored in red, and the goal is colored in black. Cassie reaches the goal if its footprint is inside the black ring with radius 0.5[m].

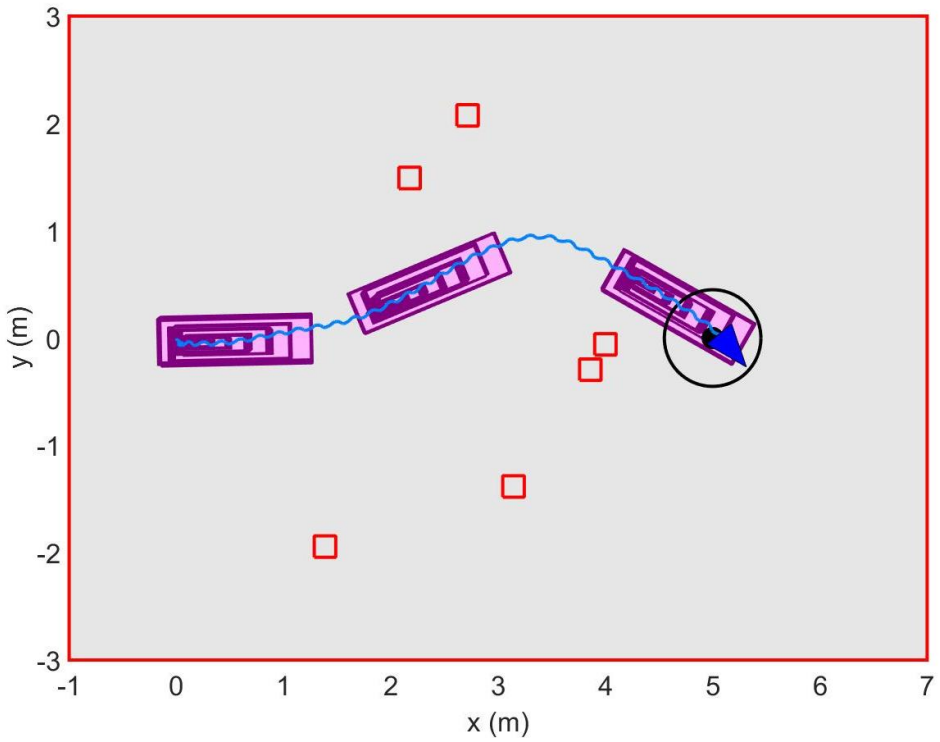


Figure 5.5: An illustration of global error bound in the xy-plane at 3 arbitrary time instances with the proposed method with respect to Fig. 5.4. Cassie is marked as the blue triangle, and its trajectory is colored in blue. Obstacles are colored in red, goal is colored in black, and global error bound is colored in pink. The boxes in dark pink stands for the error bound on pelvis position at each touch-down.

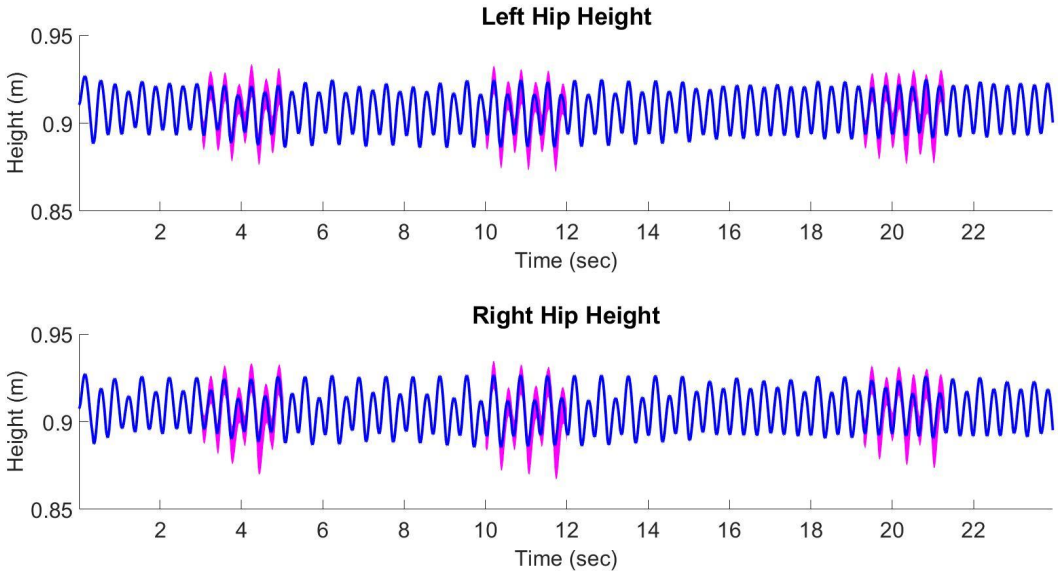


Figure 5.6: An illustration of global error bound of hip heights at the same time instances with the proposed method with respect to Fig. 5.4 and 5.5. Hip heights are colored in blue, and their corresponding global error bounds are colored in magenta.

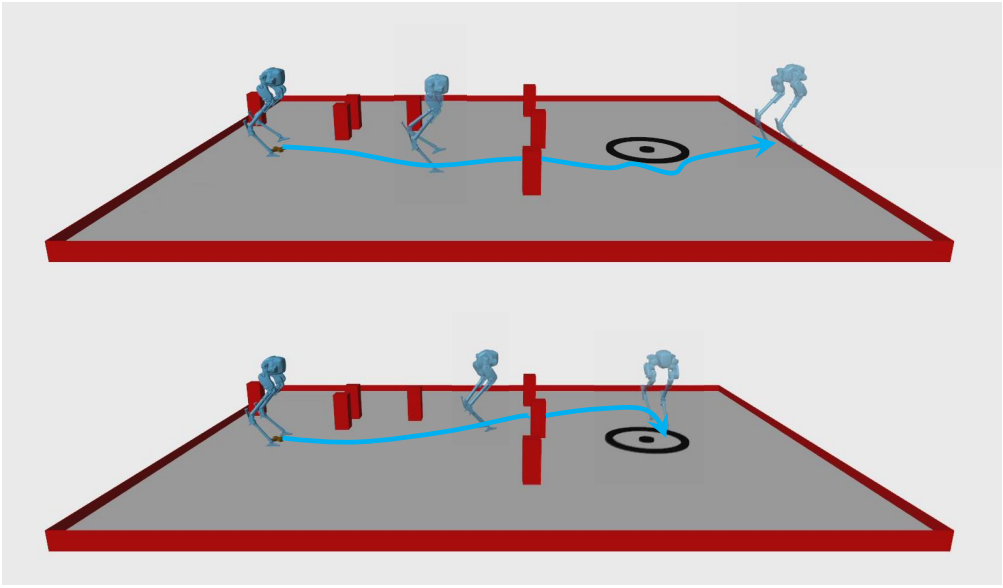


Figure 5.7: An illustration of the performance of the direct method (top) and the proposed method (bottom). Cassie is colored in blue, its trajectory is colored in light blue. Obstacles and the boundaries are colored in red, and the goal is colored in black. Cassie reaches the goal if its footprint is inside the black ring with radius 0.5[m].

	Direct Method			Proposed Method		
obs #	Fall	Crash	Reach	Fall	Crash	Reach
4	0	4	3	0	0	18
5	0	2	3	0	0	13
6	0	6	1	0	0	17
7	0	5	3	0	0	14
8	0	5	3	0	0	11

Table 5.1: A quantitative comparison of the direct method to the method developed in this chapter while controlling Cassie.

Among all the 100 trials, the statistics of directed method and proposed method is summarized in Tab. 5.1. For each value of number of obstacles, we randomly generate 20 trials to evaluate the performance of both methods. Notice the numbers in the row do not always sum to 20, since it is possible that Cassie is unable to find a feasible way to reach the goal in some circumstances.

The average solving time of one optimization in the direct method is 59.2393 seconds, while the average solving time of the proposed method is 5.726 seconds. Notice that the solving time is larger than τ_1 and therefore Algorithm 8 is currently unable to synthesize control input in real-time. The next chapter describes potential ways to improve the computational speed of Algorithm 8.

CHAPTER 6

General Conclusions and Future Directions

In this dissertation, we have made improvements to the properties of control synthesis methods for bipedal robots along three dimensions: optimality, safety, and computational speed. To understand if an optimal control strategy can be found for generic hybrid system, Chapter 2 develops an algorithm to solve the hybrid optimal control problem in an ideal case, where we assume perfect knowledge of the system but no knowledge of the optimal solution. Next in Chapter 3, we establish formal safety guarantees for 2D bipedal robots in an online fashion by performing reachability analysis on a simplified model, where the difference between the simplified model and the true model is assumed to be bounded. In Chapter 4, we developed a parallel algorithm capable of solving polynomial optimization problems up to arbitrary tolerance. Moreover, we derived practical bounds on the computation time and memory usage. Finally, in Chapter 5, we develop a guaranteed safe online MPC framework that accommodates 3D robots and obstacles.

6.1 Future Work

Although this dissertation makes significant strides in improving optimality, safety, and computational speed for control synthesis of bipedal robot models, many adaptations and extensions can be made to improve performance. We summarize these potential directions below.

6.1.1 Real-Time Computation on GPUs

As describe in Section 5.7, the proposed trajectory optimization method is not fast enough to be performed in an online fashion. However, to improve computational speeds, several potential improvements are worth exploring. First, note that evaluating the dynamics of a bipedal robot is computationally expensive. However, by placing practical bounds on certain portions of the dynamics (e.g. Coriolis term) rather than computing them exactly one could reduce the computational overhead associated with evaluating the dynamics. Second, note that the size of the error bounds

computed as in Section 5.3 decreases as the order of numerical integrator increases. Therefore by choosing a numerical integrator of higher order, one may select a larger step size h such that each evaluation of constraints requires fewer integration iterations. Finally, note that the current implementation on GPU relies on representing the state of the robot using double precision arithmetic. However, GPUs are currently optimized to perform single precision arithmetic and often can achieve 4x speed improvements by just switching from double to single precision arithmetic.

6.1.2 Planning Under Uncertainty

The online MPC frameworks proposed in Chapter 3 and 5 either require perfect knowledge of the dynamical model or rely heavily on a a bounding function that is assumed to be given *a* priori. However, such assumptions can often be unrealistic in practice. For example, modeling and simulating locomotion in detail requires careful study of compliance, motor dynamics, and joint frictions, which can add significant complexity to the planning algorithm; Generating a function that bounds the difference between a complex model and a simplified model, on the other hand, requires sampling over the entire state space. It is therefore of interest to instead allow for uncertainty in the dynamical model and accommodate this uncertainty during trajectory planning and control synthesis.

6.1.3 Walking on Uneven Terrains

The trajectory planning and control synthesis frameworks proposed in this dissertation are only concerned with bipedal robots walking on *flat ground*. Even in the case where obstacles are present, the robot is tasked to walk around them, rather than stepping on or over them. One challenge that arises due to uneven terrain is that the smoothness assumptions of guards and reset maps as in Section 5.3 may no longer be satisfied. Devising methods to accommodates such challenges will be be critical before any of the algorithms developed in this thesis can be deployed to control bipeds operating in the real-world.

APPENDIX A

Connecting Occupation Measure With Flow Map of Smooth Vector Field

Let $F : [0,T] \times \mathbb{R}^n \to \mathbb{R}^n$ be a pointwise bounded vector field, such that $F(t,\cdot)$ is Lipschitz for all $t \in [0,T]$. Consider a non-homogeneous PDE $\partial_t \mu_{t,x} = \sigma - \eta - D_x \cdot (F\mu_{t,x})$, where $\mu_{t,x}, \sigma, \eta \in \mathcal{M}([0,T] \times \mathbb{R}^n)$. Applying integration by parts and Lemma 8, this PDE becomes:

$$\begin{aligned}\int_{[0,T]\times\mathbb{R}^n} \partial_t v(t,x) d\mu_{x|t}(x)dt = \int_{[0,T]\times\mathbb{R}^n} v(t,x)\, d\big(\eta(t,x)+ \\ - \sigma(t,x)\big) + \int_{[0,T]\times\mathbb{R}^n} \nabla_x v(t,x) \cdot F d\mu_{x|t}(x)\, dt\end{aligned} \tag{A.1}$$

for any $v \in C^1([0,T] \times \mathbb{R}^n)$. To establish a relationship between F and this PDE, let Φ satisfy (B.3) with $\bar{F}_i^\epsilon$ replaced by F. Since F is pointwise bounded and $F(t,\cdot)$ is Lipschitz for all $t \in [0,T]$, the solutions of the ODE are unique [Hal09, Theorem 5.3]. By differentiating the identity $\Phi_i(t,s,\Phi_i(s,\tau,z)) = \Phi_i(t,\tau,z)$ with respect to s, we can show that $\Phi(t,\cdot,\cdot)$ is a solution to $\frac{d}{ds}\Phi_i(t,s,x) + \nabla_x \Phi_i(t,s,x) \cdot F(s,x) = 0$. This leads to:

Corollary 78. *Let $F : [0,T] \times \mathbb{R}^n \to \mathbb{R}^n$ be pointwise bounded and suppose $F(t,\cdot)$ is Lipschitz for all $t \in [0,T]$. Let σ and η satisfy* (A.1)*, and let Φ be the a.e. solution to the ODE with vector field F, then for any $w \in L^1(\mathbb{R}^n)$,*

$$\int_{[0,T]\times\mathbb{R}^n} w(\Phi(T,s,x))\, d\left(\sigma(s,x) - \eta(s,x)\right) = 0 \tag{A.2}$$

Proof. The result for $w \in C_b^1(\mathbb{R}^n)$ follows by substituting $v(s,x) := w(\Phi(T,s,x))$ and the equation $\frac{d}{ds}\Phi(t,s,x) + \nabla_x \Phi(t,s,x) \cdot F(s,x) = 0$ into (A.1). Since $C_b^1(\mathbb{R}^n)$ is dense in $L^1(\mathbb{R}^n)$ [Bog07, Corollary 4.2.2], the statement is true for all $w \in L^1(\mathbb{R}^n)$. □

We can now establish a relationship between $\mu_{x|t}$ and Φ:

Theorem 79. *Let $F : [0,T] \times \mathbb{R}^n \to \mathbb{R}^n$ be pointwise bounded and suppose $F(t,\cdot)$ is Lipschitz for all $t \in [0,T]$. Given $\sigma, \eta \in \mathcal{M}_+([0,T] \times \mathbb{R}^n)$, the solution to* (A.1) *is given by $\mu_{x|t} = \Phi(t,\cdot,\cdot)_{\#}\,(\sigma - \eta)$ for almost every $t \in [0,T]$, where $\Phi(t,\cdot,\cdot) : [0,t] \times \mathbb{R}_n \to \mathbb{R}^n$ is defined in* (B.3) *with $\bar{F}_i^\epsilon$ replaced by F.*

Proof. We first verify $\mu_{x|t} = \Phi(t,\cdot,\cdot)_{\#}\,(\sigma - \eta)$ satisfies (A.1). We need to check the equality only on test functions of the form $\psi(t)w(x)$. We substitute $\mu_{x|t} = \Phi(t,\cdot,\cdot)_{\#}\,(\sigma - \eta)$ into the left-hand side of (A.1) and show it is equal to the right-hand side of (A.1):

$$\begin{aligned}
&\int_0^T \dot{\psi}(t) \int_{\mathbb{R}^n} w(x)\, d\mu_{x|t}(x)\, dt \\
&= \int_{[0,T]\times\mathbb{R}^n} \Big(\int_s^T \dot{\psi}(t) w(\Phi_i(t,s,x)) dt \Big) d\left(\sigma^i(s,x) - \eta^i(s,x)\right) && \text{(A.3)} \\
&= \int_{[0,T]\times\mathbb{R}^n} \Big(\psi(T) w(\Phi(T,s,x)) - \psi(s) w(\Phi(s,s,x)) + \\
&\quad - \int_s^T \psi(t) \frac{d}{dt} w(\Phi(t,s,x))\, dt \Big)\, d\left(\sigma(s,x) - \eta(s,x)\right) && \text{(A.4)} \\
&= \int_{[0,T]\times\mathbb{R}^n} \psi(s) w(x)\, d\left(\eta(s,x) - \sigma(s,x)\right) + \\
&\quad - \int_0^T \psi(t) \int_{[0,t]\times\mathbb{R}^n} \nabla_x w(\Phi_i(t,s,x)) \cdot F(t, \Phi(t,s,x))\, d\left(\sigma(s,x) - \eta(s,x)\right)\, dt && \text{(A.5)} \\
&= \int_{[0,T]\times\mathbb{R}^n} \psi(s) w(x)\, d\left(\eta(s,x) - \sigma(s,x)\right) + \int_0^T \psi(t) \langle \mu_{x|t}, \nabla_x w \cdot F \rangle\, dt && \text{(A.6)}
\end{aligned}$$

where (A.3) follows from Fubini's Theorem; (A.4) follows from integration by parts; (A.5) follows from Corollary 78 and Fubini's Theorem; (A.6) follows from $\mu_{x|t} = \Phi(t,\cdot,\cdot)_{\#}\,(\sigma - \eta)$. As a result, $\mu_{x|t} = \Phi(t,\cdot,\cdot)_{\#}\,(\sigma - \eta)$ is a solution to (A.1). To show the solution is unique dt-almost everywhere, suppose there exists measures $\mu_{x|t,1}$, $\mu_{x|t,2} \in \mathcal{M}_+(\mathbb{R}^n)$ defined for $t \in [0,T]$ that satisfy (A.1). Let $\mu_{x|t,3} := \mu_{x|t,1} - \mu_{x|t,2} \in \mathcal{M}(\mathbb{R}^n)$, then $\int_0^T \int_{\mathbb{R}^n} \left(\partial_t v(t,x) + \nabla_x v(t,x) \cdot F\right)\, d\mu_{x|t,3}\, dt = 0$, which has the zero measure as a solution. Using the proof of [HK14, Lemma 3], such $\mu_{x|t,3}$ is defined uniquely dt-a.e. Therefore $\mu_{x|t,3}$ is zero for a.e. $t \in [0,T]$, which proves the result. □

APPENDIX B

Proof of Theorem 9

Proof. This proof consists of several steps: in Step 1, we use a family of mollifiers parameterized by ϵ to smooth the vector field and all relevant measures and establish a relationship between the smooth measures using the solution to the smooth vector field via Theorem 79; in Step 2, we prove that all trajectories that satisfy this smooth vector field and enter the domain, eventually leave the domain, and vice versa; in Steps 3 and 4, we prove a connection between the time at which each trajectory enters and leaves; since Steps 2-4 are all proven for the "smoothed" versions of the vector field and measures, in Step 5 we prove that there exists a limiting measure as the parameter controlling smoothness, ϵ, goes to zero; in Step 6, we prove that this limit satisfies (b); in Step 7, we prove (a) when the vector field is continuous; in Step 8, we approximate the discontinuous vector field with a sequence of smooth functions and bound the approximation error; in Step 9, we prove (a) for arbitrary bounded vector fields.

