C^1-Smooth Multi-Contact Dynamics for Robotics

by

Marion Anderson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical & Computer Engineering (Robotics))
in the University of Michigan
2025

Doctoral Committee:

Associate Professor Shai Revzen, Chair
Assistant Professor Daniel Bruder
Assistant Professor Jing Shuang (Lisa) Li
Associate Professor Peter Seiler

Lee M. Anderson, Jr.
marand@umich.edu
ORCID iD: 0000-0002-7578-1257

To my parents.
This is as much a credit to your work as it is to mine.

ACKNOWLEDGEMENTS

I owe the greatest debt to my parents, to whom this dissertation is dedicated. As any parent must know, they have dedicated hours upon untold hours to my life and well-being writ large. I cannot imagine it was easy, though they insist otherwise. I have had the good fortune to know my parents into my adulthood, and been similarly blessed with the perspective that can accompany such aging. To my mother I owe my sense of justice, my capacity for wonder, and my curiosity. These gifts provided light when the path was dark, as it so often is during a Ph.D. I do not know that I would have found my way to this dissertation without them. To my father I owe my love of the outdoors and the near-unshakable belief that any problem can be solved with only enough time or the right team – and if it cannot, there is a workaround. They gave me the courage and the peace of mind to look for my way here. I owe similar debts to my Grandmother Joan, Aunt Annie, Uncle Gerry, and Aunt Alison. I would like to acknowledge especially Martha Hewitt, who was like family and taught me how to be kind when I was a child. She lived to hear that I passed my defense, but not to the submission of this document. May she rest in peace.

There are still a great many people to acknowledge, and I hope you will bear with me as I engage in the Sisyphean attempt to name a fraction of them. I should begin with my advisor, Shai Revzen, whose patience and duck-like calm allowed me to take this work where I found interesting and and my committee who helped make it intelligible. Thank you to my lab mates: Daphne (Yishun) Zhou, Ziyou Wu, Brian Bitter, Daniel Kennedy, Erick Vega, Taylor McLaughlin, and Nicholas Pagliocca for reminding me what makes academia worthwhile.

I have been incredibly fortunate to make so many friends along the way who have helped me through this process – Jennifer with weekly phone calls, my DND group giving me something to laugh about every weekend, AJ, Triton, Ryan, Ryan, and Ryan, Bradley, Haley, Sidd, Sidh, Stacy, Natasha, Lieb for reminding me that there is life both beside and outside of graduate school. Thank you to Nana, Matt, Andrew, and Laksh for helping me make it through the writing. Thank you to my roommates over the years: Andrew, Jeff, Trevor, Jim, Declan, Daniel and our frequent guests Nate, Emilia, Shelley, and Olivette for making home a friendly place. Thank you to my teachers, Mr. Guldin, Mr. Huber, and Mr. Yoh for always believing I would get up to something interesting. A particular thanks to my three-time physics teacher, Mr. Born for introducing me to electrical engineering and providing perhaps the single largest push towards this journey. Thank you to Mrs. Bessom for introducing me to science fiction as a genre. Thank you to Justin, Katie, and Faith for always keeping me on my toes and challenging me to dream of concrete ways to make the world a better place. Thank you to my students for their trust in me and everything I have learned from them. Thank you to everyone who gave me a change to try something new, from rocketry to stage management. Finally, thank you, dear reader, for making it this far. I hope you find this dissertation digestible and that you learn something new.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vii

LIST OF TABLES xi

ABSTRACT xii

CHAPTER

1 Introduction 1

1.1 Conceptual Overview & Relevance Checklist 2
1.2 Background 3
1.3 Contributions 6
1.3.1 Chapter 2: Experimental Confirmation of Once-Differentiable Multi-Contact Dynamics in Nature 6
1.3.2 Chapter 3: Mode-Schedule-Agnostic Real-Time Optimal Control of Multi-Contact Dynamics 7
1.3.3 Chapter 3: Tail-Like Actuation Scheme 7
1.3.4 Chapter 4: Proof and Conditions of 3rd-Order Event Localization Accuracy Using Hermite Splines 8
1.3.5 Chapter 4: Software Interface for Rapidly Integrating Multi-Contact Dynamics 8
1.3.6 Chapter 4: Computational Benchmarks for Multi-Contact Integrators 9
1.3.7 Chapter 4: Performance Assessment Against MuJoCo 10
1.4 Discussion 10

2 Evidence of C^1-Smooth Multi-Contact Dynamics on a Physical Robot 12

2.1 Introduction 12
2.1.1 Motivation 12
2.1.2 Background 13
2.1.3 Hypothesis 16
2.2 Methods 18

2.2.1 Experimental Design & Data Collection 18
2.2.2 Selecting Poincaré Sections 19
2.2.3 Determining Touchdown Contact Order (Mode Schedule) . . 21
2.2.4 Impact Map Estimation & Comparison 22
2.3 Results 24
2.4 Conclusion 28

3 Mode-Schedule-Agnostic Steering of a Three-Legged Hopping Robot 29

3.1 Introduction 29
3.2 Background 30
3.3 Robot 33
3.4 Control Scheme 35
3.4.1 Dynamical Model 35
3.4.2 System Identification 36
3.4.3 Transfer to Hardware 37
3.5 Experimental Results 44
3.6 Discussion 44

4 A Numerical Integrator Specialized for Event-Selected Multi-Contact Dynamics 51

4.1 Introduction 51
4.2 Improving Interpolating Splines of Scalar Functions Composed on Event-Selected Trajectories for Event Localization 53
4.2.1 Motivation 53
4.2.2 Background 54
4.2.3 Proof of 3rd-Order Accuracy 55
4.3 Algorithm 60
4.3.1 Discontinuity Locking 60
4.3.2 Single Multi-Contact Resolution 61
4.3.3 Usage 61
4.3.4 Modeling Example 65
4.4 Numerical Experiments 65
4.4.1 Algorithm Implementation 65
4.4.2 Order Test 65
4.4.3 Calibration against MuJoCo 67
4.4.4 Scaling v.s. MuJoCo with Multiple Contacts 70
4.5 Discussion 71

LIST OF FIGURES

FIGURE

2.1 The three-legged hopping robot we developed for this experimental work. The robot "preloads" its elastic legs prior to ground contact using a cable assembly so that the legs immediately exert a force on the ground at the moment of contact. As a result, the contact force model is affine and cannot be readily solved using the machinery for linear complementarity problems [Bro03]. 19

2.2 Data-driven selection of Poincaré sections using marginal probability distributions. Adjacent plots share axes. Vertical dashed black lines correspond to the Poincaré sections used to segment the motion capture data. Horizontal dashed black lines correspond to the minimum (maximum) foot height thresholds of the body height conditional probabilities. Clockwise from the bottom right: minimum (maximum) foot height versus body height scatter plot, probability density of minimum (maximum) foot height, conditional probability densities of body height conditioned on minimum (maximum) foot height. 21

2.3 Determining the foot contact ordering (mode schedule) of a touchdown. Foot contact occurs when the foot z-velocity (blue dashed lines) attains its minimum during a touchdown transition ((solid yellow fill). The order of foot z-velocity minima (black plus signs) determines the touchdown's mode schedule. All feet must contact the ground during the touchdown transition, or else the touchdown is excluded from the dataset. 22

2.4 Boxplots of the bootstrapped combined prediction mean-squared error (MSE) of impact map models describing the 3D positions and 3D velocities of the three body markers. From left to right, affine model specialized for each mode schedule, unspecialized affine model near the triple-contact trajectory, unspecialized affine model ignoring distance from the triple-contact trajectory, unspecialized affine model far from the triple-contact trajectory, mean predictor specialized for each class, unspecialized mean predictor. Length units were normalized to 29.13mm and 27 milliseconds. 26

2.5 Cumulative density (top) and box (bottom) plots showing the spread of arc-length distances in our dataset. The significant overlap between the cross-class and in-class distances suggests that the dominant dynamics are identical between mode schedules, and the hopper dynamics are C^1smooth. Distances between identical matrices subject to additive noise ("$I_{34,17}$ + noise") emulates the spread of distances expected from identical matrices estimated from experimental data (e.g. motion capture of a hopping robot) with a signal-to-noise ratio of 10. Distances between random matrix pairs ("random") emulates the spread of unrelated dynamics. The dataset distances' overlap with identical matrices and its distance from random matrices further suggests that the hopper dynamics are C^1smooth. 27

3.1 Diagram of the three-legged hopping robot used in the experiments. Left-to-right: (a) overall assembly, (b) detail of a leg actuator, (c) detail of the counterweight-boom actuator. All motors are MX-106 brushed DC motors from Dynamixel. (a) A slot-together frame made of quarter-inch lasercut ABS forms the chassis. It is held together by Kevlar-embedded tape wrapped horizontally around the frame. Actuators are secured to the chassis via M2.5, M3 screws and locknuts. (b) The leg is a 1075 spring steel beam connected to the quarter-inch ABS lever arm via a thin metal cable. The motor raises and lowers the lever arm to store and release energy in the leg. (c) The boom actuator moves a counterweight around an axis not aligned with the body frame. We used a 4S LiPo battery for the counterweight. 34

3.2 Photograph of the hopping robot. Power is supplied by the battery counterweight. Motor communication travels through the attached Ethernet cable (white). 35

3.3 Heatmaps of the identified frame-change per cycle based on actuator state (ϕ) and control input $\Delta\phi$. The hopper is able to turn, drift right or left, but cannot hop "backwards." This indicates the hopper's motion is nonholonomic and not immediately straightforward to classical linear control. Note that each heatmap has a different scale. ϕ ranges from -2π to 4π showing that the regressions are periodic or nearly so. $\Delta\phi$ ranges from $-\pi/4$ to $\pi/4$. 39

3.4 Hopping control process flow diagram. Left, a thread asynchronously polls for new data from the motion capture system and stores the hopper's SE(2) state in a buffer. Right, a separate thread controls the hopper's motors, reading the SE(2) state from the buffer only when needed for planning. 43

3.5 The hopper's motion in the unimpeded origin stabilization experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. Our control scheme was able to adapt to eventual deviation from the pure simulation response, showing that our simplified control scheme is effective. 45
3.6 The hopper's motion in the unimpeded turning experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. Our control scheme was able to adapt to eventual deviation from the pure simulation response, showing that our simplified control scheme is effective. 46
3.7 The foamcore board blocking the hopper's path as it move towards the origin. The foamcore caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller. 47
3.8 The hopper's motion in the impeded origin stabilization experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. The foamcore caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller. 48
3.9 The hopper's motion in the impeded turning experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. The path obstructions caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller. 49

4.1 Densely integrating $h_k(x(t))$ directly to resolve the kth guard impact. By abuse of notation, h_k refers to the value of $h_k(\cdot)$ as a variable. The system obeys $\dot{x} = f_1(x)$ for $h_k < 0$ and $\dot{x} = f_2(x)$ otherwise. Using the multivariable chain rule, h_k follows the scalar differential equation $\dot{h}_k = \nabla h_k \cdot f_1(x)$ until guard impact ($h_k = 0$). Thus the value of $\nabla h_k \cdot f_1(x)$ at (unknown) t^*, is identical to "true" solution. Furthermore, $\nabla h_k \cdot f_1(x)$ is smooth, and can be densely integrated using a cubic spline [HWN93, p.190, Eqn. 6.7]. This transforms 64

4.2 Log-log plot of our algorithm's RMS integration error versus $1/\varepsilon$ (dotted line) for an initial value problem. The data interval used to determine the order (gray rectangle) and the fit extended over the entire interval considered (dashed line) are overlaid. The integrator achieved third-order accuracy, indicated by the slope of the dashed black lined being above 4. The tails for small and large $1/\varepsilon$ is expected. 68
4.3 Time series output of our calibration: integrating a the trajectory of a 1D hopping robot. The top row of plots show the exact trajectory (dotted gray), MuJoCo's result (dashed blue), and our result (red) overlaid. Due to our integrator's large timestep, the top left plot is numerically accurate, but visually undersampled. The bottom row shows the residual from the exact trajectory for both MuJoCo and our solver. The left plots show height above the ground (contact height = 1) and the right plots show vertical velocity. 69
4.4 The marginal cost of our algorithm (red) and MuJoCo (dashed blue) for increasing number of spring-damper contacts. We integrated the same $N = 150$ random initial conditions for each number of contacts. Our algorithm implemented in pure Python has roughly the same median marginal cost per contact as MuJoCo. This suggests that our numerical method would easily outperform MuJoCo in a more optimized implementation. 71

LIST OF TABLES

TABLE

1.1 Conceptual checklist for determining if a region of a system is ESS (left) or FoPAS (right). FoPAS [CRB22] possess a classical Jacobian, but ESS apply to a broader range of dynamics [Bur+16]. Coarsely, FoPAS are mechanical ESS with spring-like (not necessarily soft) contact forces. Both classes describe local phenomena, and periodic dynamics can be represented as a combination of multiple ESS/FoPAS. 3

4.1 Description of variables and relevant constants used in this proof. 55
4.2 Notation used in the integration algorithm. 60
4.3 The 3D system of affine ODEs used for our order test. The x velocity (a), y velocity (b), and z velocity (c) are grouped separately. 67
4.4 Calibration parameters determined by simulating the same 1D hopper initial value problem with MuJoCo and our algorithm. The runtime in seconds of each method is tabulated in the right two columns. MuJoCo used a timestep of 0.002 seconds. We were unable to find parameters which made the MuJoCo simulation comparably accurate to ours, so we used a maximum timestep of 1 second for our integrator. 70

ABSTRACT

When a robot forms multiple contact points with its environment near-simultaneously, small perturbations can change the anticipated order of contact, which can lead to vastly different outcomes. These "multi-contact dynamics" have eluded convenient mathematical analysis, despite posing no issue for a plethora of terrestrial organisms ranging from ants to elephants. Classically, simulating a three-legged hopping robot a mere 2 hops into the future would require solving over 1,000 separate differential equations. This dissertation leverages recent theoretical work to show that many multi-contact dynamics are – in practice – easier to model, control, and simulate than previously believed. We begin by introducing a three-legged hopping robot and showing experimentally that its flight-to-triple-stance impact map is linearizable (C^1) near the triple-contact trajectory. Using this result, we employ smooth system identification and control tools to successfully steer the hopper in the plane in real-time, even in the presence of significant disturbances. Finally, we present a numerical method for rapidly and accurately solving initial value problems belonging to the more general class of "event-selected [multi-contact] systems" that performs on-par with the state-of-the-art via a comparison with MuJoCo.

CHAPTER I

Introduction

Physical contact, from picking up a glass of water to foot placement while walking is an ever-present aspect of the human experience. It is unsurprising that the field of robotics – which broadly seeks to build devices capable of performing human tasks – has had a sustained interest in modeling contact for many decades. This interest has borne significant fruit. Legged robots like Boston Dynamics' Spot [Hut+16] are the obvious example, but assembly line manipulators [Bic00], drone "sky cranes" [AGI], and even cars [Wil+16] rely on contact and contact models to perform their tasks. However, most of these examples are specially designed to simplify the kinds of contact they might experience. Nearly every robot quadruped walks according to a Buehler Clock [Bue+99], different kinds of cars have different tires [F124], and no commercially available robot biped has toes despite their ubiquity in primates.

Analyzing contact, both mathematically and physically, is awkward. Tools for robot planning and control almost categorically rely on smoothness – the existence of first order approximations, typically in the form of a derivative. Contact usually introduces jump discontinuities which require further analysis beyond the familiar derivative tooling. The ensuing awkwardness compounds when multiple contacts are formed in an indeterminate order. It is generally unclear if such *multi-contact dynamics* are indeed governed by a first-order approximation, as is the case for the more familiar class of smooth systems. In the extreme, multi-contact systems may even lack a continuous dependence on initial conditions. As a result each contact

ordering, called a *mode schedule*, is usually treated separately, leading to a combinatorial analysis. However, nearly all terrestrial animals successfully control and plan for multi-contact dynamics. This suggests that multi-contact dynamics may be much simpler than meets the eye.

Recent theoretical advances claim that many physically-common multi-contact dynamics are governed by a first order approximation [Bur+16], and a large subset is even classically once-differentiable (C^1 or C^1-smooth) [CRB22, Sec. 3.2]. If present in practice, this result could dramatically simplify the analysis required to control and analyze such systems. This dissertation attempts to bring these results into the physical realm. It is my hope that the experiments herein will make the control of multi-contact dynamics less resource-intensive, thereby lowering the barrier of entry to building robots and developing stranger actuation topologies and motion plans.

1.1 Conceptual Overview & Relevance Checklist

Before proceeding with theoretical minutia, we present a conceptual overview to help the reader decide if this work is relevant to their system. This dissertation is based on the theory of *Event-Selected Systems* (ESS) [Bur+16]. At its core, ESS theory describes local properties of dynamics that are smooth except at the moment of contact. Each contact is represented by a smooth surface in the state space (a *manifold*) called a *guard*. If all trajectories in a neighborhood containing the guards are guaranteed to cross them exactly once, then a system is likely ESS. We call such a neighborhood an *ESS complex*. Periodic dynamics can be described with ESS theory by subdividing the state space into multiple ESS complexes. Since an ESS must cross its guards, it cannot experience "pure" hard contact such as a perfectly rigid ball bouncing on a perfectly rigid table. Regardless, many hard contacts of interest are ESS. A legged robot's center of mass dynamics can be ESS even when the feet make hard contact

with the ground so long as the guards describing foot contact in the relevant state space are crossed. Further, many second-order mechanical ESS with position-based contacts are fully C^1-smooth. To simplify Checklists are laid out in Tab. 1.1.

☐ Trajectories are continuous	☐ Is ESS
☐ Collisions appear as smooth surfaces in the state space	☐ States can be grouped into "position" and "velocity" coordinates
☐ Every contact is locally made/broken exactly once	☐ Contacts are functions of only the position variables
☐ Contact does not completely annihilate penetration velocity	☐ Contacts introduce additive "accelerations"

Table 1.1: Conceptual checklist for determining if a region of a system is ESS (left) or FoPAS (right). FoPAS [CRB22] possess a classical Jacobian, but ESS apply to a broader range of dynamics [Bur+16]. Coarsely, FoPAS are mechanical ESS with spring-like (not necessarily soft) contact forces. Both classes describe local phenomena, and periodic dynamics can be represented as a combination of multiple ESS/FoPAS.

1.2 Background

Contact dynamics have been studied since at least the time of Newton [WM92], and can be treated with many different formalisms. We refer the reader to [Bro16] for a detailed treatment. The most general approaches use differential inclusions [Mor88; Fil88; Bro+06] and specialized applications of hybrid automata [GST09; Lyg+01]. In manipulation, the need to understand the forces necessary to keep two rigid bodies in constant contact (e.g. holding an object between the thumb and forefinger) has popularized the linear complementarity formulation of contact [ST00b; AP97; Faz+17]. This formalism allows ready computation of contact forces using familiar optimization tools [Löt82; CPS09], can treat friction with sufficient fidelity [ST00b; AP97], and in some cases can even be used to plan over multiple mode

schedules [PCT14]. As a result, linear complementarity has become the de-facto standard for solving contact forces in simulation [Lee+18; CB16; TET12]. However, despite some stability results [PTT16], the complementarity formulation does not generally allow for dynamical analysis through contact.

In legged locomotion, the ratio between a robot's weight and its lateral velocity makes the assumption of sticking friction more robust [WGK03]. In this context, understanding the instantaneous effects of foot contact becomes of greater importance. Usually, *hard contact*, where any penetrating velocity is destroyed at the moment of contact, is assumed. This has brought significant attention to the use of impact maps (sometimes called reset maps) [Lyg+01; WGK03; GS09]. Impact maps describe the sudden change in motion that occurs when two objects collide. This change usually makes the underlying dynamics discontinuous [GS09]. Despite being discontinuous, the response to perturbations near a single contact is still governed by a linear approximation. The *saltation matrix*, introduced in [AG58] and described in practice by [Kon+24], describes the effect of a pre-contact perturbation on the post-contact state. It functions similarly to a Jacobian matrix. So much so that new kinds of Kalman filter have been built around it [Kon+21; PKJ22a]. However, this linearization only holds for individual contacts and specific mode schedules. [RBS10, Sec. 5.1] observed that hard simultaneous contact creates a discontinuous transfer function which prohibits tools like Jacobian and Floquet analysis. Here is the core of the combinatorial explosion – the dynamics of different mode schedules can be completely unrelated, and so each of the combinatorially-many possible mode schedules must be considered.

Resolving the combinatorial explosion requires relaxing the assumption of perfectly rigid bodies. [Bur+16; PB17] observed that when some penetrating velocity is preserved after contact, many multi-contact dynamics become *Event Selected Systems* (ESS). ESS have a piecewise-smooth structure in the form of a *Bouligand derivative* [Sch12, Chp. 3]. The Bouligand derivative, when it exists, is numerically identical

to the directional derivative [Sch12, p. 65]. Each smooth "piece" corresponds to a mode schedule. As above, the linearization for each mode schedule is an ordered product of Jacobian and Saltation matrices. The core insight of ESS was continuity and Bouligand differentiability; the individually smooth pieces meet continuously at the simultaneous contact trajectory. The Bouligand derivative preserves chain rule and product rule, allowing the familiar tools of control and dynamical systems to be used on ESS, albeit in a slightly unfamiliar way. Extending this, [CRB22, Sec. 3.1] introduced *Force-Position Additive Systems* (FoPAS). FoPAS are a subclass of ESS which are fully C^1, i.e. they possess a classical Jacobian matrix. Many mechanical ESS with springy contact are FoPAS. The springy contact forces allow the saltation matrices to commute, making each smooth "piece" meet smoothly, not just continuously, at the simultaneous trajectory.

