Increasing Reliability of Legged Robots in the Presence of Uncertainty

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Nathan J. Kong

B.S., Mechanical Engineering, University of Minnesota

Carnegie Mellon University

Pittsburgh, PA

August 2022

Acknowledgements

I would like to acknowledge the funding that has made this research possible from the U.S. Army Research Office under grant #W911NF-19-1-0080. Some of this work was done in collaboration with #IIS-1704256 and #ECCS-1924723 from the National Science Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office, National Science Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. Thank you to XPeng Robotics and Xingye Da in particular for inviting me to work on parts of my thesis with them as well as helping me run my experiments.

I would like to thank my advisor, the chair of this committee, Aaron Johnson, for encouraging me to explore my different ideas and guiding me throughout each of them, and the rest of my committee members, Mark Bedillion, Zachary Manchester, and Michael Posa, for their help throughout this process. In addition, I would like to thank my coauthors Joe Payne, James Zhu, George Council, Chuanzheng Li, and Sam Burden. Thank you to everyone from the Robomechanics lab for all their helpful discussions and making my PhD an enjoyable experience. I am thankful to my undergraduate advisor, Timothy Kowalewski, and Trevor Stephens for introducing me to robotics and teaching me how to conduct research early in my career.

Thank you to my family and friends for supporting me. Especially, thank you to my soon to be wife, Cynthia, for always being my number one fan and encouraging me along the way – without her, this thesis would probably not exist.

Abstract

Legged robots have the potential to traverse a wide variety of environments, but are currently too unreliable to use in mission critical settings. A major factor that hinders the reliability of legged robots is the hybrid dynamics that arises when their legs make varying contact with the environment. The discontinuities introduced by hybrid dynamics interfere with traditional tracking, planning, and state estimation strategies. This thesis presents several novel ideas and tools in overcoming these difficulties: creating robust trajectories through optimally convergent planning, a tutorial on the saltation matrix (the update to the sensitivity equation for hybrid transitions), Kalman filtering on hybrid systems, iterative Linear Quadratic Regulator for hybrid systems, and a model predictive controller which can continuously update the current plan given new information.

Optimally convergent planning creates trajectories that are robust to state uncertainty in undersensed and underactuated systems. Convergent planning utilizes ideas from contraction analysis and minimizes divergence to find trajectories that naturally shrink state uncertainty. This optimization framework is validated for an undersensed hill navigation problem as well as an underactuated rotary cart pole incline.

The saltation matrix tutorial provides the necessary information to get started implementing smooth tools for hybrid systems with event-triggered transitions. The tutorial contains a survey of where the saltation matrix is commonly used, as well as expanded proofs for deriving the saltation matrix. We also show an example saltation matrix calculation for a simple rigid body system and show that vital information is lost when not using the saltation matrix, and we calculate the saltation matrix for a generalized rigid body system with unilateral constraints.

The Kalman filter is a widely used optimal state estimation algorithm. We extended the Kalman filter to hybrid systems by creating the "Salted Kalman Filter" which allows hybrid transitions to occur during the a priori and a posteriori updates and by using linearizations about a hybrid transition in order to propagate uncertainty belief. The Salted Kalman Filter is compared against a version of the algorithm but with a naive method of linearization about hybrid transitions. The Salted Kalman Filter is also validated by comparing it against a hybrid particle filter benchmark.

Iterative Linear Quadratic Regulator (iLQR) is a trajectory optimization algorithm that uses locally linear models of the dynamics and uses a quadratic cost function. We extended iterative Linear Quadratic Regulator to hybrid systems with the following additions: allowing for hybrid transitions on the forward pass, handling mode mismatches with extensions on reference trajectories, and using the linearization about a hybrid transition on the backward pass. We validate the hybrid iterative Linear Quadratic Regulator on a variety of hybrid systems and show that the algorithm can optimally choose contact timing and placements.

We utilize hybrid iLQR as a Model Predictive Controller (MPC) in order to replan in realtime. By replanning in real-time, the robot will be more robust to unplanned, large disturbances because the controller can generate a new plan instead of rigidly tracking a reference trajectory. We validated the MPC on a quadruped robot both in simulation and on the real robot by applying disturbances to the robot, and we also compared the MPC against an MPC that uses simplified robot dynamics.

Overall, this thesis makes legged robots more reliable by creating a robust global plan and by reactively replanning in real time to deal with local disturbances such as contact timing errors or unplanned slips. Robustness to the global plan can be achieved through planning convergent trajectories. To alleviate the complexity of hybrid transitions, we heavily utilize linearizations to enable reactive replanning and fast state estimation through hybrid iLQR MPC and the Salted Kalman Filter.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robots are on the verge of being widely utilized in society for tasks such as home assistance, environmental monitoring, and exploration. Currently, robots are used mainly in well-controlled settings like factories and warehouses but are not widely used in "real world" tasks. Planning and control for real world robotic systems is difficult because they are often plagued with uncertainty in state and modeling parameters. These issues are further exacerbated when robots make varying contacts with their surroundings (such as in the case of robotic manipulation or legged locomotion) due to the complexities that arise from the discontinuous nature of impact. This is because small bumps can lead to drastically different outcomes, like falling, whereas for smoother systems, small disturbances lead to small differences in outcomes.

Legs offer more adaptability to a given environment, which is important given the unpredictability of the terrain. However, in mission critical scenarios, wheels or tracks are often used instead of legs because they are much more reliable; e.g. rovers are fitted with wheels even though legs could possibly broaden the searchable area on Mars. In order to increase the reliability of legged robots, they must be able to plan robust global trajectories and be able to efficiently reason about the hybrid dynamics (systems where the dynamics make discontinuous jumps, Def. 1) that occur

when making contact. By reasoning about these hybrid events, robots can effectively plan, control, and estimate around these contacts so that small perturbations do not lead to failure.

When planning trajectories, it is important to consider not only the amount of energy being used, but also how robust the trajectory is to uncertainty [Kong and Johnson, 2019]. This is especially important for systems that are underactuated and undersensed because uncertainty may not be able to be collapsed through closed-loop control alone. Planning for contact systems, in particular, is made more difficult by the drastic differences in dynamics between two configurations that are close to each other. For example, when a robot's foot is slightly off the ground, the robot cannot apply any forces, while when the foot is slightly lower and touches the ground, the robot can apply the maximum amount of force that the actuators can provide. This discontinuity disrupts standard planning algorithms because they generally depend on solutions that are close to each other, yielding close outputs. A popular optimization strategy is to use contact implicit trajectory optimization [Posa et al., 2014; Mordatch et al., 2012] where each state and input along a trajectory can be in a different contact mode as long as the distance between the rigid bodies are zero when contact forces are being applied (complementarity constraints). However, because these constraints are so discontinuous, it is very difficult for these methods to find solutions quickly and reliably. Because of these pitfalls, optimization methods which use these complementarity constraints alongside the full nonlinear dynamics are not suitable for real time use on systems with mechanical time scales. However, they can be used to create a library of offline behaviors, but the behaviors might not always be adaptable to real scenarios due to uncertainty [Bjelonic et al., 2022]. Also, the trajectories produced by standard trajectory optimization methods tend to be brittle and difficult to track. Tracking issues are further exacerbated for hybrid trajectories.

A common strategy to track trajectories is to use a high gain feedback controller. High gain feedback works well for fully controllable smooth systems because any nonlinearities can be canceled out, but does not always work for underactuated systems especially in the case of legged systems. Tracking using high gain feedback is not reliable for legged systems and hybrid systems in general because perturbations can lead to different contact modes from the reference, which can

lead to both jumps in error signals and differing control authority [Mason et al., 2016]. In the case of legged systems, mismatches in the contact mode can lead to a leg attempting to apply downward force when the leg is not in contact with the ground, which can cause the robot to slip or fall.

This issue of differing hybrid modes is a common issue in tracking for hybrid systems [Biemond et al., 2012], [Pagilla and Yu, 2004], and [Forni et al., 2013]. For contact systems, the issue of mismatching errors when the contact mode is different from the reference has been studied in [Saccon et al., 2014], where they propose to define a new local error signal that projects the reference back to the current mode. For legged robots, another common strategy is to ignore these mismatches and treat the environment as disturbances to the controller [Xie et al., 2018]. However, these two methods are fairly local and if contact timing or foot placement is perturbed far enough, these methods may become unreliable. Control near contact is sometimes simplified by lowering the magnitude of the input and gains near impact times [Yang and Posa, 2021b]. Another strategy is to move dynamically until close to an object and then move quasistatically while interacting with the object in order to reduce the uncertainty and remove the dynamics [Gibo et al., 2009]. However, these strategies will not work well for a legged system trying to maximize the amount of energy that can be put into the system when executing dynamic behaviors.

In order to successfully track legged behaviors through large contact timing mismatches, trajectories need to be replanned/modified online in order to incorporate these mismatches in contact timing. Also, the planned trajectories for underactuated or undersensed systems need to not just optimize the cost of transport but also consider the effect of the dynamics on uncertainty.

A popular framework for continuous online re-planning is called Model Predictive Control (MPC), where a new trajectory is created each control loop cycle in order to bring the current state of the robot closer to the reference. Often, a simpler model is used because the full-order dynamics are highly nonlinear due to the hybrid dynamics and are too slow to solve with standard methods, as shown in [Di Carlo et al., 2018] and [Lee et al., 2020]. For example, [Di Carlo et al., 2018] represents a quadruped model with a floating robot body with no legs, and the inputs are forces applied at the hips. However, one issue with simplified dynamics is that important information can be lost with

simplification, which can limit the explorable space of the robot. For full order dynamic planning, smooth iterative Linear Quadratic Regulator (iLQR) – a trajectory optimization algorithm which uses locally linear models of the dynamics and uses a quadratic cost function – has been used in the past for MPC but does not directly extend to legged systems because of the discontinuities that arise from hybrid dynamics. However, even though there are discontinuities, these hybrid transitions can still be linearized about, and therefore can provide the necessary variational information to algorithms like iLQR.

Linearizations about hybrid events are also useful for state estimation algorithms, because some methods already use linearizations in the smooth case to update covariances. This is especially important for state estimation algorithms like the Kalman filter (a standard fast state estimation algorithm for smooth systems), which requires the knowledge of the covariance at each timestep and the discontinuities introduced by contact severely disrupt the filter's performance. Methods like a hybrid particle filter [Koutsoukos et al., 2002; Koval et al., 2015b] (which keeps a nonparametric belief about the distribution and not just the covariance) are able to estimate the state well through hybrid transitions, but are extremely inefficient and cannot be run in real time. Having a real time, accurate estimate of the robot's state is crucial for reliably controlling them. Without an accurate state estimate at each control loop, the system will not be able to track well and could ultimately be unstable.

In this work, we show that by linearizing about hybrid events and reasoning about uncertainty, we can make planning and estimation for legged robots more efficient, which will in turn enable reactive replanning online. Being able to reactively replan trajectories online makes legged robots more robust to uncertainty and ultimately more reliable in mission critical scenarios.

1.2 Overview

Chapter 2 shows that, by taking advantage of the underlying dynamics of the system, we can produce trajectories that minimize the effect that state uncertainty has on undersensed and underactuated

systems. In this work, we create a trajectory optimization framework that minimizes divergence metrics, as well as regularizes smoothness and length of trajectories, which results in more robust trajectories to state uncertainty when compared with non-optimal planning methods such as RRT and its optimal variant RRT* for an undersensed hill navigation system [Kong and Johnson, 2019]. In Section 2.6, we apply the same idea to an underactuated swing-up behavior of a rotary cart pole. We show that perturbations applied along a convergent swing-up trajectory has less of an effect on the swing-up behavior than minimizing just for energy usage. We include physical experiments to validate the feasibility of the trajectories.

To address the complexity that hybrid dynamics adds to planning, control, and state estimation, this work uses the linearization about a piecewise smooth hybrid event called a "saltation" matrix (Def. 2) to unlock algorithms such as iLQR and Kalman filtering for these hybrid systems. In Chapter. 3, we give a tutorial on deriving the saltation matrix, showing how to use it, and analyzing saltation matrices for rigid body systems with unilateral constraints [Kong et al., 2022b].

In Chapter 4, we extend Kalman filtering to hybrid systems for fast state estimation [Kong et al., 2021b]. We validate the Kalman filter with a hybrid particle filter for a simple constant flow system and a asymmetric spring loaded inverted pendulum. We also compare using a naive method (using the Jacobian of the reset map) instead of the saltation matrix, and show that the estimation error significantly increases when the saltation matrix is not used.

We also investigate the dual problem of planning and control for hybrid systems, where instead of propagating covariance, we are now update the approximation of the value function through hybrid transitions. This is especially useful for LQR and iLQR, where the approximation of the value function is utilized to compute optimal gains. In Chapter 5, we demonstrate a hybrid iLQR algorithm which 1) updates the forward pass to allow for hybrid transitions 2) uses reference extensions when tracking with mode mismatches and 3) updates the backwards pass using the saltation matrix when the forward pass makes a hybrid transition [Kong et al., 2021a].

Hybrid iLQR can efficiently plan nonlinear hybrid dynamics and vary contact time due to the iterative nature of the algorithm. However, one drawback of Hybrid iLQR is that it is more

prone to local minima because it is a shooting method. This issue can be partially mitigated by using Hybrid iLQR as a model predictive controller where we can seed the solver with a reference behavior when tracking it. By using Hybrid iLQR in a receding horizon fashion, we can create stabilizing behaviors for large perturbations by replanning trajectories including the foot step timing and location to handle them. In Chapter 6, we extend Hybrid iLQR to model predictive control by applying hybrid fixes similar to those in Chapter 5 to the tracking problem and using several software packages for efficient rigid body computations [Kong et al., 2022a]. We compare against a popular method that utilizes a fixed gait sequence, heavy linearizations, and simplifications to the robot dynamics model and find that using the full nonlinear dynamics and allowing for changes in the contact sequence leads to more robust behaviors.

Finally, we conclude in Chapter 7 where we discuss overall insights gained from this thesis and discuss future extensions for the completed work.

Chapter 2

Contraction analysis Convergent Planning

The content of this chapter appeared previously in [Kong and Johnson, 2019].

2.1 Abstract

This paper investigates optimization-based planning methods for generating trajectories which are robust to state uncertainty in undersensed and underactuated systems. Specifically, these methods are applied to an undersensed robotic hill climbing system. In previous work, divergence metrics based on contraction analysis were used to quantify robustness of a trajectory to state uncertainty in conjunction with a kinodynamic RRT planner to guide the planner towards more convergent directions. Resulting trajectories were sub-optimal or needed to be smoothed prior to implementation. This work proposes an optimization framework to plan optimally robust and smooth trajectories which can also be readily implemented on the robotic hill climbing problem. A new hill climbing controller is also presented which can guarantee for the first time the strongest result of contraction analysis, global asymptotic convergence, where possible. Trajectories created using the new trajectory optimization framework and hill controller are shown to be smoother and more robust than previous methods as well as an asymptotically optimal versions of previous methods.

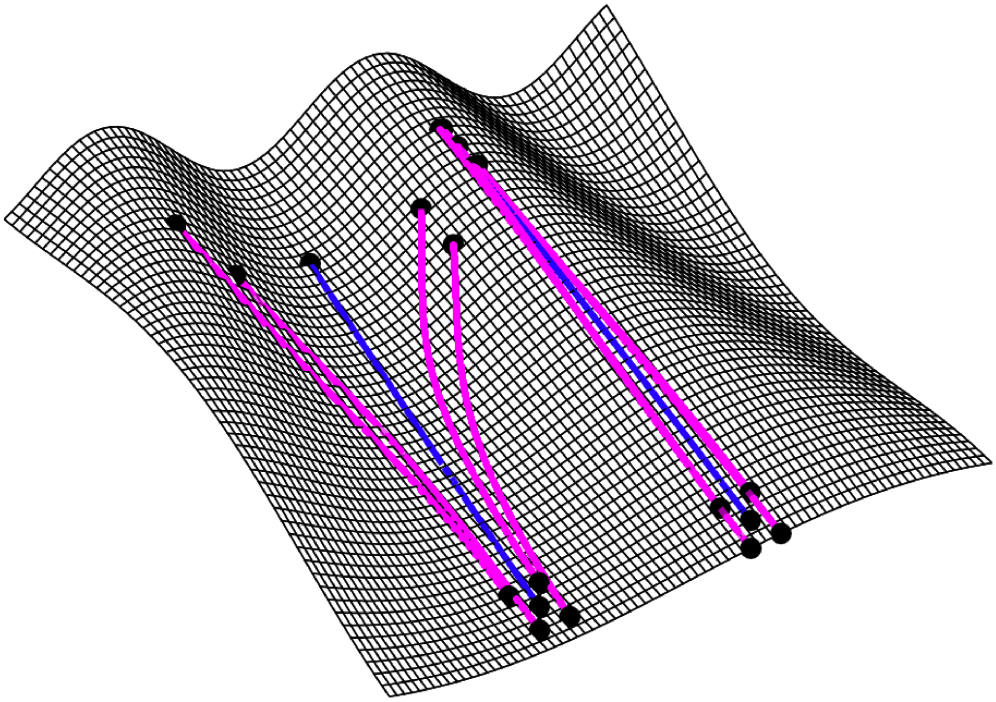

Figure 2.1: Diverging (left) versus converging (right) trajectories. The nominal trajectory is shown in (blue), and 4 points of state uncertainty (black) are simulated forwards (magenta).

2.2 Introduction

Consider a mobile robot on hilly terrain that has precise heading control and an accurate map. The robot is tasked to traverse from its current state to a goal region, as in [Johnson et al., 2011; Ilhan et al., 2018]. However, the robot may not be well localized, and has only a rough understanding of where it is currently located on the hill, e.g. in GPS-denied settings. If a robot path planner only considers the shortest path to the goal region, the resulting trajectory may diverge due to continuous growth of uncertainty, as in the left side of Fig. 2.1. However, in some regions the shortest path will lead to convergent behaviors, as in the right side of Fig. 2.1, where the uncertainty shrinks over time.

In many robotic applications, state uncertainty growth is dealt with by using closed-loop feedback. A controller can be used to sense and reduce errors in the state. However, robotic systems are often undersensed or underactuated and cannot close the loop on certain state uncertainties. In these cases they must rely on planning robust behaviors through leveraging the underlying geometry

of their dynamics. This issue is especially prevalent for the hill climbing robot example in [Johnson et al., 2011] and planning for robust manipulator pushing motions [Koval et al., 2015a], but arises in many more general settings.

Contraction analysis [Lohmiller and Slotine, 1998] considers robustness to state uncertainty through geometric analysis, and provides a proof for global asymptotic convergence for dynamical systems with a vector field with a strictly negative Jacobian, a contraction region. This states that if all the eigenvalues of the Jacobian are strictly negative in a region, all trajectories converge to a single trajectory and the resulting behavior is robust to state uncertainty because there is global asymptotic convergence.

To plan for robust trajectories, *convergent planning* [Johnson et al., 2016b] utilizes a kinodynamic rapidly-exploring random tree (RRT) [LaValle and Kuffner Jr, 2001] which is biased towards behaviors that are on average convergent. Average convergence is defined in [Johnson et al., 2016b] to be trajectories where the average eigenvalue of the vector field's Jacobian to be negative instead of the maximum eigenvalue being negative in the case of strict contraction. This corresponds to contraction down to a set of zero volume, instead of a single point [Lohmiller and Slotine, 1998]. In [Johnson et al., 2016b] these trajectories are created for both robotic hill climbing and manipulator pushing motions. Another RRT method which may be modified to include directions of divergence to reduce state uncertainty is the Vector Field RRT (VF-RRT) [Ko et al., 2013] algorithm which uses the vector field of the system to minimize upstream cost. However, since these trajectories are created through an RRT, they are not optimal. Asymptotically near optimal trajectories are created in [Liu et al., 2019] by modifying the average divergence metric used in [Johnson et al., 2016b] to have optimality guarantees, and more optimal RRT methods include multiple restarts RRT (C-MRRT) and stable sparse RRT (C-SSRT) [Li et al., 2016].

One issue with trajectories produced by RRT is that they are often not smooth. In [Liu et al., 2019] it is noted that the near optimal trajectories produced by C-MRRT and C-SRRT can be post-processed to feasibly work on a robot, because optimal and smooth RRT methods such as RRT* [Karaman and Frazzoli, 2010] are not suitable in these contexts. This is because a steering function

may not always be available for kinodynamic planning, which is required for RRT* to rewire the tree to improve solution quality [Liu et al., 2019]. However, if the near optimal trajectories are post-processed, they will no longer be optimal and may no longer satisfy contraction constraints. In this paper, RRT* is implemented and compared against previous methods because there exists a steering function for this specific problem. Although these trajectories are asymptotically optimal and more directly usable than RRT methods, optimality is only guaranteed as time approaches infinity.

This paper presents a new trajectory optimization (T-OPT) framework based on [Hargraves and Paris, 1987] that can produce optimally convergent and smooth trajectories which can readily be implemented on a robot and converge to an optimal solution in finite time. The framework uses analytical calculations of divergence and also includes options to create trajectories which guarantee asymptotic convergence or average convergence. In addition to independently creating optimal trajectories which can satisfy constraints, the proposed T-OPT framework can post-process trajectories generated using RRT methods while maintaining optimality and specified constraints.

In the context of the hill climbing example, both [Johnson et al., 2016b] and [Liu et al., 2019] do not consider the case of strict contraction, and only considers an estimate of average convergence. In fact, the hill climbing controller used can never guarantee strict convergence and at best only guarantee average convergence to a set of zero volume. To resolve this issue, a new hill climbing controller that varies speed as a function of local hill steepness is proposed in this paper. This change allows the guarantee of strict convergence, and ultimately increases the robustness of the controller to state uncertainty.

2.2.1 Related Work

This work focuses on planning trajectories where state uncertainty is expected. One popular method of planning under state uncertainty is to formulate the planning problem as a partially observable Markov decision process (POMDP) [Kaelbling et al., 1998]. However, POMDP solvers do not perform well when the action space is continuous and do not scale well in higher dimensions.

Probabilistic conformant planning is another method to plan for state uncertainty. The goal of the planner is to maximize the likelihood of success given an expectation of state uncertainty [Lozano-Perez et al., 1984; Hyafil and Bacchus, 2003]. Although conformant planning is useful to plan robust trajectories in some contexts, it requires an accurate model of state uncertainty and doesn't utilize sensor feedback to increase robustness. The methods proposed here do not require a model of the state uncertainty, and instead try to continually reduce whatever uncertainty there is.

The hill climbing problem used in this paper is also used in [Johnson et al., 2016b; Liu et al., 2019; Johnson et al., 2011; Ilhan et al., 2018], where a robot is tasked to navigate from a start position to a goal region using the gradient of the hill as feedback. The hill gradient controller in these examples were inspired by [Arkin, 1992]. This paper analyzes contraction properties of this controller and proposes modifications to create a contraction region, satisfying the strongest results from contraction analysis [Lohmiller and Slotine, 1998].