Step 1 (Regularization). We first mollify $\mu^i_{x|t}$, σ^i, and η^i with respect to the space variable using a family of strictly positive mollifiers $\{\theta_\epsilon\} \subset C^\infty(\mathbb{R}^{n_i})$ with unit mass, zero mean, and uniformly bounded second moment, obtaining smooth measures $\mu^i_{x|t;\epsilon} := \mu^i_{x|t} * \theta_\epsilon$, $\sigma^i_\epsilon := \sigma^i * \theta_\epsilon$, and $\eta^i_\epsilon := \eta^i * \theta_\epsilon$. We also define a smooth vector field $\bar{F}^\epsilon_i$ by

$$\bar{F}^\epsilon_i(t,\cdot) := \begin{cases} \frac{\bar{F}_i(t,\cdot)\mu^i_{x|t} * \theta_\epsilon}{\mu^i_{x|t} * \theta_\epsilon}, & \text{if } \|\mu^i_{x|t}\| > 0; \\ 0, & \text{if } \|\mu^i_{x|t}\| = 0. \end{cases} \tag{B.1}$$

Notice the smooth vector field $\bar{F}^\epsilon_i$ is pointwise bounded: Let $M < +\infty$ be a pointwise bound for $\bar{F}_i$, then

$$\left|\bar{F}^\epsilon_i(t,x)\right| \leq \frac{M\mu^i_{x|t} * \theta_\epsilon}{\mu^i_{x|t} * \theta_\epsilon} \leq M\frac{\mu^i_{x|t} * \theta_\epsilon}{\mu^i_{x|t} * \theta_\epsilon} = M \tag{B.2}$$

for all $(t,x) \in [0,T] \times \mathbb{R}^{n_i}$. By applying Young's convolution inequality, one can prove $\|\mu^i_{x|t;\epsilon}\| \leq \|\mu^i_{x|t}\|$, $\|\sigma^i_\epsilon\| \leq \|\sigma^i\|$, and $\|\eta^i_\epsilon\| \leq \|\eta^i\|$.

Such $\mu^i_{x|t;\epsilon}$ is a solution of (2.16) with respect to $\bar{F}^\epsilon_i$, σ^i_ϵ, and η^i_ϵ. Since $\bar{F}^\epsilon_i$ is pointwise bounded and $\bar{F}^\epsilon_i(t,\cdot)$ is Lipschitz, Theorem 79 implies that $\mu^i_{x|t;\epsilon} = \Phi^\epsilon_i(t,\cdot,\cdot)_\#\left(\sigma^i_\epsilon - \eta^i_\epsilon\right)$ for a.e. $t \in [0,T]$,

where $\Phi_i^\epsilon(t,s,x)$ satisfies:

$$\Phi_i^\epsilon(t,s,x) = x + \int_s^t \bar{F}_i^\epsilon(\tau, \Phi_i^\epsilon(\tau,s,x))\, d\tau, \quad 0 \le s \le t \le T \tag{B.3}$$

The function $\Phi_i^\epsilon(\cdot,s,x)$ can be extended to $[0,T]$ (as opposed to $[s,T]$) due to the regularity of $\bar{F}_i^\epsilon$. Denote the extended version as $\hat{\Phi}_i^\epsilon(\cdot,s,x) \in \Gamma_i$ for any $(s,x) \in [0,T] \times \mathbb{R}^{n_i}$. The space of all such functions is denoted as $\Gamma_i^\epsilon := \{\hat{\Phi}_i^\epsilon(\cdot,s,x) \mid (s,x) \in [0,T] \times \mathbb{R}^{n_i}\} \subset \Gamma_i$ endowed with the subspace topology. It follows by the existence and uniqueness theorem for ODE that the evaluation map $e_t(0,T,\cdot)$ restricted to Γ_i^ϵ is an isomorphism for any $t \in [0,T]$. Define $\Psi^\epsilon : (t,x) \mapsto \hat{\Phi}^\epsilon(\cdot,t,x)$ from $[0,T] \times \mathbb{R}^{n_i}$ to Γ_i^ϵ, and also a projection map $\pi^1 : (s,x) \mapsto s$ from $[0,T] \times \mathbb{R}^{n_i}$ to $[0,T]$. Define

$$\begin{aligned} \rho_\epsilon^{i,+} &:= \left(\pi^1 \times \Psi^\epsilon\right)_\# \sigma_\epsilon^i \in \mathcal{M}_+([0,T] \times \Gamma_i^\epsilon), \\ \rho_\epsilon^{i,-} &:= \left(\pi^1 \times \Psi^\epsilon\right)_\# \eta_\epsilon^i \in \mathcal{M}_+([0,T] \times \Gamma_i^\epsilon). \end{aligned} \tag{B.4}$$

Step 2 (Marginals of $\rho_\epsilon^{i,+}$ and $\rho_\epsilon^{i,-}$). This step shows that all trajectories that enter the domain via σ_ϵ^i leave through η_ϵ^i by proving that the γ-marginals of $\rho_\epsilon^{i,+}$ and $\rho_\epsilon^{i,-}$ are equal. Since ρ_ϵ^{i+} and ρ_ϵ^{i-} are finite measures and $\mathbb{R} \times \Gamma_i^\epsilon$ is Radon separable metric space, using [AGS08, Theorem 5.3.1], the measures $\rho_\epsilon^{i,+}$ and $\rho_\epsilon^{i,-}$ can be disintegrated as

$$\begin{aligned} d\rho_\epsilon^{i,+}(s,\gamma) &= d\rho_{s|\gamma;\epsilon}^{i,+}(s)\, d\rho_{\gamma;\epsilon}^{i,+}(\gamma), \\ d\rho_\epsilon^{i,-}(\tau,\gamma) &= d\rho_{\tau|\gamma;\epsilon}^{i,-}(\tau)\, d\rho_{\gamma;\epsilon}^{i,-}(\gamma), \end{aligned} \tag{B.5}$$

where $\rho_{s|\gamma;\epsilon}^{i,+}$ and $\rho_{\tau|\gamma;\epsilon}^{i,-}$ are probability measures for all $\gamma \in \mathrm{spt}(\rho_{\gamma;\epsilon}^{i,+})$ and $\gamma \in \mathrm{spt}(\rho_{\gamma;\epsilon}^{i,-})$, respectively. We next show the γ-marginals are equal. Let $w \in L^1(\mathbb{R}^{n_i})$ be arbitrary. Notice

$$0 = \int_{[0,T]\times\mathbb{R}^{n_i}} w(\Phi_i^\epsilon(T,s,x))\, d\left(\sigma_\epsilon^i(s,x) - \eta_\epsilon^i(s,x)\right) \tag{B.6}$$

$$= \int_{[0,T]\times\Gamma_i^\epsilon} w(e_T(0,T,\gamma))\, d\left(\rho_\epsilon^{i,+}(s,\gamma) - \rho_\epsilon^{i,-}(s,\gamma)\right) \tag{B.7}$$

$$= \int_{\Gamma_i^\epsilon} w(e_T(0,T,\gamma))\, d\left(\rho_{\gamma;\epsilon}^{i,+}(\gamma) - \rho_{\gamma;\epsilon}^{i,-}(\gamma)\right) \tag{B.8}$$

where (B.6) follows from Corollary 78; (B.7) follows from definition of Ψ^ϵ and (B.4); (B.8) follows from (B.5). Since $e_T(0,T,\cdot)$ is an isomorphism and $w \in L^1(\mathbb{R}^{n_i})$ is arbitrary, $\rho_{\gamma;\epsilon}^{i,+} = \rho_{\gamma;\epsilon}^{i,-}$. For convenience, we denote them both by $\rho_{\gamma;\epsilon}^i$.

Step 3 (Construct $\rho_{\epsilon,\delta}^i$). We now want to combine $\rho_\epsilon^{i,+}$ and $\rho_\epsilon^{i,-}$ to generate a measure $\rho_\epsilon^i \in \mathcal{M}_+([0,T] \times [0,T] \times \Gamma_i^\epsilon)$ that describes the trajectories that evolve in the domain as well as their

entering and exiting time. Such a measure can be defined by pushing forward $\rho^{i,+}_\epsilon$ through a map that associates entering and exiting times. However, such a map may not be well defined; for example, two trajectories can enter the domain at the same time but leave at different times. To address such issues, we mollify the t-component and define a sequence of measures $\rho^i_{\epsilon,\delta}$ first, and then define ρ^i_ϵ as the limit of this sequence as $\delta \downarrow 0$ which is done in Step 4. Let $\{\theta_\delta\} \subset C^\infty(\mathbb{R})$ be a family of smooth mollifiers with unit mass and zero mean, and define $\rho^{i,+}_{s|\gamma;\epsilon,\delta} := \rho^{i,+}_{s|\gamma;\epsilon} * \theta_\delta$ and $\rho^{i,-}_{\tau|\gamma;\epsilon,\delta} = \rho^{i,-}_{\tau|\gamma;\epsilon} * \theta_\delta$. We further define measures $\rho^{i,+}_{\epsilon,\delta}, \rho^{i,-}_{\epsilon,\delta} \in \mathcal{M}_+(\mathbb{R} \times \Gamma^\epsilon_i)$ as $d\rho^{i,+}_{\epsilon,\delta}(s,\gamma) := d\rho^{i,+}_{s|\gamma;\epsilon,\delta}(s)\, d\rho^{i,+}_{\gamma;\epsilon}(\gamma)$ and $d\rho^{i,-}_{\epsilon,\delta}(\tau,\gamma) := d\rho^{i,-}_{\tau|\gamma;\epsilon,\delta}(\tau)\, d\rho^{i,-}_{\gamma;\epsilon}(\gamma)$. For a.e. $t \in [0,T]$ and any non-negative $w \in L^1(\mathbb{R}^{n_i})$:

$$
\begin{aligned}
0 &\le \langle \mu^i_{x|t;\epsilon}, w\rangle && \text{(B.9)}\\
&= \int_{[0,t]\times\mathbb{R}^{n_i}} w(\Phi^\epsilon_i(t,s,x))\, d\left(\sigma^i_\epsilon(s,x) - \eta^i_\epsilon(s,x)\right) && \text{(B.10)}\\
&= \int_{\Gamma^\epsilon_i} w(e_t(0,T,\gamma)) \left(\rho^{i,+}_{s|\gamma;\epsilon}([0,t]) - \rho^{i,-}_{\tau|\gamma;\epsilon}([0,t])\right)\, d\rho^i_{\gamma;\epsilon}(\gamma), && \text{(B.11)}
\end{aligned}
$$

where (B.9) follows from the fact that $\mu^i_{x|t;\epsilon}$ is an unsigned measure; (B.10) follows by substituting in $\mu^i_{x|t;\epsilon} = \Phi^\epsilon_i(t,\cdot,\cdot)_\# (\sigma^i_\epsilon - \eta^i_\epsilon)$; (B.11) follows from (B.4) and (B.5).

Equivalently, given any Borel set $E_\Gamma \subset \Gamma^\epsilon_i$,

$$
\int_{E_\Gamma} \left(\rho^{i,+}_{s|\gamma;\epsilon}([0,t]) - \rho^{i,-}_{\tau|\gamma;\epsilon}([0,t])\right)\, d\rho^i_{\gamma;\epsilon}(\gamma) \ge 0. \tag{B.12}
$$

Since the functions $t \mapsto \rho^{i,+}_{s|\gamma;\epsilon}([0,t])$ and $t \mapsto \rho^{i,-}_{\tau|\gamma;\epsilon}([0,t])$ are absolutely continuous, $\rho^{i,+}_{s|\gamma;\epsilon}([0,t]) \ge \rho^{i,-}_{\tau|\gamma;\epsilon}([0,t])$ is satisfied for all $t \in [0,T]$ for all $\gamma \in \text{spt}(\rho^i_{\epsilon,\Gamma})$. Using the definition of convolution and Fubini's theorem, one can prove a similar result for the mollified measures $\rho^{i,+}_{s|\gamma;\epsilon,\delta}$ and $\rho^{i,-}_{s|\gamma;\epsilon,\delta}$, i.e. $d\rho^{i,+}_{s|\gamma;\epsilon,\delta}((-\infty,t]) \ge \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}((-\infty,t])$ for all $\gamma \in \text{spt}(\rho^i_{\gamma;\epsilon})$.

Since $\rho^{i,+}_{s|\gamma;\epsilon,\delta}$ and $\rho^{i,-}_{\tau|\gamma;\epsilon,\delta}$ are smooth non-negative measures, the functions $t \mapsto \rho^{i,+}_{s|\gamma;\epsilon,\delta}((-\infty,t])$ and $t \mapsto \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}((-\infty,t])$ are continuous and non-decreasing. Also, $0 \le \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}((-\infty,t]) \le \rho^{i,+}_{s|\gamma;\epsilon,\delta}((-\infty,t]) \le \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}(\mathbb{R}) = 1$, where the last equality follows because $\rho^{i,-}_{\tau|\gamma;\epsilon,\delta}$ is a probability measure; by the Mean Value Theorem, for any $\gamma \in \text{spt}(\rho^i_{\gamma;\epsilon})$ there exists a function $r_\gamma : \mathbb{R} \to \mathbb{R}$ such that $r_\gamma(t) \ge t$ and $\rho^{i,+}_{s|\gamma;\epsilon,\delta}((-\infty,t]) = \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}((-\infty,r_\gamma(t)])$ for every $\gamma \in \text{spt}(\rho^i_{\gamma;\epsilon})$. Moreover, the function r_γ is strictly increasing and therefore invertible, i.e., there exists a function $r^{-1}_\gamma : \mathbb{R} \to \mathbb{R}$ such that $r_\gamma(r^{-1}_\gamma(t)) = r^{-1}_\gamma(r_\gamma(t)) = t$. Using Step 2, $\rho^{i,+}_{s|\gamma;\epsilon,\delta}((-\infty,t]) = \rho^{i,-}_{\tau|\gamma;\epsilon,\delta}((-\infty,r_\gamma(t)])$ can be written as

$$
\int_{\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,t]}(s) d\rho^{i,+}_{\epsilon,\delta}(s,\gamma) = \int_{\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,r_\gamma(t)]}(\tau) d\rho^{i,-}_{\epsilon,\delta}(\tau,\gamma) \tag{B.13}
$$

for any $t \in \mathbb{R}$ and any Borel subset $E_\Gamma \subset \Gamma_i^\epsilon$.

We now abuse notation and define a map $r : \mathbb{R} \times \text{spt}(\rho^i_{\gamma;\epsilon}) \to \mathbb{R}$ by letting $r(s,\gamma) := r_\gamma(s)$ for all $\gamma \in \text{spt}(\rho^i_{\gamma;\epsilon})$, and also projection maps $\pi^1 : (s,\gamma) \in \mathbb{R} \times \Gamma_i^\epsilon \mapsto s \in \mathbb{R}$, $\pi^2 : (s,\gamma) \in \mathbb{R} \times \Gamma_i^\epsilon \mapsto \gamma \in \Gamma_i^\epsilon$. We can then define a measure $\rho^i_{\epsilon,\delta} \in \mathcal{M}_+(\mathbb{R} \times \mathbb{R} \times \Gamma_i^\epsilon)$ as $\rho^i_{\epsilon,\delta} = (\pi^1 \times r \times \pi^2)_\# \rho^{i,+}_{\epsilon,\delta}$. Notice for any triplet $(s,\tau,\gamma) \in \text{spt}(\rho^i_{\epsilon,\delta})$ we know $s \le \tau$ since $r_\gamma(t) \ge t$.