We now recapitulate the technical conditions and discuss their physical intuition.

Definition 1.2.1. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$ and a function $h \in C^r(U, \mathbb{R})$ defined on an open subset $U \subset D$, we say that h is an event function for F on U if there exists a positive constant $f > 0$ such that $Dh(x)F(x) \geq f$ elementwise, for all $x \in U$. A codimension-1 embedded submanifold $\Sigma \subset U$ for which $h|_\Sigma$ is constant is refereed to as a local section for F. [Bur+16, Defn. 1].*

Defn. 3.2.1 establishes the "liveness condition" of ESS. It ensures that all guards are (eventually) crossed, restricting the dynamics from exhibiting sliding modes or branching. The event functions can be chosen such that each $h(x) = 0$ is a guard. The specific choice of constant is arbitrary, but $h|_\Sigma = 0$ aligns with canonical notation in the literature [ST00a].

Definition 1.2.2. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$, we say that F is event-selected C^r at $\rho \in D$ if there exists an open set $U \subset D$ containing ρ and a collection $\{h_j\}_{j=1}^n\} \subset C^r(U, \mathbb{R})$ such that*

1. *(event functions)* h_j *is an event function for* F *on* U *for all* $j \in \{1, ..., n\}$

2. *(*C^r *extension) for all* $b \in \{-1, +1\}^n = B_n$*, with*

$$D_b = \{x \in U : b_j(h_j(x) - h_j(\rho)) \geq 0\},$$

$F|_{IntD_b}$ *admits a* C^r *extension* $F_b : U \to TU$

[Bur+16, Defn. 2].

Defn. 3.2.2 establishes that event-selectedness is a local property. Requiring that each guard be crossed exactly once, irrevocably prevents periodic and rhythmic behaviors, such as hopping. By constructing event-selectedness as a local property, periodic systems can be treated as several event-selected systems joined together, preserving both the mathematical convenience of event-selectedness and the generality of mechanical collisions.

1.3 Contributions

1.3.1 Chapter 2: Experimental Confirmation of Once-Differentiable Multi-Contact Dynamics in Nature

In this chapter, we show that a physical robot with multi-contact dynamics has a single, C^1-smooth impact map. We developed a three-legged hopping robot which makes hard contact with the ground via springy contacts, measured its motion, and estimated its flight-stance impact maps. Classically, the hopper would be considered to have six distinct impact maps, one for each of the $3! = 6$ possible mode schedules during the flight-stance impact. We showed that each of these six impact maps are practically the same. We then concluded that the hopper's impact dynamics are once-differentiable, suggesting that smooth tools for estimation, control, and optimization

are fully applicable to such systems. This result forms the basis of the control scheme in Chp. 3. We also provide an overview of the signal processing and statistics involved for those curious.

1.3.2 Chapter 3: Mode-Schedule-Agnostic Real-Time Optimal Control of Multi-Contact Dynamics

Simulating the hopper from Chp. 2 a mere two hops into the future involves over 1,000 distinct mode schedules. Familiar optimization approaches rapidly become untenable in real-time under this framework. We demonstrated real-time receding-horizon optimal control of the hopper over a 3-hop horizon. Our approach did not consider mode schedules at all, reducing 46,656 differential equations to a set of 3 difference equations. We demonstrated the hopper's ability to turn and approach waypoints, even in environments with very inconsistent friction. The system identification and optimization tools used were largely textbook, but are presented for replicability's sake.

1.3.3 Chapter 3: Tail-Like Actuation Scheme

We steered the hopper using a tail-like counterweight-boom actuator. Rather than control the hopper's motion by directly modulating the contact forces, we moved the hopper's center-of-mass using the counterweight. This proved more effective for turning the hopper and controlling the direction of its hops. To improve turning, we moved the boom only during flight to take advantage of the conservation of angular momentum.

1.3.4 Chapter 4: Proof and Conditions of 3rd-Order Event Localization Accuracy Using Hermite Splines

Solving initial value problems (IVPs) for nonsmooth mechanical systems, e.g. for multi-contact dynamics, typically involves solving familiar smooth IVPs and joining them at the moments of contact, called *events*. Determining when these events occurred – *event localization* – is critical to accurately simulating nonsmooth mechanics. Event localization is usually computationally expensive, and numerous schemes have been employed to trade off runtime performance and numerical accuracy. We proved that the dynamics considered in this work admit rapid and accurate event localization using familiar Hermite splines without detailed consideration to the structure of the event functions. We highlight the specific property that ensures the approximation error is bounded by a 3rd-order polynomial in both time and space and provide a proof using a triangle inequality.

1.3.5 Chapter 4: Software Interface for Rapidly Integrating Multi-Contact Dynamics

We developed a software interface which allows our event localization algorithm to be used with any ordinary differential equation solver that supports event detection. When an event is detected, the trajectory information is passed to our event localization algorithm, which efficiently resolves the possibly-multiple event(s) before returning the updated trajectory to the classical integrator. When multiple events occur within a single timestep, our algorithm resolves them using a mix of Hermite spline interpolation and single-step 3rd-order Runge-Kutta integration.

1.3.6 Chapter 4: Computational Benchmarks for Multi-Contact Integrators

We sought to evaluate our integration scheme's performance, both on its own and in comparison to MuJoCo. Due to a lack of standard benchmarks in the literature, we devised our own. None of these benchmarks is particularly complex, but due to a lack of literature we consider them important contributions. We first designed a three-dimensional piecewise-affine system with three contact events, each corresponding to the trajectory crossing the xy, xz, or yz-planes, respectively. All piecewise-affine systems have a known closed-form solution through piecewise application of the matrix exponential under a coordinate shift, making them suitable for assessing the accuracy of numerical methods. Having each event correspond to crossing a coordinate plane was an artifact of convenience. Coordinate planes are visually apparent and orthogonal, making them suitable for qualitative troubleshooting and more detailed mathematical analysis. We then developed a simulation of a one-dimensional vertical hopper with a springy contact. This is a mechanical system with straightforward constraints and a readily-computable closed-form solution that any robot simulator should be able to handle. Since this system contains exactly one contact event, it is a suitable "calibration" problem to assess how a numerical method's performance scales with additional contact points. Further, being a conservative mechanical system, it can be used to troubleshoot numerical codes by tracking where they violate the conservation of energy. Finally, we devised a test problem which can be procedurally generated to the effect of arbitrarily-many contacts: an infinitely-long bar falling onto a bed of evenly-spaced frictionless spring contacts. In concert with the vertical hopper, this problem can be used to compare the computational-cost-per-contact of simulators regardless of simulator implementation or overhead.

1.3.7 Chapter 4: Performance Assessment Against MuJoCo

We first confirmed that our pure-Python implementation was 3rd-order accurate using the three-dimensional piecewise-affine benchmark. We then sought to calibrate its runtime performance against MuJoCo using the vertical hopper benchmark. Interestingly, the efficiency of MuJoCo's contact solver comes at the cost of numerical stability: the vertical hopper gained energy in the MuJoCo simulation. After calibration, we compared our pure-Python implementation against MuJoCo, finding that it performed nearly on-par with superior numerical accuracy.

1.4 Discussion

We have demonstrated that, with a mild relaxation of perfectly rigid impacts, evaluating dynamics across different mode schedules in practice becomes straightforward and, in many cases, trivial. This relaxation can be as simple as a robot leg exhibiting a **reactive** impedance between the body and ground. Although rigid impacts are more popular in the robotics community [Lee+18; CB16; WGK03], this approach provides great gains. Chiefly, these systems are linearizable. Prior to this work, it was unknown if any physical multi-contact dynamics could be differentiated near simultaneous impact [RBS10]. As a result, multi-contact has generally been avoided [Bic00], with only some approaches using specific kinds of contact models [PCT14].

Many mechanical systems with impact become FoPAS when the assumption of perfectly rigid impact is relaxed. We have shown that a physical FoPAS with three contacts is classically linearizable, despite our system experiencing hard (but not rigid) ground impact. It was not previously known that such a system could possess a Jacobian matrix. Although we needed 3,000 impacts to confirm this result, an accurate data-driven model should require a dataset of only 100 impacts of arbitrary mode schedule. We leveraged this insight to effectively steer a three-legged hopper

in the plane. Classically, a predictive controller looking three hop into to the future would need to evaluate $6^{2\times 3} = 46,656$ separate differential equations. We were able to reduce the computation to three difference equations. This allowed real-time optimization and data-lean system identification. We estimated a control model from only 100 hops and then successfully controlled the hopper's motion in real-time. Finally, we introduced a numerical method for solving ESS initial value problems whose pure-Python implementation nearly matches MuJoCo's performance. Using the benign "liveness" condition of ESS, we were able to demonstrate accuracy bounds on efficient splining methods for resolving contact.

This work shows that multi-contact is ripe for control and dynamical study. Despite the apparent difficulty of multi-contact, we have found great improvements to control strategies using the slight additional structure imposed by FoPAS and ESS From system identification to analysis to numerical methods, effective models and controllers can be built with no more effort and data than for smooth systems. It is human nature to stay close to the familiar. There is a tendency in engineering to "brute force" a problem and mathematics prefers to observe from one hundred miles up. The pursuit of this work has allowed us to go up and down that ladder across many different fields. We feel it is better for it and encourage you to do the same.

CHAPTER II

Evidence of C^1-Smooth Multi-Contact Dynamics on a Physical Robot

2.1 Introduction

The theoretical results of ESS and FoPAS are theoretically certain because they were presented with mathematical proof ([Bur+16] and [CRB22], respectively). However, it is unclear how sensitive the underlying mathematics are to the perturbations introduced by physical reality. This chapter shows that a physical robot that could be reasonably modeled as a FoPAS is linearizable in practice as predicted by ESS theory, establishing the practical relevance of event-selected dynamics. It is primarily concerned with experimental methodology and the philosophy of linearization.

2.1.1 Motivation

When robots form several contacts (*collide*) with their environment in an uncertain order, it is generally unclear whether perturbations around this trajectory are governed by a linearization, as they would be for smooth dynamical systems. Dynamics through a single collision can be linearized using a *saltation matrix* [Kon+24]. This result extends to multiple contacts only when the order of in which contacts are made is fixed (called a *mode schedule*). There is no general relationship between the dynamics passing through different mode schedules, and this directly causes the

"combinatorial explosion," where the resulting stability and analysis problems become combinatorially hard [Bur+16; PCT14]. The successful and prevalent use of multi-collision dynamics by virtually all terrestrial organisms, including arthropods with relatively minuscule neural processing capabilities, suggests that modeling multi-collision dynamics may be much simpler than seems at first.

Event-selected systems (ESS) theory [Bur+16] explores physically common multi-collision dynamics where each collision occurs exactly once. Under this structure, ESS flows are continuous, piecewise-differentiable maps. Each differentiable "piece" corresponds to a single mode schedule. ESS provides a generalized way to construct first-order models around multi-collision trajectories even when collisions are simultaneous. Surprisingly, a recent result observes that a common class of ESS has (C^1) continuously-differentiable dynamics [CRB22, Sec. 3.2.2]. These *Force-Position Additive Systems* (FoPAS) subsume many dynamics typically treated as linear complementarity problems (LCPs), suggesting that many contact-rich planning problems could be analyzed and optimized through classical multivariate differentiation. The chief contribution of this work is demonstrating continuous differentiability through multi-collision in an experimental system on a physical robot, highlighting that these mathematical predictions do appear in practice.

2.1.2 Background

Multi-collision planning is hard because perturbations can change the mode schedule, and the change in trajectories caused by a change in mode schedule is generally unknown. This leads to a "combinatorial explosion," where motion planners must optimize over each of the (combinatorially-many) mode sequences individually. This has been somewhat ameliorated using the complementarity formulation of contact [PCT14]. In the complementarity formulation of contact [AP97], contact forces are solved for using a linear complementarity problem (LCP) [Cot79; CPS09]. LCPs are a

type of matrix optimization problem and as such benefit from powerful linear algebra solvers [CPS09]. [PCT14] used this framework to frame multi-collision trajectory optimization as a "Mathematical Program with Complementarity Constraints" which is able to solve over the different mode sequences implicitly, removing the emphasis on mode schedules.

It is not generally known what happens to the trajectory when the mode schedule changes, even though the complementarity formulation has proofs of stability, such as in [PTT16]. At its core, collision dynamics are hybrid dynamical systems, and LCPs are only one way to represent them [Bro+06]. [Bur+16] showed that under certain common technical conditions, a hybrid dynamical with multi-collision dynamics has its flow vary continuously as mode schedules change, and admits a first-order approximation through the Bouligand Derivative which preserves chain rule and product rule. The presence of a first-order approximation reduces the complications of contact to straightforward analysis with Jacobian matrices.

We now recapitulate the technical conditions and discuss their physical intuition.

Definition 2.1.1. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$ and a function $h \in C^r(U, \mathbb{R})$ defined on an open subset $U \subset D$, we say that h is an event function for F on U if there exists a positive constant $f > 0$ such that $Dh(x)F(x) \geq f$ elementwise, for all $x \in U$. A codimension-1 embedded submanifold $\Sigma \subset U$ for which $h|_\Sigma$ is constant is refereed to as a local section for F. [Bur+16, Defn. 1].*

Defn. 3.2.1 establishes the "liveness condition" of ESS. It ensures that all guards are (eventually) crossed, restricting the dynamics from exhibiting sliding modes or branching. The event functions can be chosen such that each $h(x) = 0$ is a guard. The specific choice of constant is arbitrary, but $h|_\Sigma = 0$ aligns with canonical notation in the literature [ST00a].

Definition 2.1.2. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$, we say that F is event-selected C^r at $\rho \in D$ if there exists an open set $U \subset D$ containing ρ and a collection $\{h_j\}_{j=1}^n\} \subset C^r(U, \mathbb{R})$ such that*

1. *(event functions) h_j is an event function for F on U for all $j \in \{1, ..., n\}$*
2. *(C^r extension) for all $b \in \{-1, +1\}^n = B_n$, with*

$$D_b = \{x \in U : b_j(h_j(x) - h_j(\rho)) \geq 0\},$$

$F|_{IntD_b}$ admits a C^r extension $F_b : U \to TU$

[Bur+16, Defn. 2].

Defn. 3.2.2 establishes that event-selectedness is a local property. Requiring that each guard be crossed exactly once, irrevocably prevents periodic and rhythmic behaviors, such as hopping. By constructing event-selectedness as a local property, periodic systems can be treated as several event-selected systems joined together, preserving both the mathematical convenience of event-selectedness and the generality of mechanical collisions.

Defns. 3.2.1 and 3.2.2 form the "check-list" for determining if a dynamical system is event-selected. [Bur+16] showed that such systems possess a first-order approximation, although not in the sense of a traditional derivative. Instead, ESS possess a Bouligand Derivative. The Bouligand Derivative, when it exists, is numerically identical to the directional derivative [Sch12]. Intuitively, perturbing a trajectory which crosses 2+ guards simultaneously will cause the perturbed trajectory to have a specific mode schedule, and the correct first-order approximation will be the Bouligand Derivative corresponding to said specific mode schedule. Furthermore, these first-order approximations can be constructed from saltation matrices, a well-established tool for dynamical analysis.

Surprisingly, [CRB22, Sec. 3.2] shows that some ESS are C^1 smooth. In general ESS, the loss of smoothness when changing mode schedule comes from the fact that the saltation matrices do not generally commute. However, when an ESS

1. is "second-order mechanical" in the sense that the state space can be evenly partitioned into coordinates and their time derivatives ("positions" and "velocities")
2. has only guards which exclusively depend on position coordinates
3. crossing guards (the "collisions") introduce only additive forces, e.g. "soft" spring-like collisions,
4. and the trajectory is continuous (no jumps in velocity)

the different saltation matrices are commutative and the dynamics are linearizable in the traditional sense. This implies that many soft and firm contact problems typically treated with LCP are classically smooth, and are amenable to analysis for more well-behaved nonlinear dynamics.

2.1.3 Hypothesis

This work is concerned with the empirical C^1-smoothness of physical FoPAS flows. If the C^1 property holds, then the linearizations of nearby trajectories should be similar, even when they belong to different mode schedules. We expect the linearizations to be so similar that the practitioner can simply use a single linearization for planning and control – an extreme simplification. In this case it would be difficult to distinguish between two linearizations – they would be, in practice, identical. This is the core of our hypothesis: if a FoPAS flow is (practically) C^1, then there will be no significant difference between nearby linearizations, even when they belong to different mode schedules.

This follows from C^1 by definition: the slope (derivative) and offset of the linearizations are continuous curves. Therefore for small ϵ, trajectories governed by a C^1 flow and originating in the same ϵ-ball have δ-close linearizations by $\epsilon - \delta$ continuity of the derivative. An empirical test for C^1-smoothness then determines if the δ (distance) between linearizations is large enough to violate the C^1 property.

Multi-contact flows obeying liveness are already C^1 for individual mode schedules (c.f. Sec. 2.1.2), so if the C^1 property is violated, it is violated at a boundary between mode schedules. Conveniently, the trajectory which crosses all guards simultaneously is a boundary between all the mode schedules. Sampling around this simultaneous trajectory samples an ϵ-ball containing trajectories for every mode schedule. Linearizations corresponding to the same mode schedule in this ϵ-ball are highly similar, so the distance between them is the baseline distance between (practically) identical linearizations. We can determine if other linearizations are also highly similar by comparing the distance between them to this "in-class" distance.

We chose to measure the distance between linearizations using both the operator norm and the subspace angle. For constant ϵ, the operator norm is the prediction error. If the flow is indeed C^1, we would expect the linearization built by combining data from all the mode schedules to have a prediction error similar to the linearizations specialized for each mode schedule. Depending on the underlying higher-order terms, accurate linearizations may be able to be built from data excluding certain mode schedules. This aligns with our interest in practical application – the typical use of linearization is to more readily predict dynamical behavior, and being able to ignore the added complexity of different mode schedules is a significant simplification.

To measure the similarity of the affine models directly, we measured the angle between them – the subspace angle. Such angle measurements are more robust to noise in the independent variable(s) and here provide insight into the similarity of the lower-dimensional dominant dynamics. We expect that, for a C^1 flow, the distance (in terms

of subspace angle) between linearizations specialized for the same mode schedule is similar to the distance between linearizations specialized for different mode schedules. If the both the distances and the prediction errors of experimentally determined affine models are similar, we will say that a FoPAS flow is empirically C^1.

2.2 Methods

2.2.1 Experimental Design & Data Collection

We built a 3-legged hopping robot with springy legs, shown in Fig. 2.1. For stability, the legs were equal-length and arranged symmetrically about the body z-axis. We chose to study the triple collision at landing (*touchdown*). To ensure the dynamical state of the robot persists through touchdown, we performed a drop test releasing the robot from a height comparable to its hopping height. The robot bounced back to more than 40% of the initial height. We conclude that the collision losses at landing do not annihilate the dynamical state of the robot[1]. We placed 3 motion capture markers on the body to track its rigid body pose and a marker at the distal end of each leg (foot) to track contact state. We recorded the hopper hopping[2], compensated for the slight curvature of the floor, and partitioned the data into touchdowns with pre- and post- touchdown Poincaré sections, described in Sec. 2.2.2.

[1]It should be noted that many hopping robots do no meet this criterion – they convert the entirety of their potential energy to heat and electrical energy, then "jump" using stored electrical power. Such systems are "deadbeat" in a control-theoretic sense, but provide no insight into the mechanical dynamics.

[2]We collected data over 4 days at a frame rate of 100Hz, totaling 182611 frames and 2860 hops. The measured position noise was 0.6mm on average, as determined by the error in the pairwise rigid body distances between body markers on the hopper. Motion capture data was collected using Qualysis QTM version 217 (build 4000) on Oqus 310+ cameras.

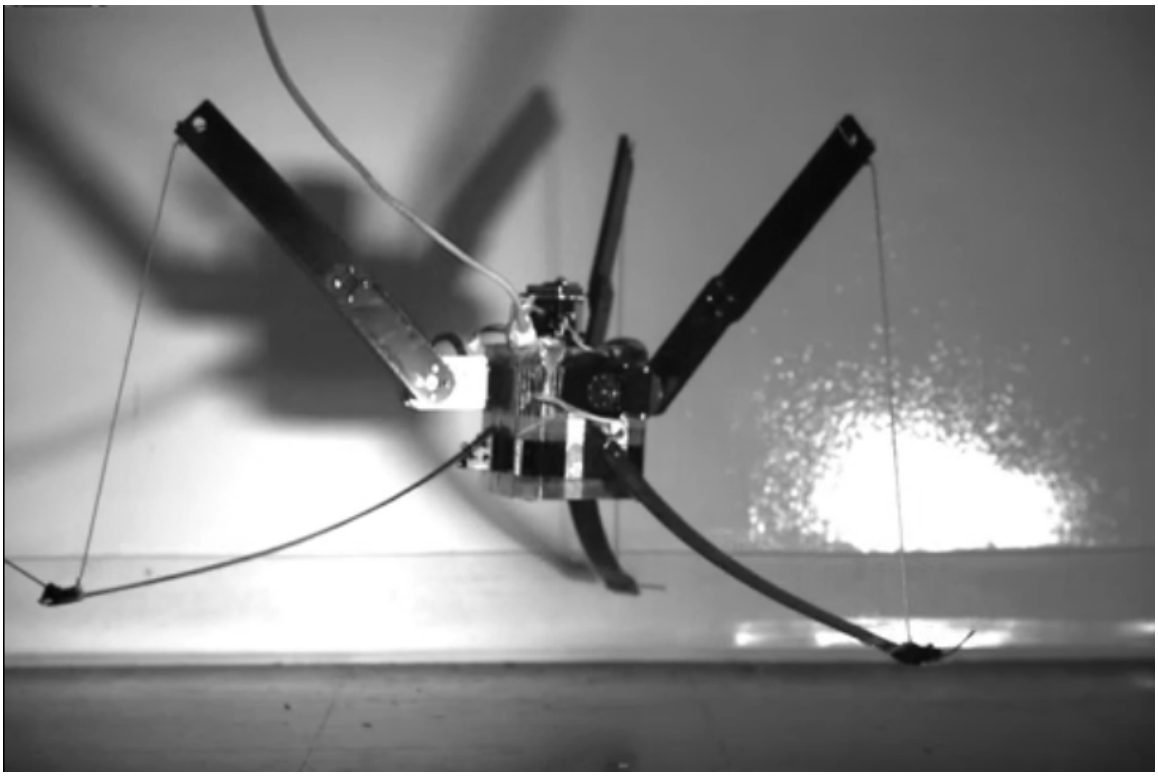

Figure 2.1: The three-legged hopping robot we developed for this experimental work. The robot "preloads" its elastic legs prior to ground contact using a cable assembly so that the legs immediately exert a force on the ground at the moment of contact. As a result, the contact force model is affine and cannot be readily solved using the machinery for linear complementarity problems [Bro03].