2.3 Methods

2.3.1 Contraction Analysis and Divergence Metrics

Converging behaviors of a dynamical system can be analyzed through contraction analysis [Lohmiller and Slotine, 1998]. Consider a dynamical system with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, control input $u \in \mathcal{U} \subseteq \mathbb{R}^m$, and a continuously differentiable vector field $f(x, u, t)$ such that $f : \mathcal{X} \times \mathcal{U} \times \mathbb{R} \to T\mathcal{X}$. Define F as the symmetric part of the Jacobian of f [Lohmiller and Slotine, 1998, Definition 1],

$$F(x, u, t) = \frac{1}{2}\left(\frac{\partial f(x, u, t)}{\partial x} + \frac{\partial f(x, u, t)}{\partial x}^T\right) \tag{2.1}$$

Now consider two neighboring trajectories in the vector field f. Define δx to be the virtual displacement between the neighboring trajectories, and the squared distance to be $\delta x^T \delta x$. The rate of change of $\delta x^T \delta x$ is bounded by the max eigenvalue $\lambda_{max}(x, u, t)$ of the symmetric part of the

Jacobian F,

$$\frac{\delta}{\delta t}(\delta x^T \delta x) \leq 2\lambda_{max}(x, u, t)\delta x^T \delta x \tag{2.2}$$

This implies that the magnitude of the virtual displacement δx can also be bounded by the max eigenvalue $\lambda_{max}(x, u, t)$ [Lohmiller and Slotine, 1998, Eqn. 3],

$$\|\delta x\| \leq \|\delta x_0\| e^{\int_0^t \lambda_{max}(x,u,t)dt} \tag{2.3}$$

If λ_{max} is uniformly negative definite ($\exists \beta > 0, \forall t \geq 0, \lambda_{max}(x, u, t) \leq -\beta < 0$), then any infinitesimal length $\|\delta x\|$ converges exponentially to zero [Lohmiller and Slotine, 1998, Thm. 1]. Similarly, consider a differential volume δV around the trajectory, and the evolution of δV is defined as [Lohmiller and Slotine, 1998, Sec. 3.9],

$$\|\delta V\| = \|\delta V(t_0)\| e^{\int_0^t \mathrm{div} f(x,u,t)dt} \tag{2.4}$$

As a relaxation to the maximum eigenvalue λ_{max} being uniformly negative definite, [Lohmiller and Slotine, 1998] considers the case where the average eigenvalue λ_{avg} is uniformly negative definite. Since the divergence of f is just the sum of the eigenvalues in F [Lohmiller and Slotine, 1998], then if the average eigenvalue λ_{avg} is uniformly negative definite then so is the divergence of f. This implies that the magnitude of a differential volume δV converges exponentially to zero if λ_{avg} is uniformly negative definite.

These results motivated the creation of two divergence metrics in [Johnson et al., 2016b]: the maximal divergence metric $D_m := \lambda_{max}$ and the average divergence metric $D_a := \lambda_{avg}$. If D_m is uniformly negative definite for an entire trajectory, then all neighboring trajectories converge to a single trajectory [Lohmiller and Slotine, 1998, Thm. 1]. If D_a is uniformly negative definite for an entire trajectory, then on average all neighboring trajectories converge to a set of zero volume [Lohmiller and Slotine, 1998, Sec. 3.9].

2.3.2 Hill Climbing Problem

The well established hill climbing problem [Johnson et al., 2011; Ilhan et al., 2018; Johnson et al., 2016b; Liu et al., 2019] considers a mobile robot navigating hilly terrain as shown in Fig. 2.1. Since contraction analysis [Lohmiller and Slotine, 1998] requires a smooth system, the hill is modeled as a smooth height function, $h(x, y)$.

The hill gradient controller from [Johnson et al., 2016b] follows a constant speed α while choosing any arbitrary angle θ relative to the hill gradient ∇h,

$$f(x, y, \theta) = \alpha R(\theta) \frac{\nabla h(x, y)}{||\nabla h(x, y)||}, \tag{2.5}$$

where $R(\theta)$ is a rotation matrix. In [Johnson et al., 2016b], it is stated that one eigenvalue of the vector field Jacobian is zero. Therefore, the maximum divergence is equal to the average divergence, $D_m = D_a$, or equal to 0, $D_m = 0$. However, this is not entirely correct. While it is true that there is always some direction that is neither converging or diverging, it is possible that one eigenvalue is positive and one eigenvalue is negative. The controller can never have eigenvalues with the same sign, and thus the main conclusion that the controller can never have a negative max eigenvalue is still valid.

Lemma 1. *Given a vector field f defined by* (2.5) *and obtaining F through* (2.1)*, the maximum eigenvalue of F, λ_{max}, is non-negative, $\lambda_{max} \geq 0$.*

The proof is included in Lemma 2.

2.3.3 Power Controller

Define a new controller where the gradient is scaled by its norm to the p^{th}-power,

$$f(x, y, \theta, p) = \alpha R(\theta) \frac{\nabla h}{||\nabla h||^p}, \quad p \in \mathbb{R} \tag{2.6}$$

This new controller changes speed with the local steepness of the hill, and following it there can exist a negative maximum eigenvalue for certain choices of p with respect to the local curvature of the hill.

Lemma 2. *Given a vector field f defined by* (2.6) *and obtaining F through* (2.1)*, there exists a heading θ such that the maximum eigenvalue of F is negative, $\lambda_{max} < 0$, if one of the following condition is true: $p < 1$ for regions with positive hill curvature or $p > 1$ for regions with negative hill curvature.*

Proof. Define G to be the un-rotated part of the controller in (2.6),

$$G = \frac{\nabla h}{\|\nabla h\|^p} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} \tag{2.7}$$

Set θ to be piecewise constant, and thus there is no gradient of theta with respect to the states (x, y). Furthermore, because the speed scaling term α does not affect the analysis, it is set to be unity. Therefore, the symmetric part of the Jacobian F of the vector field f is,

$$\begin{aligned} F &= \begin{bmatrix} a & c \\ c & b \end{bmatrix} := \frac{1}{2}(D_f + D_f^T) \\ a &= \frac{\partial G_x}{\partial x}\cos\theta - \frac{\partial G_y}{\partial x}\sin\theta, \qquad b = \frac{\partial G_y}{\partial y}\cos\theta + \frac{\partial G_x}{\partial y}\sin\theta \\ c &= \frac{1}{2}\left((\frac{\partial G_x}{\partial y} + \frac{\partial G_y}{\partial x})\cos\theta + (\frac{\partial G_x}{\partial x} - \frac{\partial G_y}{\partial y})\sin\theta\right) \end{aligned} \tag{2.8}$$

The characteristic equation of F is given by

$$\lambda_{1,2} = \frac{a + b \pm \sqrt{a^2 - 2\,a\,b + b^2 + 4\,c^2}}{2} \tag{2.9}$$

By (2.9), the maximum eigenvalue of F is defined as,

$$\lambda_{max} = \frac{e\cos\theta + g\sin\theta + d}{2} \tag{2.10}$$

where d, e and g are defined as,

$$d = \sqrt{\frac{\partial G_x}{\partial x}^2 + \frac{\partial G_y}{\partial y}^2 + \left[\frac{\partial G_y}{\partial x}^2 + 2\frac{\partial G_y}{\partial x}\frac{\partial G_x}{\partial y} + \frac{\partial G_x}{\partial y}^2\right] - 2\frac{\partial G_x}{\partial x}\frac{\partial G_y}{\partial y}} \tag{2.11}$$

$$e = \frac{\partial G_x}{\partial x} + \frac{\partial G_y}{\partial y}, \qquad g = \frac{\partial G_x}{\partial y} - \frac{\partial G_y}{\partial x} \tag{2.12}$$

Then solve for θ when the maximum eigenvalue is zero and define $\phi = \tan^{-1}\left(\frac{g}{e}\right)$,

$$0 = \tan^{-1}\left(\frac{g}{e}\right) \pm \cos^{-1}\left(-\frac{d}{\sqrt{e^2 + g^2}}\right) \tag{2.13}$$

In (2.13), the $\pm$ term describes the range of diverging directions, and the first component ϕ is in the direction of $\max(\lambda_{max})$. A negative maximum eigenvalue exists when the term in the inverse cosine is within the open set from -1 to 1 and so,

$$d^2 - (e^2 + g^2) < 0 \tag{2.14}$$

By substituting values of d, e, and g and plugging in the derivatives of G (2.7) for the power controller (2.6) into the constraint (2.14), the existence of a negative λ_{max} can be evaluated using constraint,

$$\left(\frac{\partial^2 h}{\partial x^2}\frac{\partial^2 h}{\partial y^2} - \frac{\partial^2 h}{\partial x \partial y}^2\right)(p-1) < 0 \tag{2.15}$$

The original controller (2.5) has a p value of 1, and never satisfies (2.15). Therefore, there does not exist a θ which results in a negative λ_{max} for any position (x, y) on any hill $h(x, y)$ with $p = 1$.

The Gaussian curvature for a hill function $h(x, y)$ is,

$$K = \frac{\frac{\partial^2 h}{\partial x^2}\frac{\partial^2 h}{\partial y^2} - \frac{\partial^2 h}{\partial x \partial y}^2}{\left(1 + \frac{\partial h}{\partial x}^2 + \frac{\partial h}{\partial y}^2\right)^2} \tag{2.16}$$

The numerator in (2.16) determines the sign of the Gaussian curvature at a point (x, y) on a hill.

By using the constraint in (2.15), it is clear that there exists a negative λ_{max} for regions in positive curvature with a p value less than 1, and for regions of negative curvature with a p value greater than 1, that is,

$$\begin{aligned} (\text{sign}(K) > 0 \ \wedge \ p < 1) \ &\rightarrow \ \exists\, \theta \ \text{s.t} \ \lambda_{max} < 0 \\ (\text{sign}(K) < 0 \ \wedge \ p > 1) \ &\rightarrow \ \exists\, \theta \ \text{s.t} \ \lambda_{max} < 0 \end{aligned} \tag{2.17}$$

□

2.3.4 Trajectory Optimization Framework

This section presents a method of automatically finding optimally convergent and smooth trajectories through trajectory optimization (T-OPT) using direct collocation [Hargraves and Paris, 1987]. The resulting trajectories approximate a smooth trajectory with N piecewise-smooth trajectories. The trajectory optimization framework uses a cost function which seeks to maximize convergence and smoothness of a trajectory. The first order dynamics of the mobile robot are enforced through linear collocation constraints. The trajectory's start and end positions are bounded to match the problems specifications. Nonlinear constraints are used to enforce uniformly negative definite divergence metrics $D_m < 0$ or $D_a < 0$.

The trajectories consist of N waypoints and 5 decision variables per waypoint i: the position $(x^{(i)}, y^{(i)})$, velocity $(\dot{x}^{(i)}, \dot{y}^{(i)})$, and controller power $p^{(i)}$. The heading angle $\theta^{(i)}$ and forward speed are encoded as a velocity vector $(\dot{x}, \dot{y})$ to avoid using nonlinear constraints for collocation. The velocity is calculated in a global frame and converted to a hill-relative angle θ as a post processing step. In this paper MathWorks MATLAB's nonlinear programming solver `fmincon` [MATLAB] is used with the sequential quadratic programming (SQP) algorithm. The trajectory optimization framework is initialized with a straight line trajectory from the starting point to the center of the goal region.

Cost Function The cost is defined as,

$$J(x, y, \dot{x}, \dot{y}, p) = \frac{1}{N-1} \sum_{i=1}^{N-1} \left(J_{D_a}^{(i)} + J_{D_m}^{(i)} + J_{accel}^{(i)} + J_{path}^{(i)} \right) \tag{2.18}$$

where

$$\begin{aligned} J_{D_a}^{(i)} &= \sigma D_a(x^{(i)}, y^{(i)}, \dot{x}^{(i)}, \dot{y}^{(i)}, p^{(i)}) \\ J_{D_m}^{(i)} &= \upsilon \max(D_m(x^{(i)}, y^{(i)}, \dot{x}^{(i)}, \dot{y}^{(i)}, p^{(i)}), -\epsilon) \\ J_{accel}^{(i)} &= \gamma[(\dot{x}^{(i+1)} - \dot{x}^{(i)})^2 + (\dot{y}^{(i+1)} - \dot{y}^{(i)})^2] \\ J_{path}^{(i)} &= \rho[(\dot{x}^{(i)})^2 + (\dot{y}^{(i)})^2] \end{aligned} \tag{2.19}$$

The cost function is designed to minimize average divergence J_{D_a}, encourage negative maximum eigenvalues J_{D_m}, smooth the trajectory by adding cost to big changes in velocity J_{accel}, and to minimize path length J_{path}, based on some user desired weighting $(\sigma, \upsilon, \gamma, \rho, \epsilon)$. The sum is evaluated up to $N-1$, because the cost associated to the Nth waypoint does not affect the trajectory.

Linear Constraints and Bounds Linear equality constraints are used to ensure first order dynamics.

$$\begin{aligned} x^{(i+1)} - x^{(i)} - \dot{x}^{(i)} dt &= 0 \\ y^{(i+1)} - y^{(i)} - \dot{y}^{(i)} dt &= 0 \end{aligned} \tag{2.20}$$

In addition, upper and lower bounds are defined to be:

$$\begin{aligned} x_{initial} \leq x^{(0)} \leq x_{initial} \\ y_{initial} \leq y^{(0)} \leq y_{initial} \\ x_{end} - \eta \leq x^{(n)} \leq x_{end} + \eta \\ y_{end} - \eta \leq x^{(n)} \leq y_{end} + \eta \\ x_{min} \leq x \leq x_{max} \\ y_{min} \leq y \leq y_{max} \\ p_{min} \leq p \leq p_{max} \end{aligned} \tag{2.21}$$

Initial and ending positions are bounded instead of using linear equality constraints to help the solver find solutions. The initial position is bounded to be the exact desired starting position, and the end position is bounded to be within a square with side length 2η around the desired end location. All positions are bounded to be within the limits of the defined map. Bounds on the power variable p are placed to ensure stable and realistic values.

Nonlinear constraints Nonlinear constraints are used to enforce the strict convergence results of [Lohmiller and Slotine, 1998, Thm. 1] and [Lohmiller and Slotine, 1998, Sec. 3.9]. The three different methods considered are:

1. Applying [Lohmiller and Slotine, 1998, Thm. 1] to ensure strict convergence at each waypoint:

$$D_m^{(i)} < 0 \quad \forall \quad 1 \leq i \leq N-1 \tag{2.22}$$

2. Applying [Lohmiller and Slotine, 1998, Sec 3.9] to ensure average convergence at each waypoint:

$$D_a^{(i)} < 0 \quad \forall \quad 1 \leq i \leq N-1 \tag{2.23}$$

3. No nonlinear constraints (unconstrained)

Trajectories where D_m are uniformly negative definite are resilient to uncertainty in all directions at all times and are guaranteed to exponentially converge to the nominal trajectory. In the case that D_a is uniformly negative definite, the trajectories are on average robust to uncertainties, but may suffer from uncertainty in the direction of maximum divergence. Lastly, the unconstrained case is considered because there will more often be a feasible solution and the average divergence is minimized by the cost function, while in the constrained cases, feasible solution may often not exist due to the harshness of the constraints. The analytical solutions of D_m and D_a were calculated using the eigenvalues from (2.9) as well as their associated gradients with respect to the states. Due to the strictness and complexity of satisfying these constraints, the gradient played a crucial role in finding solutions and naively calling `eig` [MATLAB] did not find any feasible solutions for the

($D_m < 0$) constrained case.

2.3.5 RRT methods

The proposed trajectory optimization (T-OPT) methods are compared against the rapidly-exploring random tree (RRT) methods as shown in [Johnson et al., 2016b]. Specifically, biased RRT (B-RRT) and contraction region RRT (CR-RRT) are implemented. The T-OPT methods were also compared against asymptotic optimal variants of the RRT methods by utilizing (RRT*) [Karaman and Frazzoli, 2010]. Since both B-RRT and CR-RRT are extensions of Kinodynamic RRT (KD-RRT) [LaValle and Kuffner Jr, 2001], an RRT* version of each method was developed: B-RRT* and CR-RRT*.

Kinodynamic RRT (KD-RRT) KD-RRT for the hill climbing problem follows the same collocation constraints (2.20) and bounds on start position and goal position set (2.21) as for trajectory optimization. The planner builds a tree which starts at the desired starting position. At each iteration the planner samples a random position and finds the nearest neighbor in the RRT using the Euclidean distance metric. Random directions and powers p are sampled to generate a set of candidate actions. Each action is evaluated by finding the end point of a trajectory that follows that direction for a fixed distance. An action a is selected and a new node is added to the RRT based on which action reaches closest to the sampled point, using the same Euclidian distance metric. This is repeated until a node is within the goal position set (2.21).

Biased RRT (B-RRT) B-RRT attempts to minimize divergence by scaling the Euclidean distance metric used for action selection by e^{bD_i}, where $b \in \mathbb{R}$ is a bias term and D_i is the specified divergence metric to minimize. In our implementation, average divergence D_a is used.

Contraction Region RRT (CR-RRT) CR-RRT applies the same constraint as (2.22) at the action selection step to ensure a trajectory that meets the contraction region requirements at every step. However, in [Johnson et al., 2016b] there can never exist a true contraction region because a

constant velocity controller was used (as proven in Lemma 2). Here, both D_m (2.22) and D_a (2.23) are used.

Biased RRT* (B-RRT*) and Contraction Region RRT* (CR-RRT*) RRT* follows the same algorithm as KD-RRT except that the goal was randomly sampled 5% of the time and before adding the node to the RRT, there is a rewiring step which optimizes which node in the tree should connect to the candidate node. Nodes that are within a distance of r away from the candidate node are considered for rewiring. Branches are grown from all neighboring nodes to the candidate node, and the entire path length for each branch is calculated. The power variable p for each branch is chosen from the set $\{0, 1, 2\}$ to minimize the cost of the branch. The branch that leads to the minimum path length from the candidate node to the beginning node is added to the tree. Once a node is within the goal position set (2.21), the algorithm is iterated k more times to continue rewiring. B-RRT* uses the same biased distance metric as B-RRT in the rewiring step and CR-RRT* can apply either constraint (2.22) or (2.23) during the rewire step.

2.3.6 Simulation

To evaluate convergence of the generated trajectories (from either the trajectory optimization or RRT methods), a circle of 40 points of uncertainty with radius $r = \eta$ (half the side length of the desired goal region) was added to the trajectories' starting positions and simulated forwards by integrating the power controller's (2.6) dynamics using MathWorks MATLAB's `ode45` [MATLAB]. To ensure heading and velocity were nominally constant between each waypoint, heading angles θ and velocity scaling variable α were calculated in between each waypoint. The starting area of the circle and radius of the circle is compared against the convex hull of the particles at the end of the trajectory and the particle furthest from the nominal trajectories end position. Define two metrics of path convergence to be the ratio between the end particle area and the starting circle area E_a and the ratio between the maximum end particle and the starting circle radius E_m [Johnson et al., 2016b]. The area ratio E_a corresponds to average divergence D_a and the maximum distance ratio corresponds

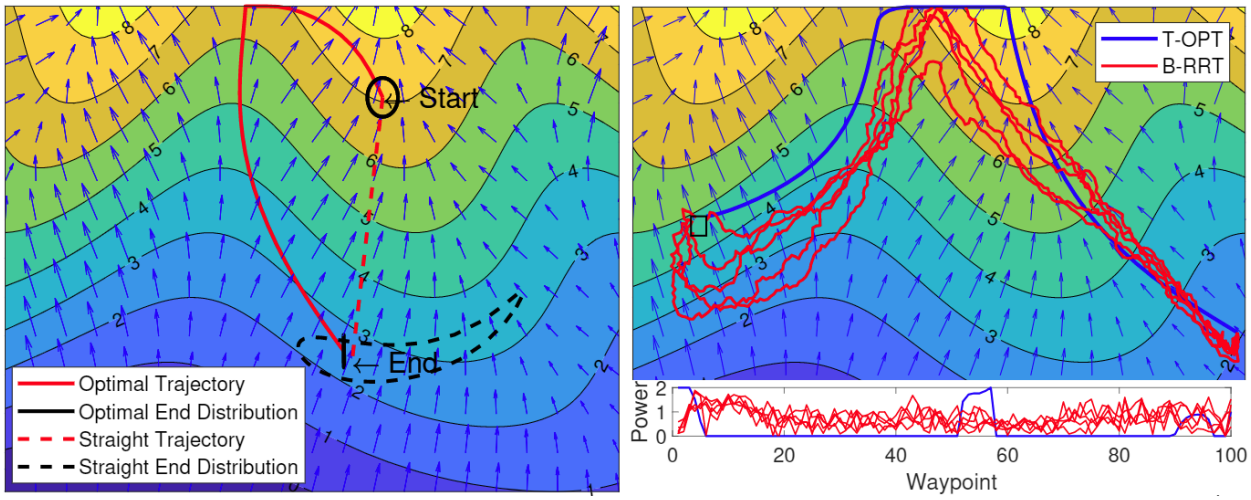

Figure 2.2: Topographic map of a hill showing elevation lines and hill gradient (blue arrows). Left: Simulating forward a circle of uncertainty (black) around the nominal trajectory (red). An optimal trajectory (red solid) converged to a line (black solid) with an area ratio of $E_a = 0.02$ and an max of distance ratio $E_m = 0.88$ and a straight line method (red dashed) diverged to an end distribution (black dashed) with $E_a = 11.32$ and $E_m = 11.66$. Top Right: T-OPT (blue line) trajectory, 5 B-RRT (red line) trajectories, and the goal region (black square). Bottom Right: Hill power controller's (2.6) power value p at each waypoint for T-OPT (blue) and the 5 B-RRT (red).

to the maximum divergence D_m. An example output of the simulation for a trajectory optimized using the unconstrained trajectory optimization method is shown in Fig. 2.2.

2.4 Experiments

This section demonstrates the effectiveness of the proposed trajectory optimization framework and the new power controller (2.6) for a mobile robot traversing hilly terrain. The hill function is chosen to be consistent with [Johnson et al., 2016b; Liu et al., 2019],

$$h(x, y) := 3y + \sin(x + xy) \quad \text{s.t.} \quad (x, y) \in [-2, 2] \times [0, 2.5] \tag{2.24}$$

This hill contains a range of curvature content. Therefore, sampling random starting and ending positions evaluates the efficacy of each planner and controller on a collection of diverse landscapes.