We now establish the relationship between the marginals of $\rho^i_{\epsilon,\delta}$ and the measures $\rho^{i,+}_{\epsilon,\delta}$ and $\rho^{i,-}_{\epsilon,\delta}$. We use variables $(s,\tau,\gamma) \in \mathbb{R} \times \mathbb{R} \times \Gamma_i^\epsilon$ to denote any point in $\text{spt}(\rho^i_{\epsilon,\delta})$. Since $\pi^1 \times \pi^2$ is identity map, the (s,γ)-marginal of $\rho^i_{\epsilon,\delta}$ is equal to $\rho^{i,+}_{\epsilon,\delta}$. To show the (τ,γ)-marginal of $\rho^i_{\epsilon,\delta}$ is equal to $\rho^{i,-}_{\epsilon,\delta}$, it is then sufficient to show $\int_{\mathbb{R}\times\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,t]}(\tau) d\rho^i_{\epsilon,\delta}(s,\tau,\gamma) = \int_{\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,t]}(\tau) d\rho^{i,-}_{\epsilon,\delta}(\tau,\gamma)$ holds for all $t \in \mathbb{R}$ and all Borel subsets $E_\Gamma \subset \Gamma_i^\epsilon$. The equation is true because

$$\begin{aligned}
&\int_{\mathbb{R}\times\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,t]}(\tau)\, d\rho^i_{\epsilon,\delta}(s,\tau,\gamma) \\
&= \int_{\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty, r_\gamma^{-1}(t)]}(s)\, d\rho^{i,+}_{\epsilon,\delta}(s,\gamma) \qquad \text{(B.14)} \\
&= \int_{\mathbb{R}\times E_\Gamma} \mathbb{1}_{(-\infty,t]}(\tau)\, d\rho^{i,-}_{\epsilon,\delta}(\tau,\gamma), \qquad \text{(B.15)}
\end{aligned}$$

where (B.14) follows by the definition of $\rho^i_{\epsilon,\delta}$ and because r_γ is strictly monotonic and therefore $r_\gamma(s) \in (-\infty,t]$ if and only if $s \in (-\infty, r_\gamma^{-1}(t)]$; (B.15) follows by substituting in (B.13) and from the fact that r_γ is invertible;

Step 4 (Properties of the limiting measure of $\{\rho^i_{\epsilon,\delta}\}_\delta$). We now show that the limit of $\rho^i_{\epsilon,\delta}$ exists as $\delta \downarrow 0$ and that for this limiting measure $\mu^i_{x|t;\epsilon} = (e_t)_\# \rho^i_\epsilon$ for a.e. $t \in [0,T]$. We also show that specific marginals of this limiting measure are equal to $\rho^{i,+}_\epsilon$ and $\rho^{i,-}_\epsilon$ and that for any (s,τ,γ) in the support of this limiting measure, $s \le \tau$. To prove this condition, we use the notion of tightness of measures [Man07, pp. 605-606]:

Integral Condition for Tightness: Let X be a separable metric space. A family $\mathcal{K} \subset \mathcal{M}_+(X)$ is tight if and only if there exists a function $\Theta : X \to [0,+\infty]$ whose sublevel sets are compact in X such that $\sup_{\mu\in\mathcal{K}} \int_X \Theta(x)\, d\mu(x)$ is finite.

Tightness Criterion: Let X, X_1, X_2 be separable metric spaces and let $r^i : X \to X_i$, $i = 1,2$ be continuous maps such that the product map $r : r^1 \times r^2 : X \to X_1 \times X_2$ is proper. Let $\mathcal{K} \subset \mathcal{M}_+(X)$ be such that $\mathcal{K}_i := r^i_\#(\mathcal{K})$ is tight in $\mathcal{M}_+(X_i)$ for $i = 1,2$. Then also $\mathcal{K}$ is tight in $\mathcal{M}_+(X)$. Notice the statement also holds for finitely many maps by induction.

Choosing maps r^1, r^2 defined on $\mathbb{R} \times \mathbb{R} \times \Gamma_i^\epsilon$ as $r^1 : (s,\tau,\gamma) \mapsto (s,\gamma) \in \mathbb{R} \times \Gamma_i^\epsilon$ and $r^2 : (s,\tau,\gamma) \mapsto \tau \in \mathbb{R}$. Notice that $r = r^1 \times r^2$ is an isomorphism and therefore proper. The family $\{r^1_\# \rho^i_{\epsilon,\delta}\}_\delta$ is given by $\{\rho^{i,+}_{\epsilon,\delta}\}_\delta$ which are tight by definition, and the family $\{r^2_\# \rho^i_{\epsilon,\delta}\}_\delta$ is given by the first marginal of $\{\rho^{i,-}_{\epsilon,\delta}\}_\delta$ which are also tight. Applying the tightness criterion, the family $\{\rho^i_{\epsilon,\delta}\}_\delta$ is

tight, and therefore narrowly sequentially relatively compact according to Prokhorov Compactness Theorem. Let ρ^i_ϵ be any limit of the family $\{\rho^i_{\epsilon,\delta}\}$ as $\delta \downarrow 0$. Since the (s,γ)-marginal of $\rho^i_{\epsilon,\delta}$ is equal to $\rho^{i,+}_{\epsilon,\delta}$ and the (τ,γ)-marginal of $\rho^i_{\epsilon,\delta}$ is equal to $\rho^{i,-}_{\epsilon,\delta}$, we let $\delta \downarrow 0$ and therefore the (s,γ)-marginal of ρ^i_ϵ is equal to $\rho^{i,+}_\epsilon$ and the (τ,γ)-marginal of ρ^i_ϵ is equal to $\rho^{i,-}_\epsilon$, i.e.,

$$\begin{aligned}\int_{[0,T]\times[0,T]\times\Gamma^\epsilon_i} \varphi(s,\gamma)\, d\rho^i_\epsilon(s,\tau,\gamma) &= \int_{[0,T]\times\Gamma^\epsilon_i} \varphi(s,\gamma)\, d\rho^{i,+}_\epsilon(s,\gamma)\\ \int_{[0,T]\times[0,T]\times\Gamma^\epsilon_i} \varphi(\tau,\gamma)\, d\rho^i_\epsilon(s,\tau,\gamma) &= \int_{[0,T]\times\Gamma^\epsilon_i} \varphi(\tau,\gamma)\, d\rho^{i,-}_\epsilon(\tau,\gamma)\end{aligned} \tag{B.16}$$

for all $\varphi \in L^1(\mathbb{R}\times\Gamma_i)$.

Let $(s,\tau,\gamma) \in \mathrm{spt}(\rho^i_\epsilon)$ be arbitrary. To show $s \le \tau$, let $\varphi' \in C_b(\mathbb{R}^2)$ be such that $\mathrm{spt}(\varphi') \subset \{(s,\tau)\in\mathbb{R}^2 \mid s > \tau\}$. Since $\int_{\mathbb{R}\times\mathbb{R}\times\Gamma_i} \varphi'(s,\tau)\, d\rho^i_{\epsilon,\delta}(s,\tau,\gamma) = 0$ for all δ, it follows from narrow convergence that $\int_{[0,T]\times[0,T]\times\Gamma_i} \varphi'(s,\tau)\, d\rho^i_\epsilon(s,\tau,\gamma) = 0$. Since $\mathbb{1}_{\{(s,\tau)\in[0,T]^2 | s>\tau+\Delta\}}$ is a limit point of such functions φ' with respect to $L_1(\rho^i_\epsilon;\mathbb{R})$ for any $\Delta > 0$ [Bog07, Corollary 4.2.2], ρ^i_ϵ is supported on (s,τ,γ) such that $s \le \tau$.

For a.e. $t\in[0,T]$ and any $w \in L^1(\mathbb{R}^{n_i})$,

$$\begin{aligned}&\int_{\mathbb{R}^{n_i}} w(x)\, d\mu^i_{x|t;\epsilon}(x)\\ =&\int_{[0,t]\times[0,T]\times\Gamma^\epsilon_i} w(e_t(s,T,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma) - \int_{[0,t]\times[0,t]\times\Gamma^\epsilon_i} w(e_t(\tau,T,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma) &\text{(B.17)}\\ =&\int_{[0,t]\times[0,t]\times\Gamma^\epsilon_i} \left(w(e_t(s,T,\gamma)) - w(e_t(\tau,T,\gamma))\right)\, d\rho^i_\epsilon(s,\tau,\gamma) +\\ &+\int_{[0,t]\times(t,T]\times\Gamma^\epsilon_i} w(e_t(s,T,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma) &\text{(B.18)}\\ =&0+\int_{[0,t]\times[t,T]\times\Gamma^\epsilon_i} w(e_t(s,\tau,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma) - \int_{[0,t]\times\{t\}\times\Gamma^\epsilon_i} w(e_t(0,T,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma), &\text{(B.19)}\end{aligned}$$

where (B.17) follows from $\mu^i_{x|t;\epsilon} = \Phi^\epsilon_i(t,\cdot,\cdot)_\#(\sigma^i_\epsilon - \eta^i_\epsilon)$, (B.4), and (B.16); (B.18) follows by splitting the domain of integration; Since $e_t(t_1,T,\cdot) = e_t(0,T,\cdot)$ and $e_t(t_1,T,\cdot) = e_t(t_1,t_2,\cdot)$ for all $0 \le t_1 \le t \le t_2 \le T$, the first term of (B.18) is zero because the integrand is zero, (B.19) follows by adding and subtracting $[0,t]\times\{t\}\times\Gamma^\epsilon_i$ to the domain of integration. Since $\rho^i_\epsilon([0,t]\times\{t\}\times\Gamma^\epsilon_i)$ is non-zero for at most countably many t's (otherwise ρ^i_ϵ would not be bounded), $\mu^i_{x|t;\epsilon} = (e_t)_\#\rho^i_\epsilon$ for a.e. $t\in[0,T]$.

Step 5 (Tightness of the family $\{\rho^i_\epsilon\}_\epsilon$). We show that the limit of ρ^i_ϵ exists as $\epsilon \downarrow 0$. To begin, choose maps r^1, r^2, r^3 defined in $[0,T]\times[0,T]\times\Gamma_i$ as $r^1:(s,\tau,\gamma)\mapsto s\in[0,T]$, $r^2:(s,\tau,\gamma)\mapsto \tau\in[0,T]$, and $r^3:(s,\tau,\gamma)\mapsto\gamma\in\Gamma_i$. Observe that $r = r^1\times r^2\times r^3$ is the identity map and

therefore proper. The family $\{r^1_{\#}\rho^i_\epsilon\}_\epsilon$ and $\{r^2_{\#}\rho^i_\epsilon\}_\epsilon$ are given by the first marginals of σ^i_ϵ and η^i_ϵ, respectively, which are tight and are independent of ϵ. To establish a similar result for $r^3_{\#}\rho^i_\epsilon$, let $\Theta : \Gamma_i \to \mathbb{R}\bigcup\{+\infty\}$ as $\Theta(\gamma) = \|\gamma\|$ if $|\dot\gamma(t)| \le M$ a.e., and $\Theta(\gamma) = +\infty$ otherwise. We next show this function Θ satisfies the requirement of the integral condition for tightness. Let $S := \{\gamma \in \Gamma_i \mid \Theta(\gamma) \le C\}$. Since any sequence $\{\gamma_n\} \subset S$ is uniformly bounded and equicontinuous, S is precompact according to Arzela-Ascoli Theorem. To show S is closed, let $\{\gamma_n\}$ be a convergent sequence in S, and by definition $\dot\gamma_n \to \dot\gamma$ in $L^1([0,T])$. There is a subsequence of $\dot\gamma_n$ that converges pointwise a.e. to $\dot\gamma$ [Fol13, Proposition 2.29], therefore $|\dot\gamma(t)| \le M$ a.e., which implies that the set S is closed. Notice

$$\int_{\Gamma_i} \Theta(\gamma)\, d\left(r^3_{\#}\rho^i_\epsilon\right)(\gamma) = \int_{[0,T]\times\Gamma^i_\epsilon} \Theta(\gamma)\, d\rho^{i,+}_\epsilon(s,\gamma) \tag{B.20}$$

$$= \int_{[0,T]\times\mathbb{R}^{n_i}} \left(\left|\hat\Phi^\epsilon_i(0,s,x)\right| + \int_0^T \left|\dot{\hat\Phi}^i_\epsilon(t,s,x)\right| dt \right) d\sigma^i_\epsilon(s,x) \tag{B.21}$$

$$\le \int_{[0,T]\times\mathbb{R}^{n_i}} \left(|\hat\Phi^\epsilon_i(s,s,x)| + \int_0^s \left|\bar F^\epsilon_i(\hat\Phi^i_\epsilon(t,s,x))\right| dt + \int_0^T \left|\bar F^\epsilon_i(\hat\Phi^i_\epsilon(t,s,x))\right| dt \right) d\sigma^i_\epsilon(s,x) \tag{B.22}$$

$$\le \int_{[0,T]\times\mathbb{R}^{n_i}} |x|\, d\sigma^i_\epsilon(s,x) + 2MT\|\sigma^i_\epsilon\| \tag{B.23}$$

$$\le \int_{[0,T]\times\mathbb{R}^{n_i}} (|x|^2+1)\, d\sigma^i_\epsilon(s,x) + 2MT\|\sigma^i_\epsilon\| \tag{B.24}$$

$$\le \int_{[0,T]\times X_i}\int_{\mathbb{R}^{n_i}} |x+y|^2\theta_\epsilon(y)dyd\sigma^i_\epsilon(s,x) + (1+2MT)\|\sigma^i\| \tag{B.25}$$

$$\begin{aligned} &= \int_{[0,T]\times X_i} |x|^2\, d\sigma^i(s,x) + \left(\int_{\mathbb{R}^{n_i}} |y|^2\theta_\epsilon(y)\, dy\right)\|\sigma^i\| + \\ &\quad + \int_{[0,T]\times X_i}\int_{\mathbb{R}^{n_i}} 2x^T y\theta_\epsilon(y)dyd\sigma^i(s,x) + (1+2MT)\|\sigma^i\|, \end{aligned} \tag{B.26}$$

where (B.20) follows from (B.16); (B.21) follows from (B.4) and (B.3); (B.22) follows from triangle inequality; (B.23) follows from (B.2); (B.24) is true because $|x|^2 + 1 \ge |x|$ for all $x \in \mathbb{R}^{n_i}$, and σ^i_ϵ is non-negative; (B.25) follows from the definition of convolution and $\|\sigma^i_\epsilon\| \le \|\sigma^i\|$; Since σ^i is bounded and X_i is compact therefore $|x|^2$ is bounded for all $x \in X_i$, the first and last term in (B.26) are bounded. Because θ_ϵ is assumed to have zero mean and bounded second moment, the second term in (B.26) is bounded and the third term in (B.26) is zero. As a result, the left hand side of (B.20) is bounded. Using the integral condition for tightness, $\{r^3_{\#}\rho^i_\epsilon\}_\epsilon$ is tight, and $\{\rho^i_\epsilon\}_\epsilon$ is tight via the tightness criterion.

Step 6 (Part (b)). We prove the limit of ρ^i_ϵ as ϵ goes to zero satisfies Part (b). Using the Prokhorov Compactness Theorem, the family ρ^i_ϵ is narrowly sequentially relatively compact. We

choose a narrowly convergent sequence in $\{\rho^i_\epsilon\}_\epsilon$ and define its limit by $\rho^i \in \mathcal{M}_+([0,T]\times[0,T]\times\Gamma_i)$. For a.e. $t\in[0,T]$ and all $w\in C_b(\mathbb{R}^{n_i})$, it follows from $\mu^i_{x|t;\epsilon} = (e_t)_\# \rho^i_\epsilon$ that

$$\int_{\mathbb{R}^{n_i}} w(x)\, d\mu^i_{x|t;\epsilon}(x) = \int_{[0,T]\times[0,T]\times\Gamma_i} w(e_t(s,\tau,\gamma))\, d\rho^i_\epsilon(s,\tau,\gamma). \tag{B.27}$$

Since e_t is continuous, $w\circ e_t \in C_b([0,T]\times[0,T]\times\Gamma_i)$. We then pass to the limit $\epsilon\downarrow 0$ on both sides of (B.27) to obtain $\int_{X_i} w(x)\, d\mu^i_{x|t}(x) = \int_{[0,T]\times[0,T]\times\Gamma_i} w(e_t(s,\tau,\gamma))\, d\rho^i(s,\tau,\gamma)$ for a.e. $t\in[0,T]$. Since $C_b(\mathbb{R}^{n_i})$ is dense in $L^1(\mathbb{R}^{n_i})$ [Bog07, Corollary 4.2.2], $\mu^i_{x|t} = (e_t)_\#\rho^i$ for a.e. $t\in[0,T]$.