2.2.2 Selecting Poincaré Sections

To regress the 6 possible touchdown impact maps, we needed to define their domain and range. We elected to use Poincaré sections for this purpose. In dynamical analysis, a Poincaré section is a manifold in the state space which the trajectories of interest are guaranteed to cross. Poincaré sections allow a (continuous) differential equation to be studied from discrete observations, specifically through where trajectories cross a Poincaré section. Since the hopper's motion was primarily vertical, we sought level sets of the hopper world-frame z-position for our Poincaré sections.

This produced a selectivity trade-off. Placing the Poincaré sections close together would sample close to the simultaneous triple-contact trajectory, but would exclude relevant data. We resolved this trade off by examining the joint distributions of body

height and foot height, illustrated in Fig. 2.2. We sought two values of body height close together, one corresponding to all feet were off the ground (pre-touchdown) and the other corresponding to all feet were off the ground (post-touchdown). We examined distributions of minimum and maximum foot height. Both distributions were bi-modal with the lower (and larger) mode corresponding to ground contact. We took the upper knee of the lower modes as thresholding when all 3 feet were on (or off) the ground. To relate these thresholds to body height, we looked at the conditional distribution of body height, conditioned on the foot height thresholds. For our post-touchdown Poincaré section, we selected the lowest knee of body height conditioned on the foot height being off the ground (above the maximum foot height threshold). This ensured that the any body height below the post-touchdown Poincaré section had all feet on the ground. We did the reverse for our pre-touchdown Poincaré section – we selected the body height above which even the lowest foot was on the ground (below the minimum foot height threshold). This ensured that even the lowest foot was off the ground when the body height was above our pre-touchdown Poincaré section.

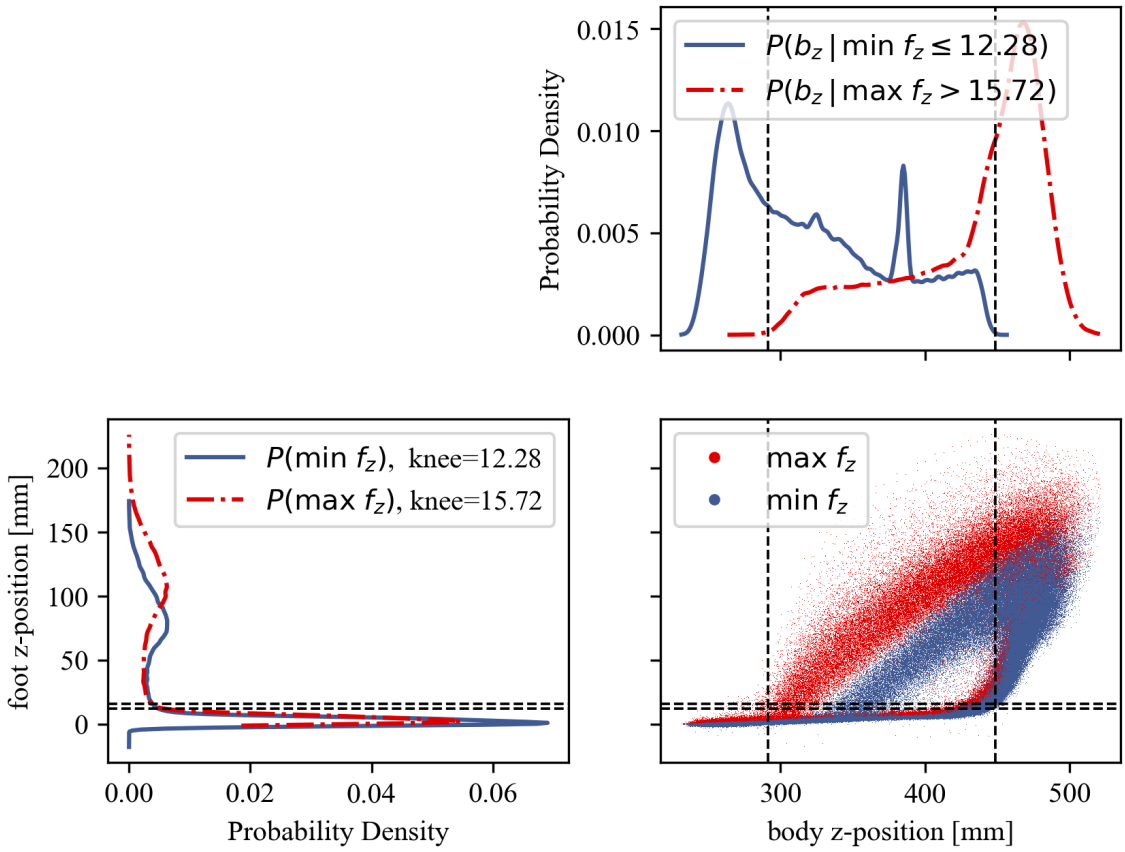

Figure 2.2: Data-driven selection of Poincaré sections using marginal probability distributions. Adjacent plots share axes. Vertical dashed black lines correspond to the Poincaré sections used to segment the motion capture data. Horizontal dashed black lines correspond to the minimum (maximum) foot height thresholds of the body height conditional probabilities. Clockwise from the bottom right: minimum (maximum) foot height versus body height scatter plot, probability density of minimum (maximum) foot height, conditional probability densities of body height conditioned on minimum (maximum) foot height.

2.2.3 Determining Touchdown Contact Order (Mode Schedule)

Although foot height physically determined when a foot was in contact with the ground, we found foot z-velocity to be a more robust indicator of ground contact. The feet could not pass through the ground, so foot z-velocity experienced a large local minimum at the moment of ground contact, regardless of the measured floor height. The order in which the feet experienced these local minima determined the

mode schedule, illustrated in Fig. 2.3. If a foot contacted the ground before or after a touchdown, that touchdown was excluded from the dataset.

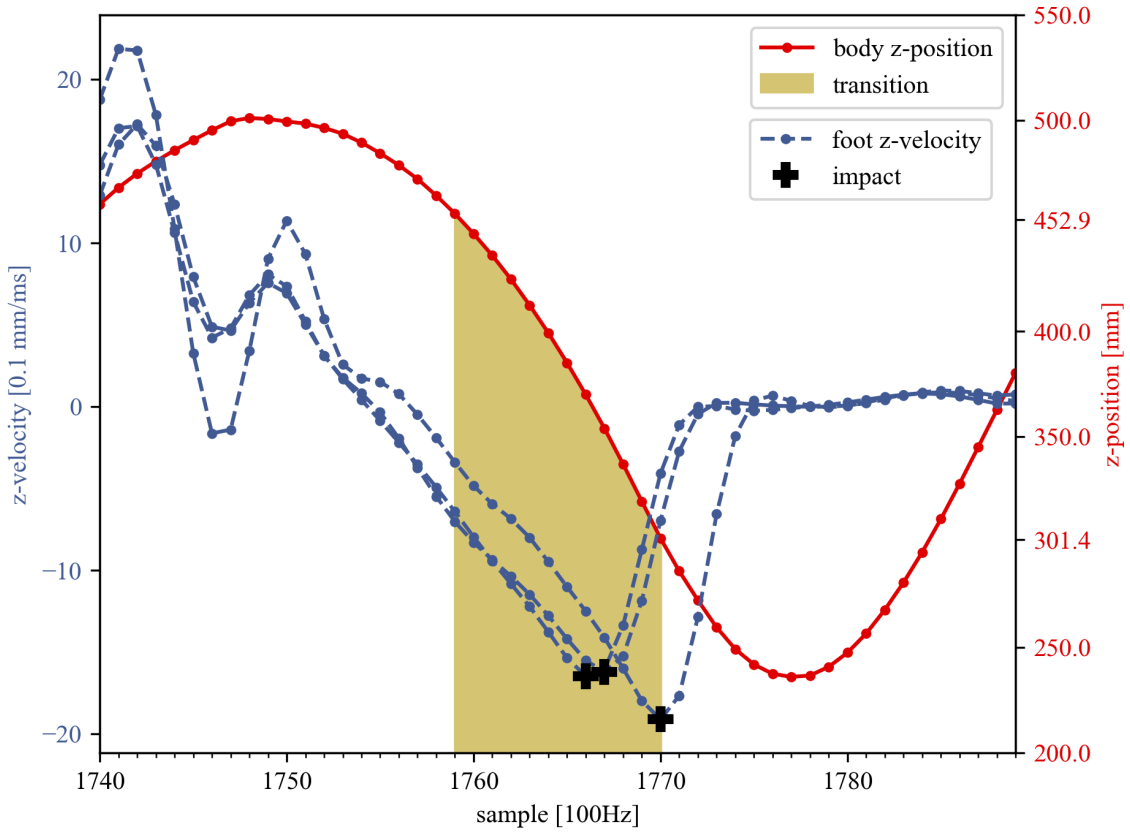

Figure 2.3: Determining the foot contact ordering (mode schedule) of a touchdown. Foot contact occurs when the foot z-velocity (blue dashed lines) attains its minimum during a touchdown transition ((solid yellow fill). The order of foot z-velocity minima (black plus signs) determines the touchdown's mode schedule. All feet must contact the ground during the touchdown transition, or else the touchdown is excluded from the dataset.

2.2.4 Impact Map Estimation & Comparison

Since we could reasonably expect the floor-compensated dynamics to be independent of (world-frame) xy position and z rotation, we regularized each touchdown to an SE(2)-invariant "home" frame. This frame had its origin centered at the average body position during touchdown, and its x-axis passed through the body marker a specific body marker. Each frame used the same body marker. Each touchdown had a point on the pre- and a point on the post-touchdown Poincaré section. We took these pairs

of points to be input-output pairs describing the flow induced by the multi-contact touchdown dynamics. After regularizing the data, we selected units of position and time so that prediction error in position and velocity coordinates would be equally represented. We computed our results on the normalized regularized input-output pairs.

We sought to understand the different linear approximations to the flow during the touchdown transition. Linear regression is the standard tool to compute a linear relationship from data. We regressed affine models using ordinary least squares. To compare the dominant linearized dynamics, we needed (1) a way to determine "dominance" and (2) a means to compare them. Principal component analysis (PCA) describes the directions of variation in a dataset, ordered from highest to lowest variability. For 18-dimensional data, the principal components are the axes of the "best-fit" 18-ellipse. From this perspective, low-variance principal components describe directions along which the data is "constrained," or varies little.

Although our dataset had 36 dimensions (9 input position coordinates, 9 input velocity coordinates, 9 output position coordinates, and 9 output velocity coordinates), we found the two least significant principal components to correspond to the Poincaré sections. This simplified our expected flow to have a state space in the principal components of dimension 17. Since the flow is likely a full-rank map from 17 dimensions to 17 dimensions, it is approximately an 17-dimensional hyperplane in 34-dimensional space. Therefore, we expect to find 17 constraints, i.e. 17 "small" principal components. We took these principal components to describe the dominant dynamics. The 17 small principal components are a collection of orthonormal vectors represented as a matrix. However, comparing matrices does not necessarily compare the underlying subspaces. Each matrix is a particular basis representation for its subspace. Spurious differences in noise between mode schedules could cause basis vectors to change position or "rotate" slightly in the PCA matrix. These minor

changes could cause a numerical difference in the matrices which has no connection to the underlying dynamics. To compare the underlying dynamics, we need a means to convert from the PCA basis representations to distance between subspaces.

This problem has received a great deal of attention, and [EAS98, Sec 4.6] collates several notions of subspace distance which can be computed from orthonormal tall matrices. Each distance has a different geometric interpretation, as the set of subspaces forms a manifold embedded in Euclidean space, known as the Grassmann Manifold.

Regardless of geometry, we seek to make statistical claims. Our constraint subspaces are samples from some underlying probability distributions influenced by our experimental noise. We seek to determine how similar these underlying distributions are between the different mode schedules. If the distributions are statistically similar, we can conclude that the different mode schedules' linearizations are likewise similar. Each distance is a transformation from matrix pairs to the real numbers. Depending on the nature of the transformation, a distance may make two distributions seem more or less similar. A similarity result would be made stronger by persisting under a distance which makes similar matrices appear distant, i.e. is sensitive to noise. We evaluated each distance's sensitivity to noise and found the "arc length" distance to be most sensitive to noise. The arc length is the L_2 norm of the subspace angles between subspaces, and corresponds to the geodesic distance on the Grassmann manifold, effectively characterizing the size of the rotations required to bring two subspaces into alignment [EAS98, Sec 4.6].

2.3 Results

We trained affine OLS and mean-predictor models predicting the three body markers' 3D positions and 3D velocities after touchdown. We evaluated their performance by

the prediction mean-squared error (MSE) using 1000 bootstraps with a 120/41 sample train/test split. The worst case prediction error was on the order of 4mm or 0.14 m/s depending on the quantity. The hopper traveled at least 140mm per touchdown, suggesting that all models were accurate predictors of the hopper's touchdown motion.

The performance of affine models trained on data selected without attention to mode schedule ("unspecialized") versus affine models specialized for each mode schedule strongly suggests that the hopper's multi-contact dynamics are C^1 (c.f. Fig. 2.4). Both specialized and unspecialized models comparably outperformed their respective mean predictors, suggesting that the dynamical information gained from modeling the "slope" was the same regardless of mode schedule. Further, the performance of unspecialized affine models improved closer to the simultaneous contact trajectory and worsened further away. This mimics the anticipated effect of higher order terms, which theoretically differ for each mode schedule. We trained "near" unspecialized models selecting impacts whose standard deviation if the respective three foot contact times was below the median of the entire dataset, and "far" unspecialized models from data whose contact time standard deviation was above the median.

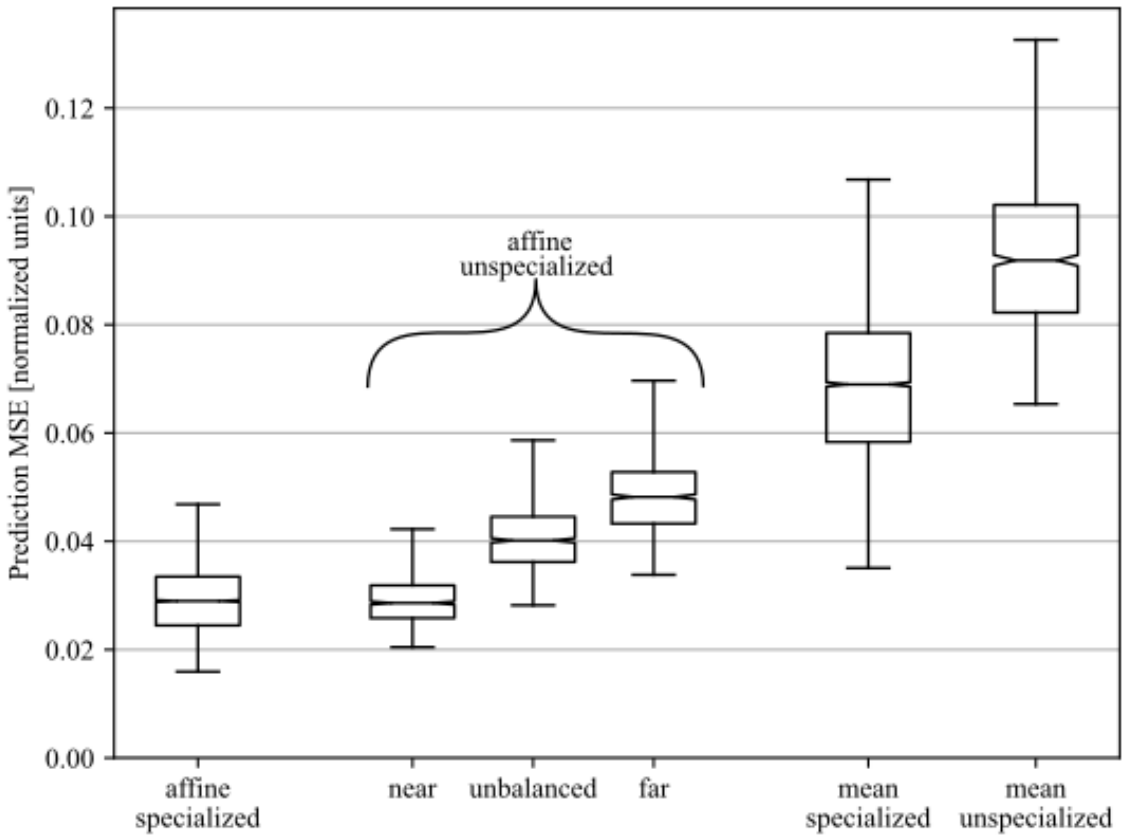

Figure 2.4: Boxplots of the bootstrapped combined prediction mean-squared error (MSE) of impact map models describing the 3D positions and 3D velocities of the three body markers. From left to right, affine model specialized for each mode schedule, unspecialized affine model near the triple-contact trajectory, unspecialized affine model ignoring distance from the triple-contact trajectory, unspecialized affine model far from the triple-contact trajectory, mean predictor specialized for each class, unspecialized mean predictor. Length units were normalized to 29.13mm and 27 milliseconds.

To evaluate the dynamical similarity of the dynamics, we bootstrapped 1000 PCAs for each mode schedule. Due to asymmetry in the dataset, we trained each PCA on 30 samples rather than 120 as for the affine models. The arc-length distance between the most significant half of the principal components for the same mode schedule and for differing mode schedules were highly similar, and were comparable to the distance between noised identical matrices (c.f. Fig. 2.5). The noised identical matrices acted as a control to establish what the distance between identical matrices estimated from data should be. As a further control, we evaluated the spread of random matrices of appropriate type and found it to be well above the distance between all mode schedule

PCAs.

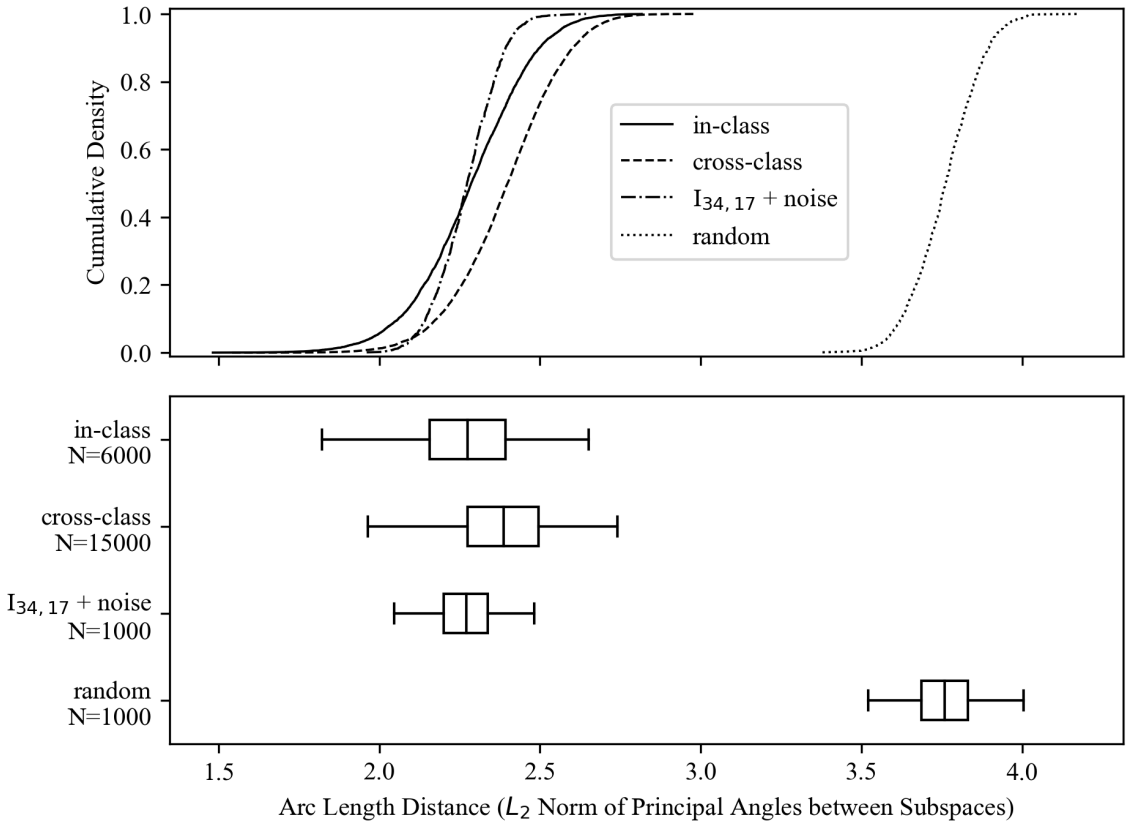

Figure 2.5: Cumulative density (top) and box (bottom) plots showing the spread of arc-length distances in our dataset. The significant overlap between the cross-class and in-class distances suggests that the dominant dynamics are identical between mode schedules, and the hopper dynamics are C^1smooth. Distances between identical matrices subject to additive noise ("$I_{34,17}$ + noise") emulates the spread of distances expected from identical matrices estimated from experimental data (e.g. motion capture of a hopping robot) with a signal-to-noise ratio of 10. Distances between random matrix pairs ("random") emulates the spread of unrelated dynamics. The dataset distances' overlap with identical matrices and its distance from random matrices further suggests that the hopper dynamics are C^1smooth.