2.4.1 Experiment Methods

To evaluate the convergence and smoothness of the trajectories generated using T-OPT, B-RRT, CR-RRT, B-RRT*, and CR-RRT*, 200 random start and end points were used to calculate an average area ratio E_a, max distance ratio E_m, and acceleration cost J_{accel}. Eleven different trajectory planning methods with different algorithms, power controller bounds, constraints on divergence metrics (2.22)-(2.23), and cost functions were tested:

1. (T-OPT), $p = 1$, unconstrained, minimize path length
2. (T-OPT), $p \in [0, 2]$, unconstrained, minimize cost function (2.19)
3. (B-RRT), $p \in [0, 2]$, unconstrained, bias $b = 1.5$ action size $a = 0.025$
4. (B-RRT*), $p \in [0, 2]$, bias $b = 1.5$ action size $a = 0.005$ rewire size $r = 0.025$
5. (T-OPT), $p \in [0, 2]$, $D_m < 0$, minimize cost function (2.19)
6. (CR-RRT), $p \in [0, 2]$, $D_m < 0$, $b = 1.5$, $a = 0.025$
7. (CR-RRT*), $p \in [0, 2]$, $D_m < 0$, $b = 1.5$, $a = 0.005$, $r = 0.025$
8. (T-OPT), $p \in [0, 2]$, $D_a < 0$, minimize cost function (2.19)
9. (T-OPT), $p = 1$, $D_a < 0$, minimize cost function (2.19)
10. (CR-RRT), $p \in [0, 2]$, $D_a < 0$, $b = 1.5$, $a = 0.025$
11. (CR-RRT*), $p \in [0, 2]$, $D_a < 0$, $b = 1.5$, $a = 0.005$, $r = 0.025$

The constants in the cost function (2.19) were heuristically tuned to be $\sigma = 30$, $\upsilon = 2$, $\gamma = 10^5$, $\rho = 10^4$, and $\epsilon = 1$. In the upper and lower bounds, the square goal region's side length $\eta = 0.05$ was set to be half of the acceptable goal region in [Johnson et al., 2016b]. Powers were bounded to allow a change of ± 1 from the nominal $p = 1$ since higher powers lead to instability, $p_{min} = 0$ and $p_{max} = 2$.

Table 2.1: Results for trajectory optimization (T-OPT), biased RRT (B-RRT), and contraction region RRT (CR-RRT) path planning methods over 200 random trials where the power of the controller was either $p = 1$ or in the closed set $p \in [0, 2]$. Results include success rate (S), log area ratio (E_a), log distance ratio (E_m), acceleration cost (J_{accel}), and planning time (T) in seconds. Mean ± one standard deviation are listed.

Method	p	S	$\log(E_a)$	$\log(E_m)$	J_{accel}	T
shortest						
1 T-OPT	1	100	0.09 ± 1.12	0.685 ± 0.686	0.406 ± 0.325	0.741 ± 0.478
uncon						
2 T-OPT	$[0, 2]$	100	-3.26 ± 2.66	0.425 ± 0.736	5.89 ± 2.66	219 ± 37.8
3 B-RRT	$[0, 2]$	96.5	-3.03 ± 1.67	-0.00146 ± 0.170	31.8 ± 7.62	73 ± 108
4 B-RRT*	$[0, 2]$	94	-3.08 ± 1.55	0.133 ± 0.243	7.96 ± 1.79	181 ± 259
$Dm < 0$						
5 T-OPT	$[0, 2]$	4.5	-5.17 ± 1.52	-0.221 ± 0.167	76.3 ± 95.8	244 ± 81.3
6 CR-RRT	$[0, 2]$	7.5	-1.78 ± 1.89	-0.124 ± 0.116	20 ± 10.4	32.3 ± 65
7 CR-RRT*	$[0, 2]$	1	-0.20 ± 0.13	-0.005 ± 0.033	5.55 ± 2.63	2160 ± 3060
$Da < 0$						
8 T-OPT	$[0, 2]$	92.5	-4.2 ± 1.67	0.0633 ± 0.168	6.52 ± 3.28	714 ± 440
9 T-OPT	1	84	-3.06 ± 1.74	0.0921 ± 0.156	6.37 ± 12.7	121 ± 35.5
10 CR-RRT	$[0, 2]$	93	-3.17 ± 1.69	-0.0176 ± 0.174	29.2 ± 8.28	96.6 ± 151
11 CR-RRT*	$[0, 2]$	83.5	-3.01 ± 1.5	0.028 ± 0.278	9.2 ± 2.27	708 ± 939

2.4.2 Results

The success rate S, average log area ratio E_a, average log max distance ratio E_m, and planning time for each tested method is shown in Table 2.1. As expected, planning for the shortest path led to an average positive log area ratio E_a, and using converging planning methods led to negative log area ratio E_a averages. A paired difference test is used for mean comparisons. All comparisons discussed are significant ($p < 0.05$) unless noted.

Unconstrained Problems Method 2 (unconstrained T-OPT) on average created trajectories with lower log area ratios E_a than method 3 (unconstrained B-RRT) and method 4 (unconstrained B-RRT*) as shown in Table 2.2. However, methods 3 and 4 had a lower mean log area ratio E_a than method 2 when using paired differences. This resulted from the few divergent paths created from method 2 dominating the comparisons. Divergent paths occurred because in some rare cases, T-OPT picked an early divergent step in a trajectory to gain access to more convergent steps later

in the trajectory. B-RRT and B-RRT* are not affected by this issue because it they are inherently greedy search algorithms, and are biased towards the most convergent direction at each waypoint. To avoid this issue for T-OPT, a saturation limit can be placed on the average divergence cost J_{D_a} or the $D_a < 0$ constraint can be applied. Method 2 was able to find a feasible solution for all 200 random trials while methods 3 and 4 could not. Methods 3 and 4 also had a lower mean log E_m than method 2, likely because RRT inherently randomizes the direction of max divergence, while the optimal solution tends to keep the worst direction aligned in a similar direction. Because this hill is smooth, and the hill gradient does not rapidly change, resulting T-OPT solutions tend to grow uncertainty in the same direction. Method 4 is smoother than method 3 when comparing acceleration cost J_{accel} which in turn likely made method 4 have a greater mean log E_m than method 3.

Constrained Maximum Convergence For this hill climbing problem, these are the first results where trajectories can be produced with a contraction region. Method 6 (CR-RRT, $D_m < 0$) found 15 feasible solutions while method 5 (T-OPT, $D_m < 0$) only found 9 and method 7 (CR-RRT*, $D_m < 0$) found 2. However, all but one trajectory has a negative log maximum distance ratio E_m for method 5 and all trajectories for method 7 have a negative log maximum distance ratio E_m, while 3 of the 15 trajectories produced by method 6 have a positive log maximum distance ratio E_m. The resulting positive log maximum distance ratio E_m is due to the action size a being too large in comparison to the geometry of the hill. Since planning time grew exponentially with increasing tree size, the action size could not be further reduced. In the single case that method 4 had a positive log distance ratio E_m, the maximum action size was also large ($a = 0.4518$ on average).

Constrained Average Convergence In the case of constraining the average divergence to be negative ($D_a < 0$), method 8 (T-OPT, $p \in [0, 2]$, $D_m < 0$) was able to find 9 % more feasible solutions than method 9 (T-OPT, $p = 1$, $D_m < 0$), 9.5 % more feasible solutions than method 11 (CR-RRT*, $D_m < 0$), and 0.5 % fewer feasible solutions than method 10 (CR-RRT, $D_m < 0$). Method 8 has a lower mean log area ratio E_a than methods 9, 10, and 11. However, methods 8

Table 2.2: Comparing the percentage of trials that the end area ratio E_a is smaller using the method on the vertical axis compared to the method on the horizontal axis.

Method	Compared Method Number: 8	2	4	9	11	10	3
8 T-OPT, $p \in [0,2]$, $D_a < 0$	0	0.66	0.67	0.9	0.73	0.67	0.72
2 T-OPT, $p \in [0,2]$, uncon	0.34	0	0.6	0.72	0.63	0.6	0.63
4 B-RRT*, $p \in [0,2]$, uncon	0.33	0.4	0	0.5	0.66	0.64	0.64
9 T-OPT, $p = 1$, $D_a < 0$	0.099	0.28	0.5	0	0.59	0.57	0.57
11 CR-RRT*, $p \in [0,2]$, $D_a < 0$	0.27	0.37	0.34	0.41	0	0.51	0.53
10 CR-RRT, $p \in [0,2]$, $D_a < 0$	0.33	0.4	0.36	0.43	0.49	0	0.57
3 B-RRT, $p \in [0,2]$, uncon	0.28	0.37	0.36	0.42	0.47	0.43	0

and 11 took about 7 times longer to compute than methods 9 and 10, and like the unconstrained problems, the RRT methods have a lower maximum distance ratio E_m, Table 2.1.

Log Area Ratio E_a Direct comparisons For a more fair comparison between methods, direct trial comparisons for log area ratio E_a are made between the unconstrained and the average divergence constrained ($D_a < 0$) methods as shown in Table 2.2. Only runs where both methods had a feasible solution were compared. Because constraining ($D_m < 0$) led to few solutions, the strict convergence case was not directly compared against the other methods. For all problems, method 8 (T-OPT, $p \in [0,2]$, $D_a < 0$) has on average a lower log area ratio E_a for each direct comparison. In second place, method 2 (T-OPT, $p \in [0,2]$, unconstrained) has a lower log area ratio E_a for the majority of problems except when compared to method 8. Method 9 (T-OPT, $p = 1$, $D_a < 0$) is worse than all other T-OPT cases, but is slightly better than or equal to the RRT and RRT* cases methods (3,4,10,11). When directly comparing the power controller (2.6), method 8, versus the constant speed controller (2.5), method 9, the power controller is more convergent 90.1% of the time.

Trajectory Smoothness Trajectories generated by trajectory optimization were smoother than both RRT* and RRT methods in both position choices and power choices. An example run shown in the top side of Fig. 2.2 compares trajectories generated using T-OPT and B-RRT. The bottom side of Fig. 2.2 shows that the powers chosen by T-OPT are smoother and more consistent

than the powers chosen by the B-RRT runs. The acceleration cost J_{accel} was imposed to the trajectory optimization framework to reduce sudden changes in linear and rotational velocity. When comparing the acceleration cost over the 200 trials between the unconstrained (methods 2, 3, and 4), and constrained ($D_a < 0$, methods 8, 9, 10, and 11), T-OPT had a smaller mean acceleration cost J_{accel} than B-RRT and B-RRT*. There were not enough successful trials for the ($D_m < 0$) constrained cases to draw meaningful conclusions on smoothness.

2.5 Conclusion

Planning trajectories which are robust to uncertainty is critical for creating reliable robotic systems. Typically, uncertainty is reduced by using closed-loop feedback which can sense and reduce errors. However, in situations like the hill climbing problem, the robot is under-sensed and cannot reduce error by using closed-loop feedback. Instead it must rely on reducing error by leveraging the geometry of the underlying vector field.

Prior work [Johnson et al., 2016b; Liu et al., 2019] to create robust trajectories using convergent rapidly-exploring random trees. However, the resulting trajectories are not smooth and at best could only enforce average convergence. The work in this paper creates a new convergent optimization framework which generates optimally smooth and convergent trajectories.

This work also introduces a new hill navigation controller (2.6) which enables the possibility for strictly convergent trajectories while a constant speed controller (2.5) can only produce average convergent trajectories. The new power controller was also more convergent 90% of the time and found 9% more feasible solutions than the constant speed controller when solving for average convergent trajectories using T-OPT. Although a low number of trials emitted a feasible solution when applying the strict convergence constraint, the work in this paper is the first example of finding contraction regions for this hill climbing problem. The low number of solutions is due to the specifics of the problem and how strict this requirement is. Problems that more readily admit such solutions would see higher success rate.

Trajectories generated through the trajectory optimization framework on average are more convergent and smoother than the ones produced using RRT and RRT* methods. However, RRT and RRT* methods produced trajectories with smaller max distance ratios E_m than the T-OPT methods when not constraining maximum divergence to be negative. We believe the randomness when picking heading directions from RRT methods help shrink the maximum distance ratio, because the maximum divergence direction is rapidly changing at each waypoint, while the optimal solution keeps the direction of maximum divergence aligned in a similar direction throughout the trajectory. However, solutions produced by RRT methods would need to first be smoothed out before implementation, and may lose the benefits of randomized alignment. Post-processing these trajectories may lead to less optimal solutions or violation of the convergence constraints. These 2 drawbacks are apparent in the RRT* trajectories. The trajectories produced by RRT* are smoother than the ones produced by RRT, but in the process, it is likely that the trajectories aligned the maximum divergence direction in one direction more and worsened the max distance ratio E_m. In the constrained RRT* cases, RRT* could not exploit creating jagged paths to satisfy the constraints, and ultimately emitted few or poor convergence trajectories. On the other hand, trajectories generated from T-OPT can readily be implemented on a robot with simple unicycle dynamics while guaranteeing optimality and constraints on divergence.

In general, optimizing for convergence is a useful tool for tasks where robots are undersensed or underactuated. The presented trajectory optimization framework and convergent controller analysis can also be expanded to more dynamic path planning problems. The framework will directly translate to other 2 dimensional state space problems. Modifications to eigenvalue analytical calculations must be made for higher dimensions because they are trivial in the 2D case. In the future, we plan to analyze the uncertainty growth in non-smooth systems such as hybrid dynamical systems [Burden et al., 2018a], and to utilize a similar optimization framework to plan robust walking behaviors for legged robots.

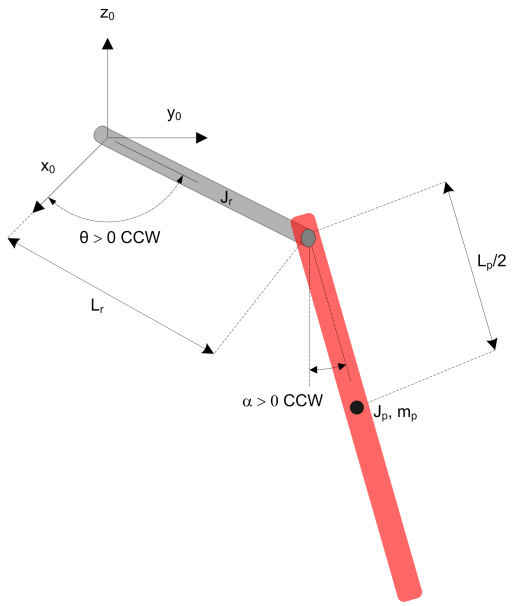

Figure 2.3: Rotary Cart Pole Coordinates [Quanser Inc.]

2.6 Optimally Convergent Swing Up for Rotary Cart Pole

2.6.1 Rotary Cart Pole System Definition

The rotary cart pole coordinates are defined in Fig. 2.3. Where the rotary cart angle is θ and the pole angle is defined to be α. The states are defined to be $x = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^T$. The system parameters of the rotary cart pole are listed in Table 2.3.

The equations of motion for the rotary cart pole system are derived using Lagrangian dynamics.

$$\begin{aligned}
&\left(m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos(\alpha)^2 + J_r\right) \ddot{\theta} \cdots \\
&- \left(\frac{1}{2} m_p L_p L_r \cos(\alpha)\right) \ddot{\alpha} + \left(\frac{1}{2} m_p L_p^2 \sin(\alpha) \cos(\alpha)\right) \dot{\theta}\dot{\alpha} \cdots \\
&+ \left(\frac{1}{2} m_p L_p L_r \sin(\alpha)\right) \dot{\alpha}^2 = \tau - D_r \dot{\theta}
\end{aligned} \tag{2.25}$$

DC Motor		
R_m	Terminal resistance	8.4Ω
k_t	Torque constant	$0.042N.m/A$
k_m	Motor back-emf constant	$0.042V/(rad/s)$
J_m	Rotor inertia	$4.0 \times 10^{-6} kg.m^2$
L_m	Rotor inductance	$1.16mH$
m_h	Load hub mass	$0.0106kg$
r_h	Load hub mass	$0.0111m$
J_h	Load hub inertia	$0.6 \times 10^{-6} kg.m^2$
Load Disk		
m_d	Mass of disk load	$0.053kg$
r_d	Radius of disk load	$0.0248m$

Table 2.3: Rotary Cart Pole system parameters [Quanser Inc.]

and

$$\frac{1}{2}m_p L_p L_r \cos(\alpha)\ddot{\theta} + \left(J_p + \frac{1}{4}m_p L_p^2\right)\ddot{\alpha} - \frac{1}{4}m_p L_p^2 \cos(\alpha)\sin(\alpha)\dot{\theta}^2 + \frac{1}{2}m_p L_p g \sin(\alpha) = -D_p\dot{\alpha}. \tag{2.26}$$

By solving for accelerations $(\ddot{\theta}, \ddot{\alpha})$ the dynamics are defined to be

$$f(x, \tau, t) = \begin{bmatrix} \dot{\theta} & \dot{\alpha} & \ddot{\theta} & \ddot{\alpha} \end{bmatrix}^T \tag{2.27}$$

The maximum divergence metric for the system D_m is too large to display, but is positive semi definite (positive definite in non singular configurations). Therefore, this system cannot produce a contraction region:

$$D_m \geq 0 \tag{2.28}$$

However, like the hill navigation example, we can minimize the average divergence to reduce the effects of perturbation on a closed loop controller.

2.6.2 Trajectory Optimization

The trajectory optimization for the rotary cart pole swing up is the same as the hill navigation example as shown in Sec. 2.3.4, with the modifications of removing the cost on maximum divergence and terms related to the power controller.

The trajectories consist of N waypoints and 5 decision variables per waypoint i: the position $(\theta^{(i)}, \alpha^{(i)})$, velocity $(\dot{\theta}^{(i)}, \dot{\alpha}^{(i)})$, and input torque $\tau^{(i)}$. In this paper MathWorks MATLAB's nonlinear programming solver `fmincon` [MATLAB] is used with the sequential quadratic programming (SQP) algorithm. The trajectory optimization framework is initialized with zeros for the minimal energy trajectory. The minimal energy trajectory is used to seed the creation of a smooth minimal energy trajectory. Lastly, the smooth minimal energy trajectory is used to initialize the optimally convergent trajectory.

2.6.3 Controller design

Since a contraction region $D_m < 0$ cannot be produced, we must design a controller to stabilize the trajectory. In this work, Linear Time Varying Linear Quadratic Regulator (LTVLQR) is used to stabilize the trajectory. The feedback-stabilizing controller is numerically calculated by solving the differential equation:

$$-\dot{s}(t) = Q - s(t)B(t)R^{-1}B(t)^T s(t) + sA(t) + A(t)^T s \tag{2.29}$$

and setting the scheduled gain $K(t)$ to

$$K(t) = R^{-1}B^T s(t) \tag{2.30}$$

The linear time varying matrices $A(t)$ and $B(t)$ are defined by linearizing the vector field (2.27).

By using `ode45` [MATLAB], the ODE for $s(t)$ is solved backwards in time for the duration of the trajectory where the end condition is set to be $s(T) = Q_f$. By combining the feedforward (open

loop) torques and closed loop control, the resulting input torque is defined to be:

$$u(t) = u_{open}(t) + K(t)(x_{des} - x_{actual}) \tag{2.31}$$

2.6.4 Hardware Experiments

To validate the feasibility of the resulting trajectories, they were evaluated on hardware. The weight on each state was heuristically tuned in Q until the trajectory could successfully be tracked. As expected, in both cases, the penalty for pole angle deviation was the largest. Once the pole is in the vertical position, the controller was switched to infinite time horizon LQR.

2.6.5 Simulating Perturbations Experiment

Perturbations were used to evaluate robustness of trajectories. The goal was to perturb the trajectory in the direction that would cause the most divergence. Since the linearizations are highly dependent on pole angle α, perturbations were applied directly to α. In this test, perturbations were applied at every time step of the trajectory as shown in Figs. 2.9 and 2.10, and the perturbed trajectory with a LTVLQR controller was simulated forward using ode45. To keep comparisons consistent, Q, Q_f, and R were set to be the same for both trajectories and an actuator limit of 0.1 Nm was set to better reflect the real system. Success was defined to be if the simulated trajectory was able to converge back to the nominal and end with the vertical pole position. Robustness was quantified by dividing the number of successful trials by the total.

2.6.6 Trajectory Optimization Results

Producing optimal trajectories that could be ran on the real system heavily depended on several factors.

1. Providing the gradient of the cost function and nonlinear constraints.

2. Adding penalty to change in torque

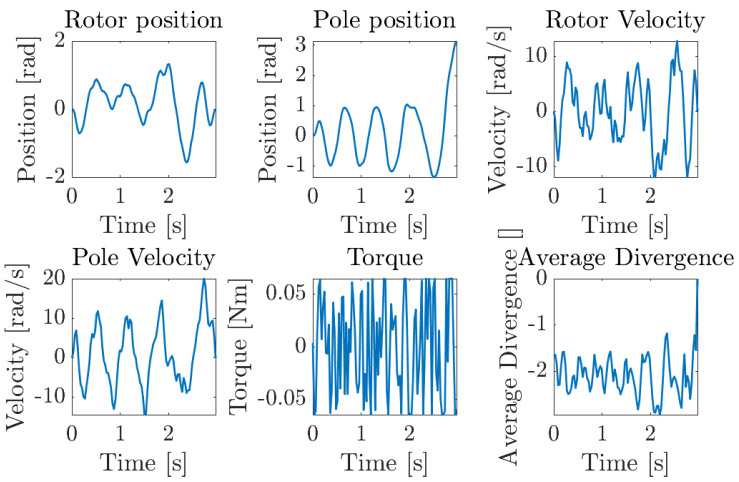

Figure 2.4: Minimal energy trajectory

3. Initialzing the optimization with previous feasible solutions

Without providing the gradient information, the solver often did not converge. When minimizing for energy, the optimizer produced trajectories with high frequency inputs that would not be trackable due to the bandwidth of the actuator as shown in Fig. 2.4. Adding penalty to change in torque drastically improved quality of the trajectory by reducing the frequency of the input signal as shown in Fig. 2.5.

Lastly, introducing additional terms to the cost function added complexity. When only optimizing for energy, the optimization would solve successfully converge to an optimal solution when seeded with just zeros, but adding any additional terms would cause the optimizer not to converge. The minimal energy was first solved for by seeding with zeros Fig. 2.4 and was used to initialize the optimization for optimizing for smoothness and minimal energy trajectory Fig. 2.5. The smooth and minimal energy trajectory was then seeded to create the optimally convergent trajectory as shown in Fig. 2.6 by adding the divergence term to the cost.

2.6.7 Hardware Results

The smooth minimal energy trajectory as shown in Fig. 2.5 and the optimally convergent trajectory as shown in Fig. 2.6 were both tested on the real hardware as shown in Figs. 2.7 and 2.8.

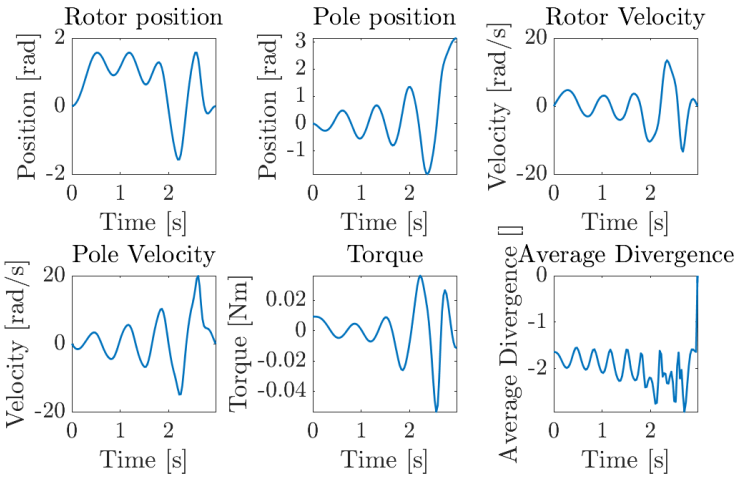

Figure 2.5: Optimally smooth and minimal energy trajectory.