Step 7 (Part (a) with continuous vector field). Using a similar argument in Step 4, we may show $s\le\tau$ for any triplet $(s,\tau,\gamma)\in \mathrm{spt}(\rho^i)$. Moreover, it follows from $\mu^i_{x|t} = (e_t)_\#\rho^i$ that $\gamma(t)\in \mathrm{spt}((e_t)_\#\rho^i)\subset X_i$ for a.e. $t\in[s,\tau]$. Since γ is absolutely continuous and X_i is compact, $\gamma(t)$ stays in X_i for all $t\in[s,\tau]$. To prove the rest of (a), we only need to show

$$\int_{[0,t]\times[t,T]\times\Gamma_i} \left|\gamma(t)-\gamma(s)-\int_s^t \bar F_i(\tau',\gamma(\tau'))d\tau'\right| d\rho^i(s,\tau,\gamma) = 0 \tag{B.28}$$

for all $t\in[0,T]$. Let $v\in C_b([0,T]\times X_i;\mathbb{R}^{n_i})$, then

$$\begin{aligned}
&\int_{[0,t]\times[t,T]\times\Gamma_i} \left|\gamma(t)-\gamma(s)-\int_s^t v(\tau',\gamma(\tau'))d\tau'\right| d\rho^i_\epsilon(s,\tau,\gamma)\\
\le&\int_{[0,t]\times[t,T]\times\Gamma_i}\int_s^t \left|\bar F^\epsilon_i(\tau',\gamma(\tau')) - v(\tau',\gamma(\tau'))\right| d\tau' d\rho^i_\epsilon(s,\tau,\gamma) &\text{(B.29)}\\
\le&\int_0^t \int_{[0,\tau']\times[\tau',T]\times\Gamma_i} \left|\bar F^\epsilon_i(\tau',\gamma(\tau')) - v(\tau',\gamma(\tau'))\right| d\rho^i_\epsilon(s,\tau,\gamma)d\tau' &\text{(B.30)}\\
=&\int_0^t\int_{\mathbb{R}^{n_i}} \left|\bar F^\epsilon_i(\tau,x) - v(\tau,x)\right| d\mu^i_{x|\tau;\epsilon}(x)\, d\tau &\text{(B.31)}
\end{aligned}$$

$$\le \int_{[0,T]\times\mathbb{R}^{n_i}} \left|\bar F_i(\tau,x)-v(\tau,x)\right| d\mu^i_{\tau,x}(\tau,x) + \Big(\sup_{\substack{\tau\in[0,T]\\ x\in\mathbb{R}^{n_i}}} |v^\epsilon(\tau,x)-v(\tau,x)|\Big)\,\|\mu^i_{\tau,x}\| \tag{B.32}$$

for any $t\in[0,T]$ where (B.29) follows by substituting in $\int_s^t \bar F^\epsilon_i(\tau',\gamma(\tau'))d\tau' = \gamma(t)-\gamma(s)$ and applying the triangle inequality for integrals; (B.30) follows by first applying Fubini's theorem to change the order of integration, and then relaxing the domain of integration (since ρ^i_ϵ is nonnegative); (B.31) follows from $\mu^i_{x|t;\epsilon} = (e_t)_\#\rho^i_\epsilon$ and a change of variables $\tau'=\tau$; in (B.32) we add and subtract $v^\epsilon(\tau,\cdot) := \frac{(v(\tau,\cdot)\,\mu^i_{x|\tau})*\theta_\epsilon}{\mu^i_{x|\tau;\epsilon}}$, and then apply the triangle inequality and [Man07, Lemma 3.9]. Since the family $\{\rho^i_\epsilon\}_\epsilon$ is tight and the integrand is a bounded continuous function, and v is uni-

formly continuous v^ϵ converges to v uniformly as $\epsilon \downarrow 0$, and the second term of (B.32) converges to 0, therefore for a.e. $t \in [0,T]$,

$$\begin{aligned}\int_{[0,t]\times[t,T]\times\Gamma_i} \left|\gamma(t)-\gamma(s)-\int_s^t v(\tau',\gamma(\tau'))d\tau'\right| d\rho^i(s,\tau,\gamma) \\ \leq \int_{[0,T]\times X_i} |\bar{F}_i(\tau,x)-v(\tau,x)|\, d\mu^i_{\tau,x}(\tau,x).\end{aligned} \tag{B.33}$$

If $\bar{F}_i$ is uniformly continuous, let $v := \bar{F}_i$, and (B.28) follows.

Step 8 (Error bound of vector field approximation). When there is no regularity in $\bar{F}_i$ other than boundedness, we choose a sequence of continuous functions converging to $\bar{F}_i$ in $L^1(\mu^i_{t,x};\mathbb{R}^{n_i})$, and prove an error bound of the approximation: Let $\{v_k\}_{k\in\mathbb{N}} \subset C([0,T]\times X_i;\mathbb{R}^{n_i})$ be a sequence of continuous functions converging to $\bar{F}_i$ in $L^1(\mu^i_{t,x};\mathbb{R}^{n_i})$, whose existence is guaranteed by [Bog07, Corollary 4.2.2]. Given any $t \in [0,T]$, we compute the following error between v_k and $\bar{F}_i$:

$$\begin{aligned}&\int_{[0,t]\times[t,T]\times\Gamma_i}\int_s^t \left|v_k(\tau',\gamma(\tau'))-\bar{F}_i(\tau',\gamma(\tau'))\right| d\tau'\, d\rho^i(s,\tau,\gamma) \\ \leq &\int_0^t \int_{[0,\tau']\times[\tau',T]\times\Gamma_i} \left|v_k(\tau',\gamma(\tau'))-\bar{F}_i(\tau',\gamma(\tau'))\right| d\rho^i(s,\tau,\gamma)\, d\tau' \qquad \text{(B.34)} \\ = &\int_{[0,T]\times X_i} \left|v_k(\tau,x)-\bar{F}_i(\tau,x)\right| d\mu^i_{\tau,x}(\tau,x), \qquad \text{(B.35)}\end{aligned}$$

where (B.34) follows by first applying Fubini's Theorem to change the order of integrations, and then relaxing the domain of integration (since ρ^i is nonnegative); (B.35) follows by substituting in $\mu^i_{x|t} = (e_t)_\# \rho^i$. Observe that as $k \to \infty$ this error goes to zero.

Step 9 (Condition (a) with bounded vector field). We may now combine Step 7 and Step 8 together and prove Part (a) in a more general setting. Using the results in Step 7 and Step 8, we obtain for any $t \in [0,T]$,

$$\begin{aligned}\int_{[0,t]\times[t,T]\times\Gamma_i} \left|\gamma(t)-\gamma(s)-\int_s^t \bar{F}_i(\tau',\gamma(\tau'))d\tau'\right| d\rho^i(s,\tau,\gamma) \\ \leq 2\int_{[0,T]\times X_i} \left|\bar{F}_i(\tau,x)-v_k(\tau,x)\right| d\mu^i_{\tau,x}(\tau,x),\end{aligned} \tag{B.36}$$

where (B.36) follows by adding and subtracting the term $\int_s^t v_k(\tau',\gamma(\tau'))d\tau'$, applying the triangle inequality, and using the results in Step 7 and Step 8. When we let $k \to \infty$, (B.36) goes to zero, therefore Part (a) holds. □

APPENDIX C

Proof of Theorem 12

Proof. This proof consists of several steps: in Step 1 we show that trajectories defined in support of ρ^i and ρ^j satisfy the reset map for all $(i,j) \in \mathcal{E}$; in Step 2 we show trajectories in each mode can be connected to obtain hybrid trajectories that are defined on $[0,T]$; in Step 3 we prove that those hybrid trajectories are admissible by showing they all start from $\text{spt}(\mu_0^i)$ and end in $\text{spt}(\mu_T^i)$ thus proving (a); in Step 4 we define a measure ρ and prove that it satisfies (b) and (c); in Step 5 we prove (d) using (b) and (c).

Step 1 (Reset maps are satisfied). According to Corollary 10, it suffices to show $\sigma^j = \delta_0 \otimes \mu_0^j + \sum_{(i,j)\in\mathcal{E}} \tilde{R}_{(i,j)\#}\eta^i, \forall j \in \mathcal{I}$. This can be proved by using (2.13) and Assumption 2 to obtain $\tilde{R}_{(i,j)\#}\eta^i = \tilde{R}_{(i,j)\#}\mu^{S_{(i,j)}}$. As a result, all trajectories in the support of ρ^i are reinitialized to another trajectory in the support of ρ^j after it reaches the guard $S_{(i,j)}$; On the other hand, a trajectory can only start in mode i either from the given initial condition x_0 at time 0, or by transitioning from another mode j if $(j,i) \in \mathcal{E}$. We can therefore connect trajectories in each mode together to obtain hybrid trajectories.

Step 2 (Hybrid trajectories are defined on $[0,T]$). This step shows that all hybrid trajectories are defined on $[0,T]$. To prove this, we first show that there is a $\Delta t > 0$ such that $\tau - s \geq \Delta t$ for any $i \in \mathcal{I}$ and $(s,\tau,\gamma) \in \text{spt}(\rho^i)$, $\tau \neq T$. Let $(s,\tau,\gamma) \in \text{spt}(\rho^i)$ for some $i \in \mathcal{I}$, and let $0 \leq s \leq \tau < T$. According to Corollary 10, $\gamma(s) \in \{x_0\} \bigcup_{(i',i)\in\mathcal{E}} R_{(i',i)}(S_{(i',i)})$ and $\gamma(\tau) \in \bigcup_{(i,i')\in\mathcal{E}} S_{(i,i')}$. According to Definition 1 and Assumptions 2 and 5, $\gamma(s)$ and $\gamma(\tau)$ belong to disjoint compact sets (since the image of a compact set under a continuous map is compact) and therefore there exists a $d_i > 0$ such that $|\gamma(\tau) - \gamma(s)| \geq d_i$. Let $M_i > 0$ be a bound for $\bar{F}_i(t,x)$ over $[0,T] \times X_i$, and define $\Delta t := \min_{i\in\mathcal{I}}(d_i/M_i)$. Then it follows from the Fundamental Theorem of Calculus that $(\tau - s) \geq \Delta t$.

We can apply proof by contradiction to show all hybrid trajectories are defined on $[0,T]$. Let a hybrid trajectory be defined on a strict subinterval of $[0,T]$, then according to Corollary 10 its endpoints must belong to either S_e or $R_{e\#}\mu^{S_e}$ for some $e \in \mathcal{E}$. It then follows from Step 1 that its domain can always be extended by at least Δt due to transitioning from or to another point

Notice it follows from the above discussion that for any $i \in \mathcal{I}$ and $(0, \tau, \gamma) \in \text{spt}(\rho^i)$, $\tau \geq \Delta t$. As a result, $\text{spt}(\mu^{S_e}) \subset [\Delta t, T] \times S_e$ for all $e \in \mathcal{E}$. Then, as a result of Step 1, for all $e \in \mathcal{E}$, $\text{spt}(\tilde{R}_{e\#}\mu^{S_e}) \subset [\Delta t, T] \times R_e(S_e)$.

Step 3 (Part (a)). For any triplet $(0, \tau, \gamma) \in \text{spt}(\rho^i)$, $(0, \gamma(0)) \in \text{spt}(\sigma^i)$ according to Corollary 10. It then follows from $\text{spt}(\tilde{R}_{e\#}\mu^{S_e}) \subset [\Delta t, T] \times R_e(S_e)$ that $\gamma(0) \in \text{spt}(\mu_0^i)$. Now suppose $(s, T, \gamma) \in \text{spt}(\rho^i)$ but $\gamma(T) \notin \text{spt}(\mu_T^i)$. According to Corollary 10 and Step 1 γ is reinitialized to another trajectory γ' in some mode $i' \in \mathcal{I}$. As a result of Corollary 10, $(T, \gamma'(T)) \in \text{spt}(\sigma^{i'}) \bigcup \text{spt}(\eta^{i'})$, therefore as a result of Assumptions 2 and 4, $\gamma'(T) \in \text{spt}(\mu_T^{i'})$.

Step 4 (Part (b) and (c)). As a result of Step 3, there exists a measure $\rho \in \mathcal{M}_+(\mathcal{X})$ such that $(e_t^i)_{\#}\rho = (e_t)_{\#}\rho^i$ for all $t \in [0, T]$. Therefore, Part (b) follows from Theorem 9. To prove Part (c), notice $\rho(\mathcal{X}) = \sum_{i\in\mathcal{I}} \left((e_0^i)_{\#}\rho\right)(X_i) = \sum_{i\in\mathcal{I}} \rho^i(\{0\} \times [0, T] \times \Gamma_i)$. According to Corollary 10, $\int_{[0,T]\times[0,T]\times\Gamma_i} \mathbb{1}_{\{0\}}(s)\, d\rho^i(s, \tau, \gamma) = \int_{[0,T]\times X_i} \mathbb{1}_{\{0\}}(s)\, d\sigma^i(s, x) = \sigma^i(\{0\} \times X_i)$. If $\sum_{i\in\mathcal{I}} \mu_0^i(X_i) = 1$, then $\rho(\mathcal{X}) = \sum_{i\in\mathcal{I}} \sigma^i(\{0\} \times X_i) = \sum_{i\in\mathcal{I}} \mu_0^i(X_i) = 1$.

Step 5 (Part (d)). Let $A \times B$ be in the Borel σ-algebra of $[0, T] \times X_i$, then

$$\mu_{t,x}^i(A \times B) = \int_{\mathcal{X}_T} \int_0^T \mathbb{1}_{A\times B}(t, \gamma_i(t))\, dt\, d\rho(\gamma), \tag{C.1}$$

which follows by substituting in $(e_t^i)_{\#}\rho = \mu_{x|t}^i$ and applying Fubini's Theorem. Since ρ is a probability measure, $\sum_{i\in\mathcal{I}} \mu_{t,x}^i([0, T] \times X_i) = T$.

For all B in the Borel σ-algebra of X_i,

$$\mu_0^i(B) = \int_{[0,T]\times X_i} \mathbb{1}_{\{0\}\times B}(s, x)\, d\sigma^i(s, x) \tag{C.2}$$

$$= \int_{\mathcal{X}} \mathbb{1}_B(\gamma_i(0))\, d\rho(\gamma), \tag{C.3}$$

where (C.2) follows from definition of δ_0, from (2.13) and $\text{spt}(\tilde{R}_{e\#}\mu^{S_e}) \subset [\Delta t, T] \times R_e(S_e)$; (C.3) follows from Corollary 10 and because $(e_t^i)_{\#}\rho = (e_t)_{\#}\rho^i$. Similarly, for all B in the Borel σ-algebra of X_{T_i}, $\mu_T^i(B) = \int_{\mathcal{X}_T} \mathbb{1}_B(\gamma_i(T))\, d\rho(\gamma)$. Since ρ is a probability measure, $\sum_{i\in\mathcal{I}} \mu_T^i(X_{T_i}) = 1$.

Finally, for all $(i, i') \in \mathcal{S}$ and $A \times B$ in the Borel σ-algebra of $[0, T] \times S_{(i,i')}$,

$$\mu^{S_{(i,i')}}(A \times B) = \int_{[0,T]\times X_i} \mathbb{1}_{A\times B}(\tau, x)\, d\eta^i(\tau, x) \tag{C.4}$$

$$= \int_{[0,T]\times[0,T]\times\Gamma_i} \mathbb{1}_{A\times B}(\tau, \gamma(\tau))\, d\rho^i(s, \tau, \gamma) \tag{C.5}$$

$$= \int_{[0,T]\times[0,T]\times\Gamma_i} \#\{(t, e_t(s, \tau, \gamma)) \in A \times B\}\, d\rho^i(s, \tau, \gamma) \tag{C.6}$$

$$= \int_{\mathcal{X}} \#\{t \in A \mid \lim_{\tau\to t^-} \gamma_i(\tau) \in B\}\, d\rho(\gamma), \tag{C.7}$$

where (C.4) follows from (2.13), Assumption 4, and the fact that $B \subset S_{(i,i')}$; (C.5) follows from Corollary 10; (C.6) follows from Assumption 2; (C.7) follows because $(e^i_t)_{\#}\rho = (e_t)_{\#}\rho^i$ and because all $\gamma_i \in \Gamma_i$ are absolutely continuous. From Step 2, each $\gamma \in \text{spt}(\rho)$ undergoes at most $\frac{T}{\Delta t}$ transitions, where Δt is defined as in Step 2. Therefore $\sum_{(i,i')\in\mathcal{E}} \#\{t \in [0, T] \mid \lim_{\tau\to t^-} \gamma_i(\tau) \in S_{(i,i')}\} \leq \frac{T}{\Delta t}$ for all $\gamma \in \text{spt}(\rho)$. Since ρ is a probability measure, $\sum_{e\in\mathcal{E}} \mu^{S_e}([0, T] \times S_e) \leq \frac{T}{\Delta t}$. $\square$

APPENDIX D

Proofs of Theorems 44, 45, and 46

D.1 Proof of Theorem 43 (Unconstrained Rate of Convergence)

Proof. The first term in (4.24) follows from Theorem 42 and guarantees $|\mathbf{x}| \leq \delta$. To prove $\max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \epsilon$ for all $\mathbf{y} \in \mathcal{L}$ using the second and third terms, we need to define C_1 and C_2. First, we use the fact that x^* is not on the boundary of $\mathbf{u}$ to find $R > 0$ such that $\mathbf{x} \subset \mathcal{B}_R(x^*)$ (the closed Euclidean-norm ball centered at x^* with radius R). Using the first-order necessary condition for optimality [NW06, Theorem 2.2] we know $\nabla p(x^*) = 0$. Let $\sigma_{\max}$ be the maximum singular value of $\nabla^2 p(x^*)$. Consider an arbitrary point $x \in \mathbf{u}$ and let $t := x - x^*$. Using Taylor's theorem, we have

$$p(x) = p(x^*) + \nabla p(x^*)^\top t + \frac{1}{2} t^\top \nabla^2 p(x^*)\, t + o(\|t\|^2) \tag{D.1}$$

Since $\nabla p(x^*) = 0$, (D.1) implies that $p(x) - p(x^*) \leq \frac{1}{2}\sigma_{\max}\|t\|^2 + o(\|t\|^2)$. For sufficiently small t, the second order term dominates higher order terms. That is, there exists $R > 0$ such that

$$p(x) - p(x^*) \leq \sigma_{\max}\|t\|^2 \tag{D.2}$$

for all $x \in \mathcal{B}_R(x^*)$.