We consider all numerical results to strongly suggest that the dynamics near the triple-contact touchdown can be both effectively predicted and described by a single linearization trained from data ignoring mode schedule.

2.4 Conclusion

Conventional wisdom has indicated that multi-contact dynamics are prohibitively expensive to control, and are best treated using optimization. We have shown the opposite on a physical robot: many multi-contact dynamics appear to be conventionally linearizable and should be, for practical purposes C^1-smooth. This result has two key implications. First, the theoretical analysis of physically-realistic multi-contact dynamics is more straightforward than previously believed. Classical linear and nonlinear techniques should be completely applicable. Second, less-complex models can be built from less data for the same accuracy. Even in the case of "hard" contact dynamics, there is often an impedance between the contact point and the center-of-mass, so such center-of-mass dynamics are affected by this result.

We believe this has broad consequences for robot control and optimization. Not only should introductory nonlinear control tools be useful on multi-contact robots, but gradient-based optimization methods could work nearly "out of the box." These tools have yet to be deployed on multi-contact robots and we plan to continue to probe this boundary between theory and practice in future work.

CHAPTER III

Mode-Schedule-Agnostic Steering of a Three-Legged Hopping Robot

Having shown that the hopper is classically linearizable, we turn to the question of controlling its motion. This chapter demonstrates that classical (smooth) system identification and linearized models are sufficient to steer the three-legged hopping robot around an arena.

3.1 Introduction

The relationship between a robot's motion and its environment is generally[1] governed by contact forces. Effective task planners should then readily handle the effects of contact. This becomes complicated when a robot can form multiple contacts in an indeterminate order. The change in dynamics caused by a change in contact order (called a *mode schedule*) is generally unknown. A task planner must consider each of the combinatorially-many mode schedules, a phenomenon termed the *combinatorial explosion*. Planning is already a time-consuming aspect of robot control. Having to consider combinatorially-many planning problems simultaneously prohibits direct analysis of these *multi-contact dynamics*.

[1]Many legged robots do have a ballistic segment to their motion, and drones and satellites are sometimes regarded as robots.

Historically, roboticists have employed three strategies to mitigate the combinatorial explosion. When appropriate, soft contact models cause the multi-contact dynamics to become continuously once-differentiable (C^1-smooth), immediately enabling the use of classical smooth tools [RR23]. In manipulation, contact forces are usually computed from Linear Complementarity Problems (LCPs) [AP97; TT10]. LCPs, being optimization problems themselves, can be used to implicitly optimize over multiple mode schedules in a "Mathematical Program with Complementarity Constraints" [PCT14]. However, in the absence of LCP models or soft contact, a planner is stuck with the combinatorial explosion. It is for this reason many commercial legged robots have rubber feet and appear to use gait libraries: online planning for multi-contact is simply expensive.

Recent work has shown theoretically [CRB22, Sec. 3.2] and experimentally (Chp. 2) that many multi-contact dynamics previously thought to be intractable are actually C^1. As in case of soft contact, this admits the use of classical smooth tools for planning, control, and optimization. We demonstrate this simplification on hardware by effectively steering a three-legged robot in the plane using direct single-shooting model predictive control in real-time. The remainder of this work is organized as follows: the next section summarizes the theoretical foundations motivating this work, Sec. 3.3 describes the robot, and Sec. 3.4 describes the control process from concept to hardware. Experimental results and setup are laid out in Sec. 3.5.

3.2 Background

[Bur+16] showed that under mild technical conditions, systems with multi-contact dynamics have flows that vary continuously as mode schedules change, and admit a first-order approximation through the Bouligand Derivative which preserves chain rule and product rule. These systems are termed *Event-Selected Systems* (ESS). Even

stronger, [CRB22, Sec. 3.2], showed that many common ESS are fully C^1-smooth, possessing a classical derivative instead of a Bouligand derivative. This work leverages the structure of these smooth systems. We begin by describing the conditions for a dynamical system to be ESS, and then provide a "checklist" to determine when an ESS is smooth.

Definition 3.2.1. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$ and a function $h \in C^r(U, \mathbb{R})$ defined on an open subset $U \subset D$, we say that h is an event function for F on U if there exists a positive constant $f > 0$ such that $Dh(x)F(x) \geq f$ element-wise, for all $x \in U$. [Bur+16, Defn. 1].*

Defn. 3.2.1 establishes the "liveness condition" of ESS. It ensures that all guards are (eventually) crossed, restricting the dynamics from exhibiting sliding modes or branching. The event functions can be chosen such that each $h(x) = 0$ is a guard. The specific choice of constant is arbitrary, but $h|_\Sigma = 0$ aligns with canonical notation in the literature [ST00a].

Definition 3.2.2. *Given a vector field $F : D \to TD$ over an open domain $D \subset \mathbb{R}^d$, we say that F is event-selected C^r at $\rho \in D$ if there exists an open set $U \subset D$ containing ρ and a collection $\{h_j\}_{j=1}^n\} \subset C^r(U, \mathbb{R})$ such that*

1. *(event functions) h_j is an event function for F on U for all $j \in \{1, ..., n\}$*
2. *(C^r extension) for all $b \in \{-1, +1\}^n = B_n$, with*

$$D_b = \{x \in U : b_j(h_j(x) - h_j(\rho)) \geq 0\},$$

$F|_{IntD_b}$ admits a C^r extension $F_b : U \to TU$

[Bur+16, Defn. 2]. Note that this requires continuous solution trajectories, i.e. ESS trajectories cannot experience "jumps."

Defn. 3.2.2 establishes that event-selectedness is a local property. Requiring that each guard be crossed exactly once, irrevocably prevents periodic and rhythmic behaviors, such as hopping. By constructing event-selectedness as a local property, periodic systems can be treated as several event-selected systems joined together, preserving both the mathematical convenience of event-selectedness and the generality of mechanical collisions.

Defns. 3.2.1 and 3.2.2 form the "check-list" for determining if a dynamical system is event-selected. [Bur+16] showed that such systems possess a first-order approximation, although not in the sense of a traditional derivative. Instead, ESS possess a Bouligand Derivative. The Bouligand Derivative, when it exists, is numerically identical to the directional derivative [Sch12]. Intuitively, perturbing a trajectory which crosses 2+ guards simultaneously will cause the perturbed trajectory to have a specific mode schedule, and the correct first-order approximation will be the Bouligand Derivative corresponding to said specific mode schedule. Furthermore, these first-order approximations can be constructed from saltation matrices [AG58; Kon+24], a well-established tool for dynamical analysis.

In general ESS, the loss of smoothness when changing mode schedule comes from the fact that the saltation matrices do not generally commute. However, when an ESS

1. is "second-order mechanical" in the sense that the state space can be evenly partitioned into coordinates and their time derivatives ("positions" and "velocities")

2. has guards which only depend on positions

3. crossing guards (the "collisions") introduce only additive forces, e.g. "soft" spring-like collisions,

4. the "forces" introduced by the discontinuities are additive

the different saltation matrices are commutative and the dynamics are linearizable in the traditional sense. This implies that many soft and firm contact problems typically treated with LCP are classically smooth, and are amenable to conventional nonlinear dynamical analysis.

3.3 Robot

We conducted the experiments using the three-legged robot diagrammed in Fig. 3.1 and shown in Fig. 3.2. It used two sets of actuators to move: three independently-driven gated parallel elastic actuators for legs, and a counter-weight boom actuator to move its center of mass (CoM). Motion capture cameras provided feedback, and a process diagram of the motion plan is shown in Fig. 3.4. Each leg actuator was composed of a Dynamixel MX-106 motor, a lever arm, and a 1075 spring steel beam "leg." A thin metal cable connected the distal ends of the lever arm and leg. We divided the hopper's motion into a flight phase, when no legs contacted the ground, and a stance phase, when all three legs contacted the ground. During flight, the motors raised the lever arms, "preloading" the spring steel to store energy. When the hopper detected[2] stance, it lowered the lever arms to release the stored energy into the next hop.

Modulating the contact forces by varying the amount of preload failed to induce meaningful hop-to-hop translation or rotation. However, placing counterweights on the hopper's chassis introduced a consistent hop-to-hop drift. This observation led us to design the counterweight-boom actuator, shown in Fig. 3.1c. It consisted of a motor mounted at a slant angle to the chassis, a boom mounted to the motor at a slant angle, and a counterweight attached to the boom. This geometry avoided mechanical interference with the leg actuators and could return the CoM to the (horizontal)

[2]Stance was detected when the current consumed by each of the three motors all dropped below a fixed threshold.

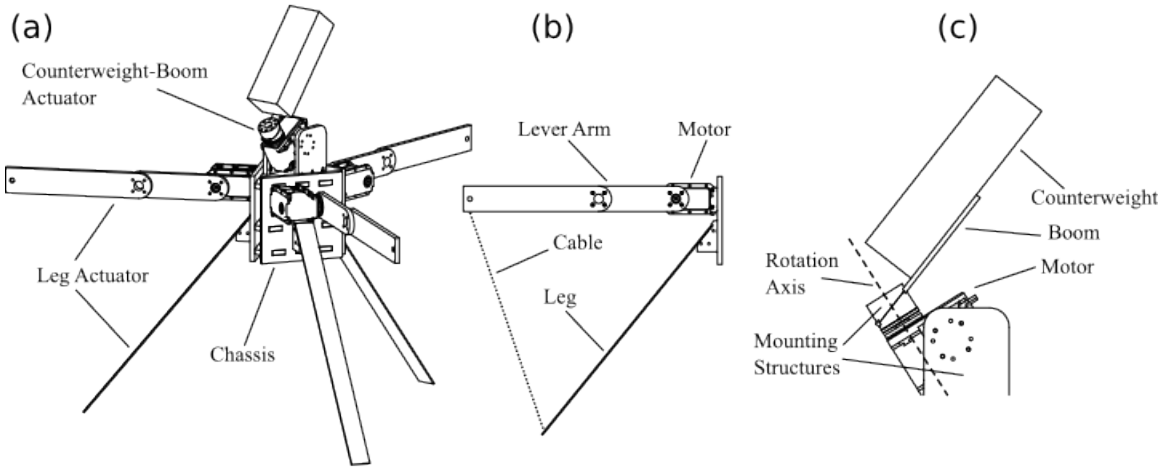

Figure 3.1: Diagram of the three-legged hopping robot used in the experiments. Left-to-right: (a) overall assembly, (b) detail of a leg actuator, (c) detail of the counterweight-boom actuator. All motors are MX-106 brushed DC motors from Dynamixel. (a) A slot-together frame made of quarter-inch lasercut ABS forms the chassis. It is held together by Kevlar-embedded tape wrapped horizontally around the frame. Actuators are secured to the chassis via M2.5, M3 screws and locknuts. (b) The leg is a 1075 spring steel beam connected to the quarter-inch ABS lever arm via a thin metal cable. The motor raises and lowers the lever arm to store and release energy in the leg. (c) The boom actuator moves a counterweight around an axis not aligned with the body frame. We used a 4S LiPo battery for the counterweight.

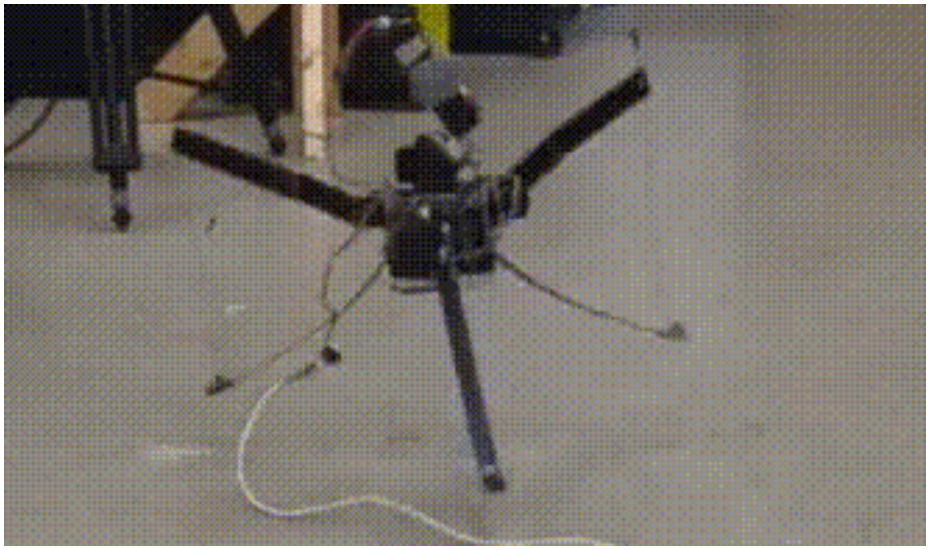

Figure 3.2: Photograph of the hopping robot. Power is supplied by the battery counterweight. Motor communication travels through the attached Ethernet cable (white).

center of the robot during stance as a "neutral" position.

The hopper was able to rotate by moving the boom during flight. However, moving the boom too quickly caused the hopper to flip over and land in an unrecoverable configuration. This limited the boom's motion to $\pm 90^\circ$ per hop. Since the boom actuator could not achieve an arbitrary boom angle during a hop, we modeled it as having an internal state (the boom angle) and receiving a velocity control input (the change in angle). This allowed us to build a model of the hop-to-hop motion as a function of the actuator state and input, detailed in the next section.

3.4 Control Scheme

3.4.1 Dynamical Model

We seek to control the hopper as a mobile robot, i.e. its SE(2) configuration during stance. We will borrow from the bipeds literature and study the "frame-change per [gait] cycle": we fix an origin to the most recent stance frame and express the next

stance configuration relative to this origin. Previous analysis (Chp. 2) showed[3] that the frame-change dynamics were zero-order; they depended principally on the control input and not any dynamical history. To highlight their simplicity, we modeled the dynamics as three independent C^1 functions (2 for translation and 1 for orientation) of the boom actuator state (practical details in Sec. 3.4.3). These dynamics did not depend on mode schedule and can be expressed in the world frame as a discrete-state update equation:

$$
\begin{aligned}
x_{k+1} &= x_k + \Delta x^b(\phi_k, \Delta\phi_k) \cdot \cos\theta_k - \Delta y^b(\phi_k, \Delta\phi_k) \cdot \sin\theta_k \\
y_{k+1} &= y_k + \Delta x^b(\phi_k, \Delta\phi_k) \cdot \sin\theta_k + \Delta y^b(\phi_k, \Delta\phi_k) \cdot \cos\theta_k \\
\theta_{k+1} &= \theta_k + \Delta\theta^b(\phi_k, \Delta\phi_k) \\
\phi_{k+1} &= \phi_k + \Delta\phi_k
\end{aligned}
$$

with stance frame $(x, y, \theta) \in SE(2)$, boom angle $\phi \in \mathcal{S}^1$, control input (boom angle change) $\Delta\phi \in \mathcal{S}^1$, and (smooth) frame-change dynamics $\Delta\,\cdot^b(\cdot,\cdot)$.

3.4.2 System Identification

We used Gaussian Process regression [RW06] to model the control dynamics based on physical data. Described above, the control model depended on the current boom angle (ϕ, the boom actuator state) and the change in boom angle (swing) during flight ($\Delta\phi$, the control input). We recorded the hopper hopping while executing control inputs using a Qualysis motion capture system at 100 Hz. The hopper selected the control inputs based on the semi-random grid traversal algorithm described in Alg. 1. From the recorded data, we extracted instantaneous SE(2) pose (Alg. 2), instantaneous boom angle (Alg. 3), and identified motion capture frames corresponding

[3]The hopper's dynamics were invariant to its location in the motion capture arena, and contact friction largely destroyed the effects of momentum.

to consecutive stance phases (Alg. 4). We then calculated the ϕ[4], $\Delta\phi$[5], and frame change[6] for each pair of consecutive stance phases.

We randomly selected 100 stance pairs to train 3 Gaussian process regressors in scikit-learn [Ped+11, Sec. 1.7], one for each SE(2) coordinate. To enforce a periodic dependence on ϕ, we duplicated the data in the training set with a change in ϕ of -2π, 0, and 2π radians. Building models which considered mode schedules would have required at least 3600 stance pairs[7]. Each regression used a Matérn kernel with $\eta = 1.5$ and a white noise kernel with noise level 0.1,

$$\frac{1}{\Gamma(1.5)\sqrt{2}}\left(\sqrt{3}\,d(x_i,x_j)\right)^{1.5} K_{1.5}\left(\sqrt{3}\,d(x_i,x_j)\right) + \begin{cases} 0.1 & , i = j \\ 0 & , \text{else.} \end{cases}$$

This particular Matérn kernel describes C^1 functions, making it suitable for the expected dynamics. Interestingly, the Matérn kernel for $\mathcal{C}^2$ functions broadly failed to model the data, suggesting that the control dynamics had some directly measurable C^1 behavior. Fig. 3.3 shows the general structure of the regressions.

3.4.3 Transfer to Hardware

The hop-to-hop dynamics were nonholonomic (the hopper could not hop "backwards"), which complicated the use of trivial linear controllers. To highlight the benefits of our approach, we turned to online predictive optimal controllers, in particular direct-shooting model predictive control (MPC). MPC uses simulation to find a sequence of control inputs that approximates the optimal control input an amount

[4]A stance's boom angle (ϕ) is the median instantaneous boom angle in the 11 (motion capture) frames immediately following.

[5]The boom swing between two consecutive stance phases ($\Delta\phi$) is the difference of their boom angles.

[6]Given two consecutive stance phases with SE(2) configurations, g_1 and g_2 respectively, their frame-change per cycle is $g_1^{-1}g_2$.

[7]With three legs, each flight or stance transition has $3! = 6$ mode schedules, and so the hop-to-hop dynamics had 36 mode schedules (stance-flight, flight-stance).

Algorithm 1: Generate a near-minimal sequence of boom angles that covers the entire (discretized) control space.

Result: $Nx1$, boom angle command for each hop
$N_p \leftarrow 16$;
$N_v \leftarrow 3$;
$N_c \leftarrow 3$;
; `/* Build transition matrix */`
$M \leftarrow \text{zeros}((N_p, N_v))$;
for k *in* N_p **do**
    $s \leftarrow k - N_v$;
    $e \leftarrow k + N_v + 1$;
    if $s < 0$ **then**
        $M[k, s:] \leftarrow N_c$;
        $M[k, :e] \leftarrow N_c$;
    else if $e \geq N_p$ **then**
        $M[k, s:] \leftarrow N_c$;
        $M[k, : e - N_p] \leftarrow N_c$;
    else
        $M[k, s:e] \leftarrow N_c$;
    end
end
; `/* Generate traversal sequence */`
$k \leftarrow 0$;
$q \leftarrow$ empty array;
$nM \leftarrow \text{sum}(M, \text{axis} = 1)$;
$N = \text{arange}(Np)$;
$nnM \leftarrow \text{sum}(M)$;
while $nnM > 0$ **do**
    $q.append(k)$;
    if $nM[k] > 0$ **then**
        $n \leftarrow N[M[k] > 0]$;
    else
        $n \leftarrow N[M_0[k] > 0]$;
    end
    $w \leftarrow nM[n] + 1/N_v$;
    $cw = \text{cumsum}(w)$;
    if $nM[k] > 0$ **then**
        $M[k, nk] \leftarrow M[k, nk] - 1$;
        $nM[k] \leftarrow nM[k] - 1$;
        $nnM \leftarrow nnM - 1$;
    end
    $k \leftarrow nk$;
end
return q

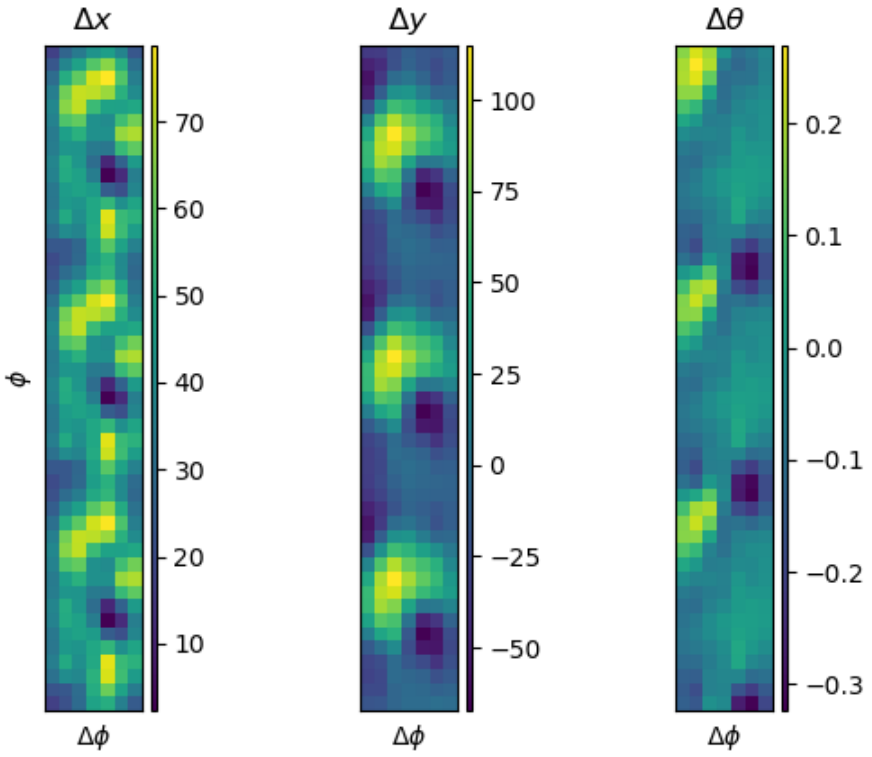

Figure 3.3: Heatmaps of the identified frame-change per cycle based on actuator state (ϕ) and control input $\Delta\phi$. The hopper is able to turn, drift right or left, but cannot hop "backwards." This indicates the hopper's motion is nonholonomic and not immediately straightforward to classical linear control. Note that each heatmap has a different scale. ϕ ranges from -2π to 4π showing that the regressions are periodic or nearly so. $\Delta\phi$ ranges from $-\pi/4$ to $\pi/4$.