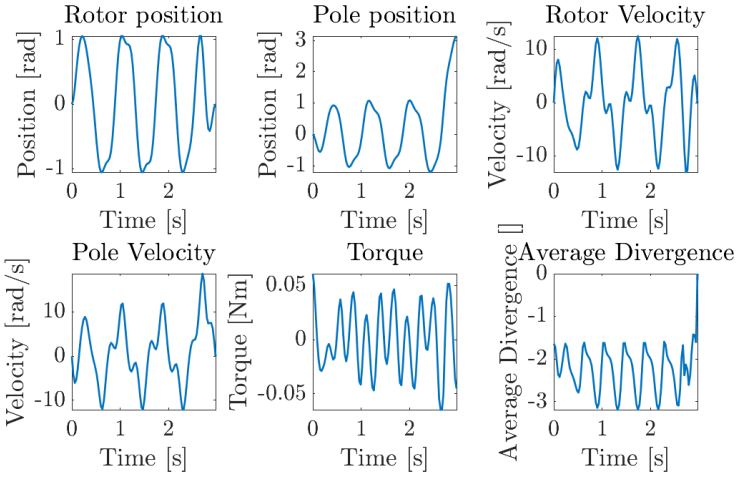

Figure 2.6: Optimally convergent trajectory

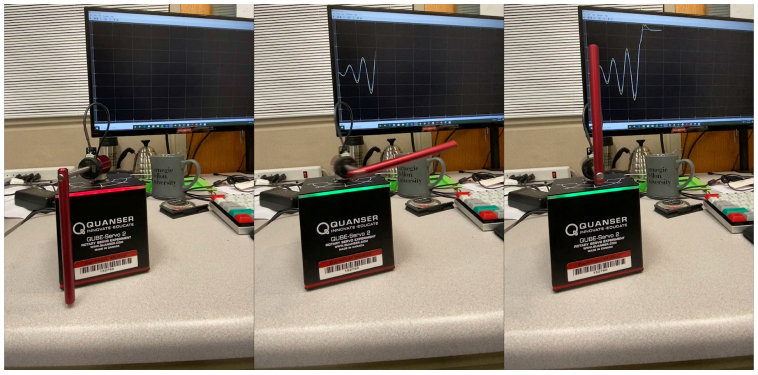

Figure 2.7: Tracking the smooth minimal energy trajectory on the real system with pole angle plotted in the background

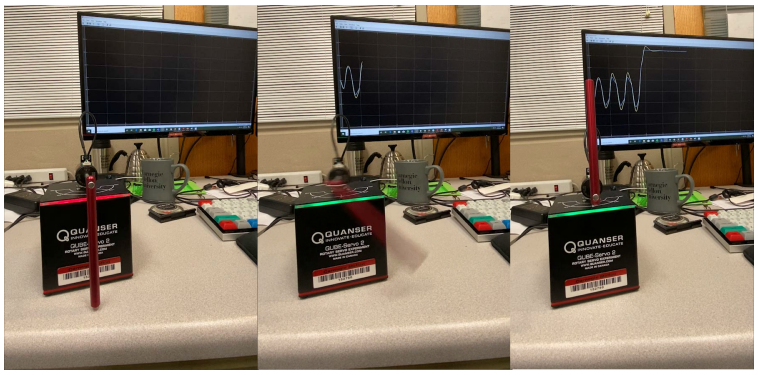

Figure 2.8: Tracking the optimally convergent trajectory on the real system with pole angle plotted in the background

However, due to unmodeled nonlinearities introduced through the encoder cabling (acted as a nonlinear spring), both trajectories on average only succeeded half of the time. It was necessary to reset the cables initial condition to a favorable location before each trial. This cable also caused issues with other swing up behaviors such as energy shaping.

2.6.8 Perturbation Results

Both trajectories had negative average divergence as shown in Figs. 2.5 and 2.6: this was attributed to damping in the system, since damping naturally leads to convergence. As expected, both trajectories were less convergent during the final swing up and were expected to fail when perturbations were added to the final swing up.

In this test, perturbations with magnitude 1 were applied along the trajectory. When adding perturbations to the final swing up, both trajectories started to diverge. However, the minimal energy trajectory also failed on the second to last swing. This is evident in Figs. 2.9 and 2.10 where the trajectories diverged in the minimal energy case both on the second to last swing and the last swing while in the case of optimally convergent only diverged on the last swing. These results suggests that planning to minimize average divergence led to avoiding areas in the state space that are hard to recover from.

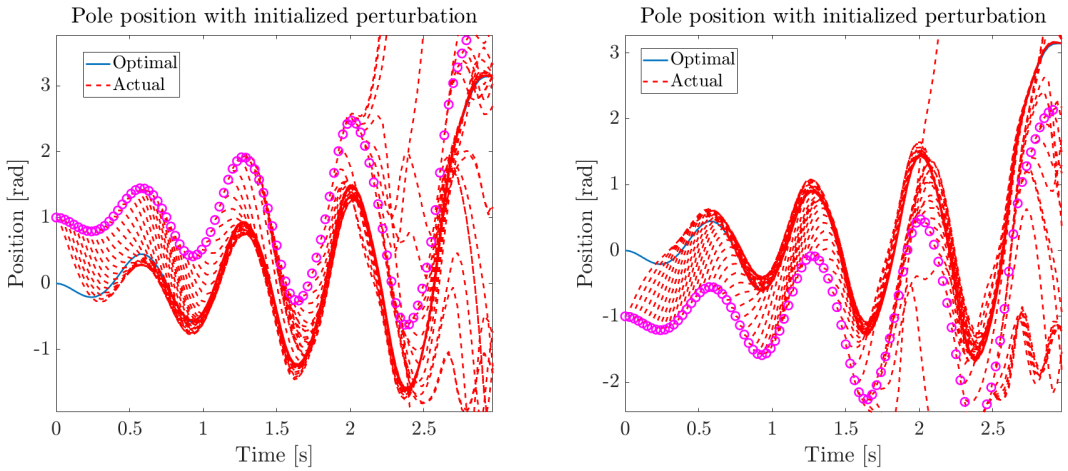

Figure 2.9: Flowing ±1 radian perturbations with LTVLQR on the smooth minimal energy trajectory

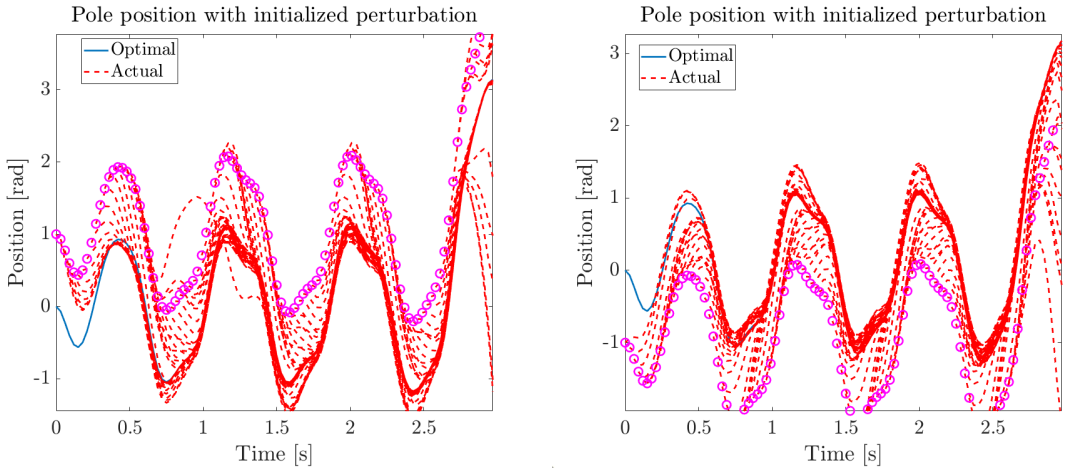

Figure 2.10: Flowing ±1 radian perturbations with LTVLQR on the optimally convergent trajectory

2.6.9 Conclusion

In this work, optimizing for average divergence rather than just minimum energy directly improved the success rate for a swing up behavior on a rotary cart pole system. In this underactuated scenario, optimizing for average divergence seemed to be enough to improve the robustness of the trajectory.

Chapter 3

Saltation Matrices: The Essential Tool for Linearizing Hybrid Systems

3.1 Abstract

Contact is ubiquitous in robotic systems that interact with the world and it introduces complexity due to the hybrid nature of contact. For example, a robot's leg swinging in the air is able to exert very little control effort compared to when it is on the ground. When the leg hits the ground, the penetrating velocity instantaneously collapses to zero. These changes in dynamics and discontinuities (or jumps) in state make standard smooth tools for planning, estimation, and control difficult for hybrid systems. Many strategies seek to smooth these discontinuous events by utilizing significant control effort to cancel them out so that standard strategies can be applied once they are nullified – like slowing down before impact to ensure the discontinuity is small. This strategy may be acceptable for robotic systems which are fully actuated, but many tough and interesting problems are not fully actuated (legged robots, dexterous manipulation, etc) and require making use of the natural dynamics of the system. This means that we must allow for jumps in our methods. One of the key tools for accounting for these jumps is called the saltation matrix. In this paper, we present an intuitive derivation of the saltation matrix and what it captures, where the saltation matrix has

been used in the past, how to use it for linear and quadratic forms, and how to compute it for rigid body systems with unilateral constraints.

3.2 Introduction

Many interesting problems in engineering can be modeled as hybrid dynamical systems, meaning that they involve both continuous and discrete states [Back et al., 1993; Lygeros et al., 2003; Goebel et al., 2009]. These systems can be hybrid due to physical contact, such as robotics manipulation or legged locomotion [Johnson et al., 2016a], or they can be triggered by control – reacting to sensor feedback or switching control modes. Meanwhile, most of the tools we have for planning, control, state estimation, and learning assume continuous (if not smooth) systems. A common strategy to apply tools that were made for smooth systems to hybrid systems is to minimize the effect of discontinuities [Yang and Posa, 2021a; Council et al., 2014], e.g. by slowing down to near zero velocity at the time of an impact event [Raibert et al., 1989] or controllers that work aroudn the discontinuity. However, these strategies do not make use of the underlying dynamics of the system and only seek to remove them. This strategy may work out for fully actuated systems, but generally hybrid systems of interest are underactuated and cannot always cancel out the discontinuous dynamics.

Rather than trying to assume continuous dynamics, we should instead develop new tools that account for the effects of discrete events. Often, discrete events are called "jumps" or "resets" and they map the state from one continuous domain to another. The key to developing new tools is to both model what happens at the moment of reset, but also account for what happens to neighboring trajectories (variations) that reset at different times. One might analyze the evolution of variations through linearization by taking the Jacobian of the reset map. However, this only captures part of the story. It is just as important to also capture the variation that arises from changes in reset timing. For example, if the hybrid modes have different dynamics at the boundary, then flowing trajectories a different amount of time in each mode will result in changes in the variation. The

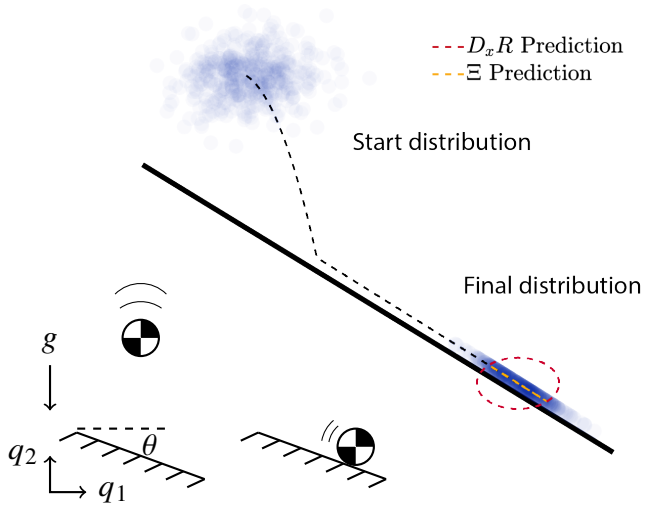

Figure 3.1: Example drop on a slanted surface with initial covariance. Using the saltation matrix (Ξ), we correctly estimate the end distribution's covariance where covariance in the direction of the constraint is eliminated. Using the incorrect update, only the Jacobian of the reset map (D_xR) leads to retaining belief in the direction of the constraint.

saltation matrix, also known as ..., which captures the total variation caused by both impact timing and reset dynamics, is the key tool to understanding what happens near a hybrid event up to first order.

An illustrative example of how the saltation matrix can help us analyze a common hybrid system, rigid bodies with contact, is shown Fig. 3.1. Here we drop a distribution of balls (centered about a nominal drop) on a slanted surface. When the balls make contact with the surface, a plastic impact law is applied which resets the system into a sliding mode on the surface by zeroing out the velocity into the surface. For this system, we expect the distribution to start out in the full 2D space and to end up constrained to the 1D surface after all particles have made impact. However, since the reset map only changes the velocity of the ball, its Jacobian will not capture how variations in position are mapped. The saltation matrix does capture this information, and accurately predicts the resulting covariance by accounting for the difference in timing. We show in this tutorial that a similar trend is found in the generalized case for rigid body contact systems where the configuration variation needs to also be considered.

The saltation matrix originally appeared in [Aizerman and Gantmakher, 1958, Eq. 3.5] which

used it to analyze the stability of periodic motions. Other major works include [Filippov, 1988, Pg. 118 Eq. 6], [Ivanov, 1998], and [Leine and Nijmeijer, 2004]. The word "saltation" directly translates to "leap" from Latin – which closely matches to the "jump" name for the hybrid events. Saltation is also used describe how sand particles "jump" along the ground when blown by wind in the desert [Owen, 1964].

Recently, there has been an increasing use of saltation matrices for a wide variety of applications from robotics to computational neuroscience as discussed in Sec. 3.3. The saltation matrix provides necessary information about event driven hybrid systems for stability analysis and creating efficient algorithms. It is crucial for anyone dealing with these types of systems to know about the saltation matrix. In this paper, we present:

- (Sec. 3.3) A survey of where the saltation matrix is being used in a variety of application areas.
- (Sec. 3.4) The definition of the saltation matrix (Sec. 3.4.1), its derivation (Sec. 3.4.2), and a tutorial on how it appears in linear (Sec. 3.4.3) and quadratic forms (Sec. 3.4.4).
- (Sec. 3.5) An example showing the saltation matrix calculation for a simple contact system.
- (Sec. 3.6) The derivation of the saltation matrix for a common class of hybrid systems: rigid body dynamics with contact and friction.

We not only provide a survey and tutorial for the saltation matrix, but also a chain rule derivation of it (App. 3.8.1), a derivation of using it for propagating covariances through hybrid systems (App. 3.8.3), and a derivation of using it for updating the Riccati equation for hybrid systems (App. 3.8.4). All of which has not been found anywhere. Different components of the rigid body dynamics with contact and friction (Sec. 3.6) have been scattered across different texts. In this work, we collect and unify saltation matrices contact and friction into one general model.

3.3 Survey of saltation matrix applications

The saltation matrix has been a valuable tool for analysis and control for a wide variety of fields from general bifurcations theory [Leine and van Campen, 1999; Leine and Van Campen, 2002, 2006; Di Bernardo et al., 2008; Kowalczyk and Glendinning, 2011], power circuits [Hiskens and Pai, 2000; Maity et al., 2007; Giaouris et al., 2008; Okafor et al., 2010a; Ivanov, 2000; Mallik et al., 2020; Bizzarri et al., 2013a, 2011b; Chakrabarty and Kar, 2012; Giaouris et al., 2011; Biggio et al., 2013], rigid body systems [Jiang et al., 2017; Chawla et al., 2022; Banerjee et al., 2009; Revzen and Kvalheim, 2015; Bizzarri et al., 2016; Suda and Banerjee, 2016], to hybrid neuron models [Nobukawa et al., 2015, 2017; Park et al., 2018; Bizzarri et al., 2013b; Coombes et al., 2018; Lai et al., 2018]. Often, the saltation matrix is used to assess the stability of hybrid dynamical systems.

In general, the stability of periodic systems has been extensively analyzed using the saltation matrix [Aizerman and Gantmakher, 1958; Leine and Nijmeijer, 2004; Ivanov, 1998]. The most popular method for analyzing stability of periodic hybrid systems is to analyze the fundamental matrix solution (as shown in Sec. 3.4.3) which is called the monodromy matrix [Asahara and Kousaka, 2018; Muñoz et al., 2019; Mandal et al., 2017; Giaouris et al., 2009; Bizzarri et al., 2014; Bernardo et al., 2008; Elbkosh et al., 2008a; Okafor et al., 2010b; Abusorrah et al., 2017; Daho et al., 2008; Chakrabarty and Kar, 2020; Morel et al., 2011; Mandal, 2013; Elbkosh et al., 2008b; Mandal and Banerjee, 2014; Imrayed, 2012; Wu and Pickert, 2014; El Aroudi et al., 2018; Chen et al., 2019; Bizzarri et al., 2012; Jiang et al., 2017; Gkizas, 2018; Wu et al., 2018; Giaouris et al., 2006; Kuznyetsov, 2021; Nicks et al., 2018; El Aroudi et al., 2020; Bizzarri et al., 2011a; Lopez et al., 2004; Maity and Sahu, 2015; Morel et al., 2022; Fečkan and Pospíšil, 2010; El Aroudi et al., 2017; Chawla et al., 2022; Giaouris et al., 2013; Mandal et al., 2013; Muñoz et al., 2021; Daho, 2012; Banerjee et al., 2011; El Aroudi et al., 2015; Dieci and Elia, 2021; Banerjee et al., 2009; Ageno and Sinopoli, 2005; Cortés et al., 2013]. The monodromy matrix is heavily used in the circuits field specifically for determining local stability of converters (due to their switching nature) and determining if bifurcations will occur [Muñoz et al., 2019; Giaouris et al., 2009; Bizzarri et al., 2014; Bernardo et al., 2008; Elbkosh et al., 2008a; Okafor et al., 2010b; Abusorrah et al., 2017;

Daho et al., 2008; Chakrabarty and Kar, 2020; Morel et al., 2011; Mandal, 2013; Elbkosh et al., 2008b; Mandal and Banerjee, 2014; Imrayed, 2012; Wu and Pickert, 2014; El Aroudi et al., 2018; Chen et al., 2019; Bizzarri et al., 2012; Jiang et al., 2017; Gkizas, 2018; Wu et al., 2018; Giaouris et al., 2006; Kuznyetsov, 2021; Nicks et al., 2018; El Aroudi et al., 2020; Bizzarri et al., 2011a; Lopez et al., 2004; Maity and Sahu, 2015; Morel et al., 2022; Fečkan and Pospíšil, 2010; El Aroudi et al., 2017; Chawla et al., 2022; Giaouris et al., 2013; Mandal et al., 2013; Muñoz et al., 2021; Daho, 2012; Banerjee et al., 2011; El Aroudi et al., 2015; Dieci and Elia, 2021; Banerjee et al., 2009; Ageno and Sinopoli, 2005; Cortés et al., 2013]. See [El Aroudi et al., 2015] for an in depth review for analyzing the stability of switching mode power converters. For more information on bifurcations in periodic systems, see [Müller, 1995; Bockman, 1991] where they discuss Lyapunov exponents (the rate of separation of infinitesimally close trajectories) for hybrid systems.

In [Zhu et al., 2022], the saltation matrix components of the monodromy matrix are used to analyze known robotic stabilizing phenomena such as Raibert stepping controller, paddle juggling, and swing leg retraction. The saltation matrix formulation reveals "shape" parameters, which are terms in the saltation matrix that are independent from the system's dynamics, but have an effect on the stability of the system. We show that these shape parameters can be optimized to generate stable open loop trajectories for complex hybrid systems that undergo periodic orbits.

A more restrictive but stronger form of stability analysis can be done by anlyzing the convergence of neighboring trajectories through hybrid events [Burden et al., 2018b] – where global asymptotic convergence is guaranteed if we are able to show that both the continuous-time flow and the saltation matrix are infinitesimally contractive.

In addition to stability analysis, saltation matrices are also useful for generating controllers. In optimal control, value functions are propagated along a trajectory to generate feedback controllers. For linear time-varying LQR, sensitivity information about a trajectory is used to schedule optimal gains along that trajectory. To implement optimal trajectory tracking for a hybrid system, [Saccon et al., 2014] utilized the saltation matrix to update the sensitivity equation (as shown in Sec. 3.4.4). Due to the sudden jump from the reset map, the optimal controller will also have a jump in the gain

schedule, as first noted in [Schwerin et al., 1996]. Other work further expanding and improving on [Saccon et al., 2014] include [Rijnen et al., 2015, 2017a,b, 2019]. A key concept from these works is "reference spreading" or "reference extension" which creates a new references by extending the pre-transition state through the guard and the post-transition state backwards in time. If there is a mode mismatch, the correct reference extension is selected to track.

Using similar value function approximations and reference spreading, [Kong et al., 2021a] proposed a contact implicit trajectory optimization method by extending these ideas to iterative LQR (iLQR). This approach is able to generate both the nominal state trajectory and the feedback controller without having to specify the mode sequence in advance, as in [Von Stryk, 1999; Kelly, 2017; Schultz and Mombaur, 2009; Posa et al., 2016], or depend on complementarity constraints that are difficult to solve, as in [Posa et al., 2014; Mordatch et al., 2012]. Recently, this hybrid iLQR has also been used as an online Model Predictive Controller (MPC) [Kong et al., 2022a].

State estimation also uses sensitivity information in an analogous way, where the saltation matrix can be used to propagate covariance through a hybrid transition (Sec. 3.4.4). The first paper to do this is [Biggio et al., 2014], which considers covariance propagation for power-spectral density calculation in circuits. We utilize this covariance propagation law to extend Kalman filtering to hybrid dynamical systems [Kong et al., 2021c]. We have also investigated covariance propagation with noisy guards and uncertainty in the reset map [Payne et al., 2022a]. Using covariance propagation is powerful for state estimation because it efficiently maintains the belief of a distribution through hybrid events. In [Kong et al., 2021c], this "Salted Kalman Filter" runs with comparable accuracy to a hybrid particle filter [Koutsoukos et al., 2002] at a fraction of the computation time. The main drawbacks are that it uses a Gaussian approximation, that the entire distribution is propagated instantaneously, and that it is not capable of keeping track of a split distribution that exists near a hybrid transition (whereas non-parametric filters like the particle filter can maintain a non-Gaussian and split distribution).