Now we use R and $\sigma_{\max}$ to construct C_1 and C_2. Let n be the current iteration number such that

$$n \geq \left\lceil \max\left\{ -\log_2\left(\frac{R}{\sqrt{l}}\right), -\frac{1}{2}\log_2\left(\frac{\epsilon}{\sigma_{\max} l + 2\zeta_p}\right)\right\}\right\rceil. \tag{D.3}$$

Notice $\mathbf{x} \subset \mathcal{B}_R(x^*)$ because $\max_{x\in\mathbf{x}} \|x - x^*\| \leq |\mathbf{x}| = \sqrt{l}\cdot 2^{-n} \leq R$. Therefore, (D.2) is satisfied

for all $x \in \mathbf{x}$. We also have

$$\max B(\mathbf{x}) - \min B(\mathbf{y}) \leq \max_{x \in \mathbf{x}} p(x) - p(x^*) + 2\zeta_p \cdot 2^{-2n} \tag{D.4}$$

$$\leq \sigma_{\max} \max_{x \in \mathbf{x}} \|x - x^*\|^2 + 2\zeta_p \cdot 2^{-2n} \tag{D.5}$$

$$\leq \sigma_{\max} l \cdot 2^{-2n} + 2\zeta_p \cdot 2^{-2n} \tag{D.6}$$

$$\leq \epsilon \tag{D.7}$$

where (D.4) is a verbatim copy of (4.21); (D.5) follows from (D.2); (D.6) follows from Remark 36; and (D.7) follows from (D.3). This implies Algorithm 1 terminates no later than iteration n. To conclude the proof, we define

$$C_1 := -\log_2 \left(\frac{R}{\sqrt{l}} \right), \quad C_2 := \frac{1}{2} \log_2 \left(\sigma_{\max} l + 2\zeta_p \right) \tag{D.8}$$

□

D.2 Proof of Theorem 44 (Unconstrained Memory Usage)

Proof. Since $m < \infty$, let $\sigma_{\max}$ and $\sigma_{\min}$ be the maximum and minimum singular values, respectively, of $\nabla^2 p(x_s^*)$ over all $s = 1, \cdots, m$. Notice $\sigma_{\max} \geq \sigma_{\min} > 0$. Using similar arguments in the proof of Theorem 43, an analogue of (D.2) can be obtained. That is, there exists $R > 0$ such that

$$\frac{1}{4}\sigma_{\min} \|x_s - x_s^*\|^2 \leq p(x_s) - p(x_s^*) \leq \sigma_{\max} \|x_s - x_s^*\|^2 \tag{D.9}$$

for all $s = 1, \cdots, m$ and $x_s \in \mathcal{B}_R(x_s^*)$.

Without loss of generality, let x^* represent any minimizer $x_1^*, \cdots, x_m^*$ of (4.16). Let $\mathbf{y} \subset \mathcal{B}_R(x^*)$ be a subbox in $\mathbf{u}$ such that

$$\min_{y \in \mathbf{y}} \|y - x^*\| > \sqrt{\frac{4(\sigma_{\max} l + 2\zeta_p)}{\sigma_{\min}}} \cdot 2^{-n} =: R'. \tag{D.10}$$

We claim that $\mathbf{y}$ cannot be in the list $\mathcal{L}$. To prove this, let $\mathbf{x} \subset \mathcal{B}_R(x^*)$ be the subbox that contains

x^*. Then

$$\min B(\mathbf{y}) \geq \min_{y \in \mathbf{y}} p(y) - \zeta_p \cdot 2^{-2n} \tag{D.11}$$

$$\geq p(x^*) + \frac{1}{4}\sigma_{\min}\left(\min_{y \in \mathbf{y}} \|y - x^*\|^2\right) - \zeta_p \cdot 2^{-2n} \tag{D.12}$$

$$> p(x^*) + \sigma_{\max} \cdot \left(\sqrt{l} \cdot 2^{-n}\right)^2 + \zeta_p \cdot 2^{-2n} \tag{D.13}$$

$$\geq \max_{x \in \mathcal{B}_{\sqrt{l}\cdot 2^{-n}}(x^*)} p(x) + \zeta_p \cdot 2^{-2n} \tag{D.14}$$

$$\geq \max_{x \in \mathbf{x}} p(x) + \zeta_p \cdot 2^{-2n} \tag{D.15}$$

$$\geq \max B(\mathbf{x}) \tag{D.16}$$

where ζ_p is the constant in Theorem 32 corresponding to polynomial p; (D.11) follows from Theorem 32; (D.12) follows from (D.9); (D.13) follows from (D.10) and the strict monotonicity of quadratic functions restricted to non-negative numbers; (D.14) follows from (D.9); (D.15) follows from Remark 36; (D.16) follows from Corollary 33. According to Theorem 40, such a subbox $\mathbf{y}$ cannot be in $\mathcal{L}$. In other words (by taking the contrapositive), the distance between x^* and any subbox contained in $\mathcal{B}_R(x^*)$ cannot exceed R', meaning $\min_{y \in \mathbf{y}} \|y - x^*\| \leq R' \; \forall \, \mathbf{y} \in \mathcal{L}$ and $\mathbf{y} \subset \mathcal{B}_R(x^*)$. Using (D.10), we can choose a large enough n such that $R' < R$. Therefore, all subboxes contained in $\mathcal{B}_R(x^*)$ must also be contained in a hypercube centered at x^* with widths $2R' + 2 \cdot 2^{-n}$. Since the maximum width of all subboxes is 2^{-n} (Remark 36), the number of subboxes contained in $\mathcal{B}_R(x^*)$ is bounded from above by

$$\left(\frac{2R' + 2 \cdot 2^{-n}}{2^{-n}}\right)^l = \left(4\sqrt{\frac{(\sigma_{\max} l + 2\zeta_p)}{\sigma_{\min}}} + 2\right)^l =: C_3. \tag{D.17}$$

Since there are m global minimizers, the total number of subboxes remaining in $\mathcal{L}$ is bounded from above by $m \cdot C_3$, which is a constant. □

D.3 Proof of Theorem 45 (Constrained Rate of Convergence)

Proof. Let $n \geq 1$ be the current iteration number. To prove Algorithm 1 solves (P) in n^{th} iteration, we must show that there exists a subbox $\mathbf{x} \in \mathcal{L}$ that meets the stopping criterion in Definition 35. That is, $\mathbf{x}$ should satisfy

(a) $\mathbf{x} \in \mathcal{L}$,

(b) $|\mathbf{x}| \leq \delta$,

(c) $\max B_{gi}(\mathbf{x}) \leq 0$ for all $i = 1, \cdots, \alpha$,

(d) $-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}}$ for all $j = 1, \cdots, \beta$, and

(e) $\max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \epsilon$ for all $\mathbf{y} \in \mathcal{L}$.

We define such a subbox $\mathbf{x}$ as a function of n using the implicit function theorem, then construct C_7, C_8, and C_9 using these criteria.

Let $\hat{\alpha}$ ($\hat{\alpha} \leq \alpha$) be the number of active inequality constraints at x^*. Without loss of generality, suppose $g_1, \cdots, g_{\hat{\alpha}}$ are active. Define

$$A := \begin{bmatrix} \nabla g_1(x^*) & \cdots & \nabla g_{\hat{\alpha}}(x^*) & \nabla h_1(x^*) & \cdots & \nabla h_\beta(x^*) \end{bmatrix}^\top \tag{D.18}$$

It follows from LICQ that A is full rank. Let Z be a matrix whose columns are a basis for the null space of A; that is,

$$Z \in \mathbb{R}^{l \times (l - \hat{\alpha} - \beta)}, \quad Z \text{ is full rank}, \quad AZ = 0^{(\hat{\alpha}+\beta) \times (l-\hat{\alpha}-\beta)}. \tag{D.19}$$

For convenience, define

$$\tilde{A} := \begin{bmatrix} A \\ Z^\top \end{bmatrix} \tag{D.20}$$

where $\tilde{A}$ is a square matrix with full rank. Define a parameterized system of equations $F : \mathbb{R}^l \times \mathbb{R} \to \mathbb{R}^l$ by

$$F(a,t) = \begin{bmatrix} g_1(a) + (\zeta_{g1} + L_{g1}\sqrt{l}) \cdot t \\ \vdots \\ g_{\hat{\alpha}}(a) + (\zeta_{g\hat{\alpha}} + L_{g\hat{\alpha}}\sqrt{l}) \cdot t \\ h_1(a) \\ \vdots \\ h_\beta(a) \\ Z^\top x \end{bmatrix} = 0^{l \times 1} \tag{D.21}$$

where ζ_{gi} is the constant in Theorem 32 corresponding to polynomial g_i, and L_{gi} is the Lipschitz constant of g_i over $\mathbf{u}$. Notice in particular that $F(x^*, 0) = 0$. At $a = x^*$, $t = 0$, the Jacobian of F with respect to a is

$$\nabla_a F(x^*, 0) = \tilde{A} \tag{D.22}$$

which is nonsingular by construction of Z. According to the implicit function theorem, the system (D.21) has a unique solution x for all values of t sufficiently small; that is, there exists a number $R_1 > 0$ and a function $\xi : [-R_1, R_1] \to \mathbb{R}^l$, such that

$$\xi(0) = x^* \tag{D.23}$$

$$F(\xi(t), t) = 0 \quad \forall\, t \in [-R_1, R_1] \tag{D.24}$$

In particular, such R_1 can be obtained [CHP03] using only the bounds of the polynomials g_i and h_j for all $i = 1, \cdots \hat{\alpha}$, $j = 1, \cdots, \beta$. We now let $t := 2^{-n}$. Since $|t|$ can be arbitrarily small as n is increased, satisfying $|t| \leq R$ requires

$$n \geq -\log_2 R_1. \tag{D.25}$$

In the remainder of this proof we define $x := \xi(2^{-n})$; that is,

$$F(x, 2^{-n}) = 0, \tag{D.26}$$

and denote the subbox that contains x as $\mathbf{x}$. We next show in Steps 1–5 that such $\mathbf{x}$ satisfies Conditions (b)–(e) for $n \geq N$. The case of $\mathbf{x} \notin \mathcal{L}$ is discussed in Step 6.

Step 1 (Condition (b)). Notice from Remark 36 that

$$|\mathbf{x}| = 2^{-n} \leq 2^{-N} \leq \delta \tag{D.27}$$

therefore Condition (b) is satisfied.

Step 2 (Condition (c)). Notice that

$$\max B_{gi}(\mathbf{x}) \leq \max_{x' \in \mathbf{x}} g_i(x') + \zeta_{gi} \cdot 2^{-2n} \tag{D.28}$$

$$\leq g_i(x) + \max_{x' \in \mathbf{x}} g_i(x') - g_i(x) + \zeta_{gi} \cdot 2^{-2n} \tag{D.29}$$

$$\leq -\zeta_{gi} \cdot 2^{-n} - L_{gi}\sqrt{l} \cdot 2^{-n} + L_{gi} \max_{x' \in \mathbf{x}} \|x' - x\| + \tag{D.30}$$

$$+ \zeta_{gi} \cdot 2^{-2n}$$

$$< 0 \tag{D.31}$$

for all $i = 1, \cdots, \hat{\alpha}$, where (D.28) follows from Corollary 33; (D.29) follows by adding and subtracting $g_i(x)$; (D.30) follows from (D.26) and the definition of L_{gi}; (D.31) holds because $2^{-2n} < 2^{-n}$. Therefore Condition (c) is satisfied.

Step 3 (Condition (d)). Let L_{hj} be a Lipschitz constant of h_j over $\mathbf{u}$. To ensure $\mathbf{x}$ satisfies the

equality constraint tolerance ϵ_{eq}, we require the following inequality to hold:

$$n \geq \max_{j=1,\cdots,\beta} \left\{ -\log_2 \left(\frac{\epsilon_{\text{eq}}}{\zeta_{hj} + L_{hj}\sqrt{l}} \right) \right\}. \tag{D.32}$$

Notice

$$\begin{aligned} \max B_{hj}(\mathbf{x}) &\leq \max_{x' \in \mathbf{x}} h_j(x) + \zeta_{hj} \cdot 2^{-2n} && \text{(D.33)} \\ &\leq h_j(x) + L_{hj} \cdot \max_{x' \in \mathbf{x}} \|x' - x\| + \zeta_{hj} \cdot 2^{-2n} && \text{(D.34)} \\ &\leq 0 + (L_{hj}\sqrt{l} + \zeta_{hj}) \cdot 2^{-n} && \text{(D.35)} \\ &\leq \epsilon_{\text{eq}} && \text{(D.36)} \end{aligned}$$

for all $j = 1, \cdots, \beta$, where (D.33) follows from Corollary 33; (D.34) follows from the definition of L_{hj}; (D.35) holds because $2^{-2n} \leq 2^{-n}$; and (D.36) follows from (D.32). Similarly, we may prove

$$\min B_{hj}(\mathbf{x}) \geq -\epsilon_{\text{eq}} \tag{D.37}$$

for all $j = 1, \cdots, \beta$. It then follows from (D.36), (D.37), Theorem 32, and Corollary 33 that

$$-\epsilon_{\text{eq}} \leq \min B_{hj}(\mathbf{x}) \leq 0 \leq \max B_{hj}(\mathbf{x}) \leq \epsilon_{\text{eq}} \tag{D.38}$$

for all $j = 1, \cdots, \beta$. Therefore Condition (d) is satisfied.

Step 4 (First part of Condition (e)). It is difficult to directly talk about the relationship between $B_p(\mathbf{x})$ and $B_p(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{L}$. Instead, in Step 4 we bound the difference between $\max B_p(\mathbf{x})$ and $p(x^*)$ by $\epsilon/2$; then, in Step 5, we bound the gap between $p(x^*)$ and $\min B_p(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{L}$ also by $\epsilon/2$.

Define

$$b := \begin{bmatrix} \zeta_{g1} + L_{g1}\sqrt{l} \\ \vdots \\ \zeta_{g\hat{\alpha}} + L_{g\hat{\alpha}}\sqrt{l} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \Bigg\} l - \hat{\alpha} \\ \\ \\ \end{matrix}. \tag{D.39}$$

By taking the total derivative of (D.26) with respect to t at $t = 0$ we obtain

$$0 = \nabla_a F(\xi(0), 0) \cdot \nabla_t \xi(0) + \nabla_t F(\xi(0), 0) \tag{D.40}$$

$$= \tilde{A} \cdot \nabla_t \xi(0) + b \tag{D.41}$$

Since $\tilde{A}$ has full rank, $\nabla_t \xi(0) = -\tilde{A}^{-1} b$. Using Taylor's theorem, $x := \xi(2^{-n})$ may be written as

$$x = x^* + \nabla_t \xi(0) \cdot 2^{-n} + o(2^{-n}) \tag{D.42}$$

where the first term follows from $\xi(0) = x^*$. For sufficiently small 2^{-n}, the third term in (D.42) is dominated by 2^{-n}; that is, there exists a number $R_2 > 0$, such that

$$\|x - x^*\| \leq \left\| 1^{l \times 1} - \tilde{A}^{-1} b \right\| \cdot 2^{-n} =: C_4 \cdot 2^{-n} \tag{D.43}$$

Let L_p be the Lipschitz constant of p in $\mathcal{B}_{R_2}(x^*)$. Suppose

$$n \geq -\log_2 \left(\frac{\epsilon}{2 \left(L_p \sqrt{l} + L_p C_4 + \zeta_p \right)} \right), \tag{D.44}$$

where ζ_p is the constant in Theorem 32 corresponding to polynomial p. We then claim $\max B_p(\mathbf{x}) - p(x^*)$ is bounded by $\epsilon/2$. To see this, notice

$$\max B_p(\mathbf{x}) - p(x^*) \leq \max_{x' \in \mathbf{x}} p(x') - p(x^*) + \zeta_p \cdot 2^{-2n} \tag{D.45}$$

$$\leq \max_{x' \in \mathbf{x}} |p(x') - p(x)| + |p(x) - p(x^*)| + \zeta_p \cdot 2^{-n} \tag{D.46}$$

$$\leq \left(L_p \sqrt{l} + L_p C_4 + \zeta_p \right) \cdot 2^{-n} \tag{D.47}$$

$$\leq \frac{\epsilon}{2} \tag{D.48}$$

where (D.45) follows from Corollary 33; (D.46) follows from triangle inequality and $2^{-2n} \leq 2^{-n}$; (D.47) follows from (D.43) and definition of L_p; (D.48) follows from (D.32).

For Condition (e) to be satisfied, we need to show that minimum value of every other Bernstein patch $\mathcal{L}$ is within $\epsilon/2$ of $p(x^*)$; we do this next.

Step 5 (Second part of Condition (e)). In this step, we bound the difference between $p(x^*)$ and $\min B_p(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{L}$. To do this, we first show that any $\mathbf{y} \in \mathcal{L}$ solves a relaxed POP related to the original POP and parameterized by the current iteration n. Then, we bound the difference between the solutions of the relaxed and original POPs.