Algorithm 2: Calculate the hopper's instantaneous SE(2) configuration from motion capture data.

Data: d, a $3x3xN$ array of N frames of motion capture data. The zeroth axis selects the x, y, or z coordinate, while the first axis selects the marker.

Result: A $3xN$ array of the SE(2) configuration in each frame

; `/* Take instantaneous average marker position as the frame origin */`

$x \leftarrow d[0] - \text{mean}(d[0], \text{axis} = 0)$;

$y \leftarrow d[1] - \text{mean}(d[1], \text{axis} = 0)$;

; `/* Use complex phasors to measure heading more robustly */`

$p \leftarrow x + 1j * y$;

$\alpha \leftarrow \text{mean}(p[0]/p[1])$;

$\beta \leftarrow \text{mean}(p[0]/p[2])$;

$\theta \leftarrow \text{unwrap}(\text{angle}(p[0] + \alpha * p[1] + \beta * p[2]))$;

return $[x, y, \theta]$

of time into the future, called the "horizon." This scheme is inherently feed-forward, and so MPC provides feedback by re-solving for new control inputs from updated state information at (usually) regular intervals. Our implementation used a three-hop horizon and re-solved the optimal control problem every hop. If mode schedules were considered, this would be equivalent to optimizing over 1296 differential equations every hop, or roughly every 2 seconds.

We implemented our simulation as the difference equation in Sec. 3.4.1 and used SciPy's L-BFGS-B implementation for our optimizer [Vir+20; Byr+95]. L-BFGS-B is an efficient method for bounds-constrained smooth optimization, needed for both our actuation constraints and to evaluate the underlying smoothness of the hop-to-hop dynamics. Using the simulated optimizer, we identified that the hopper was able to follow waypoints by minimizing the Euclidean distance to the waypoint, and was able to turn by minimizing the angular distance[8] between the hopper's heading and a desired heading. In both of these motions, the hopper struggled to turn counter-clockwise and accumulated a clockwise drift.

[8] $\arctan 2\left(\sin(\theta_{\text{des}} - \theta), \cos(\theta_{\text{des}} - \theta)\right)$

Algorithm 3: Calculate instantaneous boom angle.

Data: d, a $3x4xN$ array of 3D motion capture data of 4 markers for N frames, with the hopper at rest in the first 40 frames. The first 3 markers are on the body and the fourth marker is on the boom.

Result: $Nx1$, instantaneous boom angle of each frame

; `/* Create a 4th, virtual, body marker to define the hopper's SE(3) configuration */`

$v_1 \leftarrow d[:,1] - d[:,0]$;

$v_2 \leftarrow d[:,2] - d[:,0]$;

$v_3 \leftarrow$ vectors orthogonal to v_1 and v_2 with magnitude $(||v_1|| + ||v_2||)/2$;

$d_2 \leftarrow [d[:,0], d[:,1], d[:,2], v_3 + d[:,0]]$;

; `/* Define the ''home'' configuration */`

ref $\leftarrow$average position of each marker in d_2 for the first 40 frames;

ref2 $\leftarrow$ ref centered and rotated in the xy plane so that its first marker lies on the negative y-axis;

; `/* Put data in frame defined by ref2 */`

$b \leftarrow d[:,4]$;

$b_2 \leftarrow$ empty array;

for *each frame in* d_2 **do**

  $c \leftarrow$ centroid of current frame;

  $d_k \leftarrow$ current frame minus c;

  $R \leftarrow$ least-squares matrix transforming ref2 to d_k;

  $R_U, R_V \leftarrow$ left and right singular vectors of R, respectively;

  b_2[frame] $\leftarrow (b - c)(R_U R_V)^{\mathrm{T}}$;

end

; `/* Fit circle to boom data */`

$\bar{b} \leftarrow$ average position of b_2;

$u \leftarrow$ PCA of $\bar{b}$;

$u_x \leftarrow$ last principle component of u projected onto the x-axis;

$u_y \leftarrow$ last principle component of u projected onto the y-axis;

$x \leftarrow \bar{b}$ projected onto u_x;

$y \leftarrow \bar{b}$ projected onto u_y;

$c_x, c_y, r \leftarrow$ least-squares circle fitting x, y;

; `/* Angle of boom along circle is the boom angle */`

$\phi \leftarrow \arctan2(y - c_y, x - c_x)$;

return ϕ

Algorithm 4: Determine the indices of consecutive stance phase pairs in a trial. Consecutive stances must be consecutive local minima separated by local maxima.

Data: z, an array of height time series data
Result: $Nx2$ array of index pairs
; `/* Determine candidates */`
$v_z \leftarrow \text{savgol\_filter}(z, \text{window} = 7, \text{order} = 3, \text{deriv} = 1)$;
stance candidates $\leftarrow$ local minima indices of v_z;
flight candidates $\leftarrow$ local maxima indices of v_z;
; `/* Pair and apply thresholding to mitigate measurement and numerical noise */`
zenith threshold $\leftarrow \text{mean}(z) + \text{std}(z)$;
nadir threshold $\leftarrow \text{mean}(z) - \text{std}(z)$;
pairs $\leftarrow$ empty array;
$i_1 \leftarrow$ first stance candidate;
$i_2 \leftarrow$ first flight candidate;
$i_3 \leftarrow$ first stance candidate;
while *there are stance candidates after* i_1 **do**
  if $z[i_1]$ *below nadir threshold* **then**
    while $i_1 > i_2$ **do**
      $i_2 \leftarrow$ index of flight candidate after i_2
    end
    if $z[i_2]$ *above peak threshold* **then**
      while $i_2 > i_3$ **do**
        $i_3 \leftarrow$ index of stance candidate after i_3
      end
      if $z[i_3]$ *below nadir threshold* **then**
        pairs.append((i_1,i_3))
      end
    end
  end
  $i_1 \leftarrow$ index of stance candidate after i_1
end
return *pairs*

To transfer the controller onto hardware, we wrapped the control simulation code in a hardware interface on a control computer, pictured in Fig. 3.4. The hardware interface received the hopper's SE(2) configuration from the motion capture system while sending commands to the hopper's motors, both in real-time. When the interface determined the hopper was in stance, it provided the most recent Qualisys data to the controller as the current state and sent the resulting boom angle command to the boom actuator. This framework required the control simulation complete within 100 milliseconds.

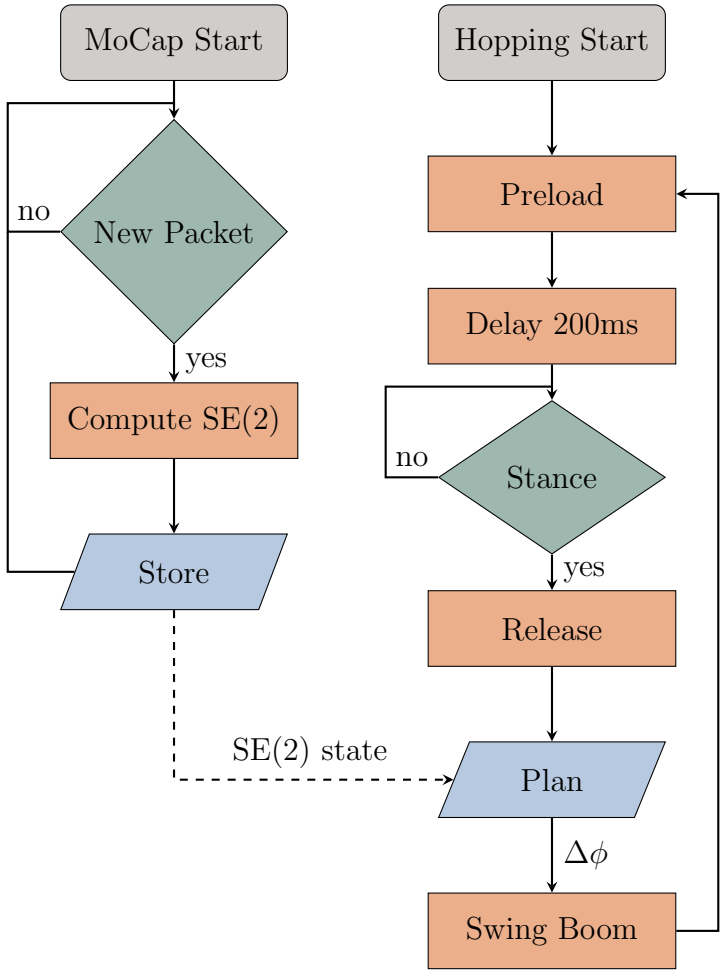

Figure 3.4: Hopping control process flow diagram. Left, a thread asynchronously polls for new data from the motion capture system and stores the hopper's SE(2) state in a buffer. Right, a separate thread controls the hopper's motors, reading the SE(2) state from the buffer only when needed for planning.

3.5 Experimental Results

We evaluated the control scheme laid out above on the robot described in Sec. 3.3. Our control scheme ignored mode schedules, reducing 1296 differential equations to (at most) 3 difference equations. We investigated the hopper's ability to perform two core steering maneuvers: approach the origin and turn 180°. The hopper's comparatively large turning radius to our motion capture arena (0.6m v.s. 2m×2m) prevented the demonstration of more complex path following.

Despite the simplified model, the hopper successfully completed both maneuvers. Fig. 3.5 shows the hopper successfully turning clockwise and as it minimized its Euclidean distance from the origin. The control response simulated on measurement data closely tracks the experimental behavior, despite deviation from the pure-simulation response. This demonstrated that the smooth control and dynamical modeling schemes were robust to model error. We further confirmed this by re-running the experiment with a foamcore board in the hopper's path. The foamcore board acted as a momentum reservoir subject to Coulomb friction, adding sudden "jerks" to the hopper's motion. Despite this, the hopper still successfully approached the origin. The experimental setup is pictured in Fig. 3.7 and the experimental results are shown in Fig. 3.8. We repeated these experiments with the hopper starting facing the positive x-axis ($\theta = 0$) and turning to face the negative x-axis by minimizing $\arctan2\left(\sin(pi - \theta), \cos(pi - \theta)\right)$. Fig. 3.6 shows the turning behavior without the foamcore board, and Fig. 3.9 shows the turning behavior with the foamcore board.

3.6 Discussion

Historically, the dynamics of different mode schedules for a single system have been to assumed to be unrelated. This non-relation leads to a combinatorial analysis – each of the (combinatorially many) mode schedules are treated as separate smooth systems.

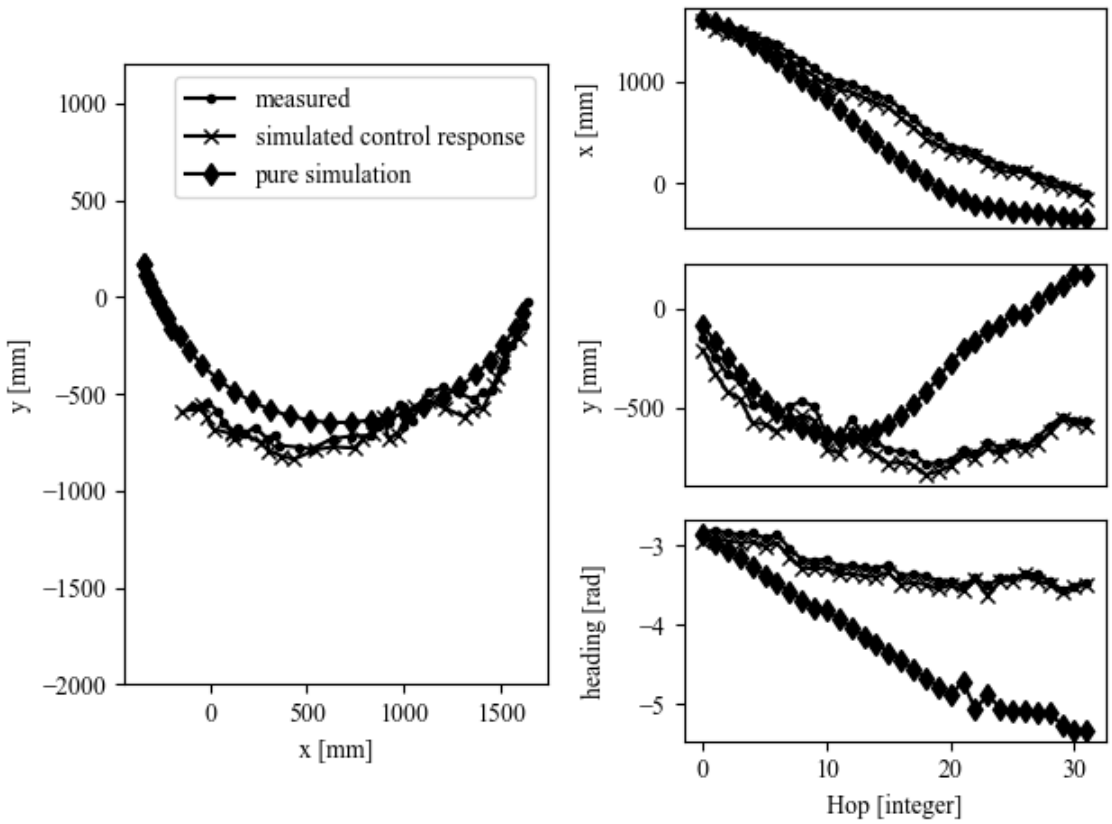

Figure 3.5: The hopper's motion in the unimpeded origin stabilization experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. Our control scheme was able to adapt to eventual deviation from the pure simulation response, showing that our simplified control scheme is effective.

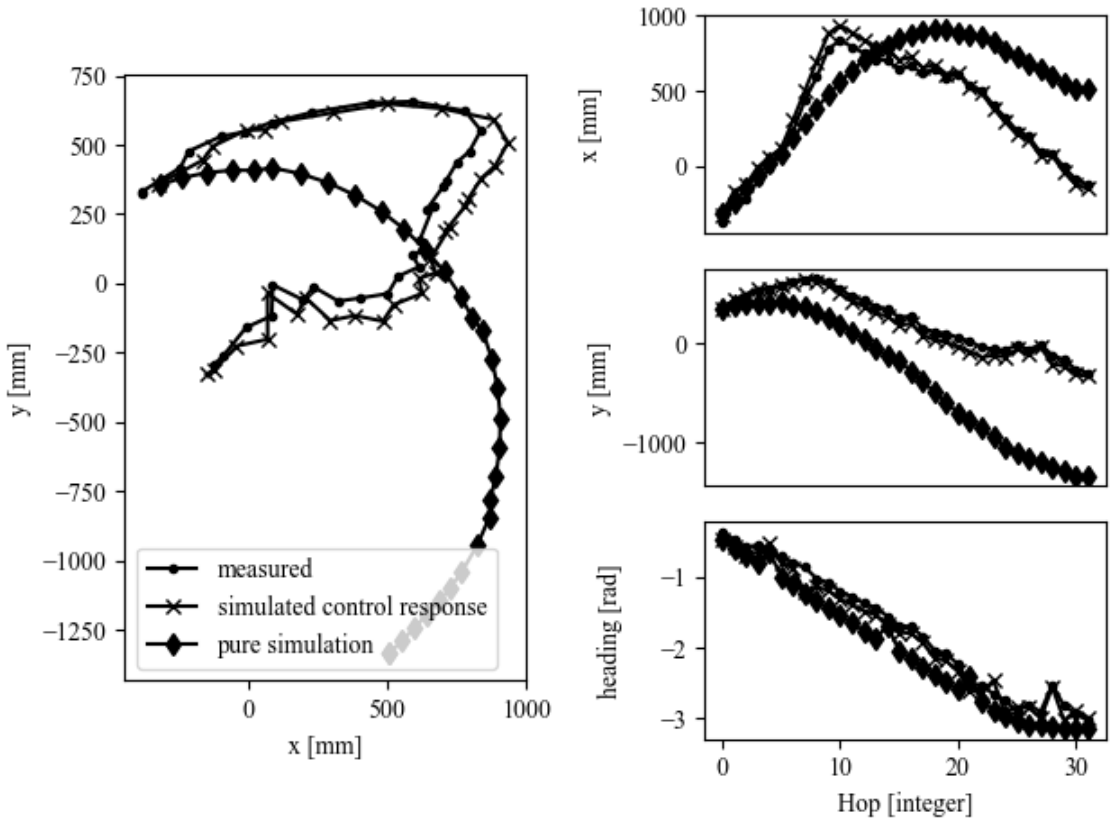

Figure 3.6: The hopper's motion in the unimpeded turning experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. Our control scheme was able to adapt to eventual deviation from the pure simulation response, showing that our simplified control scheme is effective.

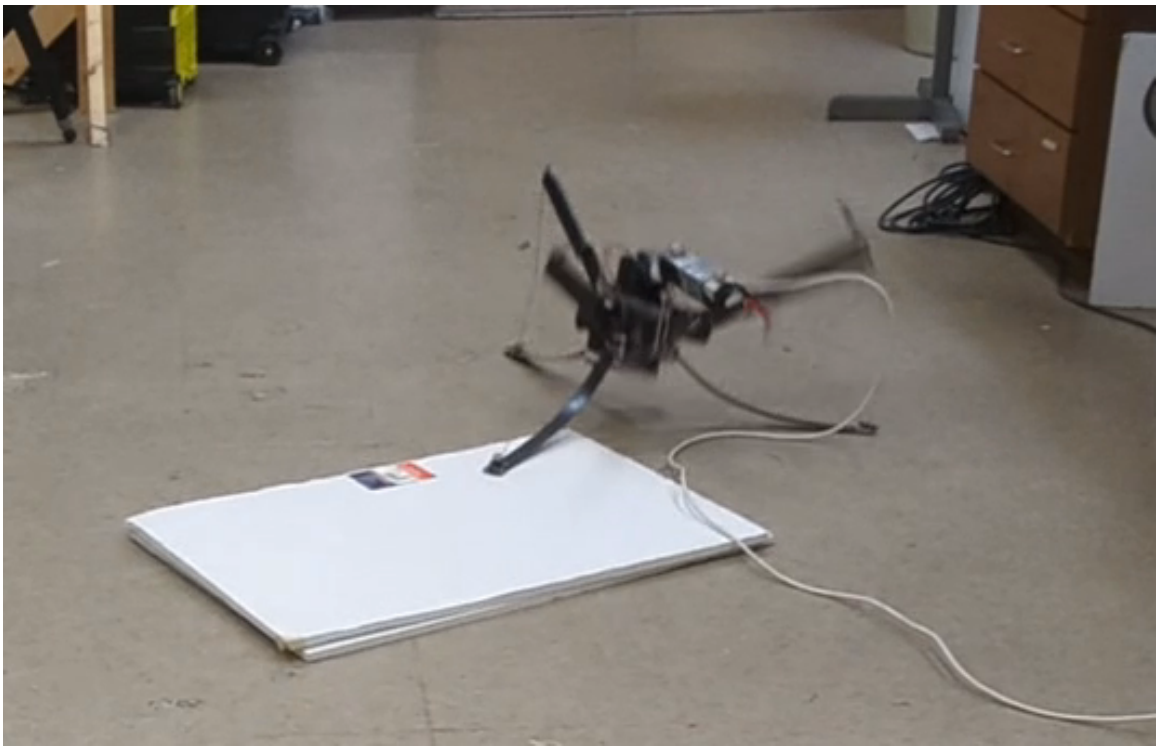

Figure 3.7: The foamcore board blocking the hopper's path as it move towards the origin. The foamcore caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller.