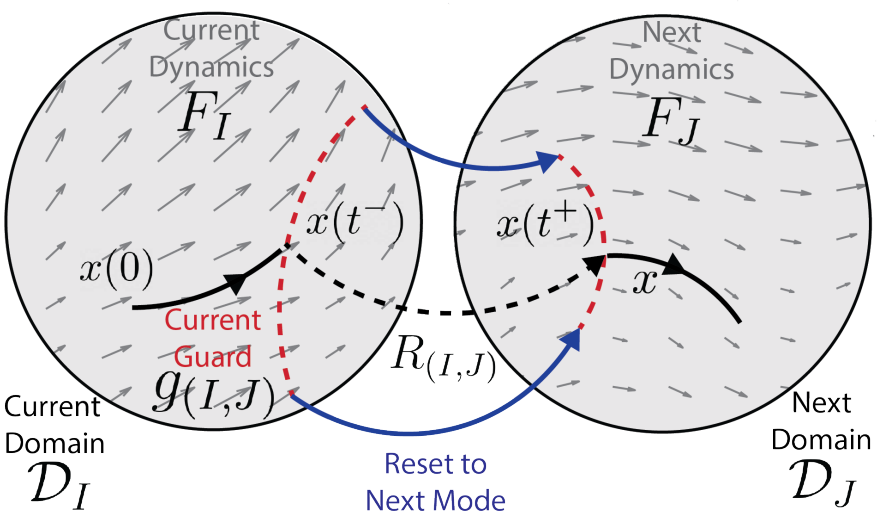

Figure 3.2: An example 2 mode hybrid system where the domains are shown in black circles $\mathcal{D}$, the dynamics are shown with gray arrows F, the guard for the current domain is shown in red dashed g, and the reset from the current mode to the next mode is shown in blue R.

3.4 What is the saltation matrix and how do you use it

In this section, we define the saltation matrix and the broad class of hybrid systems where saltation matrices exist (Sec. 3.4.1), derive the expression of the saltation matrix using a geometric approach (Sec. 3.4.2), show how they are used in linear forms (Sec. 3.4.3), and how they are used in quadratic forms (Sec. 3.4.4).

3.4.1 Saltation matrix definition

In order to carefully define the saltation matrix, we must first choose a hybrid system definition from the many versions that exist, e.g. [Back et al., 1993; Lygeros et al., 2003; Goebel et al., 2009; Johnson et al., 2016a]. Here we closely follow [Kong et al., 2021a].

Definition 1. *A C^r* ***hybrid dynamical system****, for continuity class $r \in \mathbb{N}_{>0} \cup \{\infty, \omega\}$, is a tuple $\mathcal{H} := (\mathcal{J}, \Gamma, \mathcal{D}, \mathcal{F}, \mathcal{G}, \mathcal{R})$ where the parts are defined as:*

1. *$\mathcal{J} := \{I, J, ..., K\} \subset \mathbb{N}$ is the finite set of discrete* ***modes****.*

2. $\Gamma \subset \mathcal{J} \times \mathcal{J}$ *is the set of discrete* ***transitions*** *forming a directed graph structure over* $\mathcal{J}$.

3. $\mathcal{D} := \amalg_{I \in \mathcal{J}} D_I$ *is the collection of* ***domains****, where* D_I *is a* C^r *manifold and the state* $x \in D_I$ *while in mode* I.

4. $\mathcal{F} := \amalg_{I \in \mathcal{J}} F_I$ *is a collection of* C^r *time-varying* ***vector fields****,* $F_I : \mathbb{R} \times D_I \to \mathcal{T}D_I$.

5. $\mathcal{G} := \amalg_{(I,J) \in \Gamma} G_{(I,J)}(t)$ *is the collection of* ***guards****, where* $G_{(I,J)}(t) \subset D_I$ *for each* $(I, J) \in \Gamma$ *is defined as a regular sublevel set of a* C^r *function, i.e.* $G_{(I,J)}(t) = \{x \in D_I | g_{(I,J)}(t, x) \leq 0\}$ *and* $D_x g_{(I,J)}(t, x) \neq 0 \ \forall \ g_{(I,J)}(t, x) = 0$.

6. $\mathcal{R} : \mathbb{R} \times \mathcal{G} \to \mathcal{D}$ *is a* C^r *map called the* ***reset*** *that restricts as* $R_{(I,J)} := \mathcal{R}|_{G_{(I,J)(t)}} : G_{(I,J)}(t) \to D_J$ *for each* $(I, J) \in \Gamma$.

Note that we assume that the control input $u(t, x)$ *is folded into the dynamics* $\mathcal{F}$.

Fig. 3.2 shows an example hybrid system where a hybrid execution may consist of a starting point $x(0)$ in $\mathcal{D}_I$ flowing with dynamics F_I and reaching the guard condition $g(t, x) = 0$, applying the reset map $R(t, x)$ resetting into $\mathcal{D}_J$ and then flowing with the new dynamics F_J.

Here, we would like to understand how perturbations about a nominal trajectory evolve over time. For smooth systems, the perturbations about a nominal trajectory can be approximated to first order using the derivatives of the dynamics $F(t, x)$ with respect to state.

$$\delta \dot{x} = D_x F(t, x) \delta x \tag{3.1}$$

The analogous operation can be done for hybrid systems with time triggered reset maps. In this case, the sensitivity can be found by taking the Jacobian of the reset map, $\delta x^+ = D_x R(t, x) \delta x^-$. However, this method doesn't account for discontinuities that are introduced from state triggered reset maps, when we must consider the change in dynamics between hybrid modes. To account for these discontinuities, the saltation matrix captures how perturbations map through hybrid transitions to first order. The saltation matrix, e.g. [Filippov, 1988, Pg. 118 Eq. 6], [Leine and Nijmeijer, 2004], [Aizerman and Gantmakher, 1958, Eq. 3.5], and [Burden et al., 2018b, Prop. 2]

Definition 2. *The **saltation matrix** for reset $R_{(I,J)}$ is the first order approximation of the variational update at hybrid transitions from mode I to J, defined as*

$$\boxed{\Xi_{(I,J)} := D_x R^- + \frac{\left(F_J^+ - D_x R^- \cdot F_I^- - D_t R^-\right) \cdot D_x g^-}{D_t g^- + D_x g^- \cdot F_I^-}} \tag{3.2}$$

The saltation matrix is an $n_J \times n_I$ matrix, where n_I is the dimension of the states in domain D_I and n_J is the dimension of the states in domain D_J. Note that $\cdot$ in (3.2) *represents matrix multiplication, and in particular results in an outer-product between the terms in the parentheses and $D_x g^-$ to get a rank-1 correction to the Jacobian of the reset map. The following evaluations are made for the terms in the saltation matrix*

$$F_I^- = F_I(t^-, x(t^-)) \tag{3.3}$$

$$F_J^+ = F_J(t^+, x(t^+)) \tag{3.4}$$

$$x(t^+) = R_{(I,J)}(t^-, x(t^-)) \tag{3.5}$$

$$D_x R^- = D_x R_{(I,J)}(t^-, x(t^-)) \tag{3.6}$$

$$D_t R^- = D_t R_{(I,J)}(t^-, x(t^-)) \tag{3.7}$$

$$D_x g^- = D_x g_{(I,J)}(t^-, x(t^-)) \tag{3.8}$$

$$D_t g^- = D_t g_{(I,J)}(t^-, x(t^-)) \tag{3.9}$$

where x impacts the guard $G_{(I,J)}(t)$ at time $t = t^- = t^+$, where t^- is the pre-impact time, t^+ is the post-impact time, and $x(t^\pm)$ is the limiting value of the signal x from the left $(-)$ *or right* $(+)$*. Note that by D_t in* (3.7) *and* (3.9) *we are referring to the derivative with respect to the first coordinate (and not the time dependence of x, which is captured by other terms).*

The saltation matrix maps perturbations to first order from pre-transition $\delta x(t^-)$ to post-transition $\delta x(t^+)$ as

$$\delta x(t^+) = \Xi_{(I,J)} \delta x(t^-) + h.o.t. \tag{3.10}$$

where h.o.t. represents higher order terms.

The saltation matrix in (3.2) returns a good first order approximation when the following assumptions are true, as listed in [Burden et al., 2018b]

1. Guards and resets are differentiable

2. Trajectories cannot undergo an infinite number of resets in finite time (no Zeno)

3. Trajectories must be transverse to the guard at an event

$$\frac{d}{dt} g_{(I,J)}(t, x(t)) = D_t g^- + D_x g^- F_I^- < 0 \tag{3.11}$$

The saltation matrix relies on differentiating the guards and resets so they must be differentiable. Transversality ensures that neighboring trajectories impact the same guard as the nominal if the impact point does not lie in any other guard. Transversality also ensures the denominator in (3.2) does not approach zero.

The saltation matrix maps perturbations before and after the reset, but in what case does the mapping become an identity transformation? Knowing when the saltation matrix is identity is important because we can simplify the computation and analysis for these events.

The most common reason for a saltation matrix to become identity is if both of these conditions are true:

1. the reset map is an identity transformation, $R = I_{n\times n}$, where n is the dimension of the state x in both D_I and D_J

2. the dynamics in both modes are the same before and after impact, $F_I^- = F_J^+$

$$\left.\begin{array}{l} R = I_{n\times n} \\ F_I^- = F_J^+ \end{array}\right\} \implies \Xi = I_{n\times n} \tag{3.12}$$

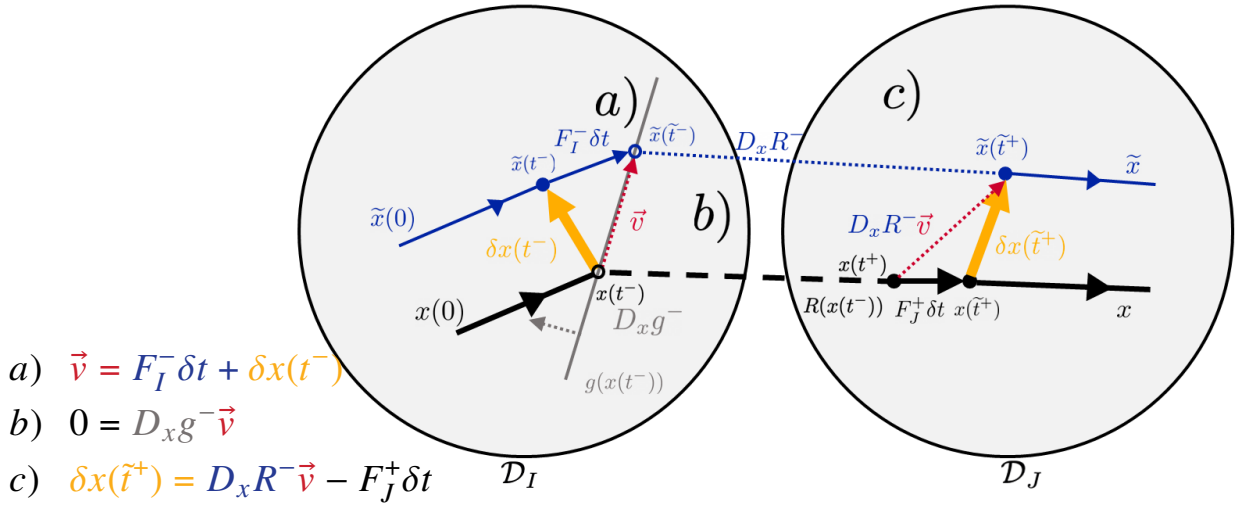

Figure 3.3: Linearizations made about the nominal trajectory shown in black where a perturbation is shown in yellow and the perturbed trajectory is shown in blue. At a) describes $\vec{v} = F_I^- \delta t + \delta x(t^-)$. At b) we get the guard condition $0 = D_x g^-(\delta x(t^-) + F_I^- \delta t)$. At c) we get $\delta x(\tilde{t}^+) = D_x R^- \vec{v} - F_J^+ \delta t$. Here δt is positive (late transition)

If the reset map is an identity transformation, then $D_x R$ is also identity and $D_t R$ is zero. Using these conditions to simplify the expression in (3.2)

$$\Xi_{(I,J)} := I_{n\times n} + \frac{\left(F_J^+ - I_{n\times n} \cdot F_I^- - 0_{n\times n}\right) \cdot D_x g^-}{D_t g^- + D_x g^- \cdot F_I^-} = I_{n\times n} \tag{3.13}$$

3.4.2 Saltation matrix derivation

Here we derive the expression for the saltation matrix (3.2), following the geometric derivation from [Leine and Nijmeijer, 2004]. In this work, we add reset maps into the derivation. There are many alternate ways to derive (3.2): a derivation using the chain rule is included in Appendix 3.8.1, and a derivation using a double limit can be found in [Burden et al., 2018b]. For simplicity of expression, here we assume a time invariant hybrid system, where the D_t terms drop out, but the full expression can be similarly derived by linearizing the time varying terms as well (as done in Appendix 3.8.1).

Suppose the nominal trajectory of interest is x as shown in Fig. 3.3. The trajectory starts in

mode I and goes through a hybrid transition to mode J at time t. The saltation matrix is a first-order approximation, and as such we take the flow as a constant in each mode, evaluated at time $t^{\pm}$ as in (3.3) and (3.4) such that

$$x(t+\delta t) = x(t^-) + F_I^- \delta t \qquad \text{in mode } I \tag{3.14}$$

$$x(t+\delta t) = x(t^+) + F_J^+ \delta t \qquad \text{in mode } J \tag{3.15}$$

We also linearize the reset and guard at t^- as in (3.6) and (3.8), such that

$$\bar{R}(t+\delta t, x+\delta x) = R_{(I,J)}(t^-, x(t^-)) + D_x R^- \delta x + D_t R^- \delta t \tag{3.16}$$

$$\bar{g}(t+\delta t, x+\delta x) = g_{(I,J)}(t^-, x(t^-)) + D_x g^- \delta x + D_t g^- \delta t \tag{3.17}$$

where $\bar{R}$ and $\bar{g}$ are the linear maps.

Trajectories that are perturbed δx away are labeled as $\widetilde{x}$. Perturbations can lead to changing the impact time $\delta t = \widetilde{t} - t$ where t is the original impact time and $\widetilde{t}$ is the perturbed impact time. If $(\delta t > 0)$ then the solution stays longer in the previous hybrid mode and if $(\delta t < 0)$ then the solution transitions early. For simplicity of notation, in this section we assume the perturbed trajectory reaches the guard surface later, but all of the analysis works equally well for earlier transitions, and the same expression (3.2) results, as shown in Appendix 3.8.2.

The perturbation at the pre-impact time of the nominal trajectory t^- and the post-impact time of the perturbed trajectory $\widetilde{t}^+$ are

$$\delta x(t^-) = \widetilde{x}(t^-) - x(t^-) \tag{3.18}$$

$$\delta x(\widetilde{t}^+) = \widetilde{x}(\widetilde{t}^+) - x(\widetilde{t}^+) \tag{3.19}$$

where $\widetilde{x}(t^-)$ is the perturbed trajectory following the previous mode dynamics until time t^-. We would like to write (3.19) in terms of the nominal trajectory at time of impact $x(t^-)$ and just after impact $x(t^+)$. Using (3.18) and (3.14), we can write $\widetilde{x}(\widetilde{t}^-)$ in terms of the flow before impact $F_I^- \delta t$

and the perturbation before impact $\delta x(t^-)$

$$\widetilde{x}(\widetilde{t^-}) = x(t^-) + \delta x(t^-) + F_I^- \delta t \tag{3.20}$$

Note that the summation of $\delta x(t^-) + F_I^- \delta t$ is labeled as $\vec{v}$ in Fig. 3.3. By using the linearized reset map (3.16) and the perturbation expressed in terms of the nominal trajectory (3.20), we can evaluate the reset at $\widetilde{x}(\widetilde{t^-})$ in terms of the nominal state $x(t^-)$, the initial perturbation $\delta x(t^-)$, and the different in impact time δt

$$\widetilde{x}(\widetilde{t^+}) = R(t^-, x(t^-)) + D_x R^- \left(\delta x(t^-) + F_I^- \delta t\right) + D_t R^- \delta t \tag{3.21}$$

To get the final term in (3.19), we can use the constant flow after the reset (3.14) to obtain $x(\widetilde{t^+})$

$$x(\widetilde{t^+}) = R(t^-, x(t^-)) + F_J^+ \delta t \tag{3.22}$$

By combining (3.19), (3.21), and (3.22) we can now write $\delta x(\widetilde{t^+})$ as a linear function of $\delta x(t^-)$ and δt

$$\begin{aligned} \delta x(\widetilde{t^+}) &= R(t^-, x(t^-)) + D_x R^- \left(\delta x(t^-) + F_I^- \delta t\right) && (3.23) \\ &\quad + D_t R^- \delta t - \left(R(t^-, x(t^-)) + F_J^+ \delta t\right) \\ &= D_x R^- \delta x(t^-) + \left(D_x R^- F_I^- + D_t R^- - F_J^+\right) \delta t && (3.24) \end{aligned}$$

This step is highlighted by the vector addition in Fig. 3.3 Eq. c.

Next, we solve for δt as a function of $\delta x(t^-)$. We can use the linear property of the guard (3.17) and the perturbation expressed in terms of the nominal trajectory (3.20) to rewrite the guard evaluated at $\widetilde{x}(\widetilde{t^-})$ as a function of the nominal (and noting that $g(t^-, x(t^-)) = 0$)

$$0 = g(t^-, x(t^-)) + D_x g^-(\delta x(t^-) + F_I \delta t) + D_t g^- \delta t \tag{3.25}$$

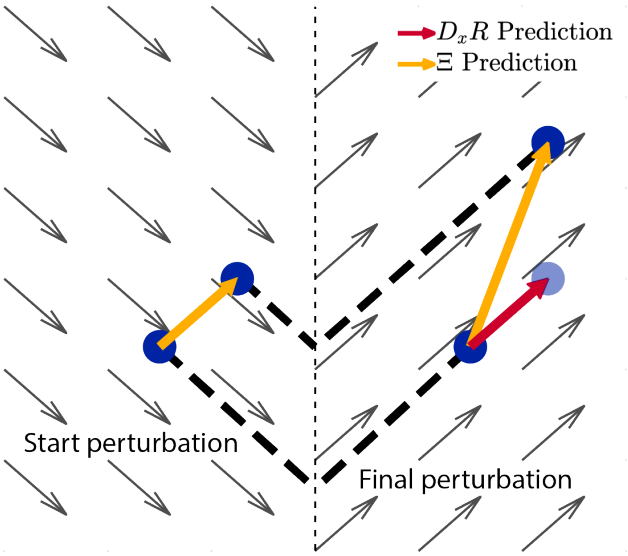

Figure 3.4: Constant flow hybrid system with identity reset map. The Jacobian of the reset map D_xR predicts no variational changes whereas using the saltation matrix Ξ predicts the correct variational changes.

$$= D_x g^- \delta x(t^-) + (D_x g^- F_I^- + D_t g^-)\delta t \tag{3.26}$$

This expansion shows up in Fig. 3.3 as Eq. b. Now we write δt as a function of $\delta x(t^-)$

$$\delta t = -\frac{D_x g^-}{D_x g^- F_I^- + D_t g^-}\delta x(t^-) \tag{3.27}$$

Plugging this δt into (3.24) and solving for $\delta x(\widetilde{t^+})$ in terms of $\delta x(t^-)$

$$\begin{aligned}\delta x(\widetilde{t^+}) &= D_x R^- \delta x(t^-) + \frac{\left(F_J^+ - D_x R^- F_I^- - D_t R^-\right) D_x g^-}{D_x g^- F_I^- + D_t g^-}\delta x(t^-) \\ &= \Xi_{(I,J)}\delta x(t^-)\end{aligned} \tag{3.28}$$

where Ξ is the saltation matrix, as in (3.10).

3.4.3 Linear forms for the saltation matrix

Understanding how trajectories behave near a trajectory of interests is crucial for many algorithms which rely on linearizations. The sensitivity equation describes how these perturbations evolve over time. For a hybrid system, the time evolution simply applies the standard smooth sensitivity equation (3.1) for the smooth dynamics and the saltation matrix when a hybrid transition occurs (3.10). For a transition from mode I to mode J at time t^- we get

$$\delta\dot{x}(t) = A_I \delta x(t) \qquad s.t.\ t \leq t^- \tag{3.29}$$

$$\delta x(t^+) = \Xi_{(I,J)} \delta x(t^-) \qquad s.t.\ t = t^- \tag{3.30}$$

$$\delta\dot{x}(t) = A_J \delta x(t) \qquad s.t.\ t \geq t^+ \tag{3.31}$$

where we denote by $A_I := D_x F_I(t, x)$ the Jacobian of the dynamics with respect to state. An example is shown in Fig. 3.4, where the sensitivity is only updated by the saltation matrix because the flows are constant in both modes. We see that if only the Jacobian of the reset is used, we get the incorrect prediction. Note that sensitivity of hybrid systems are extensively analyzed in [Hiskens and Pai, 2000] and [Saccon et al., 2014].

Many algorithms consider finite, discrete timesteps. This makes the analysis slightly different, since the hybrid transition will most likely not occur exactly at the boundary of a discrete timestep. In this case, we apply what we call a "sandwich" method, where we apply 3 (or more) smaller discrete updates during a timestep which has a hybrid transition. Consider a time interval, from t_k to $t_k + \Delta$ over which a single reset occurs at time $t_k + \Delta_1$. The system spends Δ_1 in the first mode and $\Delta_2 = \Delta - \Delta_1$ in the second mode. Let $A_{I,\Delta}$ be the Jacobian of the dynamics discretized to time duration Δ. Then a discrete approximation of the forward dynamics is,

$$\boxed{\delta x(t_{k+1}) = A_{J,\Delta_2} \Xi_{(I,J)} A_{I,\Delta_1} \delta x(t_k)} \tag{3.32}$$

which holds to first order. This result comes from the fundamental matrix solution [Leine and

Nijmeijer, 2004, Eq. 7.22].

Extending this idea, consider a periodic orbit of period Δ, such that $x(t) = x(t + \Delta)$. In this case, the fundamental matrix solution is called the monodromy matrix. If the orbit passes through n modes labeled $i = 1, 2, \ldots, n$, with mode periods Δ_i, then we define the monodromy matrix Φ, [Leine and Nijmeijer, 2004, Eq. 7.28], [Wang and Hale, 2001, Eq. 1], and [Asahara and Kousaka, 2018, Eq. 12],

$$\Phi = \Xi_{(n,1)} A_{n,\Delta_n} \Xi_{(n-1,n)} A_{n-1,\Delta_{n-1}} \cdots \Xi_{(1,2)} A_{1,\Delta_1} \tag{3.33}$$

$$\delta x(t + \Delta) = \Phi \delta x(t) \tag{3.34}$$

which holds to first order. This monodromy matrix captures the change in perturbations from one cycle through the orbit to the next and the eigenvalues (called Floquet multipliers [Leine and Nijmeijer, 2004]) determine the stability of the trajectory. Namely, if the eigenvalues all have magnitude less than one then the system is asymptotically stable [Leine and Nijmeijer, 2004].