Let $\mathbf{y} \in \mathcal{L}$ be any subbox with maximum width 2^{-n}. Then,

(f) $\min B_{gi}(\mathbf{y}) \leq 0$ for all $i = 1, \cdots, \alpha$;

(g) $\min B_{hj}(\mathbf{y}) \leq 0$ for all $j = 1, \cdots, \beta$;

(h) $\max B_{hj}(\mathbf{y}) \geq 0$ for all $j = 1, \cdots, \beta$,

from Theorem 40. Notice (f) implies

$$0 \geq \min_{y \in \mathbf{y}} g_i(y) - \zeta_{gi} \cdot 2^{-2n} \tag{D.49}$$

$$\geq \max_{y \in \mathbf{y}} g_i(y) - L_{gi}\sqrt{l} \cdot 2^{-n} - \zeta_{gi} \cdot 2^{-2n} \tag{D.50}$$

where (D.49) follows from Theorem 32; (D.50) follows from definition of L_{gi}. For convenience, denote

$$\gamma_{gi}(n) := L_{gi}\sqrt{l} \cdot 2^{-n} + \zeta_{gi} \cdot 2^{-2n}. \tag{D.51}$$

Then, (D.50) can be written as

$$\max_{y \in \mathbf{y}} g_i(y) \leq \gamma_{gi}(n). \tag{D.52}$$

Similarly, if we define

$$\gamma_{hj}(n) := L_{hj}\sqrt{l} \cdot 2^{-n} + \zeta_{hj} \cdot 2^{-2n}, \tag{D.53}$$

then (g) and (h) can be written as

$$\max_{y \in \mathbf{y}} h_j(y) \leq \gamma_{hj}(n) \tag{D.54}$$

$$\min_{y \in \mathbf{y}} h_j(y) \geq -\gamma_{hj}(n). \tag{D.55}$$

Now we are ready to introduce the relaxed POP:

$$\begin{aligned} \min_{y \in \mathbf{u}} \quad & p(y) && && (\mathrm{P}_n) \\ \text{s.t} \quad & g_i(y) \leq \gamma_{gi}(n) && i = 1, \cdots, \alpha && (\text{D.56}) \\ & h_j(y) \leq \gamma_{hj}(n) && j = 1, \cdots, \beta && (\text{D.57}) \\ & h_j(y) \geq -\gamma_{hj}(n) && j = 1, \cdots, \beta && (\text{D.58}) \end{aligned}$$

Denote y_n^* as the optimal solution to (P_n). Notice that (D.56), (D.57), and (D.58) are relaxations of (D.52), (D.54), and (D.55) respectively. That is, let $\mathbf{y} \in \mathcal{L}$ and consider any $y \in \mathbf{y}$ that satisfies

(D.52), (D.54), and (D.55). It also necessarily satisfies (D.56), (D.57), and (D.58). Therefore,

$$p(y_n^*) \leq \min_{y \in \mathbf{y}} p(z) \tag{D.59}$$

It can be obtained from Theorem 32 that

$$\min_{\mathbf{y} \in \mathcal{L}} B_p(\mathbf{y}) \geq \min_{y \in \mathbf{y}} p(y) - \zeta_p \cdot 2^{-2n} \geq p(y_n^*) - \zeta_p \cdot 2^{-2n} \tag{D.60}$$

To estimate the value $p(y_n^*)$, notice from [Fia78, Corollary 4.1],

$$\frac{\partial p(y_n^*)}{\partial \gamma_{gi}(n)} = -\lambda_{gi}^* \tag{D.61}$$

$$\frac{\partial p(y_n^*)}{\partial \gamma_{hj}(n)} = -\lambda_{hj}^* \tag{D.62}$$

where λ_{gi}^* and λ_{hj}^* are Lagrange multipliers corresponding to the constraints g_i and h_j in (P_n), respectively, for all $i = 1, \cdots, \hat{\alpha}$ and $j = 1, \cdots, \beta$. Using Taylor's theorem, we obtain

$$\begin{aligned} p(y_n^*) \geq & p(x^*) - \sum_{i=1,\cdots,\alpha} \lambda_{gi}^* \gamma_{gi}(n) - \sum_{j=1,\cdots,\beta} \left|\lambda_{hj}^* \gamma_{hj}(n)\right| + \\ & + O\left(\sum_{i=1,\cdots,\alpha} \gamma_{gi}^2(n) + \sum_{j=1,\cdots,\beta} \gamma_{hj}^2(n)\right), \end{aligned} \tag{D.63}$$

for sufficiently small $\gamma_{gi}(n)$ and $\gamma_{hj}(n)$.

Since $\gamma_{gi}(n)$ and $\gamma_{hj}(n)$ can be made arbitrarily small by increasing n, there exists a constant C_5 such that

$$p(y_n^*) \geq p(x^*) - \frac{1}{2}\left(\sum_{i=1,\cdots,\alpha} \lambda_{gi}^* \gamma_{gi}(n) + \sum_{j=1,\cdots,\beta} \left|\lambda_{hj}^* \gamma_{hj}(n)\right|\right) \tag{D.64}$$

for all $n \geq C_5$. It then follows from (D.60) and (D.64) that

$$p(x^*) - \min_{\mathbf{y} \in \mathcal{L}} B_p(\mathbf{y}) \leq \frac{1}{2}\left(\sum_{i=1,\cdots,\alpha} \lambda_{gi}^* \gamma_{gi}(n) + \sum_{j=1,\cdots,\beta} \left|\lambda_{hj}^* \gamma_{hj}(n)\right|\right) + \zeta_p \cdot 2^{-2n} \tag{D.65}$$

Suppose n satisfies

$$n \geq -\log_2 \epsilon + \log_2(C_6 + 2\zeta_p) \tag{D.66}$$

where

$$C_6 := \sum_{i=1,\cdots,\alpha} \lambda^*_{gi}(L_{gi}\sqrt{l} + \zeta_{gi}) + \sum_{j=1,\cdots,\beta} \left|\lambda^*_{hj}(L_{hj}\sqrt{l} + \zeta_{hj})\right| \tag{D.67}$$

Then it follows from (D.65) that

$$p(x^*) - \min_{\mathbf{y}\in\mathcal{L}} B_p(\mathbf{y}) \leq \frac{\epsilon}{2}. \tag{D.68}$$

By applying the triangle inequality to (D.48) and (D.68), we fulfill Condition (e).

Step 6 (Condition (a)). If $\mathbf{x} \in \mathcal{L}$ in the above discussion, then such $\mathbf{x}$ satisfies Conditions (a)–(e) and therefore the proof is finished. When $\mathbf{x} \notin \mathcal{L}$, however, we need to show there exists another subbox $\mathbf{z} \subset \mathbf{u}$ that satisfies Conditions (a)–(e). This can be done using the Cut-Off Test in Theorem 40.

Suppose $\mathbf{x} \notin \mathcal{L}$. Notice such $\mathbf{x}$ is *not infeasible* per Definition 37. According to Theorem 40, such $\mathbf{x}$ must be *suboptimal* so that it is removed from the list $\mathcal{L}$; that is, there exists a feasible $\mathbf{z} \in \mathcal{L}$ such that

$$\max B_p(\mathbf{z}) < \min B_p(\mathbf{x}) \tag{D.69}$$

Per Definition 37, such $\mathbf{z}$ satisfies Conditions (a)–(d). To show $\mathbf{z}$ satisfies Condition (e), notice from (D.48), (D.68), and (D.69) that

$$\max B_p(\mathbf{z}) - \min B_p(\mathbf{y}) < \max B_p(\mathbf{x}) - \min B_p(\mathbf{y}) \leq \epsilon \tag{D.70}$$

for all $\mathbf{y} \in \mathcal{L}$.

To conclude the proof, define constants

$$C_7 := \max\{-\log_2 R_1, C_5\} \tag{D.71}$$

$$C_8 := \max_{j=1,\cdots,\beta} \log_2\left(\zeta_{hj} + L_{hj}\sqrt{l}\right) \tag{D.72}$$

$$C_9 := \max\left\{\log_2\left(2\left(L_p\sqrt{l} + L_pC_4 + \zeta_p\right)\right), \log_2(C_6 + 2\zeta_p)\right\} \tag{D.73}$$

The result then follows. □

D.4 Proof of Theorem 46 (Constrained Memory Usage)

Proof. Consider an arbitrary global minimizer $x^* \in \mathbf{u}$. It follows from (D.47) (in the proof of Theorem 45) that for sufficiently large n there exists a subbox $\mathbf{x} \in \mathcal{L}$ such that

$$\max B_p(\mathbf{x}) - p(x^*) \leq \left(L_p\sqrt{l} + L_p C_4 + \zeta_p\right) \cdot 2^{-n} \tag{D.74}$$

where C_4 is defined as in (D.43). Consider another feasible point $z \in \mathbf{u}$ of (P), and let $\mathbf{z} \subset \mathbf{u}$ be the subbox that contains z. Such z is necessarily feasible to the relaxed problem (P_n). According to [NW06, (12.71)], we know

$$p(z) \geq p(y_n^*) + \|z - y_n^*\| \sum_{i=1,\cdots,\hat{\alpha}} \lambda_{gi}^* \nabla g_i(y_n^*)^\top d + o(\|z - y_n^*\|) \tag{D.75}$$

where y_n^* is the optimal solution to (P_n); λ_{gi}^* are the Lagrange multipliers corresponding to g_i in (P_n) for all $i = 1, \cdots, \hat{\alpha}$; and d is any unit vector such that

$$\nabla g_i(y_n^*)^\top d \geq 0 \quad \forall\, i = 1, \cdots, \hat{\alpha}, \tag{D.76}$$

$$\nabla h_j(y_n^*)^\top d = 0 \quad \forall\, j = 1, \cdots, \beta. \tag{D.77}$$

Since the critical cone of (P_n) at y_n^* is nonempty, for any d there exists $i \in \{1, \cdots, \hat{\alpha}\}$ such that

$$\lambda_{gi}^* \nabla g_i(y_n^*)^\top d > 0 \tag{D.78}$$

For notational convenience, define

$$J(d) = \sum_{i=1,\cdots,\hat{\alpha}} \lambda_{gi}^* \nabla g_i(y_n^*)^\top d \tag{D.79}$$

with minimum $q := \min_d J(d)$. In particular, it follows from the extreme value theorem that such minimum exists, since $J(d)$ is continuous with respect to d, and the set of unit vectors defined by (D.76) and (D.77) is closed and bounded (therefore compact according to Heine-Borel theorem). It also follows from (D.78) that $q > 0$. For any z that is sufficiently close to y_n^*, we may then rewrite (D.75) as

$$p(z) \geq p(y_n^*) + \frac{q}{2}\|z - y_n^*\| \tag{D.80}$$

We claim for sufficiently large n and all $z \in \mathbf{u}$ such that

$$\|z - y_n^*\| \geq \frac{2\left(2L_p\sqrt{l} + L_pC_4 + C_6/2 + 2\zeta_p\right)}{q} \cdot 2^{-n} =: C_{10} \cdot 2^{-n}, \tag{D.81}$$

the subbox $\mathbf{z}$ cannot be in the list $\mathcal{L}$. To prove this, notice

$$\begin{aligned}
\min B(\mathbf{z}) \geq & \min_{z' \in \mathbf{z}} p(z') - \zeta_p \cdot 2^{-2n} && \text{(D.82)}\\
\geq & p(z) - L_p\sqrt{l} \cdot 2^{-n} - \zeta_p \cdot 2^{-2n} && \text{(D.83)}\\
\geq & p(y_n^*) + \left(\frac{qC_{10}}{2} - L_p\sqrt{l} - \zeta_p\right) \cdot 2^{-n} && \text{(D.84)}\\
\geq & p(x^*) + \left(\frac{qC_{10}}{2} - L_p\sqrt{l} - \frac{C_6}{2} - \zeta_p\right) \cdot 2^{-n} && \text{(D.85)}\\
\geq & \max B_p(\mathbf{x}) + \left(\frac{qC_{10}}{2} - 2L_p\sqrt{l} - L_pC_4 - \frac{C_6}{2} - 2\zeta_p\right) \cdot 2^{-n} && \text{(D.86)}\\
\geq & \max B_p(\mathbf{x}) && \text{(D.87)}
\end{aligned}$$

where L_p is the Lipchitz constant of p over $\mathbf{u}$, and C_6 is defined as in (D.67). In particular, (D.82) follows from Theorem 32; (D.83) follows from definition of L_p; (D.84) follows from (D.80); (D.85) follows from (D.64); (D.86) follows from (D.74); (D.87) follows from (D.81). According to Theorem 40, $\mathbf{z}$ cannot be in the list $\mathcal{L}$.

To conclude this proof, notice the above claim implies that for sufficiently large n, all subboxes in a neighborhood of x^* must also be contained in a hypercube centered at y_n^* with widths $(2C_{10} + 2) \cdot 2^{-n}$. Since the maximum width of all subboxes is 2^{-n} (Remark 36), the number of subboxes contained in a neighborhood of x^* is bounded from above by

$$\left(\frac{(2C_{10} + 2) \cdot 2^{-n}}{2^{-n}}\right)^l = (2C_{10} + 2)^l. \tag{D.88}$$

Since there are m global minimizers, the number of subboxes remaining in $\mathcal{L}$ is bounded above by $m \cdot (2C_{10} + 2)^l$, which is a constant □

APPENDIX E

A List of Polynomial Optimization Problems

E.1 Benchmark Problems

The following problems are all from [NA11], with minor changes. To report the global optima values and locations, we solved each problem 1000 times using MATLAB's `fmincon`, with optimality and constraint tolerances set to 10^{-10}, and random initial guesses.

P1:

$$\begin{aligned} \min \ & -x_1 - x_2 \\ \text{s.t.} \ & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ & -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \end{aligned}$$

where $x_1 \in [0, 3]$ and $x_2 \in [0, 4]$. The best optimal value we found is -5.5080132636 at the location $x = (2.3295201981, 3.1784930655)$.

P2:

$$\begin{aligned} \min \ & 658500x_1^3 + 68121x_1^2 + 2349x_1 + \\ & + 1000000x_2^3 - 600000x_2^2 + 120000x_2 - 7973 \\ \text{s.t.} \ & -7569x_1^2 - 1392x_1 - 10000x_2^2 + 1000x_2 + 11 \leq 0 \\ & 7569x_1^2 + 1218x_1 + 10000x_2^2 - 1000x_2 - 8.81 \leq 0 \end{aligned}$$

where $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$. The best optimal value we found is -6961.8138816446 at $x = (0.0125862069, 0.0084296079)$.

P3:

$$\begin{aligned}
&\min\ x_1 \\
&\text{s.t. } x_1^2 - x_2 \leq 0 \\
&\qquad x_2 - x_1^2(x_1 - 2) + 10^{-5} \leq 0
\end{aligned}$$

where $x_i \in [-10, 10]$ for $i = 1, 2$. The best optimal value we found is 3.0000011115 at $x = (3.0000011115, 9.0000066709)$.

P4:

$$\begin{aligned}
&\min\ -2x_1 + x_2 - x_3 \\
&\text{s.t. } x_1 + x_2 + x_3 - 4 \leq 0 \\
&\qquad 3x_2 + x_3 - 6 \leq 0 \\
&\qquad -x^T A^T A x + 2y^T A x - \|y\|^2 + 0.25\,\|b - z\|^2 \leq 0 \\
&\qquad A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ -2 & 1 & -1 \end{pmatrix} \\
&\qquad b = (3, 0, -4)^T \\
&\qquad y = (1.5, -0.5, -5)^T \\
&\qquad z = (0, -1, -6)^T
\end{aligned}$$

where $x_1 \in [0, 2]$, $x_2 \in [0, 10]$, and $x_3 \in [0, 3]$. The global optimum is -4 at $x = (0.5, 0, 3)$.

P5:

$$\begin{aligned}
&\min\ x_3 \\
&\text{s.t. } 2x_1^2 + 4x_1x_2 - 42x_1 + 4x_1^3 - x_3 - 14 \leq 0 \\
&\qquad -2x_1^2 - 4x_1x_2 + 42x_1 - 4x_1^3 - x_3 + 14 \leq 0 \\
&\qquad 2x_1^2 + 4x_1x_2 - 26x_1 + 4x_1^3 - x_3 - 22 \leq 0 \\
&\qquad -2x_1^2 - 4x_1x_2 + 26x_1 - 4x_1^3 - x_3 + 22 \leq 0
\end{aligned}$$

where $x_i \in [-5, 5]$ for $i = 1, 2, 3$. The best optimum found is 0 at $x = (-0.305, -0.913, 0)$.

P6:

$$\begin{aligned}
\min\ & 0.6224x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1x_3 \\
\text{s.t.}\ & -x_1 + 0.0193x_3 \leq 0 \\
& -x_2 + 0.00954x_3 \leq 0 \\
& -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 750.1728 \leq 0 \\
& -240 + x_4 \leq 0
\end{aligned}$$

where $x_1 \in [1, 1.375]$, $x_2 \in [0.625, 1]$, $x_3 \in [47.5, 52.5]$, and $x_4 \in [90, 112]$. The global optimum is 6395.5078 (to the same number of decimal places as used in the cost function) at the location $x = (1, 0.625, 47.5, 90.0)$;

P7:

$$\begin{aligned}
\min\ & x_4 \\
\text{s.t.}\ & x_1^4x_2^4 - x_1^4 - x_2^4x_3 = 0 \\
& 1.4 - 0.25x_4 - x_1 \leq 0 \\
& -1.4 - 0.25x_4 + x_1 \leq 0 \\
& 1.5 - 0.2x_4 - x_2 \leq 0 \\
& -1.5 - 0.2x_4 + x_2 \leq 0 \\
& 0.8 - 0.2x_4 - x_3 \leq 0 \\
& -0.8 - 0.2x_4 + x_3 \leq 0
\end{aligned}$$

where $x_i \in [0, 5]$ for $i = 1, 2, 3, 4$. The best optimal value we found is 1.0898639714 at the location $x = (1.1275340071, 1.2820272057, 1.0179727943, 1.0898639714)$.

P8:

$$
\begin{aligned}
\min\ & 54.528x_2x_4 + 27.624x_1x_3 - 54.528x_3x_4 \\
\text{s.t. } & 61.01627586 - I \leq 0 \\
& 8x_1 - I(x) \leq 0 \\
& x_1x_2x_4 - x_2x_4^2 + x_1^2x_3 + x_3x_4^2 - 2x_1x_3x_4 - 3.5x_3I(x) \leq 0 \\
& x_1 - 3x_2 \leq 0 \\
& 2x_2 - x_1 \leq 0 \\
& x_3 - 1.5x_4 \leq 0 \\
& 0.5x_4 - x_3 \leq 0 \\
& I(x) = 6x_1^2x_2x_3 - 12x_1x_2x_3^2 + 8x_2x_3^3 + x_1^3x_4 - 6x_1^2x_3x_4 + \\
& \qquad + 12x_1x_3^2x_4 - 8x_3^3x_4
\end{aligned}
$$

where $x_1 \in [3, 20]$, $x_2 \in [2, 15]$, $x_3 \in [0.125, 0.75]$, and $x_4 \in [0.25, 1.25]$. The best optimal value we found is 42.4440570797 at the location $x = (4.9542421008, 2, 0.125, 0.25)$.