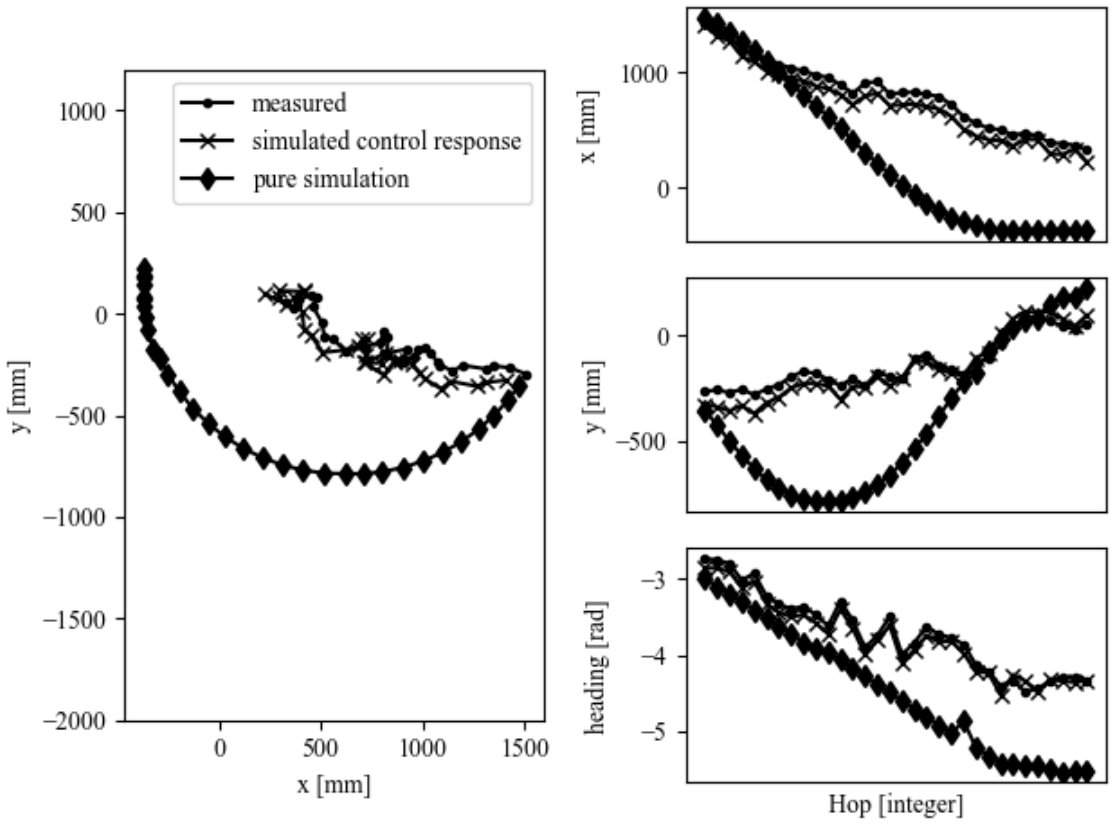

Figure 3.8: The hopper's motion in the impeded origin stabilization experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. The foamcore caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller.

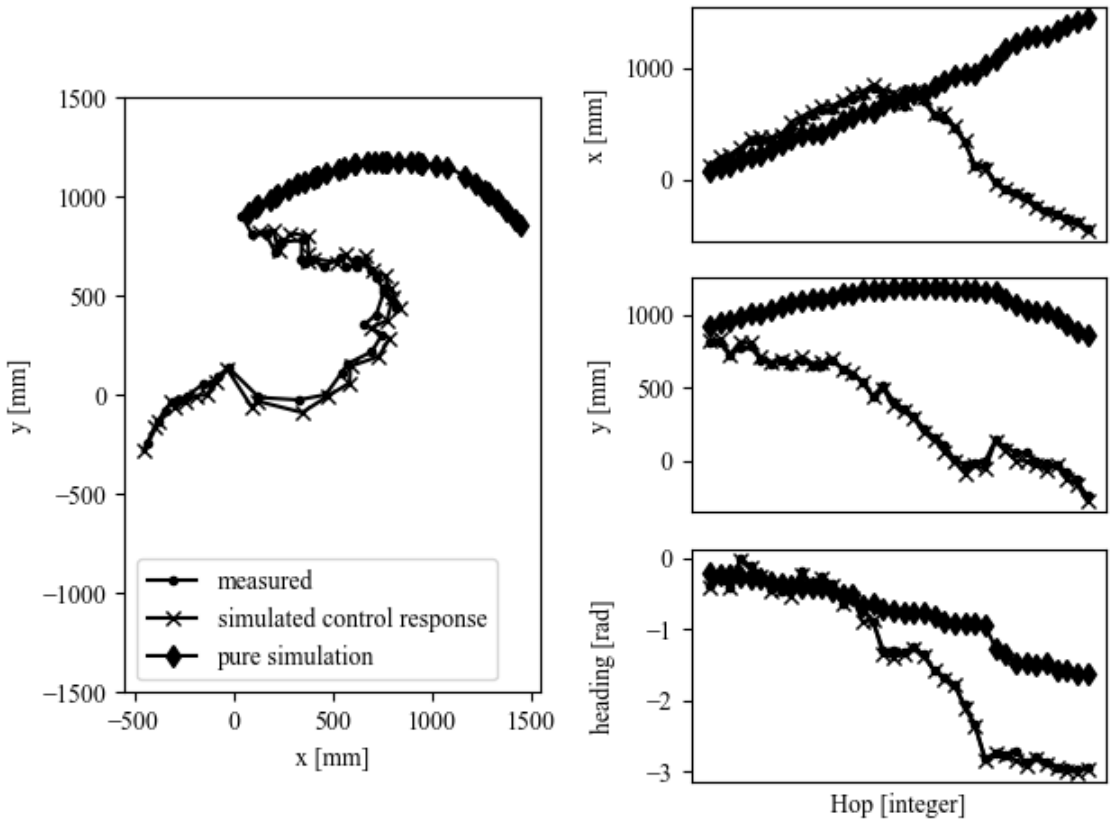

Figure 3.9: The hopper's motion in the impeded turning experiment compared against the simulated control response and the expected motion from pure simulation. The left plot shows the hopper's location in the motion capture arena while the right plots show the value of its SE(2) configuration over time. The path obstructions caused the hopper to experience inconsistent friction and hop-to-hop movement. That the hopper was still able to complete its task shows the robustness of our simplified model and controller.

This is true in the general case. What we have shown is that many mode schedule dynamics are related: they are simply pieces of one C^1-smooth system. We have demonstrated this insight by controlling a 3-legged hopping robot in the plane without considering mode schedules at all. We estimated the hop-to-hop dynamics from data using smooth system identification tools. These (smooth) estimates reduced 1296 differential equations to (at most) 3 difference equations. Our gradient-based predictive controller used the difference equations to select optimal control inputs online. In both simulation and hardware, the controller performed well despite incomplete friction and momentum modeling.

It is our hope that this work will diversify the actuation modalities present in robotics. By showing that many contact dynamics are smooth[9] we hope that more robots will make stranger use of contact with their environment in the future. Outside of robotics, this insight has potential application to many domains of hybrid dynamics, including analysis of power grids [HP00], neurobiology [Bur+16; Bro03], and economics [Bro03].

[9]Chiefly, those with spring-like (but not necessarily soft) contact forces. See Sec. 3.2 and [CRB22, Sec. 3.2] for details.

CHAPTER IV

A Numerical Integrator Specialized for Event-Selected Multi-Contact Dynamics

Having established that Event-Selected Systems (ESS) theory can be useful for the control of practical robots, we turn to the question of numerical methods specialized for ESS. An important step in the development of such methods is solving initial value problems (IVPs). This chapter treats the issue of determining when hybrid transitions (events) occur in along ESS trajectories. Herein we develop error bounds for a streamlined event localization method, implement it as an event resolver inside a classical (smooth) IVP solver, and show that it provides promising performance gains.

4.1 Introduction

Integrating ordinary differential equations (ODEs) is an expensive aspect of robot planning and optimization. Robot dynamics commonly contain discontinuities, as in legged locomotion and dexterous manipulation, which require additional special care to resolve. Many robots have a *piecewise-smooth* [GJ19] dynamical structure, where the state space is partitioned into pieces by surfaces called *guards*, and the dynamics in each piece are smooth. When a trajectory crosses or impacts a guard, the dynamics change, called an *event*. There are no general restrictions on how

the dynamics change at an event. However, the dynamics within each piece can be integrated as a classical IVP. From standard continuity arguments, if a trajectory crosses an event[1], integrating across the event can then be treated as two IVPs, one which terminates at the event and another using the state at the event as its initial value. Integrating in this way typically follows the loop:

1. Solve an IVP in one dynamical mode while checking for an event (classical integration and event detection)

2. Compute the state and time at the event (event localization)

3. Re-start classical integration from the event.

The accuracy of the resulting solution is governed by the accuracy of the integrator and the accuracy of the event localization method.

Solving IVPs numerically has a rich literature and we refer the reader to [HWN93; HW96] for a thorough treatment. It is common to describe a guard using a corresponding *event function*, a continuous scalar field which is zero only at the guard[2]. Since events occur when an event function composed on a trajectory crosses zero, the common "sign-checking methods" recast event localization as finding the root of a scalar function [ST00a; GK05; FAÅ14]. Despite their simplicity, sign-checking methods come with a set of challenges. Numerical integrators usually sample the trajectory at discrete points, called a *mesh*. For a trajectory y and event function h, robust event-localization methods will evaluate $h \circ y$ at multiple points which are a priori unknown, and this will require additional evaluations of h on additional points on the mesh, which requires additional integrations of the ODE. Many numerical codes return interpolating polynomials between mesh points, which allow efficient

[1] *Sliding modes* are an example of common piecewise-smooth phenomena which do not cross guards [GJ19, Chp. 1.1.]

[2] This is a consequence of the Implicit Function Theorem.

computation of additional values of y [Enr+86; Enr+88; Sha85; ORe70]. While interpolation of somewhat resolves the computational difficulty in finding a root of h, it is still possible that h crosses zero twice between consecutive points on the mesh [EKP01]. Prior works treat the problem of event localization very generally, spending significant theoretical and computational effort to handle pathological cases. To capture this "curvature," several methods augment the state vector to include $h \circ y$, ensuring the integration takes the curvature into account [Car78]. Sturm Sequences have been used to detect multiple roots both with and without the augmented state approach [SGB91; GK05].

Motivated by recent theoretical and practical developments, we leverage the stricter mathematical structure of ESS to develop an integration scheme specialized for event-selected robot dynamics. The next section provides proves that our approach to event localization has desirable accuracy properties. Sec. 4.3 describes our algorithm, followed by a tutorialized usage and a modeling example. Finally, Sec. 4.4 demonstrates the performance of our integration scheme wrapping a DOPRI5 method on several toy problems against standard event integrators.

4.2 Improving Interpolating Splines of Scalar Functions Composed on Event-Selected Trajectories for Event Localization

4.2.1 Motivation

Most event localization algorithms based on polynomial approximations to the solution trajectory avoid placing assumptions on the underlying structure of the hybrid dynamics. This has resulted in algorithms that focus on evaluating an event function along a spline of a trajectory, rather than a spline of the event function composed on

the trajectory. However, we are interested in building numerical methods specifically for integrating ESS. The liveness property of ESS places a lower bound on the time it will take for an event to occur. Lower bounding the event crossing time allows for the use of a triangle inequality to place error bounds on the interpolating Hermite spline of the spline of an event function composed on an event-selected trajectory. This approach appears to lower the computational cost of event localization. We provide a proof of the third-order method we implemented below, but the proof readily extends to higher orders when higher order interpolation methods are used is of order greater than three, and the underlying dynamics meet the accuracy conditions of the interpolant.

4.2.2 Background

Event localization seeks some t^* subject to $(h \circ y)(t^*) = 0$ for scalar event function $h(\cdot)$ and IVP solution $y(t)$. Computer memory is finite, so numerical integration schemes typically sample the integral curve y, returning a sequence of (time, value) pairs called a *mesh*. In practice, t^* nearly always lies between two mesh points. Determining t^* then requires interpolating between mesh points. Employing the original integration method for interpolation is costly, and most methods leverage some notion of polynomial interpolation [Sha85; Enr+86; FAÅ14]. In the event localization literature, there are two common methods which employ polynomial interpolation. The first is to search a time interval, interpolating y at each candidate time, and passes the result to a user-provided event function for evaluation, as in [SGB91]. This approach efficiently computes candidate solution values, but loses efficiency when h lacks a simple algebraic structure. The second approach, used in [GK05], treats $h \circ y$ as part of the solution curve, augmenting the dynamics to include dh/dt. This ensures that $h \circ y$ matches the specified numerical tolerance and allows direct construction of an interpolant to $h \circ y$, at the cost of additional function evaluations. [ORe70] constructed

an interpolant to $h \circ y$ for 4th-order Runge-Kutta methods directly from the mesh of y.

4.2.3 Proof of 3rd-Order Accuracy

Without loss of generality, we will consider the case of a scalar event function. Specifically, we want to show that, for a trajectory $y(t)$ with mesh $\{(t_0, y_0), (t_1, y_1)\}$ and event function h, when $(h \circ y)(t^*) = 0$, not only does the interpolating cubic Hermite spline for $h \circ y$ on the mesh have a root t' close to t^*, but the interpolating cubic Hermite spline to y at t' is close to t^*. With some algebra through the proofs below, it can be shown that the specific error bounds are

$$|t' - t^*| \leq \frac{\frac{L}{24} + L_h K}{c} \cdot \delta t^4$$

and

$$||p(t') - y(t^*)||_\infty \leq \left(\frac{\frac{L}{24} + L_h K}{c} L_p + K \right) \cdot \delta t^4$$

with variable definitions found in Tab. 4.1.

Table 4.1: Description of variables and relevant constants used in this proof.

t_0	Mesh point preceding the event
t_1	Mesh point following the event
$I := [t_0, t_1]$	The interval being interpolated for event localization
$\delta t := \|t_1 - t_0\|$	Time between mesh points
y	The exact solution curve to the IVP
h	The event function
$h_y := h \circ y$	The event function, h composed on the exact solution curve, y
p_h	The interpolating spline to h_y

p	The interpolating spline to y
$g := h \circ p$	The event function composed on the interpolating spline of y
$L := \sup_{t \in I} g^{(4)}(t)$	The upper bound of the fourth derivative of the event function composed on the true solution curve
L_h	The Lipschitz constant of h on the compact set containing $p(\cdot)$ and $y(\cdot)$
K	The error constant for p, i.e., $\|\|p(t) - y(t)\|\|_\infty \leq K\delta t^4$ for $t \in I$
$c := \inf_{t \in I} \dot{h}_y(t)$	The liveness constant of the event function h
$L_p := \sup_{t \in I} \|\|\dot{p}(t)\|\|_\infty$	Upper bound on the time derivative of the interpolating spline to y

We now recapitulate the definition of *order* for numerical integrators and the formula for interpolating cubic Hermite splines before proceeding to our proof.

Definition 4.2.1. *A numerical integration method is said to be of* ***order*** p *for a timestep* $\delta t := t_1 - t_0$*, exact solution* $y(\cdot)$*, and mesh point* (t_1, y_1) *if*

$$\|\|y(t_0 + \delta t) - y_1\|\|_\infty \leq K\delta t^{p+1} \qquad \text{[HWN93, Eq. 1.10, p.134].}$$

Definition 4.2.2. *An interpolating* ***cubic Hermite spline*** *constructed using the values at the mesh,* $\{(t_0, y_0), (t_1, y_1)\}$*, their corresponding derivatives* $f_0 := f(y_0), f_1 := f(y_1)$*, and their timestep* $\delta t := t_1 - t_0$ *has the formula*

$$u(\theta) = (1-\theta)y_0 + \theta y_1 + \theta(\theta-1)\left[(1-2\theta)(y_1 - y_0) + (\theta-1)\delta t f_0 + \theta\delta t f_1\right]$$

[HWN93, Eq. 6.7, p.190]

for $\theta \in [0, 1]$ *and* $\theta := (t - t_0)/\delta t$*.* $u(\theta)$ *is 3rd-order so long as the numerical scheme which computed the mesh is at least 3rd order.*

Theorem 4.2.1. *For a trajectory $y(t)$ subject to $\dot{y} = f(y)$ with $f \in \mathcal{C}^3$, an event function $h \in \mathcal{C}^4$, and at-least-3rd-order mesh $\{(t_0, y_0), (t_1, y_1)\}$:*

1. *p_h, the interpolating Hermite spline to h built from the mesh is 3rd-order accurate to $h_y := h \circ y$ on $I := [t_0, t_1]$.*

2. *If both p_h and h_y have roots t', t^* respectively on I, and if p is the 3rd-order interpolating Hermite spline to the mesh, then $||p(t') - y(t^*)||_\infty \leq K|t_1 - t_0|^4$ for positive K.*

Proof. (1) is a direct result from the theorem statement and Lemma 4.2.2. Let $\delta t := |t_1 - t_0|$. Since (1) is true, and h_y is strictly monotonically increasing and smooth since it is an event function (Def. 3.2.1), we immediately have $|t^* - t'| \leq K_1 \delta t^4$ for positive K_1 from Lemma 4.2.3. Then, since $|t^* - t'|$ is bounded by a 4th-order polynomial, and p is 3rd-order to y, there exists some $K > 0$ so that $||p(t') - y(t^*)||_\infty \leq K\delta t^4$. □

Lemma 4.2.2. *If $\{(t_0, y_0), (t_1, y_1)\}$ is an at-least-3rd-order mesh for some $y(t)$ subject to $\dot{y} = f(y)$ for $f : \mathbb{R}^n \to \mathbb{R}^n$ at least thrice-differentiable, and $h : \mathbb{R}^n \to \mathbb{R}$ is at least four times differentiable, then p_h, the cubic Hermite spline to $h \circ y$ with boundary conditions*

$$p_h(t_0) = h(y_0) \qquad \dot{p}_h(t_0) = Dh(y_0) \cdot f(y_0)$$
$$p_h(t_1) = h(y_1) \qquad \dot{p}_h(t_1) = Dh(y_1) \cdot f(y_1),$$

is 3rd-order to $(h \circ y)(t)$ on $I := [t_0, t_1]$.

Proof. Let p be the cubic Hermite spline built from $\{(t_0, y_0), (t_1, y_1)\}$ and define $g := h \circ p$ and $h_y := h \circ y$. This proof uses a triangle inequality with $|g(\cdot) - h_y(\cdot)|$. We seek to bound

$$|p_h(t) - g(t)| + |g(t) - h_y(t)|$$

on I by δt^4 times some positive constant, which is the necessary and sufficient condition for p_h to be 3rd-order to h_y on I.

To bound $|g(\cdot) - h_y(\cdot)|$, let $\bar{U}$ be a compact set containing both the image of $y(\cdot)$ and $p(\cdot)$ on I. h is continuously differentiable, so is Lipschitz on compact sets. Let $L_h > 0$ be the Lipschitz constant of h on $\bar{U}$. Since p is 3rd-order to y [HWN93, p.190],

$$|g(t) - h_y(t)| \leq |(h \circ p)(t) - (h \circ y)(t)| \leq L_h \, ||p(t) - y(t)||_\infty \leq L_h K \cdot \delta t^4.$$

Turning to $|p_h(\cdot) - g(\cdot)|$, [BF10, p.137] gives an error formula for Hermite splines using a (typically) unknown ξ. The formula for a spline like p_h requires that $g \in \mathcal{C}^4$. Recall that $h \in \mathcal{C}^4$ and p analytic, so $g := h \circ p \in \mathcal{C}^4$. Then $g^{(4)}$ is continuous over the compact set I and therefore bounded. Choose $L := \sup_{t \in I} |g^{(4)}|$. Observing that p_h is constructed from two points of g, the error formula becomes

$$|p_h(t) - g(t)| = \frac{(t - t_0)^2 (t - t_1)^2}{4!} \cdot |g^{(4)}(\xi(t))| \leq \frac{L}{24} \cdot \delta t^4.$$

Applying the triangle inequality,

$$|p_h(t) - h_y(t)| \leq |p_h(t) - g(t)| + |g(t) - h_y(t)| \leq \left(\frac{L}{24} + L_h K \right) \cdot \delta t^4.$$

we see that $|p_h - h_y|$ is bounded by a 4th-order polynomial on I, and so p_h is 3rd-order to $h_y := h \circ y$. □

Lemma 4.2.3. *If p_h is of order $P-1$ to the strictly monotonically increasing smooth curve $h_y : \mathbb{R} \to \mathbb{R}$ on $I := [t_0, t_1] \subset \mathbb{R}$ and $h_y(t^*) = 0$, $p_h(t') = 0$ for $t^*, t' \in I$, then $|t^* - t'|$ is bounded by $K|t_1 - t_0|^P$ for positive K.*

Proof. Let $\delta t := |t_1 - t_0|$ and $|h_y(t) - p_h(t)| \leq K_1 \delta t^P$. $p_h(t') = 0$, so

$$|h_y(t')| = |p_h(t') - h_y(t')| \leq K_1 \delta t^P.$$

h_y is strictly monotonically increasing, so $\dot{h}_y \geq c > 0$. Fix $c_2 := \inf_{t \in I} \dot{h}_y(t)$ and observe that $c|t_a - t_b| \leq c_2|t_a - t_b| \leq |h_y(t_a) - h_y(t_b)|$ for $t_a, t_b \in I$. Then

$$c|t' - t^*| \leq |h_y(t') - h_y(t^*)| = |h_y(t')| \leq K_1 \delta t^P$$

Let $K := K_1/c$ and then $|t^* - t'| \leq K\delta t^P$. □

Lemma 4.2.4. *If a smooth curve $p : \mathbb{R} \to \mathbb{R}^n$ is order $P-1$ to another curve, $y : \mathbb{R} \to \mathbb{R}^n$, on the compact interval $I := [t_0, t_1] \subset \mathbb{R}$ with constant K and the distance between two real numbers $t', t^* \in I$ is bounded by $K_1 \cdot |t_1 - t_0|^P$, then $||p(t') - y(t^*)||_\infty \leq K_2 \cdot |t_1 - t_0|^P$ for positive K, K_1, K_2.*

Proof. Let $\delta t := |t_1 - t_0|$ and recall $||p(t) - y(t)||_\infty \leq K \cdot \delta t^P$. Then

$$||p(t') - y(t^*)||_\infty \leq ||p(t') - p(t^*)||_\infty + ||p(t^*) - y(t^*)||_\infty \leq ||p(t') - p(t^*)||_\infty + K \cdot \delta t^P.$$

By the Mean Value Theorem, there is some $\eta \in I$ such that $||p(t') - p(t^*)||_\infty \leq ||\dot{p}(\eta)||_\infty \cdot |t' - t^*|$. Since $|t' - t^*| \leq K_1 \cdot \delta t^P$,

$$||p(t') - p(t^*)||_\infty \leq ||\dot{p}(\eta)||_\infty K_1 \cdot \delta t^P.$$

Choosing $L_p := \sup_{t \in I} ||\dot{p}(t)||_\infty$ and $K_2 := K + K_1 L_p$,

$$||p(t') - y(t^*)||_\infty \leq K_2 \cdot \delta t^P.$$

□

4.3 Algorithm

To our knowledge, the integration algorithm presented herein is the first to perform event localization using a polynomial approximation to $h \circ y$ between mesh points from any (suitably accurate) integrator. Furthermore, our algorithm seems to be the first tailored to the treatment of multi-contact using sign-checking. As a result of using these straightforward techniques, we see significant performance despite our pure-Python implementation in Sec. 4.4 We will use the notation in Table 4.2 in this section.

f	Vector field being integrated, i.e., $\dot{y} = f(y, ...)$
h	The m event functions represented as a vector-valued function
Dh	The Jacobian of h
h_k	kth event function
$\mathfrak{g}_k$	Guard corresponding to $\{y : h_k(y) = 0\}$

Table 4.2: Notation used in the integration algorithm.