Related to the monodromy matrix, a common technique to analyze stability of periodic systems is to analyze the return/Poincaré map. We will give a brief introduction to the Poincaré map, but more details can be found in [Leine and Nijmeijer, 2004].

A Poincaré map $P(x)$ converts the continuous-time system to a discrete map. For an autonomous system with n states and a limit cycle L, the Poincaré map is defined about a fixed point x^* on L where we define an $n - 1$ dimensional hyper-plane transverse to the flow F called the Poincaré section Ω, with $x^* \in \Omega$. The Poincaré map tells us how points move along the Poincaré section after one cycle ($P : \Omega \mapsto \Omega$). Stability of the fixed point is often computed by taking the Jacobian of the Poincaré map and by analyzing its eigenvalues. If the eigenvalues are less than one (the requirements for stability for a discrete system), the fixed point x^* is stable.

Note that for the autonomous case, the dimensionality of the system is reduced by one due to the embedding. For the non-autonomous case, we can no longer define a Poincaré section in state space because it does not regard the dependency on time. Instead, the trajectory is augmented with

a periodic time coordinate on S^1, and the Poincaré section is now defined to be at the end of each period T. In this case, the Poincaré map and its Jacobian are in the full n space, as the Poincaré section is defined on the added time coordinate.

Suppose we also construct a monodromy matrix for a cycle that starts and ends at the fixed point x^* for one cycle. In the autonomous case, the monodromy matrix will have the same eigenvalues as the Jacobian of the Poincaré map with an additional eigenvalue equal to one. This is because the monodromy matrix is still in the full n space, and perutrbations along the direction of the flow are invariant. In the non-autonomous case, the monodromy matrix and the Jacobian of the Poincaré map are equivalent, and so sometimes the monodromy matrix is defined to be the Jacobian of the Poincaré map [El Aroudi et al., 2015].

If the system is autonomous and periodic, using the Poincaré map might be more practical because the analysis is simplified by the reduction of a state variable as shown for passive dynamic walkers [McGeer, 1990]. However, the benefits of the fundamental matrix solution (monodromy matrix when periodic) is that it can be used to analyze non-cyclical behaviors. This is especially important when designing dynamic behaviors that are drastically different like for parkour or dynamic grasps.

3.4.4 Quadratic forms for the saltation matrix

Similar to linear forms, quadratic forms are often used in algorithms which rely on linearizations and the evolution of quadratic forms behave similarly to linear forms. For quadratic forms, we first zoom in at the moment of impact. We assume the input does not instantaneously change, $u(t^+) = u(t^-)$, so that we can just look at variations in state. There are 2 main updates we will look at, quadratic form of the vector (covariance) as well as the covector (value approximation).

For covariances, recall that the update law for covariance Σ through a discretized smooth system is

$$\Sigma(t_{k+1}) = A_\Delta \Sigma(t_k) A_\Delta^T \tag{3.35}$$

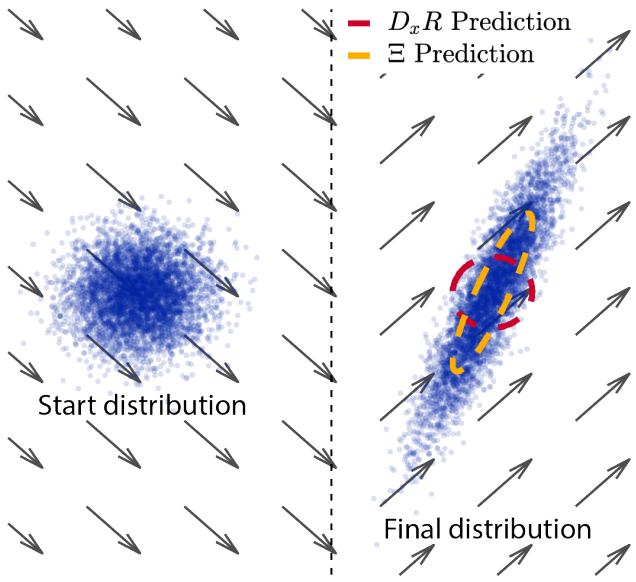

Figure 3.5: Constant flow hybrid system with identity reset map. The Jacobian of the reset map $D_x R$ predicts no covariance change whereas using the saltation matrix Ξ predicts the correct covariance.

e.g. as in [Welch and Bishop, 1995, Eqn. 1.10] or [Julier and Uhlmann, 2004, Eqn. 6]. Similarly, at hybrid transitions we use the saltation matrix in an analagous way (see derivation in Appendix 3.8.3 and validation experiments in Appendix 3.8.5)

$$\boxed{\Sigma(t^+) = \Xi_{(I,J)} \Sigma(t^-) \Xi_{(I,J)}^T} \tag{3.36}$$

[Biggio et al., 2014, Eqn. 17], [Kong et al., 2021c, Eqn. 7], which holds to first order. As with linear forms, the sandwich method (3.32) can be applied to get the covariance propagation for an entire discrete timestep

$$\Sigma(t_{k+1}) = A_{J,\Delta_2} \Xi_{(I,J)} A_{I,\Delta_1} \Sigma(t_k) A_{I,\Delta_1}^T \Xi_{(I,J)}^T A_{J,\Delta_2}^T \tag{3.37}$$

[Kong et al., 2021c, Eqn. 19]. An example is shown in Fig. 3.5, where the covariance is only updated by the saltation matrix because the flows are constant in both modes. We see that if only the Jacobian of the reset is used, we get the incorrect end covariance. Algorithms, such as a Kalman filter [Kong et al., 2021c], that propagate covariances with the dynamics can utilize this update law.

In the case of propagating a quadratic form of a co-vector, the transposes flip sides similar to how a co-vector quadratic form propagates in the smooth domain

$$P(t_k) = A_\Delta^T P(t_{k+1}) A_\Delta \tag{3.38}$$

e.g. as in [Bertsekas, 2012, Eqn. 3.40]. The covector propogation law for the hybrid transition uses the saltation matrix in an analagous way (see derivation in Appendix 3.8.4)

$$\boxed{P(t^-) = \Xi_{(I,J)}^T P(t^+) \Xi_{(I,J)}} \tag{3.39}$$

[Rijnen et al., 2015, Eqn. 23], [Kong et al., 2021a, Eqn. 31]. The main application for the covector case is in the update to the Riccati equation or Bellman update, e.g. in LQR [Kong et al., 2021a; Rijnen et al., 2015].

3.5 Example: Calculating the saltation matrix for a ball dropping on a slanted surface

One of the simplest examples of a hybrid system is a 2D point mass (ball) falling and hitting a flat surface, as shown in Fig. 3.1. Intuitively, the impact should eliminate variations normal to the constraint in both position and velocity. In this example, we show the computations for the saltation matrix and how it removes variations normal to the constraint.

3.5.1 Dynamics definition

In this example, we give a summary of the dynamics for this system, but an in-depth derivation for the general form is given in Sec. 3.6. The horizontal, vertical positions and their velocities are defined to be the states of the system $x = [q, \dot{q}] = [q_1, q_2, \dot{q}_1, \dot{q}_2]$. The ball has mass m and acceleration due to gravity a_g. For the sake of demonstrating how inputs are handled, the ball is

fully actuated with control inputs along the configuration coordinates (u_1, u_2). We consider two cases on friction, one that assume frictionless sliding when in contact with the surface, i.e. the sliding friction coefficient is zero, $\mu_s = 0$, and one where the friction is sufficient to prevent sliding, i.e. the ball sticks to a spot.

The ball impacts a sloped surface parameterized by an angle θ, where the position constraint is defined by the guard function

$$g_{(U,S)}(t, x) = \sin(\theta) q_1 + \cos(\theta) q_2 = 0 \tag{3.40}$$

where U is the unconstrained mode and S is the constrained sliding mode (the ball can slide tangentially along the constraint surface). The resulting velocity constraint Jacobian J_S in the sliding mode is

$$J_S(q) = D_q g_{(U,S)}(t, x) = \begin{bmatrix} \sin(\theta) & \cos(\theta) \end{bmatrix}, \quad s.t.\ J_S \dot{q} = 0 \tag{3.41}$$

The unconstrained mode dynamics are defined by ballistic motion

$$\dot{x} = F_U(t, x) = \left[\dot{q}_1, \dot{q}_2, \frac{u_1}{m}, \frac{u_2 - a_g m}{m} \right]^T \tag{3.42}$$

The hybrid guard for impact is defined by the constraint $g_{(U,S)}(q) \leq 0$, i.e when the constraint is met, the impact reset map is applied. The reset map is defined by plastic impact, which enforces the velocity constraint.

$$R_{(U,S)}(t, x) = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \cos^2(\theta) - \dot{q}_2 \cos(\theta) \sin(\theta) \\ \dot{q}_2 \sin(\theta)^2 - \dot{q}_1 \sin(\theta) \cos(\theta) \end{bmatrix} \tag{3.43}$$

The constrained mode dynamics are found by solving the ballistic dynamics while maintaining

the velocity constraint

$$\dot{x} = F_S(t,x) = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \frac{u_1 \cos^2(\theta)}{m} - \frac{u_2 \cos(\theta)\,\sin(\theta)}{m} + \frac{g\,m\,\cos(\theta)\,\sin(\theta)}{m} \\ \frac{u_2 \sin^2(\theta)}{m} - \frac{g\,m \sin^2(\theta)}{m} - \frac{u_1 \cos(\theta)\,\sin(\theta)}{m} \end{bmatrix} \tag{3.44}$$

In the case where we want sticking friction to be applied in a third mode C, we add a no slip condition to (3.41)

$$J_C = \begin{bmatrix} -\cos(\theta), \sin(\theta) \\ \sin(\theta), \cos(\theta) \end{bmatrix}, \quad s.t.\ J_C \dot{q} = 0 \tag{3.45}$$

such that the constrained dynamics become

$$\dot{x} = F_C(t,x) = [\dot{q}_1, \dot{q}_2, 0, 0]^T \tag{3.46}$$

The reset map ends up eliminating all velocities

$$R_{(U,C)}(t,x) = [q_1, q_2, 0, 0]^T \tag{3.47}$$

Note that this mode is fully constrained and the ball will just stick to the surface (as $\dot{q} = 0$ after impact).

3.5.2 Saltation matrix calculation

To compute the saltation matrix, the Jacobians of the guard and reset map with respect to state must be computed first. The Jacobian of the guard is simply the velocity constraint Jacobian padded

with zeros for each velocity coordinate

$$D_x g_{(U,S)}(t,x) = [J_S, 0_{1\times 2}] = [\sin(\theta), \cos(\theta), 0, 0] \tag{3.48}$$

The Jacobian of the reset map is

$$D_x R_{(U,S)}(t,x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos^2(\theta) & -\cos(\theta)\sin(\theta) \\ 0 & 0 & -\cos(\theta)\sin(\theta) & \sin^2(\theta) \end{bmatrix} \tag{3.49}$$

The saltation matrix is then computed by plugging in each component, (3.42)–(3.49), into the definition, (3.2), to get,

$$\boxed{\Xi_{(U,S)} = \begin{bmatrix} \Omega_{(U,S)} & 0_{2\times 2} \\ 0_{2\times 2} & \Omega_{(U,S)} \end{bmatrix}} \tag{3.50}$$

where $\Omega_{(U,S)}$ is a block element consisting of

$$\Omega_{(U,S)} = \begin{bmatrix} \cos^2(\theta) & -\cos(\theta)\sin(\theta) \\ -\cos(\theta)\sin(\theta) & \sin^2(\theta) \end{bmatrix} \tag{3.51}$$

For the sticking saltation matrix, similar calculations are made as in the sliding case

$$D_x g_{(U,C)}(t,x) = [\sin(\theta), \cos(\theta), 0, 0] \tag{3.52}$$

Note that the guard condition did not change which results in having the same Jacobian of the guard

as the sliding case. The Jacobian of the reset map is

$$D_x R_{(U,C)}(t,x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.53}$$

The resulting saltation matrix becomes

$$\boxed{\Xi_{(U,C)} = \begin{bmatrix} \Omega_{(U,C)} & 0_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} \end{bmatrix}} \tag{3.54}$$

where $\Omega_{(U,C)}$ is a block element consisting of

$$\Omega_{(U,C)} = \frac{1}{\dot{q}_2 \cos(\theta) + \dot{q}_1 \sin(\theta)} \begin{bmatrix} \dot{q}_2 \cos(\theta) & -\dot{q}_1 \cos(\theta) \\ -\dot{q}_2 \sin(\theta) & \dot{q}_1 \sin(\theta) \end{bmatrix} \tag{3.55}$$

3.5.3 Saltation matrix analysis

Interestingly, the saltation matrix for the sliding case $\Xi_{(U,S)}$ is a block diagonal matrix with a repeating block element as shown in (3.50)–(3.51), This implies that the variations in position get mapped equivalently to variations in velocity. By analyzing the eigenvalues and eigenvectors of this block, we see that variations in the direction of the constraint are eliminated, while variations tangential to the constraint do not change. The eigenvalues and corresponding eigenvectors of this block are,

$$\begin{aligned} \lambda_0 &= 0, & \lambda_1 &= 1 \\ v_0 &= \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix}, & v_1 &= \begin{bmatrix} -\cos\theta \\ \sin\theta \end{bmatrix} \end{aligned} \tag{3.56}$$

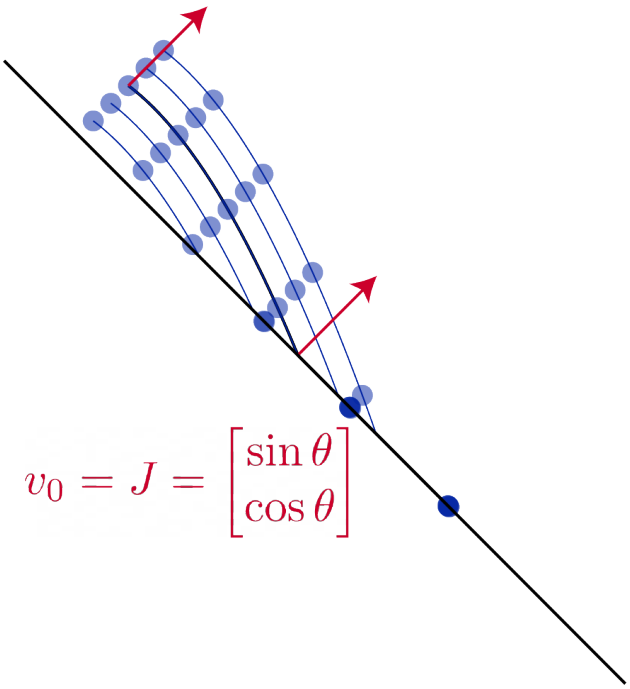

Figure 3.6: Ball drop example with sliding friction which illustrates that position variations in the direction of the constraint will be eliminated. v_0 is the eigenvector associated with the zero eigenvalue and θ is the angle of the surface.

The first eigenvalue is zero, so any variation in the direction of its eigenvector is eliminated. Note that that this eigenvector is exactly the velocity constraint Jacobian, $J_S = [\sin(\theta), \cos(\theta)]$. Thus, variations off the constraints for both position and velocity go to zero, i.e. no variations lie above or below the surface once impact is made, as shown in Fig. 3.6. Note that while the reset map zeros out velocity in this direction (and so this effect arises from the $D_x R$ term), the reset map has no effect on positions. For the position block, the effect in the constraint direction arises from the final $D_x g$ term in the numerator of the second term in (3.2), as in (3.48).

The second eigenvalue is identity, so variations in the direction of its eigenvector do not change. We see that the eigenvector is tangent to the constraint direction $[-\cos(\theta), \sin(\theta)]$, so variations tangential to the constraint are not affected by the impact. In fact, the saltation matrix is always just a rank one update to $D_x R$ in the direction of $D_x g$ and all other directions to be unaffected. Although this is a simple example, this block matrix structure exists for all rigid body systems with unilateral constraints, as explored in the next section.

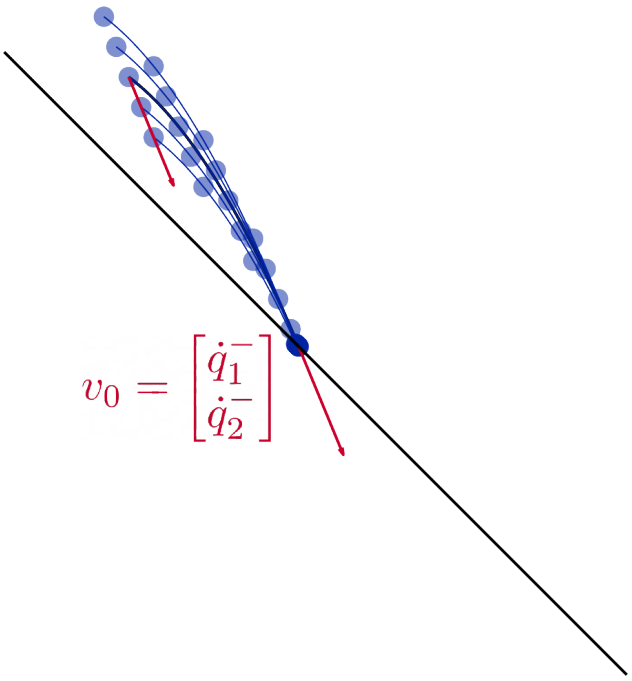

Figure 3.7: Ball drop example with sticking friction which illustrates that position variations in the direction of pre-impact velocity will be eliminated. v_0 is the eigenvector associated with the zero eigenvalue, q_1 is the horizontal configuration, and q_2 is the vertical configuration.

For plastic impact into sticking, (U, C), we expect the variations in configuration to map differently than velocity variations. This is because the tangential constraint is only applied to the velocity component and not the configuration, whereas in the normal direction, both position and velocity had the same constraint – being on the constraint surface. The sticking saltation matrix $\Xi_{(U,C)}$ reflects this change, where there is no longer a repeated element in the block diagonal. Instead, the only nonzero component is how variations in position map onto the constraint surface (3.54)–(3.55). The velocity components are all zero because velocity is fully constrained to zero. Again, we analyze the non-zero block component by computing the eigenvalues and corresponding eigenvectors,

$$\begin{aligned} &\lambda_0 = 0, && \lambda_1 = 1 \\ &v_0 = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}, && v_1 = \begin{bmatrix} -\cos(\theta) \\ \sin(\theta) \end{bmatrix} \end{aligned} \tag{3.57}$$

Similar to the sliding case, variations tangential to the constraint are preserved. However, the zero eigenvector is different. Configuration variations that are in the same direction as the impact velocity will disappear. Fig. 3.7 illustrates this idea, where position variations in the direction of pre-impact velocities are eliminated. This is intuitive because the ball impacting earlier or later has no effect if the variation is in line with the impact velocity, it will hit the same contact point and stick.

3.6 Saltation matrices for generalized rigid body systems with unilateral constraints

For rigid body systems with contacts, the hybrid modes are the enumeration of different contact conditions. In this work we adopt the same definition for a hybrid system from [Johnson et al., 2016a]. These systems are also considered in [Johnson and Koditschek, 2013; Pace and Burden, 2017; Johnson, 2021]. In this section, we define the dynamics of these systems and calculate the saltation matrix of all the common mode transitions for a single constraint.

3.6.1 Dynamics derivation

In the following examples, we consider four modes, illustrated in Fig. 3.8, where we label the unconstrained mode approaching the constraint surface to be U, the unconstrained mode leaving the surface V, a constrained mode C, and a sliding with friction mode S. The reason we include both U and V is to ensure that elastic impact is not defined with a self-reset and so we can avoid degenerate impacts just after liftoff, when the velocity is not approaching the constraint but the guard condition is satisfied $g_n \leq 0$, especially when using numerical integration.

The states of the system are the configuration coordinates q and their velocities $\dot{q}$, such that $x := [q, \dot{q}]^T$. The dimension of the configuration q is defined to be m, while the dimension of the state space x is $n = 2m$. Contacts between rigid bodies are regulated through unilateral constraints $g_n(t, x) \geq 0$. Note that $g_n(t, x)$ only depends on the configuration q and not the velocity. When

rigid bodies are in contact they must satisfy $g_n(x) = 0$.

The Jacobian of g_n with respect to the configuration coordinates is defined to be $J_n := D_q g_n(t, x)$. In the sliding mode, the constraint Jacobian consists of just this normal direction constraint, $J_S = J_n$. However, if the no slip condition is added, the constrained mode C has a constraint Jacobian consisting of

$$J_C = \begin{bmatrix} J_t \\ J_n \end{bmatrix} \tag{3.58}$$

For unconstrained modes, J is empty.

In any mode, the following acceleration constraint is applied based on that mode's J to maintain the active constraints until the next guard

$$J(t, x)\ddot{q} + \dot{J}(t, x)\dot{q} = 0 \tag{3.59}$$

The equations of motion for each mode are defined by the constrained manipulator dynamics [Murray et al., 2017], where this constraint is combined with Lagrangian dynamics

$$\begin{bmatrix} M & J^T \\ J & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} \Upsilon - N \\ 0 \end{bmatrix} - \begin{bmatrix} C \\ \dot{J} \end{bmatrix} \dot{q} \tag{3.60}$$

[Johnson and Koditschek, 2013, Eqn. 33] where λ is the constraint force (Lagrange multiplier), $M(q)$ is the mass matrix, $C(q, \dot{q})$ is the Coriolis matrix, $\Upsilon(u)$ the input vector, and $N(q, \dot{q})$ are the other nonlinear forces such as gravity and sliding friction.

To help with the following equations, the † notation from [Johnson et al., 2016a, Eqn. 8] is adopted where in each mode

$$\begin{bmatrix} M^\dagger & J^{\dagger T} \\ J^\dagger & \Lambda \end{bmatrix} := \begin{bmatrix} M & J^T \\ J & 0 \end{bmatrix}^{-1} \tag{3.61}$$

From this definition we can derive a number of identities, in particular,

$$M^{\dagger}M = I_{m\times m} - J^{\dagger T}J \tag{3.62}$$

[Johnson et al., 2016a, Eqn. 11], which will be helpful in simplifying the saltation matrix expressions.

With this notation we can now expand the state space dynamics as

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{\dagger}\left(\Upsilon - N - C\dot{q}\right) - J^{\dagger T}\dot{J}\dot{q} \end{bmatrix} \tag{3.63}$$

[Johnson et al., 2016a, Eqn. 75] where each † component is different depending on the hybrid mode based on J. For the unconstrained case, $M^{\dagger} = M^{-1}$.

When sliding, we assume Couloumb friction – frictional forces in the tangential direction F_t (included in N) are applied to resist sliding motion proportional to the normal constraint force, λ_n, and in the direction resisting the sliding velocity, $v_t = J_t\dot{q}$

$$F_t = \mu_s\lambda_n\frac{J_t\dot{q}}{\|J_t\dot{q}\|} = \mu_s\lambda_n\frac{v_t}{\|v_t\|} \tag{3.64}$$

where μ_s is the sliding coefficient of friction.