E.2 Increasing Constraints Problems

ElAttar-Vidyasagar-Dutta (E-V-D) [Gav19]:

$$(x_1^2 + x_2 - 10)^2 + (x_1 + x_2^2 - 7)^2 + (x_1^2 + x_2^3 - 1)^2$$

where $x_i \in [-100, 100]$ for $i = 1, 2$. The global optimum is 1.712780354 at $x = (3.41, -2.17)$.

Powell [Gav19]:

$$(x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

where $x_i \in [-10, 10]$ for $i = 1, 2, 3, 4$. The global optimum is 0 at $x = 0$.

Wood [Gav19]:

$$
\begin{aligned}
& (100(x_2 - x_1^2))^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + \\
& \quad + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)
\end{aligned}
$$

where $x_i \in [-10, 10]$ for $i = 1, 2, 3, 4$. The global optimum is 0 at $x = (1, 1, 1, 1)$.

Dixon-Price (D-P) [Gav19]:

$$(x_1 - 1)^2 + \sum_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$$

where $x_i \in [-10, 10]$ for $i = 1, \ldots, d$. The global optimum is 0 at $x_i = 2^{-\frac{2^i - 2}{2^i}}$ for $i = 1, \ldots, d$.

Beale:

$$(x_1 x_2 - x_1 + 1.5)^2 + (x_1 x_2^2 - x_1 + 2.25)^2 + \\ + (x_1 x_2^3 - x_1 + 2.625)^2$$

where $x_i \in [-10, 10]$ for $i = 1, 2$. The global optimum is 0 at $x = (3, 0.5)$.

Bukin02 [Gav19]:

$$100(x_2^2 - 0.01x_1^2 + 1) + 0.01(x_1 + 10)^2$$

where $x_1 \in [-15, -5]$, $x_2 \in [-3, 3]$. The global optimum is -124.75 at $x = (-15, 0)$.

Deckkers-Aarts (D-A):

$$10^5 x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4$$

where $x_i \in [-20, 20]$ for $i = 1, 2$. The global optimum is -24777 at $x = (0, \pm 15)$.

APPENDIX F

Proof of Theorem 49

We first assume the joint angles always stay within certain ranges:

Assumption 80. *During the entire experiment, Cassie's joint angles satisfy*

$$q_{4L}, q_{4R} \in [-156^\circ, -42^\circ], \tag{F.1}$$

Notice such bounds are reasonable because (F.1) represents the physical joint limit of Cassie. We may now give a formal proof of Theorem 49.

Proof. (of Theorem 49)

Without loss of generality, we only consider the left limbs of Cassie, and the arguments for the right limbs should follow in a similar way. The rest of this proof can then be outlined as follows: first, we define 3D balls around each limb joint of Cassie (shown in Fig. F.1) and argue that Cassie does not collide with the ground if certain balls stay strictly above the ground; second, we use kinematics of Cassie to prove these balls stay above the ground if the hip height stays above 0.75 [m].

We now define balls around each joint, as shown in Fig. F.1. These balls are denoted as $\mathcal{B}_s, \mathcal{B}_3, \ldots, \mathcal{B}_7$ and centered at the joints $q_{sL}, q_3, \ldots, q_7$, respectively. Denote the radii of these balls as $R_s, R_3, \ldots, R_7$. By choosing proper radii we can make sure the body of Cassie (excluding the feet) is fully contained in the convex hulls of adjacent balls, as shown in Fig. F.1. As a result, if none of the balls $\mathcal{B}_s, \mathcal{B}_3, \ldots, \mathcal{B}_6$ touches the ground, the body of Cassie (excluding the feet) stays strictly above the ground and therefore no collision with the ground happens (Definition 48). In

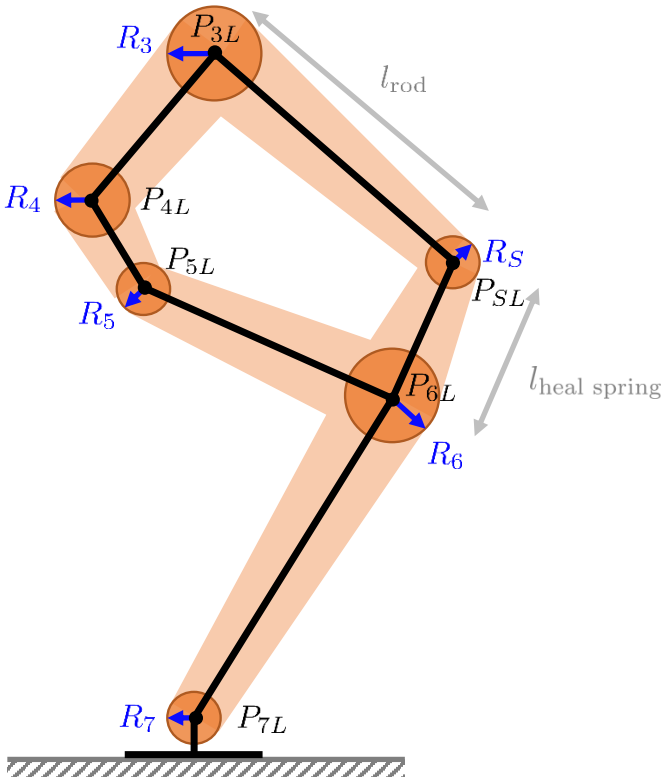


Figure F.1: Left limbs of Cassie are contained in convex hulls of adjacent balls.

practice, these radii are chosen to be

$$R_s = 0.1 \tag{F.2}$$
$$R_3 = 0.5 \tag{F.3}$$
$$R_4 = 0.3 \tag{F.4}$$
$$R_5 = 0.3 \tag{F.5}$$
$$R_6 = 0.5 \tag{F.6}$$
$$R_7 = 0.2 \tag{F.7}$$

We now prove each center of these balls, denoted as (x_i, y_i, z_i) for all $i \in \{s, 3, \ldots, 6\}$, stay above R_i. Using rigid body transformations provided in [Rob18], we may compute the 3D distances between joint q_3 and other joints as functions of the joint angles q_4 and q_5. It then follows from

elementary trigonometry that

$$\|z_3 - z_s\| \leq 0.546 \tag{F.8}$$
$$\|z_3 - z_4\| = 0.121 \tag{F.9}$$
$$\|z_3 - z_5\| \leq 0.197 \tag{F.10}$$
$$\|z_3 - z_6\| \leq 0.615 \tag{F.11}$$

Let $z_3 \geq 0.75$, it follows from triangle inequality that

$$z_s \geq 0.75 - 0.546 > R_3 \tag{F.12}$$
$$z_4 \geq 0.75 - 0.121 > R_4 \tag{F.13}$$
$$z_5 \geq 0.75 - 0.197 > R_5 \tag{F.14}$$
$$z_6 \geq 0.75 - 0.615 > R_6 \tag{F.15}$$

This concludes the proof. □

APPENDIX G

Derivation of a Balancing Controller u_0 for Cassie

Consider dynamics of Cassie

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + \tau_c \tag{G.1}$$

with configuration variables $q \in \mathcal{Q} \subset \mathbb{R}^{20}$. The system is subject to holonomic constraint $h : \mathcal{Q} \to \mathbb{R}^{12}$ such that

$$\tilde{\mathcal{Q}} := \{q \in \mathcal{Q} \mid h(q) = 0\} \tag{G.2}$$

is non-empty. τ_c are the external forces corresponding to the constraints. We assume for simplicity that the holonomic constraint can be expressed as

$$q_c = h_d(q_f) \tag{G.3}$$

for a choice of configuration variables (q_c, q_f), with the constrained coordinates $q_c \in \mathcal{Q}_c \subset \mathbb{R}^{12}$, the free coordinates $q_f \in \mathcal{Q}_f \subset \mathbb{R}^8$, and such that a diffeomorphism $F : \mathcal{Q}_c \times \mathcal{Q}_f \to \mathcal{Q}$ exists. In this case, the mapping $F_c : \mathcal{Q}_f \to \mathcal{Q}$ given by

$$F_c : q_f \mapsto F(h_d(q_f), q_f) \tag{G.4}$$

is am embedding. Moreover, its image defines the constraint manifold in the configuration space, namely

$$\tilde{\mathcal{Q}} := \{q \in \mathcal{Q} \mid q = F_c(q_f), q_f \in \mathcal{Q}_f\}, \tag{G.5}$$

which is diffeomorphic to Q_f under F_c.

Using the chain rule, we have

$$\dot{q} = \frac{\partial F_c(q_f)}{\partial q_f}\dot{q}_f =: J_c(q_f)\dot{q}_f \tag{G.6}$$

where $J_c(q_f)$ is the Jacobian of F_c at $q_f \in \mathcal{Q}_f$. Taking the time derivative of (G.6) yields

$$\ddot{q} = J_c(q_f)\ddot{q}_f + \dot{J}_c(q_f, \dot{q}_f)\dot{q}_f \tag{G.7}$$

where $\dot{J}_c(q_f, \dot{q}_f)$ is the time derivative of $J_c(q_f)$.

To eliminate the constraint force τ_c in the equation of motion, it suffices to Substituting (G.6), (G.7) into (G.1) and then left-multiplying $J_c(q_f)^\top$ on both sides. Since $J_c(q_f)^\top \tau_c = 0$ due to the principle of virtual work, the constrained dynamics can be written as

$$\begin{aligned} &J_c(q_f)^\top M(q) J_c(q_f)\ddot{q}_f + J_c(q_f)^\top M(q)\dot{J}_c(q_f, \dot{q}_f)\dot{q}_f + \\ &J_c(q_f)^\top (C(q, \dot{q})\dot{q} + G(q)) = J_c(q_f)^\top B u \Big|_{\substack{q = F_c(q_f) \\ \dot{q} = J_c(q_f)\dot{q}_f}} \end{aligned} \tag{G.8}$$

Consider a tracking controller of the form

$$u_0(q_f, \dot{q}_f) := \left(J_c(q_f)^\top B\right)^\dagger J_c(q_f)^\top \left(G - K_p\big(F_c(q_f) - q_{\text{des}}\big) - K_d J_c(q_f)\dot{q}_f\right) \tag{G.9}$$

where $K_p \in \mathbb{R}^{20\times 20}$ is symmetric and positive definite; $K_d \in \mathbb{R}^{20\times 20}$ is symmetric and positive semidefinite; $\dagger$ is the pseudo-inverse of matrix; q_{des} is a *constant* desired configuration (that is, $\dot{q}_{\text{des}} = 0$). For notational convenience, let the tracking error be $e := F_c(q_f) - q_{\text{des}}$.

Choose a Lyapunov function candidate $V : \mathbb{R}^{20} \times \mathbb{R}^{20} \to \mathbb{R}$ defined as

$$V(q_f, \dot{q}_f) := \frac{1}{2}\dot{q}_f^\top J_c(q_f)^\top M(F_c(q_f)) J_c(q_f)\dot{q}_f + \frac{1}{2}e^\top K_p e \tag{G.10}$$

Such V is positive definite for ϵ sufficiently small since K_p is positive definite. Based on (G.4) and (G.6), evaluating $\dot{V}$ along constrained trajectories of (G.1) under the feedback control law (G.9)

gives

$$\dot{V} = \dot{q}_f^\top J_c(q_f)^\top M(q) \left(J_c(q_f)\ddot{q}_f + \dot{J}_c(q_f, \dot{q}_f)\dot{q}_f \right) + \frac{1}{2}\dot{q}_f^\top J_c(q_f)^\top \dot{M}(q, \dot{q}) J_c(q_f)\dot{q}_f + \dot{q}^\top K_p e \tag{G.11}$$

$$= \dot{q}_f^\top \left(-J_c(q_f)^\top M(q)\dot{J}_c(q_f, \dot{q}_f)\dot{q}_f - J_c(q_f)^\top (C(q, \dot{q})J_c(q_f)\dot{q}_f + G(q)) + J_c(q)_f^\top Bu \right) + \\ + \dot{q}_f^\top J_c(q_f)^\top M(q)\dot{J}_c(q_f, \dot{q}_f)\dot{q}_f + \frac{1}{2}\dot{q}_f^\top J_c(q_f)^\top \dot{M}(q, \dot{q}) J_c(q_f)\dot{q}_f + \dot{q}_f^\top J_c(q_f)^\top K_p e \tag{G.12}$$

$$= \frac{1}{2}\dot{q}_f^\top J_c(q_f)^\top \left(\dot{M}(q, \dot{q}) - 2C(q, \dot{q}) \right) J_c(q_f)\dot{q}_f + \dot{q}_f^\top J_c(q_f)^\top (Bu - G(q) + K_p e) \tag{G.13}$$

$$= -\dot{q}_f^\top J_c(q_f)^\top K_d J_c(q_f)\dot{q}_f \tag{G.14}$$

$$\preceq 0 \tag{G.15}$$

where (G.11) follows from product rule and chain rule; (G.12) follows from (G.8); (G.13) is a regrouping of terms; (G.14) follows from (G.9) and the fact that $\dot{M}(q, \dot{q}) - 2C(q, \dot{q})$ is skew-symmetric for all $q \in \mathcal{Q}$; (G.15) is true because K_d is positive semidefinite. According to Lyapunov stability theorem, the constrained system is stable under the feedback controller (G.9) if (G.3) is satisfied and the input does not exceed torque limits.

BIBLIOGRAPHY

[ACG+18] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan W Hurst. Fast online trajectory optimization for the bipedal robot cassie. In *Robotics: Science and Systems*, 2018.

[AGS08] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[AGSG14] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014.

[AM14] Amir Ali Ahmadi and Anirudha Majumdar. Dsos and sdsos optimization: Lp and socp-based alternatives to sum of squares optimization. In *2014 48th annual conference on information sciences and systems (CISS)*, pages 1–5. IEEE, 2014.

[AN87] Edward J Anderson and Peter Nash. *Linear programming in infinite-dimensional spaces: theory and applications*. John Wiley & Sons, 1987.

[ApS17] MOSEK ApS. *MOSEK MATLAB Toolbox. Release 8.0.0.53.*, 2017.

[ATJ+17] Aaron D Ames, Paulo Tabuada, Austin Jones, Wen-Loong Ma, Matthias Rungger, Bastian Schürmann, Shishir Kolathaya, and Jessy W Grizzle. First steps toward formal controller synthesis for bipedal robots with experimental implementation. *Nonlinear Analysis: Hybrid Systems*, 25:155–173, 2017.

[Atk08] Kendall E Atkinson. *An introduction to numerical analysis*. John wiley & sons, 2008.

[AZGS06] Aaron D Ames, Haiyang Zheng, Robert D Gregg, and Shankar Sastry. Is there life after zeno? taking executions past the breaking (zeno) point. In *American Control Conference, 2006*, pages 6–pp. IEEE, 2006.

[BBM98] Michael S Branicky, Vivek S Borkar, and Sanjoy K Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE transactions on automatic control*, 43(1):31–45, 1998.

[BCC93] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1-3):295–324, 1993.

[BEGFB94] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[Ber95] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[BGV+15] Samuel A Burden, Humberto Gonzalez, Ramanarayan Vasudevan, Ruzena Bajcsy, and S Shankar Sastry. Metrization and simulation of controlled hybrid systems. *IEEE Transactions on Automatic Control*, 60(9):2307–2320, 2015.

[Bog07] Vladimir I Bogachev. *Measure theory*, volume 1, 2. Springer Science & Business Media, 2007.

[BSS15] Mohamed Amin Ben Sassi and Sriram Sankaranarayanan. Stability and stabilization of polynomial dynamical systems using Bernstein polynomials. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 291–292, 2015.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CAA+03] Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[CHP03] HC Chang, W He, and N Prabhu. The analytic domain in the implicit function theorem. *JIPAM. J. Inequal. Pure Appl. Math*, 4:1–5, 2003.

[CS14] Mathieu Claeys and Rodolphe Sepulchre. Reconstructing trajectories from the moments of occupation measures. In *53rd IEEE Conference on Decision and Control*, pages 6677–6682. IEEE, 2014.

[DKLP06] Etienne De Klerk, Monique Laurent, and Pablo A Parrilo. A ptas for the minimization of polynomials of fixed degree over the simplex. *Theoretical Computer Science*, 361(2-3):210–225, 2006.

[DLMO07] Geir Dahl, Jon Magne Leinaas, Jan Myrheim, and Eirik Ovrum. A tensor product matrix approximation problem in quantum physics. *Linear algebra and its applications*, 420(2-3):711–725, 2007.

[DN17] PS Dhabe and PSV Nataraj. A parallel bernstein algorithm for global optimization based on the implicit bernstein form. *International Journal of System Assurance Engineering and Management*, 8(2):1654–1671, 2017.

[DR05] Sheetal Dharmatti and Mythily Ramaswamy. Hybrid control systems and viscosity solutions. *SIAM Journal on Control and Optimization*, 44(4):1259–1288, 2005.

[Dyn18] Boston Dynamics. Parkour atlas. https://www.youtube.com/watch?v=_sBBaNYex3E, 2018.

[EL00] Michael B Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[FFT19] D. Fridovich-Keil, J. F. Fisac, and C. J. Tomlin. Safely probabilistically complete real-time planning and exploration in unknown environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7470–7476, May 2019. View online.

[Fia78] Anthony V Fiacco. Nonlinear programming sensitivity analysis results using strong second order assumptions. Technical report, George Washington University, 1978. View online.

[FK99] Robert J Full and Daniel E Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of experimental biology*, 202(23):3325–3332, 1999.

[Fol13] Gerald B Folland. *Real analysis: modern techniques and their applications*. John Wiley & Sons, 2013.

[Gar85] Jürgen Garloff. Convergent bounds for the range of multivariate polynomials. In *International Symposium on Interval Mathematics*, pages 37–56. Springer, 1985.