Our algorithm operates in two modes. The first is classical integration. The trajectory is integrated like any classical integration method (e.g. RK4), and the event function signs are compared between consecutive mesh points. When the signs differ, the algorithm performs multi-contact event localization.

4.3.1 Discontinuity Locking

Like many event-localizing integrators, our algorithm employs "discontinuity locking." Event localization requires numerical root-finding machinery, and so is subject to the difficulties of numerical tolerance. In general, any event localization algorithm can return a mesh point on the trajectory which is on the guard to within numerical tolerance, but not exactly on the guard. This phenomenon is called "discontinuity sticking." In such a case, relying on the sign of $h(y)$ to determine the hybrid mode can cause the event localization algorithm to become stuck in an infinite loop. Such

repeated executions are unnecessary. Discontinuity locking prevents discontinuity sticking by manually updating the variable which indicates the hybrid mode according to user-specified rules which describe the acceptable transitions between modes. Our algorithm represents the progression of the hybrid mode within an ESS as a length-m Boolean vector, b. When the jth guard is crossed, $b[j] \leftarrow +1$.

4.3.2 Single Multi-Contact Resolution

Given a list of "candidate" guards that a trajectory crossed on some mesh interval $(t_0, y_0), (t_1, y_1)$, we build an interpolating cubic Hermite spline on $[t_0, t_1]$ subject to the boundary conditions

$$p_{h_k}(t_0) = h_k(y_0) \qquad \dot{p}_{h_k}(t_0) = Dh_k(y_0) \cdot f(y_0)$$
$$p_{h_k}(t_1) = h_k(y_1) \qquad \dot{p}_{h_k}(t_1) = Dh_k(y_1) \cdot f(y_1).$$

where h_k is a candidate event function and Dh_k is its gradient. From Thm. 4.2.1, the root of p_h on $[t_0, t_1]$ will be 3rd-order close (Def. 4.2.1) to the true root of $(h \circ y)(t)$. We then note the guard which has the smallest impact time, t^*, interpolate to t^* using a cubic Hermite spline to the mesh, and mark the guard as crossed. We then use a 3rd-order Runge-Kutta method (RK3) to integrates the IVP $y(t^*)$ from t^* to t_1 in the new mode, check for any guard crossings, and repeat until there are no guard crossings on $[t^a st, t_1]$ This procedure is illustrated in Algorithm 5.

4.3.3 Usage

Our algorithm can be split into two parts: classical integration and multi-contact resolution. Any 3rd-order or better integrator capable of sign-change-triggered early termination is suitable for classical integration. Such classical integrators typically require definitions of f and h, with several optional parameters for handling numerical

Algorithm 5: Single multi-contact event resolution pseudocode.

Data: t_0, the time before the multi-contact event; t_1, the time after the multi-contact event; y_0, the state at t_0; y_1, the incorrect state at t_1; f, the ESS dynamics; h, the event functions as a vector; Dh the Jacobian of h

Result:

$h_0 \leftarrow h(y_0)$;
$h_1 \leftarrow h(y_1)$;
$b \leftarrow \operatorname{sgn} h_0$;
$f_0 \leftarrow f(y_0, b)$;
$f_1 \leftarrow f(y_1, b)$;
$cands \leftarrow \text{argwhere}((b < 0 \wedge h_1 \geq 0))$;
while $\neg\text{empty}(cands)$ **do**
  $\{\mathrm{t}^*{}_i\} \leftarrow \text{ImpactTimes}(cands)$ `/* See Alg. 6 */`
  $k \leftarrow \text{argmin}\{\mathrm{t}^*{}_i\}$;
  $\delta t \leftarrow t_1 - t_0$;
  $\theta \leftarrow (t_k^* - t_0)/\delta t$;
  $y_0 \leftarrow (1-\theta)y_0 + \theta y_1 + \theta(\theta-1)\ (\ (1-2\theta)(y_1 - y_0) + (\theta-1)\delta t f_0 + \theta\delta t f_1\ )$;
  $t_0 \leftarrow \mathrm{t}^*{}_k$;
  $b[cands[k]] \leftarrow +1$;
  `/* Mark the` $cands_k$`th guard as crossed. */`
  $y_1 \leftarrow \text{Rk3Step}(y_0, f(\cdot, b), t_1 - \mathrm{t}^*)$ `/* A single integration step of the "Heun order 3" method in [HWN93, Table 1.1, p. 135] */`
  $h_0 \leftarrow h(y_0)$;
  $h_1 \leftarrow h(y_1)$;
  $cands \leftarrow \text{argwhere}((b < 0 \wedge h_1 \geq 0))$;
end

Algorithm 6: Impact time calculation pseudocode.

Data: t_0, the initial time on the mesh; t_1, the final time on the mesh; y_0, the state at t_0; y_1, the state at t_1; f, the governing vector field at y_0; h, the event functions; Dh, the Jacobian of the event functions; cands, the guards crossed between y_0 and y_1

Result: Times of guard crossings

$\delta t \leftarrow t_1 - t_0$;
$h_0 \leftarrow h(y_0)$;
$h_1 \leftarrow h(y_1)$;
$\dot{h}_0 \leftarrow \text{dot}(Dh(y_0), f(y_0))$;
$\dot{h}_1 \leftarrow \text{dot}(Dh(y_1), f(y_1))$;
tarr$\leftarrow$ [];
for $i \in cands$ **do**
    $t \leftarrow \text{RootFind}(\text{Hermite}(h_0[i], h_1[i], \dot{h}_0[i], \dot{h}_1[i], \delta t), 0, 1)$;
    `/* Hermite evaluates a scalar spline as in [HWN93, Eq 6.7, p.190] and RootFind is a bracket rootfinding algorithm */`
    tarr $\leftarrow [\text{tarr}, t]$;
end
return tarr

tolerances.

Algorithm 5 requires definitions of f, h, Dh, and a root finding algorithm. We recommend using the Illinois algorithm [For95] for root finding. Note that Dh, the Jacobian of h, need not be closed form. The discontinuity locking approach provides no way to "revoke" mode transitions, i.e., set entries in b to -1. In this way, Algorithm 5 acts as a "one-shot," i.e., This interferes with the representation of rhythmic systems, since many periodic orbits in hybrid dynamics transition back and forth between hybrid modes. However, event-selectedness is a property open neighborhoods, so a dedicated integrator can be instantiated for each event-selected region. A supervisory process can then select between the different integrators to appropriately integrate each portion of the dynamics along a larger trajectory. Sec. 4.3.4 provides a modeling example for a periodic 2D ESS.

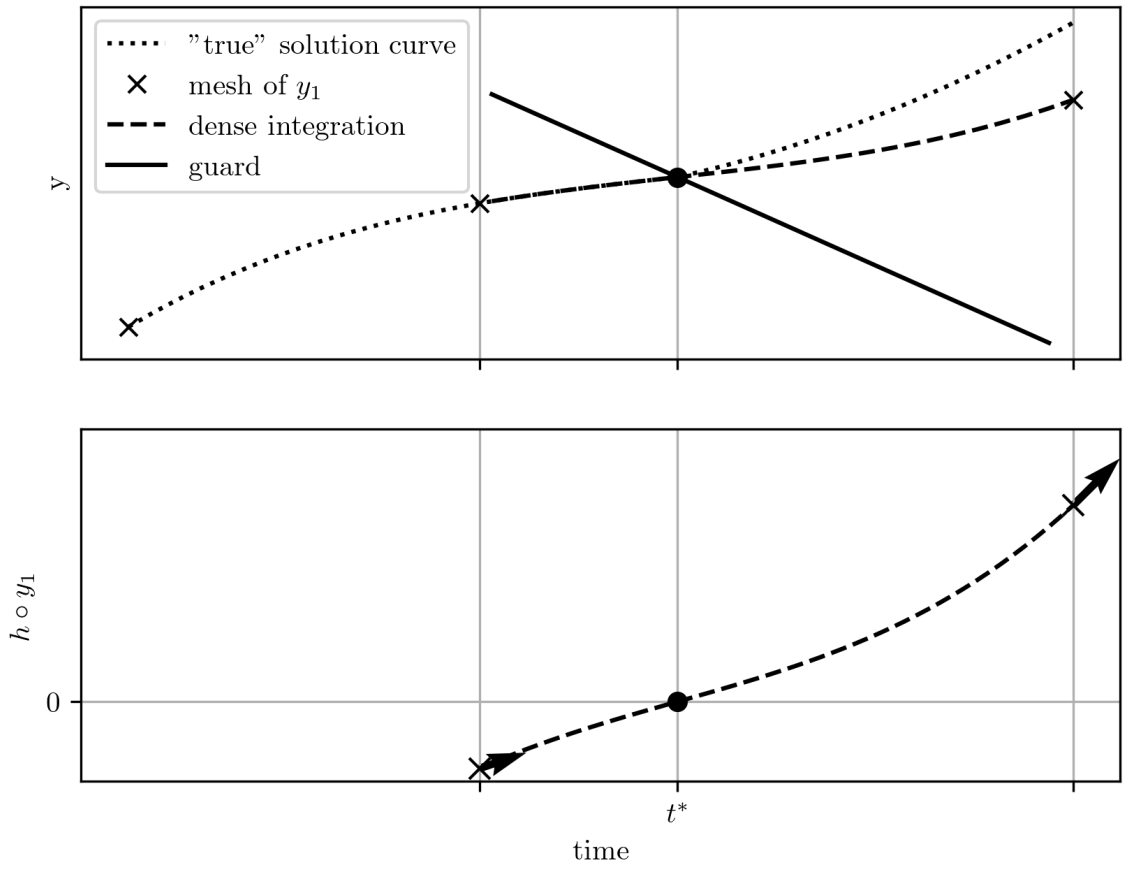

Figure 4.1: Densely integrating $h_k(x(t))$ directly to resolve the kth guard impact. By abuse of notation, h_k refers to the value of $h_k(\cdot)$ as a variable. The system obeys $\dot{x} = f_1(x)$ for $h_k < 0$ and $\dot{x} = f_2(x)$ otherwise. Using the multivariable chain rule, h_k follows the scalar differential equation $\dot{h}_k = \nabla h_k \cdot f_1(x)$ until guard impact ($h_k = 0$). Thus the value of $\nabla h_k \cdot f_1(x)$ at (unknown) t^*, is identical to "true" solution. Furthermore, $\nabla h_k \cdot f_1(x)$ is smooth, and can be densely integrated using a cubic spline [HWN93, p.190, Eqn. 6.7]. This transforms

4.3.4 Modeling Example

Consider the model of a 1D hopper with height above ground y, contact length L, spring constant k, spring rest length L_0, mass m, and gravity g:

$$\dot{y} = f(y) = \begin{cases} -g & , y > L \\ -g + k(L_0 - y)/m & , y \leq L. \end{cases}$$

The hopper dynamics have the guard $y \equiv L$, which is crossed with both positive and negative velocity. To represent the hopper as ESS, we partition it into two ESS, one with event function $h = y - L$ (takeoff) and the other with $h = L - y$ (touchdown). We can then select which ESS to integrate based off of the sign of $\dot{y}$. To ensure that we determine which ESS to integrate as infrequently as possible, we equip each ESS with an early termination function to stop integration when $\dot{y}$ changes sign. Evaluated on a mesh, this function takes the form $\dot{y}_{k-1}\dot{y}_k \leq 0$. This logic can be incorporated inside of a while loop, shown in Algorithm 7.

4.4 Numerical Experiments

4.4.1 Algorithm Implementation

We implemented our algorithm as a Python class which wraps a user-provided classical integrator. In the experiments below, we used a Python implementation of DOPRI5 ([HWN93, II.4, p.178]) for our classical integrator. We used the closed-form cubic method for our Hermite spline root finding.

4.4.2 Order Test

Since we were using a 5th-order classical integrator, we expected that any trajectory integration error should be dominated by the error of our projection step, and not

Algorithm 7: Integrate 1D hopper dynamics by connecting two ESS complexes.

Data: t_0, the starting time for integration; t_f, the ending time for integration; y_0, the initial (height, velocity) condition. IntegrateTouchdownESS, the integrator build for the touchdown dynamics; IntegrateTakeoffESS, the integrator build for the takeoff dynamics

Result: $Nx1$, boom angle command for each hop

$t_out \leftarrow [t_0]$;
$yout \leftarrow [y_0]$;
while $t_0 \leq t_f$ **do**
  `/* Decide which system to integrate based on the boundary between ESS complexes. In this case that boundary is the sign of the velocity. */`
  if $y_0[1] < 0$ **then**
    $tt, yy \leftarrow$ IntegrateTouchdownESS(t_0, t_f, y_0);
  end
  else
    $tt, yy \leftarrow$ IntegrateTakeoffESS(t_0, t_f, y_0);
  end
  $tout \leftarrow [tout, tt]$;
  $yout \leftarrow [yout, yy]$;
  $t_0 \leftarrow tt$[end] $y_0 \leftarrow yy$[end]
end

by our integration step. To experimentally verify that our algorithm is 3rd-order, we needed a differential equation of sufficient complexity which still had a known closed-form solution. Linear and affine differential equations possess readily-computable exact solutions through the matrix exponential, making them a natural choice. For simplicity, we chose our guards to be the coordinate planes. We constructed a 3D system of affine ordinary differential equations, described in Table 4.3

	$x < 0$	$x \geq 0$
$y < 0$	$-y + 1$	$-2y + 1$
$y \geq 0$	$y + 1$	$10x + 1$

(a) $\dot{x}$

	$x < 0$	$x \geq 0$
$y < 0$	$x + 1$	$x/2 + 2$
$y \geq 0$	$-x + 1$	$y + 1$

(b) $\dot{y}$

	$z \leq 0$	$z > 0$
$y < 0$	$3z - 1$	$-z - 1$
$y \geq 0$	$3z - 1$	$-z - 1$

(c) $\dot{z}$

Table 4.3: The 3D system of affine ODEs used for our order test. The x velocity (a), y velocity (b), and z velocity (c) are grouped separately.

We integrated the IVP $(x, y, z) = (-0.4, -0.15, 0.3)$ from $t = 0$ to $t = 0.5$. This trajectory crossed all guards during the time interval. We plotted the log-log integration error (Fig. 4.2) and found it to be bounded by a polynomial of order at least 4.2.

4.4.3 Calibration against MuJoCo

To assess the practical merit of our algorithm, we sought to compare it with a state-of-the-art simulation framework which explicitly treats collision phenomena. We elected to use MuJoCo, short for "Multi-Joint dynamics with Contact," for our performance baseline [Dee]. We selected MuJoCo for the simplicity of use with Python, and the ease of reading and writing modeling parameters from its XML modeling framework. MuJoCo is a simulation package which has received much attention for robot simulation [ETT15; Zha+21; Koe+15; TET12]. As such, MuJoCo has benefited from many years of dedicated optimization in C and C++ [Dee], while we implemented our integrator in Python. It is well-known that Python is less performant than C

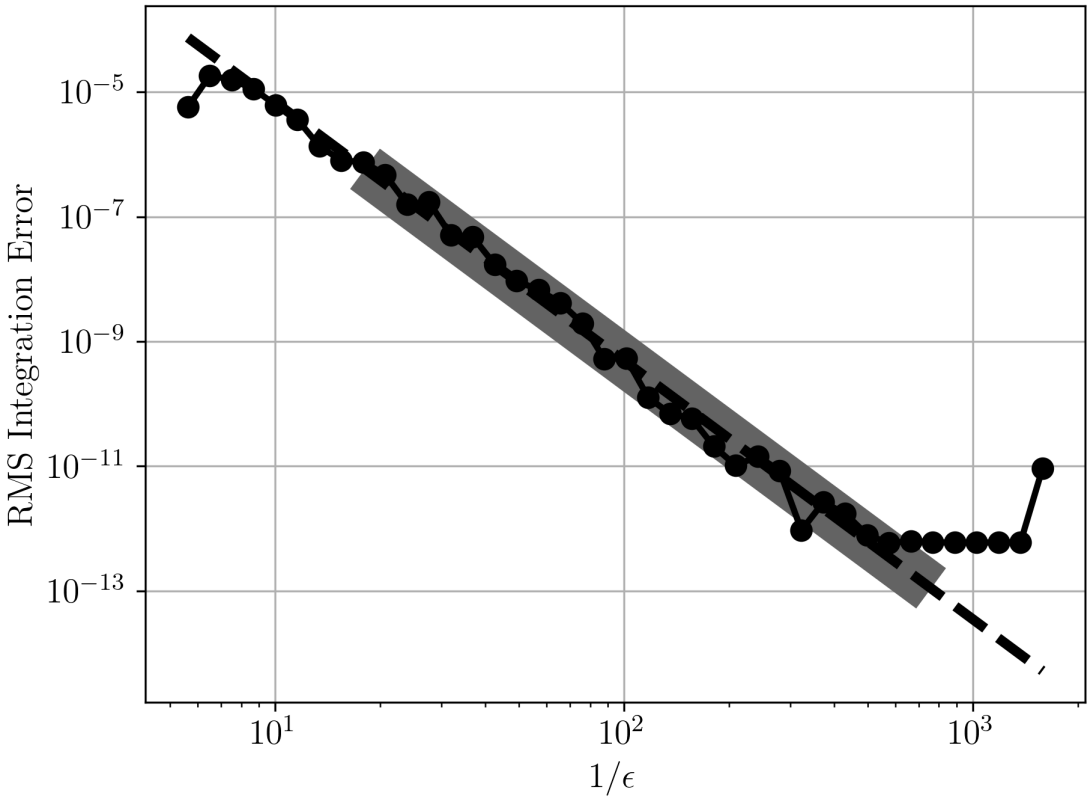

Figure 4.2: Log-log plot of our algorithm's RMS integration error versus $1/\varepsilon$ (dotted line) for an initial value problem. The data interval used to determine the order (gray rectangle) and the fit extended over the entire interval considered (dashed line) are overlaid. The integrator achieved third-order accuracy, indicated by the slope of the dashed black lined being above 4. The tails for small and large $1/\varepsilon$ is expected.

[Har+20a; Pre00], and it would be reasonable to not only expect MuJoCo to outperform our algorithm in a head-on comparison of execution time, but also that our algorithm's execution time would be dominated by the runtime performance of the Python virtual machine itself.

To establish a meaningful comparison between our algorithm and MuJoCo uncomplicated by the details of software implementation, we needed an event-selected toy robot model with a closed-form exact solution to calibrate both the runtime performance and the accuracy of each solver. We used the 1D hopping model developed in Sec. 4.3.4. We simulated an equivalent hopper model in MuJoCo starting at rest with $z_0 = 2$, $L_0 = 1$ from $t = 0$ to $t = 2$. Unfortunately, the MuJoCo simulations were

unstable in the sense that the hopper gained energy with each successive hop. We were unable to find a configuration which made MuJoCo sufficiently accurate compared to our integrator, nor a configuration which made our integrator sufficiently inaccurate to compare with MuJoCo. As such, we were unable to directly calibrate our integrator's time step parameter. Instead, we selected a maximum time step of 1 second, and recorded the runtime for our integrator and MuJoCo, show in Table. 4.4. The trajectory comparison is show in Fig. 4.3.

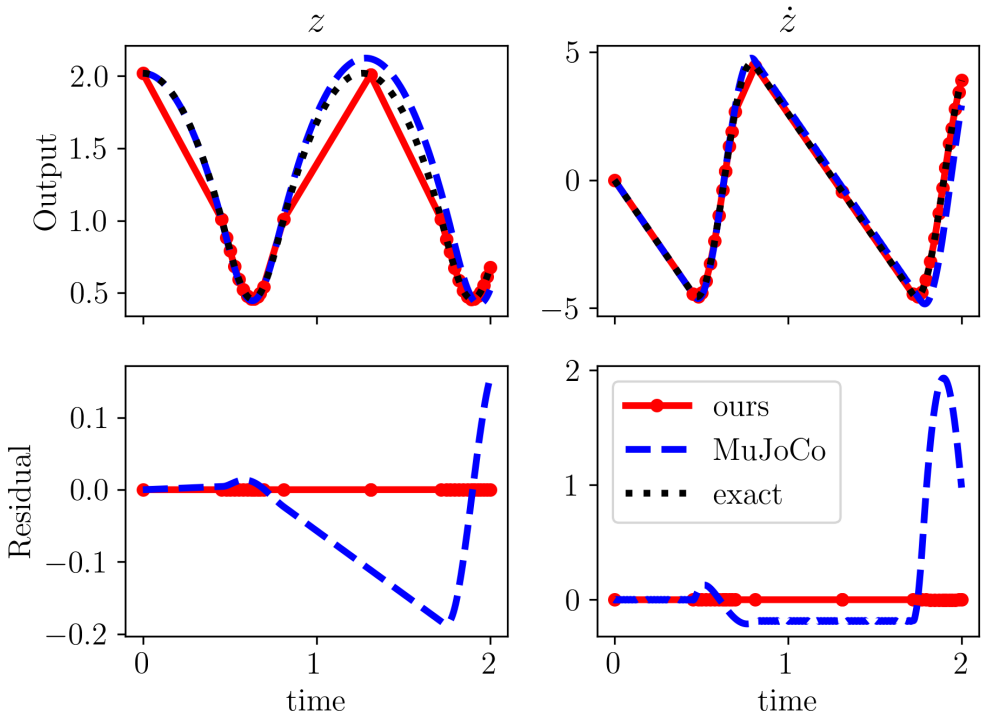

Figure 4.3: Time series output of our calibration: integrating a the trajectory of a 1D hopping robot. The top row of plots show the exact trajectory (dotted gray), MuJoCo's result (dashed blue), and our result (red) overlaid. Due to our integrator's large timestep, the top left plot is numerically accurate, but visually undersampled. The bottom row shows the residual from the exact trajectory for both MuJoCo and our solver. The left plots show height above the ground (contact height = 1) and the right plots show vertical velocity.