When a contact constraint is added, for example the normal surface constraint g_n, we apply an impact law, $J_n\dot{q}^{+} = -eJ_n\dot{q}^{-}$ (where the coefficient of restitution $e = 1$ is perfectly elastic and $e = 0$ is perfectly plastic), along with the impulse momentum equation to get

$$\begin{bmatrix} \dot{q}^{+} \\ \hat{p} \end{bmatrix} = \begin{bmatrix} M & J_n^T \\ J_n & 0 \end{bmatrix}^{-1}\begin{bmatrix} M \\ -eJ_n \end{bmatrix}\dot{q}^{-} = \begin{bmatrix} M_n^{\dagger} & J_n^{\dagger T} \\ J_n^{\dagger} & \Lambda_n \end{bmatrix}\begin{bmatrix} M \\ -eJ_n \end{bmatrix}\dot{q}^{-} \tag{3.65}$$

[Johnson et al., 2016a, Eqn. 23], [Johnson, 2021], where $\hat{p}$ is the impulse magnitude vector. Since the positions do not change instantaneously, the state space reset map for elastic, frictionless impact

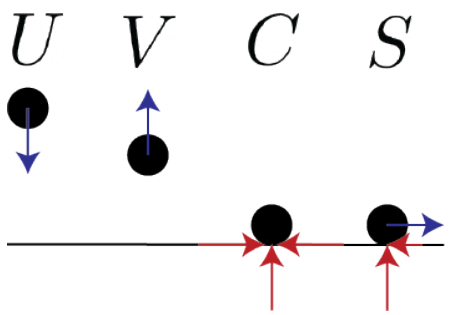

Figure 3.8: Depicting the different rigid body hybrid modes considered where blue arrows depict velocities and red arrows depict forces. U is the unconstrained mode with approaching velocity to the constraint, V is the unconstrained mode with separating velocity, C is the constrained mode, and S is the sliding mode on the constraint. A single planar point is shown here, but the system may have additional degrees of freedom.

from mode U to mode V is

$$x^+ = \begin{bmatrix} q^+ \\ \dot{q}^+ \end{bmatrix} = R_{(U,V)}(t, x^-) = \begin{bmatrix} q^- \\ M_n^\dagger M \dot{q}^- - e J_n^{\dagger T} J_n \dot{q}^- \end{bmatrix} \tag{3.66}$$

The plastic, frictionless impact reset map into mode S follows (3.66) but with $e = 0$ (and written with J_S for mode S, though $J_S = J_n$, and similarly $M_S^\dagger$)

$$x^+ = \begin{bmatrix} q^+ \\ \dot{q}^+ \end{bmatrix} = R_{(U,S)}(t, x^-) = \begin{bmatrix} q^- \\ M_S^\dagger M \dot{q}^- \end{bmatrix} \tag{3.67}$$

The frictional, plastic impact reset map, $R_{(U,C)}$, follows (3.67) but with J_C and $M_C^\dagger$ instead of J_S and $M_S^\dagger$. Similarly, the liftoff reset maps into modes U or V are the same except that there is no constraint J, and so the reset simplifies to an identity map. Note that the reset map does not depend on the prior mode, so for example $R_{(S,C)} = R_{(U,C)}$.

3.6.2 Apex

Apex is a "virtual" hybrid event – one that is specified by us and does not have a physical reset map or change in the dynamics – and is triggered when the velocity switches from going away from the

constraint to towards the constraint (V, U). As the reset map is identity, and the dynamics match before and after (since we don't specify a difference in control at this event) the saltation matrix is identity following (3.12)

$$\boxed{\Xi_{(V,U)} = I_{n\times n}} \tag{3.68}$$

3.6.3 Liftoff

Liftoff is a hybrid transition into mode V from S or C that depends on the constraint force $\lambda(t, x)$, which is both a function of time and state (and implicitly a function of control input), and is calculated from the bottom row of (3.60)

$$\lambda(t, x) = J^{\dagger}\left(\Upsilon - N - C\dot{q}\right) - \Lambda\dot{J}\dot{q} \tag{3.69}$$

The guard for liftoff is determined by λ_n, the constraint force in the J_n direction: if the force becomes non-repulsive, then the contact is released

$$g_{(C,V)}(t, x) = \lambda_n(t, x) \tag{3.70}$$

$$g_{(S,V)}(t, x) = \lambda_n(t, x) \tag{3.71}$$

Because the hybrid event occurs when the constraint force goes to zero, the dynamics at the boundary are equal. This is true even in the case of sticking friction in mode C, as the friction cone ensures that either the system will transition to sliding mode S (as discussed in Sec. 3.6.6) or the frictional force goes to zero at the same time. The state does not jump during liftoff, which meaning the reset map for liftoff is an identity transformation. Since both conditions of (3.12) are met for liftoff, the saltation matrices are identity

$$\boxed{\Xi_{(C,V)} = I_{n\times n}} \tag{3.72}$$

$$\boxed{\Xi_{(S,V)} = I_{n\times n}} \tag{3.73}$$

Due to the smooth nature of liftoff, we can safely ignore liftoff events when considering variations from liftoff.

3.6.4 Plastic impact

Plastic impact occurs when the unconstrained mode U makes contact and transitions to either the sliding mode S or the constrained mode C. We first analyze plastic impact into sliding (U, S). For simplicity, we assume frictionless sliding $\mu_s = 0$ to expose the structure in the saltation matrix, but the same calculations can be made with non-zero sliding friction $\mu_s > 0$. The dynamics for each mode is from (3.63)

$$F_U(t, x^-) = \begin{bmatrix} \dot{q}^- \\ M^{-1}(\Upsilon - C^-\dot{q}^- - N) \end{bmatrix} \tag{3.74}$$

$$F_S(t, x^+) = \begin{bmatrix} \dot{q}^+ \\ M_S^\dagger(\Upsilon - C^+\dot{q}^+ - N) - J_S^{\dagger T}\dot{J}_S^+\dot{q}^+ \end{bmatrix} \tag{3.75}$$

Note that $-$ or $+$ on C and $\dot{J}$ indicates that these functions use the pre- or post-impact velocity, $\dot{q}^-$ or $\dot{q}^+$, respectively. The Jacobian of the reset map for plastic impact, (3.67), is

$$D_x R_{(U,S)}(t, x^-) = \begin{bmatrix} I_{m\times m} & 0_{m\times m} \\ D_q(M_S^\dagger M\dot{q}^-) & M_S^\dagger M \end{bmatrix} \tag{3.76}$$

The Jacobian of the guard $D_x g_{(U,S)}(t, x^-)$ is

$$D_x g_{(U,S)}(t, x^-) = \begin{bmatrix} J_S & 0_{1\times m} \end{bmatrix} \tag{3.77}$$

while the denominator of $\Xi_{(U,S)}$, $D_x g_{(U,S)} F_U$, is

$$D_x g_{(U,S)}(t, x^-) F_U(t, x^-) = \begin{bmatrix} J_S & 0_{1\times m} \end{bmatrix} F_U(t, x^-) = J_S \dot{q}^- \tag{3.78}$$

In this example, the guard and reset map are independent of time, $D_t R = 0_{n\times 1}, D_t g = 0$. However, in other cases such as a paddle juggler [Sternad et al., 2001], the impact surface can move as a function determined by time, in which case the guard and reset would depend on the prescribed motion.

To further simplify the component of the saltation matrix (3.2) that contains the difference between dynamics, $F_S - D_x R F_U$, we apply the following steps. First, substitute in $\dot{q}^+ = M_S^\dagger M \dot{q}^- = \dot{q}^- - J_S^{\dagger T} J_S \dot{q}^-$ using the reset map (3.67) and the identity (3.62). Then, plugging into the difference between dynamics

$$F_S(t, x^+) - D_x R_{(U,S)}(t, x^-) F_U(t, x^-) = \tag{3.79}$$

$$\begin{bmatrix} -J_S^{\dagger T} J_S \dot{q}^- \\ M_S^\dagger (C^- \dot{q}^- - C^+ \dot{q}^+) - J_S^{\dagger T} \dot{J}_S^+ \dot{q}^+ - D_q (M_S^\dagger M \dot{q}^-) \dot{q}^- \end{bmatrix} \tag{3.80}$$

The saltation matrix for plastic impact is obtained by inserting all terms into (3.2) and simplifying (using (3.62) again)

$$\boxed{\Xi_{(U,S)} = \begin{bmatrix} M_S^\dagger M & 0_{m\times m} \\ Z_S + D_q (M_S^\dagger M \dot{q}^-) & M_S^\dagger M \end{bmatrix}} \tag{3.81}$$

where

$$Z_S = \left(M_S^\dagger (C^- \dot{q}^- - C^+ \dot{q}^+) - J_S^{\dagger T} \dot{J}_S^+ \dot{q}^+ - D_q (M_S^\dagger M \dot{q}^-) \dot{q}^- \right) \cdot J_S / (J_S \dot{q}^-) \tag{3.82}$$

Note that the difference between the Jacobian of the reset map $D_x R$ is in the first column of the matrix where the identity matrix is now $M_S^\dagger M$ and the element on the lower left differs significantly, as shown by (3.82).

When impacting into the frictional constrained mode C, all steps remain the same except with

J_C instead of J_S (and similarly $M_C^\dagger$ and $J_C^{\dagger T}$). However, the upper left block of the saltation matrix no longer simplifies as nicely with the Jacobian of the guard $D_x g$ terms. This is because $J_S = D_x g = J_n$ but $J_C \neq D_x g$. Rather, $D_x g = J_n$ is a row of J_C, i.e. the non-penetrating constraint. The resulting saltation matrix is

$$\boxed{\Xi_{(U,C)} = \begin{bmatrix} I_{m\times m} - \frac{J_C^{\dagger T} J_C \dot{q}^- J_n}{J_n \dot{q}^-} & 0_{m\times m} \\ Z_C + D_q(M_C^\dagger M \dot{q}^-) & M_C^\dagger M \end{bmatrix}} \tag{3.83}$$

where

$$\begin{aligned} Z_C \\ = \left(M_C^\dagger (C^- \dot{q}^- - C^+ \dot{q}^+) - J_C^{\dagger T} \dot{J}_C^+ \dot{q}^+ - D_q(M_C^\dagger M \dot{q}^-)\dot{q}^- \right) J_C / (J_C \dot{q}^-) \end{aligned} \tag{3.84}$$

Again, the difference between the saltation matrix and the Jacobian of the reset is in the left column associated with the configuration variations. However, the upper left block no longer maps configuration variations exactly the same as velocity variations in the lower right, because the tangential constraint is only a velocity constraint – the contact point can be anywhere on the contact surface, whereas the velocity of the contact point must be the same everywhere on the surface.

3.6.5 Elastic impact

When the coefficient of restitution is non-zero, states in the approaching unconstrained mode U transition directly to the separating unconstrained mode V through elastic impact. The dynamics for each mode, (3.63), are

$$F_U(t, x^-) = \begin{bmatrix} \dot{q} \\ M^{-1}(\Upsilon - C^- \dot{q}^- - N) \end{bmatrix} \tag{3.85}$$

$$F_V(t, x^+) = \begin{bmatrix} \dot{q}^+ \\ M^{-1}(\Upsilon - C^+ \dot{q}^+ - N) \end{bmatrix} \tag{3.86}$$

Again, note that $-$ or $+$ on C indicates that these functions use the pre- or post-impact velocity, $\dot{q}^-$ or $\dot{q}^+$, respectively. The Jacobian of the reset map for elastic impact, (3.66), is

$$D_x R^-_{(U,V)} = \begin{bmatrix} I_{m\times m} & 0_{m\times m} \\ D_q((M_n^\dagger M - eJ_n^{\dagger T} J_n)\dot{q}^-) & M_n^\dagger M - eJ_n^{\dagger T} J_n \end{bmatrix} \tag{3.87}$$

The Jacobian of the guard is again $D_x g = [J_n, 0_{1\times n}]$. Plugging each component back into the full saltation matrix equation results in

$$\boxed{\Xi_{(U,V)} = \begin{bmatrix} M^\dagger M - eJ^{\dagger T} J & 0_{m\times m} \\ Z_V + D_q((M^\dagger M - eJ^{\dagger T} J)\dot{q}^-) & M^\dagger M - eJ^{\dagger T} J \end{bmatrix}} \tag{3.88}$$

where J and $M^\dagger$ use the normal constraint, J_n and $M_n^\dagger$, and

$$\begin{aligned} &Z_V \\ &= \Big(\left[M^{-1}(C^- - C^+(M_n^\dagger M - eJ_n^{\dagger T} J_n)) - D_q((M_n^\dagger M - eJ_n^{\dagger T} J_n)\dot{q}^-) \right] \dot{q}^- \\ &\qquad + (1+e) J_n^{\dagger T} J_n M^{-1}(\Upsilon - C^- \dot{q}^- - N) \Big) \cdot J_n / (J_n \dot{q}^-) \end{aligned} \tag{3.89}$$

Note that the following substitution can be made $M^\dagger M - eJ^{\dagger T} J = I_{m\times m} - (1+e)J^{\dagger T} J$ by (3.62). Again, the main point is that the block diagonal terms are identical and that there are additional position variation terms that are not accounted for with just the Jacobian of the reset map.

3.6.6 Stick-slip friction

The saltation matrix for stick-slip friction has been calculated in [Leine and Nijmeijer, 2004, Sec. 7.3]. We recompute this saltation matrix for our generalized system and analyze its components.

When the friction cone is broken, the mode is switched from the constrained mode C to the sliding mode S. The guard to check for slipping is the friction cone

$$g_{(C,S)}(t,x) = \mu_s \|\lambda_n(t,x)\| - \|\lambda_t(t,x)\| = 0 \tag{3.90}$$

where μ_s is the coefficient of static friction. The reset map for these hybrid transitions is an identity transformation $x^+ = R_{(C,S)}(x^-) = x^-$, and therefore $D_x R_{(C,S)} = I_{n\times n}$.

If the guard $g_{(C,S)}$ is met, we can assume that slipping will also occur in the direction of the maximum tangential force. Therefore, at the slipping boundary, if both the coefficient of static friction and kinetic friction match, $\mu_s = \mu_k$, then $\Delta F = 0$ and the saltation matrix is identity by (3.12). Any model where the coefficients of friction match at the boundary will result in an identity saltation matrix. This includes fixed, Coulomb friction with matching coefficients and models where μ_k is a function of velocity, such as Stribeck friction, so long as at $v_t = J_t\dot{q} = 0$, $\mu_k(0) = \mu_s$, to get

$$\boxed{\mu_s = \mu_k \implies F_S - F_C = 0} \tag{3.91}$$

$$\implies \Xi_{(C,S)} = I_{n\times n} \tag{3.92}$$

If $\mu_s \neq \mu_k$, the saltation matrix is not necessarily identity, and the general computations of the saltation matrix can be made to obtain this form

$$\boxed{\mu_s \neq \mu_k \implies F_S^+ - F_C^- \neq 0} \tag{3.93}$$

$$\implies \Xi_{(C,S)} = I_{n\times n} + \frac{(F_S^+ - F_C^-)\cdot D_x g_{(C,S)}}{D_t g_{(C,S)} + D_x g_{(C,S)} F_C^-} \tag{3.94}$$

For this saltation matrix, position variations do not change because the reset map is identity and the top row of F_S and F_C are equal (i.e. the velocity $\dot{q}$ does not change between modes).

However, this saltation matrix will be very prone to modeling errors as it depends on knowing exactly how the sliding and sticking coefficients differ. From a modeling perspective, it may be advantageous to assume that at the boundaries they match.

3.6.7 Slip-stick friction

When the tangential velocity in mode S goes to zero, the sliding stops and "sticks" into the constrained mode C. Therefore, the guard[1] at slip stick friction is just the tangential velocity

$$g_{(S,C)}(t,x) = J_t\dot{q} = v_t \tag{3.95}$$

$$D_x g_{(S,C)}(t,x) = \begin{bmatrix} \dot{J}_t^- & J_t \end{bmatrix} \tag{3.96}$$

The reset is an identity transformation, $x^+ = R_{(S,C)}(x^-) = x^-$, and therefore $D_x R_{(S,C)} = I_{n\times n}$, so the saltation matrix is primarily composed of the difference between the dynamics of both modes and the tangential velocity term from the guard. For this hybrid transition, the dynamics will not be equal at the boundary because, while sliding, the tangential acceleration constraint will not be satisfied until transitioning to C. Since the guard is not directly a function of time or control input in this case, $D_t g_{(S,C)} = 0$ and can be ignored, and the saltation matrix is

$$\boxed{\Xi_{(S,C)} = I_{n\times n} + \frac{(F_C^+ - F_S^-)\cdot \begin{bmatrix} \dot{J}_t^- & J_t \end{bmatrix}}{\dot{J}_t^-\dot{q}^- + J_t\ddot{q}^-}} \tag{3.97}$$

Note that the denominator is the tangential acceleration constraint (3.59) in mode C. If this condition is met at the exact moment that the velocity guard is satisfied while in the sliding mode S, the saltation matrix is not well defined; however, this would violate the transversality assumption (3.11). For this saltation matrix, as with stick-slip, position variations do not change because the reset map is identity and the top row of F_C and F_S are equal (i.e. the velocity $\dot{q}$ does not change between modes).

[1]Note that there is an additional condition that $\lambda_t < \mu_s\lambda_n$ in order to stick, which we assume is satisfied in this section.

3.7 Conclusion

The saltation matrix is an essential tool when dealing with hybrid systems with state dependent switches. In this paper, we derive the saltation matrix with two different methods and demonstrate how the saltation matrix can be used in linear and quadratic forms for hybrid systems. We also provide a survey of where saltation matrices are used in other fields. In the past, it has been heavily utilized for analyzing the stability of periodic systems, but more recently it has been critical for analyzing and designing non-periodic behaviors. This analysis is especially useful for robotics where many important robotic motions are not periodic, but are hybrid due to the discontinuous nature of impact in rigid body systems with unilateral constraints.

To further explore the nature of contact and how variations are mapped through them, we first analyze a simple contact system where we compute the saltation matrix for plastic impact and analyze the different components of the resulting saltation matrices. We find that these saltation matrices capture the entirety of how position variations are mapped through contact, whereas the Jacobian of the reset map does not provide any information on position. In addition to this simple example, we compute saltation matrices for each of the hybrid transitions for a generalized rigid body model and give insights on them. These computations are especially useful because the rigid body model covers a wide variety of systems and will help when getting started using saltation matrices for these systems. When analyzing these saltation matrices, we found similar structures in them as the ones we found in the simple example. In particular, by only using the Jacobian of the reset map instead of the saltation matrix, the entirety of the position variational information is lost. For other hybrid transitions such as stick-slip friction, the Jacobian of the reset map provides no additional information because it is an identity transformation and all the information is contained from the saltation matrix.

By using saltation matrices for hybrid systems, we can produce efficient analysis, planning, control, and state estimation algorithms. This is especially important as hybrid systems naturally have combinatoric time complexities and through the use of these tools we can simplify these problems. The hope of this paper is to introduce the topic of saltation matrices to a broader

community so that we can, as a whole, develop better methods for dealing with the complexities of hybrid systems and their applications.

3.8 Appendicies

We derive the chain rule derivation of the saltation matrix and the early impact case for the geometric derivation. We also prove the update laws through hybrid events for both covariance propagation and the Riccati equations.

3.8.1 Saltation matrix chain rule derivation

Define the solutions of hybrid domains I and J which map an initial state state x at time t_0 to a state x_f at time t_f

$$\phi_I : (t_0 \in \mathbb{R}, t_f \in \mathbb{R}, x \in \mathcal{D}_I) \mapsto x_f \in \mathcal{D}_I \tag{3.98}$$

$$\phi_J : (t_0 \in \mathbb{R}, t_f \in \mathbb{R}, x \in \mathcal{D}_J) \mapsto x_f \in \mathcal{D}_J \tag{3.99}$$

such that the vector fields $F_I(t_0, x) = -D_{t_0}\phi_I(t_0, t_f, x)$ and $F_I(t_f, x) = D_{t_f}\phi_I(t_0, t_f, x)$ for both modes. Define the solution across a hybrid transition from mode I to J to be

$$\phi(t_0, t_f, x) := \phi_J(\tau(x), t_f, R_{(I,J)}(\tau(x), \phi_I(t_0, \tau(x), x))) \tag{3.100}$$

where $\tau(x)$ is the time to impact map, such that

$$g_{(I,J)}(\tau(x), \phi_I(t_0, \tau(x), x)) = 0 \tag{3.101}$$

It helps to look at the in between steps of the function composition in (3.100)

$$x^-(x) = \phi_I(t_0, \tau(x), x) \tag{3.102}$$

$$x^+(x) = R_{(I,J)}(\tau(x), x^-) \tag{3.103}$$

$$x_f(x) = \phi_J(\tau(x), t_f, x^+) \tag{3.104}$$

where $x_f = \phi(t, x)$ is the final state in the new mode. We want to find the derivative of ϕ with respect to x using the chain rule on each of these steps

$$D_x x^-(x) = D_{\tau(x)}\phi_I D_x\tau + D_x\phi_I \tag{3.105}$$

$$D_x x^+(x) = D_{\tau(x)}R_{(I,J)}D_x\tau + D_{x^-(x)}R_{(I,J)}D_x x^- \tag{3.106}$$

$$D_x x_f(x) = D_{\tau(x)}\phi_J D_x\tau + D_{x^+(x)}\phi_J D_x x^+ \tag{3.107}$$

where the arguments to each function are suppressed but equal to their corresponding value in (3.102)–(3.104).

Combining these, we get

$$D_x\phi = D_{\tau(x)}\phi_J D_x\tau + D_{x^+(x)}\phi_J[D_{\tau(x)}R_{(I,J)}D_x\tau + D_{x^-(x)}R_{(I,J)}(D_{\tau(x)}\phi_I D_x\tau + D_x\phi_I)] \tag{3.108}$$

As we are taking a first order approximation, the terms $D_x\phi_I$ and $D_{x^+(x)}\phi_J$ are identity, and so this simplifies to (with additional substitutions for F_I and F_J)

$$D_x\phi = (-F_J + D_{\tau(x)}R_{(I,J)} + D_{x^-(x)}R_{(I,J)}F_I)D_x\tau + D_{x^-(x)}R_{(I,J)} \tag{3.109}$$

To obtain $D_x\tau$ we use the implicit function theorem and take the chain rule on the guard condition, $0 = g_{(I,J)}(\tau(x), x^-)$, and using (3.105) to get the following relation

$$0 = D_{\tau(x)}g_{(I,J)}D_x\tau(x) + D_{x^-(x)}g_{(I,J)}D_x x^- \tag{3.110}$$

$$0 = \left(D_{\tau(x)}g_{(I,J)} + D_{x^-(x)}g_{(I,J)}F_I\right)D_x\tau + D_{x^-(x)}g_{(I,J)} \tag{3.111}$$

$$D_x\tau(x) = \frac{-D_{x^-(x)}g_{(I,J)}}{D_{\tau(x)}g_{(I,J)} + D_{x^-(x)}g_{(I,J)}F_I} \tag{3.112}$$

Plugging back into (3.109), evaluating at $t = \tau(x) = 0$, substituting the notation from (3.3)–(3.9), and simplifying

$$D_x\phi = D_x R^- + \left(D_x R^- F_I^- + D_t R^- - F_J^+\right)D_x\tau \tag{3.113}$$

$$D_x\phi = D_xR^- + \frac{\left(F_J^+ - D_xR^-F_I^- - D_tR^-\right) D_xg^-}{D_tg^- + D_xg^-F_I^-} \tag{3.114}$$

$$D_x\phi := \Xi_{(I,J)} \tag{3.115}$$

where all terms are evaluated at the time of impact and the state just before impact, except for F_J^+ which is evaluated at the state just after impact, as written out in (3.3)–(3.9).