[Gar93] Jürgen Garloff. The bernstein algorithm. *Interval computations*, 2(6):154–168, 1993.

[Gav19] Andrea Gavana. Global optimization benchmarks and ampgo, 2019. View online.

[GG15] Brent Griffin and Jessy Grizzle. Walking gait optimization for accommodation of unknown terrain height variations. In *American Control Conference (ACC), 2015*, pages 4810–4817. IEEE, 2015.

[GH12] Jesse A Grimes and Jonathan W Hurst. The design of atrias 1.0 a unique monopod, hopping robot. In *Adaptive Mobile Robotics*, pages 548–554. World Scientific, 2012.

[GHD+19] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In *2019 American Control Conference (ACC)*, pages 4559–4566. IEEE, 2019.

[Gur03] Leonid Gurvits. Classical deterministic complexity of edmonds' problem and quantum entanglement. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 2003.

[HA17] Ayonga Hereid and Aaron D Ames. Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726. IEEE, 2017.

[Hal09] Jack K Hale. *Ordinary differential equations*. Courier Corporation, 2009.

[Ham18] Tareq Hamadneh. *Bounding Polynomials and Rational Functions in the Tensorial and Simplicial Bernstein Forms*. PhD thesis, University of Konstanz, 2018. View online.

[HCH+17] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: a modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522. IEEE, 2017.

[HCHA16] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1447–1454. IEEE, 2016.

[HFKG06] Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *Siam Review*, 48(2):207–304, 2006.

[HK13] Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2):297–312, 2013.

[HK14] Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2):297–312, 2014.

[HKZ+20] Patrick Holmes, Shreyas Kousik, Bohao Zhang, Daphna Raz, Corina Barbalata, Matthew Johnson-Roberson, and Ram Vasudevan. Reachable sets for safe, real-time manipulator trajectory design. *arXiv preprint arXiv:2002.01591*, 2020.

[HLS08] Didier Henrion, Jean B Lasserre, and Carlo Savorgnan. Nonlinear optimal control synthesis via occupation measures. In *2008 47th IEEE Conference on Decision and Control*, pages 4749–4754. IEEE, 2008.

[HLZ10] Simai He, Zhening Li, and Shuzhong Zhang. Approximation algorithms for homogeneous polynomial optimization with quadratic constraints. *Mathematical Programming*, 125(2):353–383, 2010.

[HSCN07] AC Van Der Heijden, AFA Serrarens, MK Camlibel, and Henk Nijmeijer. Hybrid optimal control of dry clutch engagement. *International Journal of Control*, 80(11):1717–1728, 2007.

[HW03] Eldon Hansen and G William Walster. *Global optimization using interval analysis: revised and expanded*, volume 264. CRC Press, 2003.

[HXA15] S. C. Hsu, X. Xu, and A. D. Ames. Control barrier function based quadratic programs with application to bipedal robotic walking. In *2015 American Control Conference (ACC)*, pages 4542–4548, July 2015.

[KF11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. View online.

[KGNSZ19] Xiaolong Kuang, Bissan Ghaddar, Joe Naoum-Sawaya, and Luis F Zuluaga. Alternative sdp and socp approximations for polynomial optimization. *EURO Journal on Computational Optimization*, 7(2):153–175, 2019.

[KHJ16] Milan Korda, Didier Henrion, and Colin N Jones. Controller design and value function approximation for nonlinear dynamical systems. *Automatica*, 67:54–66, 2016.

[KHV19] Shreyas Kousik, Patrick Holmes, and Ramanarayan Vasudevan. Safe, aggressive quadrotor flight via reachability-based trajectory design. *arXiv preprint arXiv:1904.05728*, 2019.

[KP14] Reza Kamyar and Matthew Peet. Polynomial optimization with applications to stability analysis and control-alternatives to sum of squares. *arXiv preprint arXiv:1408.5119*, 2014.

[KPT16] Twan Koolen, Michael Posa, and Russ Tedrake. Balance control using center of mass height variation: limitations imposed by unilateral contact. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 8–15. IEEE, 2016.

[Kuo07] Arthur D Kuo. Choosing your steps carefully. *IEEE Robotics & Automation Magazine*, 14(2):18–29, 2007.

[Kuw07] Yoshiaki Kuwata. *Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control*. PhD thesis, Massachusetts Institute of Technology, 02 2007. View online.

[KVB+18] Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *arXiv preprint arXiv:1809.06746*, 2018.

[KVJRV17] Shreyas Kousik, Sean Vaskov, Matthew Johnson-Roberson, and Ram Vasudevan. Safe trajectory synthesis for autonomous driving in unforeseen environments. In *ASME 2017 Dynamic Systems and Control Conference*, pages V001T44A005–V001T44A005. American Society of Mechanical Engineers, 2017.

[Las01] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

[Las09] Jean Bernard Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009.

[LHPT08] Jean B Lasserre, Didier Henrion, Christophe Prieur, and Emmanuel Trélat. Nonlinear optimal control via occupation measures and lmi-relaxations. *SIAM journal on control and optimization*, 47(4):1643–1666, 2008.

[LNQY09] Chen Ling, Jiawang Nie, Liqun Qi, and Yinyu Ye. Biquadratic optimization over unit spheres and semidefinite programming relaxations. *SIAM Journal on Optimization*, 20(3):1286–1310, 2009.

[LST12] John Lygeros, Shankar Sastry, and Claire Tomlin. Hybrid systems: Foundations, advanced topics and applications. *under copyright to be published by Springer Verlag*, 2012.

[LTY17] Jean B Lasserre, Kim-Chuan Toh, and Shouguang Yang. A bounded degree sos hierarchy for polynomial optimization. *EURO Journal on Computational Optimization*, 5(1-2):87–117, 2017.

[LZ10] Zhi-Quan Luo and Shuzhong Zhang. A semidefinite relaxation scheme for multivariate quartic polynomial optimization with quadratic constraints. *SIAM Journal on Optimization*, 20(4):1716–1736, 2010.

[LZG+19] Jinsun Liu, Pengcheng Zhao, Zhenyu Gan, Matthew Johnson-Roberson, and Ram Vasudevan. Leveraging the template and anchor framework for safe, online robotic gait design. *arXiv preprint arXiv:1909.11125*, 2019.

[LZG+20] Jinsun Liu, Pengcheng Zhao, Zhenyu Gan, Matthew Johnson-Roberson, and Ram Vasudevan. Leveraging the template and anchor framework for safe, online robotic gait design. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[Man07] Stefania Maniglia. Probabilistic representation and uniqueness results for measure-valued solutions of transport equations. *Journal de mathématiques pures et appliquées*, 87(6):601–626, 2007.

[Mat19] Matlab optimization toolbox, 2019. The MathWorks, Natick, MA, USA. View online.

[MLD03] Boris Mariere, Zhi-Quan Luo, and Timothy N Davidson. Blind constant modulus equalization via convex optimization. *IEEE Transactions on Signal Processing*, 51(3):805–818, 2003.

[MLEV19] Joshua G Mangelson, Jinsun Liu, Ryan M Eustice, and Ram Vasudevan. Guaranteed globally optimal planar pose graph and landmark SLAM via sparse-bounded sums-of-squares programming. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9306–9312. IEEE, 2019.

[MLV17] Shankar Mohan, Jinsun Liu, and Ram Vasudevan. Synthesizing the optimal luenberger-type observer for nonlinear systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 3658–3663. IEEE, 2017.

[MMTG92] Stefano Malan, Mario Milanese, Michele Taragna, and Jürgen Garloff. B/sup 3/algorithm for robust performances analysis in presence of mixed parametric and dynamic perturbations. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 128–133. IEEE, 1992.

[Moo66] Ramon E Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, 1966.

[MR93] Kenneth R. Muske and James B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993. View online.

[MT17a] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[MT17b] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017. View online.

[MVP16] Mohamad Shafiee Motahar, Sushant Veer, and Ioannis Poulakakis. Composing limit cycles for motion planning of 3d bipedal walkers. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6368–6374. IEEE, 2016.

[MVTT14] Anirudha Majumdar, Ram Vasudevan, Mark M Tobenkin, and Russ Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *The International Journal of Robotics Research*, page 0278364914528059, 2014.

[NA07] Paluri SV Nataraj and M Arounassalame. A new subdivision algorithm for the bernstein polynomial approach to global optimization. *International journal of automation and computing*, 4(4):342–352, 2007.

[NA11] Paluri SV Nataraj and M Arounassalame. Constrained global optimization of multivariate polynomials using bernstein branch and prune algorithm. *Journal of global optimization*, 49(2):185–212, 2011.

[NHG+16] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath. 3d dynamic walking on stepping stones with control barrier functions. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 827–834, Dec 2016.

[Nie13] Jiawang Nie. An exact jacobian sdp relaxation for polynomial optimization. *Mathematical Programming*, 137(1-2):225–255, 2013.

[NS15] Quan Nguyen and Koushil Sreenath. Optimal robust control for bipedal robots through control lyapunov function based quadratic programs. In *Robotics: Science and Systems*, 2015.

[NS16] Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *American Control Conference (ACC), 2016*, pages 322–328. IEEE, 2016.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[Par00] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[PB17] Andrew M Pace and Samuel A Burden. Piecewise-differentiable trajectory outcomes in mechanical systems subject to unilateral constraints. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 243–252. ACM, 2017.

[PC17] Ali Pakniyat and Peter E Caines. On the relation between the minimum principle and dynamic programming for classical and hybrid control systems. *IEEE Transactions on Automatic Control*, 62(9):4347–4362, 2017.

[PCT14] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[PKK09] Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009. View online.

[PKT17] Michael Posa, Twan Koolen, and Russ Tedrake. Balancing and step recovery capturability via sums-of-squares optimization. In *2017 Robotics: Science and Systems Conference*, 2017.

[PLSB11] Benjamin Passenberg, Marion Leibold, Olaf Stursberg, and Martin Buss. The minimum principle for time-varying hybrid systems with state switching and jumps. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 6723–6729. IEEE, 2011.

[PR14] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[PY19] Dávid Papp and Sercan Yildiz. Sum-of-squares optimization without semidefinite programming. *SIAM Journal on Optimization*, 29(1):822–851, 2019.

[QT03] Liqun Qi and Kok Lay Teo. Multivariate polynomial minimization and its application in signal processing. *Journal of Global Optimization*, 26(4):419–433, 2003.

[QWY04] Liqun Qi, Zhong Wan, and Yu-Fei Yang. Global minimization of normal quartic polynomials based on global descent directions. *SIAM Journal on Optimization*, 15(1):275–302, 2004.

[RC95] Dietmar Ratz and Tibor Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7(2):183–207, 1995.

[RCBL19] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[Rob18] Agility Robotics. Cassie kinematic model. https://github.com/agilityrobotics/agility-cassie-doc/wiki/Kinematic-Model, 2018.

[Rob19a] Agility Robotics. Agility robotics. http://www.agilityrobotics.com/, 2019.

[Rob19b] Agility Robotics. Cassie: Dynamic planning on stairs. https://www.youtube.com/watch?v=qV-92Bq96Co, 2019.

[SA90] Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[SAGVR17] Nils Smit-Anseeuw, Rodney Gleason, Ram Vasudevan, and C David Remy. The energetic benefit of robotic gait selection: A case study on the robot ramone. *IEEE Robotics and Automation Letters*, 2017.

[SARV19] Nils Smit-Anseeuw, C David Remy, and Ram Vasudevan. Walking with confidence: Safety regulation for full order biped models. *arXiv preprint arXiv:1903.08327*, 2019.

[SC07] M Shahid Shaikh and Peter E Caines. On the hybrid optimal control problem: theory and algorithms. *IEEE Transactions on Automatic Control*, 52(9):1587–1603, 2007.

[SCEM07] Angela Schollig, Peter E Caines, Magnus Egerstedt, and Roland Malhamé. A hybrid bellman equation for systems with regional dynamics. In *Decision and Control, 2007 46th IEEE Conference on*, pages 3393–3398. IEEE, 2007.

[So11] Anthony Man-Cho So. Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems. *Mathematical programming*, 129(2):357–382, 2011.

[SOS10] Manuel Soler, Alberto Olivares, and Ernesto Staffetti. Hybrid optimal control approach to commercial aircraft trajectory planning. *Journal of Guidance, Control, and Dynamics*, 33(3):985–991, 2010.

[SS15] Mohamed Amin Ben Sassi and Sriram Sankaranarayanan. Bernstein Polynomial Relaxations for Polynomial Optimization Problems, 2015.

[Sus99] Héctor J Sussmann. A maximum principle for hybrid optimal control problems. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 1, pages 425–430. IEEE, 1999.

[SVBT14a] V. Shia, R. Vasudevan, R. Bajcsy, and R. Tedrake. Convex computation of the reachable set for controlled polynomial hybrid systems. In *53rd IEEE Conference on Decision and Control*, pages 1499–1506, Dec 2014.

[SVBT14b] Victor Shia, Ram Vasudevan, Ruzena Bajcsy, and Russ Tedrake. Convex computation of the reachable set for controlled polynomial hybrid systems. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 1499–1506. IEEE, 2014.

[SYA19] S.W. Smith, H. Yin, and M. Arcak. Continuous abstraction of nonlinear systems using sum-of-squares programming. In *58th Conference on Decision and Control*, 2019.

[TBM17] J. Z. Tang, A. M. Boudali, and I. R. Manchester. Invariant funnels for underactuated dynamic walking robots: New phase variable and experimental validation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3497–3504, May 2017.

[TCHL19] Matteo Tacchi, Carmen Cardozo, Didier Henrion, and Jean Lasserre. Approximating regions of attraction of a sparse polynomial differential system. *arXiv preprint arXiv:1911.09500*, 2019.

[TCL93] I Thng, Antonio Cantoni, and Yee Hong Leung. Derivative constrained optimum broad-band antenna arrays. *IEEE Transactions on Signal Processing*, 41(7):2376–2388, 1993.

[TG17] Jihad Titi and Jürgen Garloff. Fast determination of the tensorial and simplicial bernstein forms of multivariate polynomials and rational functions. 2017. View online.

[VEH94] R Vaidyanathan and M El-Halwagi. Global optimization of nonconvex nonlinear programs via interval analysis. *Computers & Chemical Engineering*, 18(10):889–897, 1994.

[Vin93] Richard Vinter. Convex duality and nonlinear optimal control. *SIAM journal on control and optimization*, 31(2):518–538, 1993.

[VKL+19] Sean Vaskov, Shreyas Kousik, Hannah Larson, Fan Bu, James Ward, Stewart Worrall, Matthew Johnson-Roberson, and Ram Vasudevan. Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments. *arXiv preprint arXiv:1902.02851*, 2019.

[VP18] Sushant Veer and Ioannis Poulakakis. Safe adaptive switching among dynamical movement primitives: Application to 3d limit-cycle walkers. *arXiv preprint arXiv:1810.00527*, 2018.

[VS72] Miomir Vukobratović and J Stepanenko. On the stability of anthropomorphic systems. *Mathematical biosciences*, 15(1-2):1–37, 1972.

[WCC+07] Eric R Westervelt, Christine Chevallereau, Jun Ho Choi, Benjamin Morris, and Jessy W Grizzle. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007.

[WG15] Tyler Westenbroek and Humberto Gonzalez. Optimal control of hybrid systems using a feedback relaxed control formulation. *arXiv preprint arXiv:1510.09127*, 2015.

[WGC+07] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*, volume 28. CRC press, 2007.

[WGC+18] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[WGK03] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE transactions on automatic control*, 48(1):42–56, 2003.

[Wie02] Pierre-Brice Wieber. On the stability of walking systems. In *Proceedings of the international workshop on humanoid and human friendly robotics*, 2002.

[WKW08] Derek L Wight, Eric G Kubica, and David W Wang. Introduction of the foot placement estimator: A dynamic measure of balance for bipedal robotics. *Journal of computational and nonlinear dynamics*, 3(1):011009, 2008.

[WO13] Patrick M Wensing and David E Orin. High-speed humanoid running through control with a 3d-slip model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140. IEEE, 2013.

[YG05] Kerim Yunt and Christoph Glocker. Trajectory optimization of mechanical hybrid systems using sumt. In *Advanced Motion Control, 2006. 9th IEEE International Workshop on*, pages 665–671. IEEE, 2005.

[YGF+16] B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller. Real time integrated vehicle dynamics control and trajectory planning with mpc for critical maneuvers. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 584–589, June 2016. View online.

[ZG98] Michael Zettler and Jürgen Garloff. Robustness analysis of polynomials with polynomial parameter dependency using bernstein expansion. *IEEE Transactions on Automatic Control*, 43(3):425–431, 1998. View online.

[ZMV16] Pengcheng Zhao, Shankar Mohan, and Ram Vasudevan. Control synthesis for nonlinear optimal control via convex relaxations. *arXiv preprint arXiv:1610.00394*, 2016.

[ZMV17a] Pengcheng Zhao, Shankar Mohan, and Ram Vasudevan. Control synthesis for nonlinear optimal control via convex relaxations. In *2017 American Control Conference (ACC)*, pages 2654–2661. IEEE, 2017.

[ZMV17b] Pengcheng Zhao, Shankar Mohan, and Ram Vasudevan. Optimal control for nonlinear hybrid systems via convex relaxations. *arXiv preprint arXiv:1702.04310*, 2017.

[ZMV19] Pengcheng Zhao, Shankar Mohan, and Ramanarayan Vasudevan. Optimal control of polynomial hybrid systems via convex relaxations. *IEEE Transactions on Automatic Control*, 2019.

[ZV19] Pengcheng Zhao and Ram Vasudevan. Nonlinear hybrid optimal control with switching costs via occupation measures and lmi-relaxations. In *2019 American Control Conference (ACC)*, pages 4293–4300. IEEE, 2019.