ε'	MuJoCo (s)	ours (s)
None	0.002054	0.001452

Table 4.4: Calibration parameters determined by simulating the same 1D hopper initial value problem with MuJoCo and our algorithm. The runtime in seconds of each method is tabulated in the right two columns. MuJoCo used a timestep of 0.002 seconds. We were unable to find parameters which made the MuJoCo simulation comparably accurate to ours, so we used a maximum timestep of 1 second for our integrator.

4.4.4 Scaling v.s. MuJoCo with Multiple Contacts

Our algorithm's potential gains are in its treatment of multiple guards. We wanted to assess how its performance scaled versus MuJoCo with increasing number of contacts. We developed a model of a plate in SE(2) falling due to gravity onto a bed of evenly-spaced vertical springs with frictionless rolling contacts. A model of this form allowed us to procedurally generate models and MuJoCo compatible XML of the plate falling onto arbitrarily many spring contacts. We assumed the plate was long enough that it was above each spring at all times. For the ith spring with x-position x_{s_i}, our flight-stance event functions and Jacobians took the form:

$$h_i = L_0 + \tan\theta \cdot (x - x_{s_i}) - z$$
$$Dh_i = \begin{bmatrix} \tan\theta, & -1, & sec^2\theta \cdot (x - x_{s_i}) \end{bmatrix}$$

We generated 150 random initial conditions starting from rest centered over the spring bed with $z_0 \in (2, 3)$ and $\theta_0 \in (-0.1, 0.1)$. For each initial condition, we simulated the plate falling onto spring beds of 2 to 35 evenly-spaced identical spring-damper contacts with the precision parameter determined from our calibration. We evaluated each algorithm on its marginal runtime cost per contact. We computed marginal cost as the derivative of the median runtime with respect to the number of contacts.

We found that the median marginal runtime cost per contact for our Python implementation was roughly the same as MuJoCo's, see Fig. 4.4. We suspect that

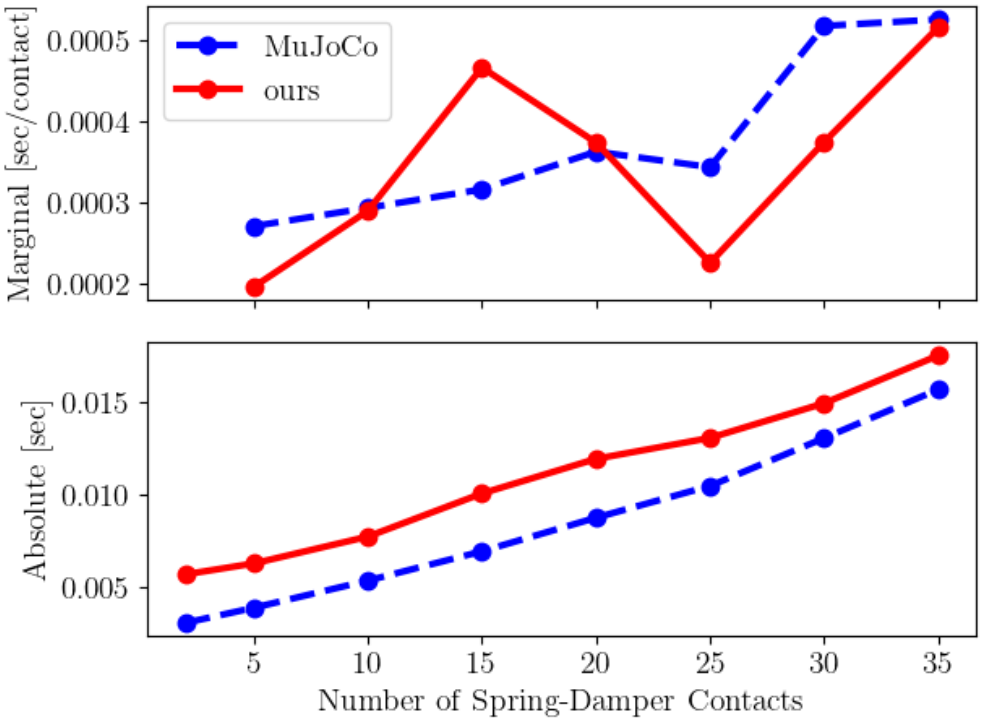

Figure 4.4: The marginal cost of our algorithm (red) and MuJoCo (dashed blue) for increasing number of spring-damper contacts. We integrated the same $N = 150$ random initial conditions for each number of contacts. Our algorithm implemented in pure Python has roughly the same median marginal cost per contact as MuJoCo. This suggests that our numerical method would easily outperform MuJoCo in a more optimized implementation.

the rapidly growing confidence interval on the runtime cost of our method is a result of our root finding implementation.

4.5 Discussion

We have presented, to our knowledge, the first event localization method using splines built only from mesh points of state that has a known error bound regardless of the integration method used. Using this method, we have developed a pure-Python numerical integration scheme for multi-contact that nearly meets MuJoCo's optimized codebase. MuJoCo is known to simulate robots faster than most other simulation suites [ETT15], and this makes our algorithm particularly well-suited to real-time applica-

tions, such as online optimization and model-predictive control of robots with 6 legs or more. We expect our algorithm could be more heavily exploited in contact-aware state estimation [Har+20b; PKJ22b]. By considering additional benign, physically-relevant mathematical structure, we have developed a numerical method that meets the state-of-the-art.

It is worth noting that ESS is more general than the types of hybrid problems MuJoCo and PyBullet treat [TET12; CB16; Bur+16], and our algorithm could find uses outside of robotics, such as in stability analysis of power grids [HP00], economics [Bro03], or in fields as far removed from engineering as ecology and neurobiology [Bur+16; Bro03].

Bibliography

[AG58] M.A. Aizerman and F.R. Gantmakher. "On the stability of periodic motions". In: *Journal of Applied Mathematics and Mechanics* 22.6 (1958), pp. 1065–1078. ISSN: 0021-8928. DOI: 10.1016/0021-8928(58)90033-9.

[AGI] Gilles Albeaino, Masoud Gheisari, and Raja R. A. Issa. "Drone-Related Deployment Limitations in Construction: A Research Roadmap". In: *Construction Research Congress 2022*, pp. 782–790. DOI: 10.1061/9780784483961.082. eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784483961.082. URL: https://ascelibrary.org/doi/abs/10.1061/9780784483961.082.

[AP97] M. Anitescu and F. A. Potra. "Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems". In: *Nonlinear Dynamics* 14.3 (Nov. 1997), pp. 231–247. ISSN: 1573-269X. DOI: 10.1023/A:1008292328909.

[BF10] Richard L. Burden and J. Douglas Faires. 9th. Boston: Brooks/Cole and Cengage Learning, 2010. ISBN: 0-538-73351-9.

[Bic00] A. Bicchi. "Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity". In: *IEEE Transactions on Robotics and Automation* 16.6 (2000), pp. 652–662. DOI: 10.1109/70.897777.

[Bro+06] B. Brogliato et al. “On the equivalence between complementarity systems, projected systems and differential inclusions”. In: *Systems & Control Letters* 55.1 (2006), pp. 45–51. ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2005.04.015.

[Bro03] B. Brogliato. “Some perspectives on the analysis and control of complementarity systems”. In: *IEEE Transactions on Automatic Control* 48.6 (2003), pp. 918–935. DOI: 10.1109/TAC.2003.812777.

[Bro16] Bernard Brogliato. 3rd ed. Communications and Control Engineering. Springer Cham, 2016. ISBN: 978-3-319-28664-8. DOI: 10.1007/978-3-319-28664-8.

[Bue+99] M. Buehler et al. “Stable open loop walking in quadruped robots with stick legs”. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 3. 1999, 2348–2353 vol.3. DOI: 10.1109/ROBOT.1999.770456.

[Bur+16] S A Burden et al. “Event-Selected Vector Field Discontinuities Yield Piecewise-Differentiable Flows”. In: *SIAM Journal of Applied Dynamical Systems* 15.2 (2016), pp. 1227–1267. DOI: 10.1137/15M1016588.

[Byr+95] Richard H. Byrd et al. “A Limited Memory Algorithm for Bound Constrained Optimization”. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: 10.1137/0916069.

[Car78] M.B. Carver. “Efficient integration over discontinuities in ordinary differential equation simulations”. In: *Mathematics and Computers in Simulation* 20.3 (1978), pp. 190–196. ISSN: 0378-4754. DOI: 10.1016/0378-4754(78)90068-X.

[CB16] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016.

[Cot79] Richard W. Cottle. "Numerical methods for complementarity problems in engineering and applied science". In: *Computing Methods in Applied Sciences and Engineering, 1977, I*. Ed. by R. Glowinski, J. L. Lions, and Iria Laboria. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 37–52. ISBN: 978-3-540-35411-6.

[CPS09] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Society for Industrial and Applied Mathematics, 2009. DOI: 10.1137/1.9780898719000.

[CRB22] George Council, Shai Revzen, and Samuel A. Burden. "Representing and Computing the B-Derivative of the Piecewise-Differentiable Flow of a Class of Nonsmooth Vector Fields". In: *Journal of Computational and Nonlinear Dynamics* 17.9 (June 2022), p. 091004. ISSN: 1555-1415. DOI: 10.1115/1.4054481.

[Dee] Deepmind Technologies. *MuJoCo Documentation*.

[EAS98] A Edelman, T A Arias, and S T Smith. "The geometry of algorithms with orthogonality constraints". In: *SIAM J Matrix Analysis and Applications* 20.2 (Oct. 1998), pp. 303–353. ISSN: 0895-4798.

[EKP01] Joel M. Esposito, Vijay Kumar, and George J. Pappas. "Accurate Event Detection for Simulating Hybrid Systems". In: *Hybrid Systems: Computation and Control*. Ed. by Maria Domenica Di Benedetto and Alberto Sangiovanni-Vincentelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 204–217. ISBN: 978-3-540-45351-2. DOI: 10.1007/3-540-45351-2_19.

[Enr+86] W. H. Enright et al. “Interpolants for Runge-Kutta formulas”. In: *ACM Trans. Math. Softw.* 12.3 (Sept. 1986), pp. 193–218. ISSN: 0098-3500. DOI: 10.1145/7921.7923.

[Enr+88] W.H. Enright et al. “Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants”. In: *Applied Mathematics and Computation* 27.4 (1988), pp. 313–335. ISSN: 0096-3003. DOI: 10.1016/0096-3003(88)90030-6.

[ETT15] Tom Erez, Yuval Tassa, and Emanuel Todorov. “Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4397–4404. DOI: 10.1109/ICRA.2015.7139807.

[F124] F1. *Beginner’s Guide to F1 tyres*. Dec. 2024. URL: https://www.formula1.com/en/latest/article/the-beginners-guide-to-formula-1-tyres.6

[FAÅ14] Emil Fredriksson, Christian Andersson, and Johan Åkesson. “Discontinuities handled with events in Assimulo, a practical approach”. In: *Proceedings of the 10th International Modelica Conference*. Ed. by Hubertus Tummescheit and Karl-Erik ´Arzèn. Link´oping Electronic Conference Proceedings 96, Mar. 2014, pp. 827–836. DOI: 10.3384/ecp14096827.

[Faz+17] Nima Fazeli et al. “Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact”. In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1437–1454. DOI: 10.1177/0278364917698749.

[Fil88] A. F. Filippov. “Existence and General Properties of Solutions of Discontinuous Systems”. In: *Differential Equations with Discontinuous Right-*

hand Sides. Ed. by F. M. Arscott. Dordrecht: Springer Netherlands, 1988, pp. 48–122. ISBN: 978-94-015-7793-9. DOI: 10.1007/978-94-015-7793-9_3.

[For95] JA Ford. "Improved algorithms of illinois-type for the numerical solution of nonlinear equations". In: *University of Essex, Department of Computer Science* (1995).

[GJ19] Paul Glendinning and Mike R. Jeffrey. Ed. by Elena Bossolini, J. Tomàs Lázaro, and Josep M. Olm. 1st ed. Advanced Courses in Mathematics - CRM Barcelona. Birkhäuser Cham, 2019. ISBN: 978-3-030-23689-2. DOI: 10.1007/978-3-030-23689-2.

[GK05] Gerald Grabner and Andrés Kecskeméthy. "An Integrated Runge–Kutta Root Finding Method for Reliable Collision Detection in Multibody Systems". In: *Multibody System Dynamics* 14.3 (Nov. 2005), pp. 301–316. ISSN: 1573-272X. DOI: 10.1007/s11044-005-2640-6.

[GS09] Robert D. Gregg and Mark W. Spong. "Bringing the compass-gait bipedal walker to three dimensions". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 4469–4474. DOI: 10.1109/IROS.2009.5354583.

[GST09] R Goebel, R Sanfelice, and A Teel. "Hybrid dynamical systems". In: *IEEE Control Systems* 29.2 (2009), pp. 28–93. ISSN: 1066-033X. DOI: 10.1109/MCS.2008.931718.

[Har+20a] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[Har+20b] Ross Hartley et al. "Contact-aided invariant extended Kalman filtering for robot state estimation". In: *The International Journal of Robotics Research* 39.4 (2020), pp. 402–430. DOI: 10.1177/0278364919894385.

[HP00] I.A. Hiskens and M.A. Pai. "Trajectory sensitivity analysis of hybrid systems". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 47.2 (2000), pp. 204–220. DOI: 10.1109/81.828574.

[Hut+16] Marco Hutter et al. "ANYmal - a highly mobile and dynamic quadrupedal robot". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 38–44. DOI: 10.1109/IROS.2016.7758092.

[HW96] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*. 2nd ed. Springer Berlin, Heidelberg, 1996. ISBN: 978-3-642-05221-7. DOI: 10.1007/978-3-642-05221-7.

[HWN93] Ernst Hairer, Gerhard Wanner, and Syvert P. Nørsett. *Solving Ordinary Differential Equations I*. 2nd ed. Springer Berlin, Heidelberg, 1993. ISBN: 978-3-540-78862-1. DOI: 10.1007/978-3-540-78862-1.

[Koe+15] J. Koenemann et al. "Whole-body model-predictive control applied to the HRP-2 humanoid". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 3346–3351. DOI: 10.1109/IROS.2015.7353843.

[Kon+21] Nathan J. Kong et al. "The Salted Kalman Filter: Kalman filtering on hybrid dynamical systems". In: *Automatica* 131 (2021), p. 109752. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2021.109752.

[Kon+24] Nathan J. Kong et al. "Saltation Matrices: The Essential Tool for Linearizing Hybrid Dynamical Systems". In: *Proceedings of the IEEE* 112.6 (2024), pp. 585–608. DOI: 10.1109/JPROC.2024.3440211.

[Lee+18] Jeongseok Lee et al. "DART: Dynamic Animation and Robotics Toolkit". In: *Journal of Open Source Software* 3.22 (2018), p. 500. DOI: 10.21105/joss.00500.

[Löt82] Per Lötstedt. "Mechanical Systems of Rigid Bodies Subject to Unilateral Constraints". In: *SIAM Journal on Applied Mathematics* 42.2 (1982), pp. 281–296. DOI: 10.1137/0142022.

[Lyg+01] J. Lygeros et al. "Continuity and invariance in hybrid automata". In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*. Vol. 1. 2001, 340–345 vol.1. DOI: 10.1109/CDC.2001.980123.

[Mor88] J. J. Moreau. "Unilateral Contact and Dry Friction in Finite Freedom Dynamics". In: *Nonsmooth Mechanics and Applications*. Ed. by J. J. Moreau and P. D. Panagiotopoulos. Vienna: Springer Vienna, 1988, pp. 1–82. ISBN: 978-3-7091-2624-0. DOI: 10.1007/978-3-7091-2624-0_1.

[ORe70] P. G. O'Regan. "Step Size Adjustment at Discontinuities for Fourth Order Runge-Kutta Methods". In: *The Computer Journal* 13.4 (Nov. 1970), pp. 401–404. ISSN: 0010-4620. DOI: 10.1093/comjnl/13.4.401.

[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*. HSCC '17. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2017, pp. 243–252. ISBN: 9781450345903. DOI: 10.1145/3049797.3049807. URL: https://doi.org/10.1145/3049797.3049807.

[PCT14] Michael Posa, Cecilia Cantu, and Russ Tedrake. "A direct method for trajectory optimization of rigid bodies through contact". In: *The In-*

ternational Journal of Robotics Research 33.1 (2014), pp. 69–81. DOI: 10.1177/0278364913506757.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PKJ22a] J. Joe Payne, Nathan J. Kong, and Aaron M. Johnson. "The Uncertainty Aware Salted Kalman Filter: State Estimation for Hybrid Systems with Uncertain Guards". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 8821–8828. DOI: 10.1109/IROS47612.2022.9981218.

[PKJ22b] J. Joe Payne, Nathan J. Kong, and Aaron M. Johnson. "The Uncertainty Aware Salted Kalman Filter: State Estimation for Hybrid Systems with Uncertain Guards". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 8821–8828. DOI: 10.1109/IROS47612.2022.9981218.

[Pre00] L. Prechelt. "An empirical comparison of seven programming languages". In: *Computer* 33.10 (2000), pp. 23–29. DOI: 10.1109/2.876288.

[PTT16] Michael Posa, Mark Tobenkin, and Russ Tedrake. "Stability Analysis and Control of Rigid-Body Systems With Impacts and Friction". In: *IEEE Transactions on Automatic Control* 61.6 (2016), pp. 1423–1437. DOI: 10.1109/TAC.2015.2459151.

[RBS10] C. David Remy, Keith Buffinton, and Roland Siegwart. "Stability Analysis of Passive Dynamic Walking of Quadrupeds". In: *The International Journal of Robotics Research* 29.9 (2010), pp. 1173–1185. DOI: 10.1177/0278364909344635. eprint: https://doi.org/10.1177/0278364909344635. URL: https://doi.org/10.1177/0278364909344635.

[RR23] M Raff and C. D. Remy. "Hard vs. Soft Contacts in Trajectory Optimization of Legged Robots: An Example with a Compliant Rubber Foot". In: (2023). URL: https://openreview.net/forum?id=k3O1RL0v_x.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. 55 Hayward Street, Cambridge, MA 02142: The MIT Press, 2006. ISBN: 026218253X.

[Sch12] S Scholtes. *Introduction to Piecewise Differentiable Equations*. SpringerBriefs in Optimization. Springer New York, 2012. ISBN: 978-1-4614-4340-7. DOI: 10.1007/978-1-4614-4340-7.

[SGB91] L. F. Shampine, I. Gladwell, and R. W. Brankin. "Reliable solution of special event location problems for ODEs". In: *ACM Trans. Math. Softw.* 17.1 (Mar. 1991), pp. 11–25. ISSN: 0098-3500. DOI: 10.1145/103147.103149.

[Sha85] Lawrence F. Shampine. "Interpolation for Runge-Kutta Methods". In: *SIAM Journal on Numerical Analysis* 22.5 (1985), pp. 1014–1027. ISSN: 00361429. (Visited on 08/06/2024).

[ST00a] L.F. Shampine and S. Thompson. "Event location for ordinary differential equations". In: *Computers & Mathematics with Applications* 39.5 (2000), pp. 43–54. ISSN: 0898-1221. DOI: 10.1016/S0898-1221(00)00045-6.

[ST00b] D. Stewart and J.C. Trinkle. "An implicit time-stepping scheme for rigid body dynamics with Coulomb friction". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. 2000, 162–169 vol.1. DOI: 10.1109/ROBOT.2000.844054.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Con-*

ference on Intelligent Robots and Systems. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[TT10] Y. Tassa and E. Todorov. "Stochastic Complementarity for Local Control of Discontinuous Dynamics". In: *Proceedings of Robotics: Science and Systems*. Zaragoza, Spain, June 2010. DOI: 10.15607/RSS.2010.VI.022.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[WGK03] ER Westervelt, JW Grizzle, and DE Koditschek. "Hybrid zero dynamics of planar biped walkers". In: *IEEE Trans Automat Contr* 48.1 (2003), pp. 42–56. DOI: 10.1109/TAC.2002.806653.

[Wil+16] Grady Williams et al. "Aggressive driving with model predictive path integral control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[WM92] Yu Wang and Matthew T. Mason. "Two-Dimensional Rigid-Body Collisions With Friction". In: *Journal of Applied Mechanics* 59.3 (Sept. 1992), pp. 635–642. ISSN: 0021-8936. DOI: 10.1115/1.2893771.

[Zha+21] Zhizheng Zhang et al. "Asynchronous Episodic Deep Deterministic Policy Gradient: Toward Continuous Control in Computationally Complex Environments". In: *IEEE Transactions on Cybernetics* 51.2 (2021), pp. 604–613. DOI: 10.1109/TCYB.2019.2939174.