3.8.2 Early impact saltation derivation

Since we assumed the perturbed trajectory impacted later to derive the geometric intuition of the saltation matrix, we also show that the saltation matrix is invariant to early or late impact by following the same derivation as before but with early impact. Again, we start by assuming the same linearizations as before and now impact occurs early at time $\widetilde{t^-}$ i.e. $\widetilde{t^-} < t^-$. It helps to visualize Fig. 3.3 with the linearization arrows flipped. Because the perturbed trajectory impacts first, we want to find the mapping from $\delta x(\widetilde{t^-})$ to $\delta x(t^+)$ instead of $\delta x(t^-)$ to $\delta x(\widetilde{t^+})$ (like in the case of late impact). In general, we want to compare states that are in the same hybrid domain, and this notation allows for this.

We define $\delta x(\widetilde{t^-})$ and $\delta x(t^+)$ to be

$$\delta x(\widetilde{t^-}) = \widetilde{x}(\widetilde{t^-}) - x(\widetilde{t^-}) \tag{3.116}$$

$$\delta x(t^+) = \widetilde{x}(t^+) - x(t^+) \tag{3.117}$$

Using the linearization of the flow before impact and the reset we can expand

$$\widetilde{x}(\widetilde{t^-}) = x(t^-) - F_I^-\delta t + \delta x(\widetilde{t^-}) \tag{3.118}$$

Since the perturbed trajectory impacts earlier, we must compute where it ends up after going through the reset map and flowing for δt time on the new dynamics. Again, we use the linearization of the

flow

$$\widetilde{x}(t^+) = R(t^-, \widetilde{x}(\widetilde{t^-})) + F_J^+ \delta t \tag{3.119}$$

Next, we want to get $R(\widetilde{t^-}), \widetilde{x}(\widetilde{t^-}))$ as a function of $x(\widetilde{t^-})$. By substituting $\widetilde{x}(\widetilde{t^-})$ into (3.118) and using the linearization of the reset we get

$$R(\widetilde{t^-}, \widetilde{x}(\widetilde{t^-})) = R(\widetilde{t^-}, x(t^-) - F_I^- \delta t + \delta x(\widetilde{t^-})) \tag{3.120}$$

$$R(\widetilde{t^-}, \widetilde{x}(\widetilde{t^-})) = R(\widetilde{t^-}, x(t^-)) + D_x R^- \left(\delta x(\widetilde{t^-}) - F_I^- \delta t)\right) \tag{3.121}$$

Now plugging back in we get

$$\widetilde{x}(t^+) = R(\widetilde{t^-}, x(t^-)) + D_x R^- \left(\delta x(\widetilde{t^-}) - F_I^- \delta t)\right) + F_J^+ \delta t \tag{3.122}$$

$$\widetilde{x}(t^+) = R(\widetilde{t^-}, x(t^-)) + D_x R^- \delta x(\widetilde{t^-}) + \left(F_J^+ - D_x R^- F_I^-\right) \delta t \tag{3.123}$$

Now we can write $\delta x(t^+)$ as a function of $\delta x(\widetilde{t^-})$ and δt by subbing $\widetilde{x}(t^+)$ into (3.117)

$$\delta x(t^+) = D_x R^- \delta x(\widetilde{t^-}) + \left(F_J^+ - D_x R^- F_I^-\right) \delta t \tag{3.124}$$

Next, we want to find δt as a function of $\delta x(\widetilde{t^-})$ using

$$0 = g(\widetilde{x}(\widetilde{t^-})) \tag{3.125}$$

Substitute in (3.118) and expand using the linearization of the guard

$$0 = g\left(x(t^-) - F_I^- \delta t + \delta x(\widetilde{t^-})\right) \tag{3.126}$$

$$0 = g(x(t^-)) + D_x g^- \left(-F_I^- \delta t + \delta x(\widetilde{t^-})\right) \tag{3.127}$$

$$0 = D_x g^- \left(-F_I^- \delta t + \delta x(\widetilde{t^-})\right) \tag{3.128}$$

Now solve for δt in terms of $\delta x(\widetilde{t^-})$

$$D_x g^- \delta x(\widetilde{t^-}) = D_x g^- F_I \delta t \tag{3.129}$$

$$\delta t = \frac{D_x g^-}{D_x g^- F_I} \delta x(\widetilde{t^-}) \tag{3.130}$$

Note that the δt for early impact is flipped when compared to late impact – as we expect.

Substitute (3.130) into (3.124) and simplify to get the saltation matrix

$$\delta x(t^+) = D_x R^- \delta x(\widetilde{t^-}) \tag{3.131}$$

$$+ \left(F_J^+ - D_x R^- F_I^-\right) \frac{D_x g^-}{D_x g^- F_I^-} \delta x(\widetilde{t^-}) \tag{3.132}$$

$$\delta x(t^+) = \left(D_x R^- + \frac{\left(F_J^+ - D_x R^- F_I^-\right) \cdot D_x g^-}{D_x g^- F_I^-} \right) \delta x(\widetilde{t^-}) \tag{3.133}$$

$$= \Xi_{(I,J)} \delta x(\widetilde{t^-}) \tag{3.134}$$

3.8.3 Covariance update through a hybrid event

In this section we derive the update for a covariance passing through a reset map, (3.36). Consider the state trajectory as a random variable $X(t)$ with mean $\mu(t) = x(t)$, the nominal trajectory, and covariance $\Sigma(t)$. We can define a perturbation as a zero mean random variable $\delta x(t)$ with the same covariance, such that $X(t) = x(t) + \delta x(t)$.

At a hybrid impact event, define the pre-impact time of the mean to be t^-, where $g(t^-, \mu(t^-), u^-) = 0$, and the corresponding post-impact time to be t^+. Consider how the distribution is updated to find $X(t^+)$ based on $X(t^-)$. To find the mean, we take the expectation of $X(t^+)$

$$\mu(t^+) = \mathbb{E}[X(t^+)] = \mathbb{E}[x(t^+) + \delta x(t^+)] \tag{3.135}$$

$$= x(t^+) + \mathbb{E}[\delta x(t^+)] \tag{3.136}$$

where the two terms are separable because expectation is a linear operator, and the expectation

of the nominal post-impact state is just its value, $\mathbb{E}[x(t^+)] = x(t^+) = R(x(t^-))$. Substituting in $\delta x(t^+) = \Xi_{(I,J)}(t^-, x(t^-))\delta x(t^-) + \text{h.o.t.}$ from (3.10)

$$\mu(t^+) = x(t^+) + \mathbb{E}[\Xi_{(I,J)}(t^-, x(t^-))\delta x(t^-) + \text{h.o.t.}] \tag{3.137}$$

$$\mu(t^+) = x(t^+) + \Xi_{(I,J)}(t^-, x(t^-))\mathbb{E}[\delta x(t^-)] + \mathbb{E}[\text{h.o.t.}] \tag{3.138}$$

Again, because of the linear properties of the expectation, we can pull out $\Xi(t^-, x(t^-))$. Then, because $\delta x(t^-)$ is centered about zero, $\mathbb{E}[\delta x(t^-)] = 0$, as are the higher order terms, $\mathbb{E}[\text{h.o.t.}] = 0$, to get

$$\mu(t^+) = x(t^+) = R(x(t^-)) \tag{3.139}$$

For covariance, we start with the definition of covariance

$$\mathbb{COV}[X] = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T] \tag{3.140}$$

the post-impact covariance $\Sigma(t^+)$ is

$$\Sigma(t^+) = \mathbb{COV}[X(t^+)] = \mathbb{COV}[x(t^+) + \delta x(t^+)] \tag{3.141}$$

$$= \mathbb{E}\Big[\Big((x(t^+) + \delta x(t^+) - \mu(t^+)) \Big(x(t^+) + \delta x(t^+) - \mu(t^+)\Big)^T\Big] \tag{3.142}$$

Since $\mu(t^+) = x(t^+)$, this simplifies to

$$\Sigma(t^+) = \mathbb{E}[\delta x(t^+)\delta x(t^+)^T] \tag{3.143}$$

Using (3.10), $\delta x(t^+)$ can be expanded as

$$\Sigma(t^+) = \mathbb{E}[(\Xi\delta x(t^-) + \text{h.o.t.})(\Xi\delta x(t^-) + \text{h.o.t.})^T] \tag{3.144}$$

$$= \Xi\Sigma(t^-)\Xi^T + 2\Xi\delta x(t^-)(\text{h.o.t.}) + (\text{h.o.t.})^2 \tag{3.145}$$

when h.o.t. is small, we get

$$\Sigma(t^+) = \Xi\Sigma(t^-)\Xi^T \tag{3.146}$$

as in (3.36), which holds to first order and is exact for linear hybrid systems.

3.8.4 Riccati update through hybrid events

In this section, we derive the update for the Riccati update through a hybrid event. See [Tedrake, 2022, Ch. 8.3.1] for a background on the smooth discrete Riccati update. Define the optimal cost-to-go $\ell^*_{t^-}$ for the reference trajectory $(x(t), u(t))$ and the optimal solution (x^*, u^*) applied at a hybrid transition at time t^- as

$$\ell_{t^-}(x^*(t^-), u^*(t^-)) = \frac{1}{2}(x^*(t^-) - x(t^-))^T Q_{t^-}(x^*(t^-) - x(t^-)) + \frac{1}{2}(u^*(t^-) - u(t^-))^T R_{t^-}(u^*(t^-) - u(t^-)) \tag{3.147}$$

where Q_t and R_t are the quadratic penalty on state and input respectively at time t^-. Define the current state to be $\widetilde{x}$ and the difference with the optimal solution to be

$$\delta x^*(t^-) := x^*(t^-) - \widetilde{x}(t^-) \tag{3.148}$$

such that (3.147) becomes

$$\begin{aligned}\ell^*_{t^-} &= \frac{1}{2}(\delta x^* + \widetilde{x} - x(t^-))^T Q_{t^-}(\delta x^* + \widetilde{x} - x(t^-)) \\ &+ \frac{1}{2}(u^*(t^-) - u(t^-))^T R_{t^-}(u^*(t^-) - u(t^-))\end{aligned} \tag{3.149}$$

Because the transition is instantaneous, we assume that the input has no affect $u(t^-) = u^*(t^-)$ and we can simplify the optimal cost-to-go

$$\ell^*_{t^-} = \frac{1}{2}(\delta x^* + \widetilde{x} - x(t^-))^T Q_{t^-}(\delta x^* + \widetilde{x} - x(t^-)) \tag{3.150}$$

The Hamiltonian for the hybrid transition.

$$\begin{aligned} H_{t^-} &:= H(x^*(t^-), u^*(t^-), \lambda^*(t^+)) \\ &:= \ell^*_{t^-} + \lambda^*(t^+)^T R_{(I,J)}(t^-, x^*(t^-)) \end{aligned} \tag{3.151}$$

where $\lambda^*(t^+)$ is the optimal costate. Using the expansion (3.10) about $R_{(I,J)}(t^-, \widetilde{x}(t^-) + \delta x^*(t^-))$

$$R_{(I,J)}(t^-, x^*(t^-)) = R_{(I,J)}(t^-, \widetilde{x}(t^-)) + \Xi \delta x^*(t^-) + \text{h.o.t.} \tag{3.152}$$

where $\Xi = \Xi_{(I,J)}(t^-, \widetilde{x}(t^-))$ The Hamiltonian [Bertsekas, 2012] for the hybrid transition is then

$$\begin{aligned} H \\ = \frac{1}{2}(\delta x^*(t^-))^T Q_{t^-} \delta x^*(t^-) + \lambda^*(t^+)^T \left(R_{(I,J)}(t^-, \widetilde{x}(t^-)) + \Xi \delta x^*(t^-) + \text{h.o.t.} \right) \end{aligned} \tag{3.153}$$

Using Pontryagin's Maximum principle, we derive the optimal state update and costate update.

$$x^*(t^+) = \frac{\delta}{\delta \lambda^*} H = R_{(I,J)}(t^-, \widetilde{x}(t^-)) + \Xi \delta x^*(t^-) \tag{3.154}$$

$$\lambda^*(t^-) = \frac{\delta}{\delta x^*} H = Q_{t^-} \delta x^* + \Xi^T \lambda^*(t^+) + \text{h.o.t.} \tag{3.155}$$

Therefore, the update for P can be derived for hybrid transitions. We start with a standard costate guess of $\lambda(t^+) = P(t^+)\delta x(t^+)$ [Tedrake, 2022]

$$P(t^-)\delta x^*(t^-) = Q_{t^-} \delta x^*(t^-) + \Xi^T P(t^+) \delta x^*(t^+) + \text{h.o.t.} \tag{3.156}$$

We substitute $\delta x^*(t^+) = \Xi \delta x^*(t^-) + \text{h.o.t.}$

$$P(t^-)\delta x^*(t^-) = Q_{t^-} \delta x^*(t^-) + \Xi^T P(t^+)(\Xi \delta x^*(t^-) + \text{h.o.t.}) + \text{h.o.t.} \tag{3.157}$$

$$P(t^-)\delta x^*(t^-) = Q_{t^-} \delta x^*(t^-) + \Xi (x^*(t^-))^T P(t^+) \Xi \delta x^*(t^-) + \Xi (x^*(t^-))^T P(t^+) \text{h.o.t.} + \text{h.o.t.} \tag{3.158}$$

See that the update for $P(t^-)$ is recursive and cannot be computed as is. However, when higher order terms are small, then we cancel $\delta x(t^-)$ from both sides and write the Bellman update for $P(t^-)$

$$P(t^-)\delta x^*(t^-) \approx Q_{t^-} \delta x^*(t^-) + \Xi^T P(t^+) \Xi \delta x^*(t^-) \tag{3.159}$$

$$P(t^-) \approx Q_{t^-} + \Xi^T P(t^+)\Xi \tag{3.160}$$

3.8.5 Covariance Propagation Validation

This experiment seeks to validate the hybrid transition covariance (3.36) propagation law experimentally by analyzing the distributions of particles simulated through hybrid transitions. Distributions are created by randomly sampling 1000 particles from a known mean and covariance. The particles are then simulated forwards through the hybrid dynamical system using Matlab's ODE45. The starting mean and covariance of the population are calculated then propagated for each timestep using the smooth update law for eachtimestep and using the hybrid transition update when a hybrid transition occurs. We also test using the Jacobian of the reset map $D_x R$ instead of the saltation matrix. The final covariance of the population and the estimated covariances are compared using Kullback–Leibler (KL) divergence [Van Erven and Harremos, 2014] – where the output is a natural unit of information (nat) therefore, the smaller the KL divergence, the more closely the distributions match.

$$KL(\Sigma_0 \| \Sigma_1) = \frac{1}{2}\left(\Sigma_1^{-1}\Sigma_0 - \dim(\Sigma_0) + \ln\frac{|\Sigma_1|}{|\Sigma_0|}\right) \tag{3.161}$$

Each hybrid system example is tested with two different initial covariances – a higher and lower value – to capture any nonlinearities. In this section we present the results of these experiments on a series of hybrid systems.

Constant Flow

The simplest hybrid system we examine is the case where there are two hybrid modes that are linearly separated and which have constant, but distinct, dynamics in each mode. The dynamics in the hybrid modes are defined:

$$F_1 = [1, -1]^T, \quad F_2 = [1, 1]^T \tag{3.162}$$

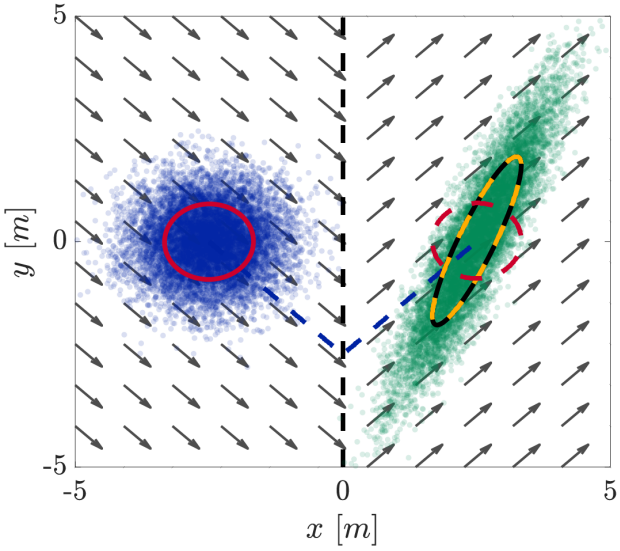

Figure 3.9: Flowing an initial distribution (blue dots) with covariance (red solid line) along a nominal trajectory (blue dashed line) through hybrid systems with dynamics (arrows) and a single guard (black dashed line). The final distribution (green dots) is overlaid with the actual covariance (black line). Estimated covariance using the Jacobian of the reset map (red dashed line) is compared against our proposed estimate using the saltation matrix (gold dashed line).

The guard sets are defined at $x_1 = 0$, such that the domain of F_1 is the left half plane and the domain of F_2 is the right half plane (Fig. 3.9). The reset is an identity map.

For the propagation experiment, the starting mean was $x = [-2.5, 0]^T$, the total simulation time was 5 seconds, and the time steps were 0.01 seconds as shown in Fig. 3.9. The high covariance level was $0.1I$ and the low covariance level was $0.005I$. All samples from the system began in hybrid mode 1 and ended in hybrid mode 2.

This scenario is interesting because the covariance does not change in either hybrid mode as the flow is simply translation in those regions – the only covariance changes are a result of the hybrid transition. The results, listed in Table 3.1 and apparent in Fig. 3.9, show that this change in covariance is captured well using the saltation matrix (as the KL-divergence is almost zero) but not well using the Jacobian of the reset map. Note that, as expected for a linear system, the KL-divergence does not significantly depend on the initial covariance and in this case the difference is practically zero.

Bouncing Ball

The bouncing ball is a hybrid dynamical system which consists of 2 hybrid domains in the $[y, \dot{y}]^T$ plane, where the first domain is defined when the ball has negative velocity $\dot{y} < 0$ and the second domain is defined when the ball has non-negative velocity $\dot{y} \geq 0$. The guard sets are defined to be when the ball hits the ground $g_{1,2}(t, y, \dot{y}) := y$ and when the velocity changes sign $g_{2,1}(t, y, \dot{y}) := \dot{y}$. Note that this could equivalently be defined as a single domain with a self-reset, however to match our definition of a hybrid dynamical system (Definition 1) we treat it as a system with separate domains. The dynamics are standard ballistic dynamics in both domains

$$F_1 = F_2 = [\dot{y}, -a_g]^T \tag{3.163}$$

where a_g is the acceleration from gravity. The reset from 1 to 2 is defined by elastic impact

$$R_{1,2} = [y, -e\dot{y}]^T \tag{3.164}$$

where α is the coefficient of restitution. The reset from 2 to 1 is an identity transformation. For the experiments, the gravitational acceleration is $a_g = 9.8$ and the coefficient of restitution is $\alpha = 0.75$. This system is an example of a system with linear dynamics, guards, and linear but non-identity resets.

For the propagation experiment, the starting mean was $x = [3, -2]^T$, the total simulation time was 1 second, and the time steps were 0.001 seconds as shown in Fig. 3.10. The high covariance level was $0.05I$ and the low covariance level was $0.001I$. Most of the samples from the system began in hybrid mode 1 and ended in hybrid mode 2, while some samples ended back in hybrid mode 1 (after a (2,1) transition). There is no effect on the mean or covariance through the (2,1) transition as $R_{(2,1)}$ is identity and $F_1 = F_2$ which means that $\Xi_{(2,1)} = D_x R + 0$ is also identity.

The results are listed in Table 3.1 and an example trial is shown in Fig. 3.10. Interestingly, even though the final distribution is no longer Gaussian for the bouncing ball, the second moment

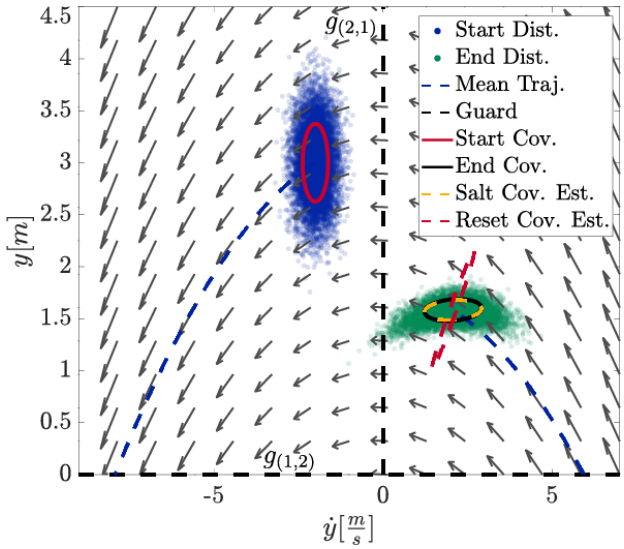

Figure 3.10: Flowing an initial distribution (blue dots) with covariance (red solid line) along a nominal trajectory (blue dashed line) through the bouncing ball hybrid system's dynamics (gray arrows) with guards (black dashed line and labeled). The final distribution (green dots) is overlaid with the actual covariance (black line). Estimated covariance using the Jacobian of the reset map (red dashed line) is compared against our proposed estimate using the saltation matrix (gold dashed line).

(covariance) is still tracked accurately through the impact using the saltation matrix (as the KL-divergence is small) but poorly with the Jacobian of the reset map. Again, as this is a linear system, the KL-divergence is independent of the initial covariance.

Pendulum hitting a spring damper

The pendulum hitting a spring damper, as shown in Fig. 3.12, is a hybrid system which consists of 2 hybrid domains over the $[\theta, \dot{\theta}]^T$ state space. The first domain is defined when the pendulum's angular position is positive $\theta > 0$ and the second domain is defined when the angular position is non-positive $\theta \leq 0$, such that the guard functions are $g_{1,2}(t, \theta, \dot{\theta}) = \theta$ and $g_{2,1}(t, \theta, \dot{\theta}) = -\theta$. The dynamics are standard pendulum dynamics while in domain 1, and while in domain 2 the pendulum is in contact with a torsional spring and damper. The resulting dynamics are,

$$F_1 = \left[\dot{\theta}, -\frac{a_g}{l}\sin(\theta)\right]^T \tag{3.165}$$

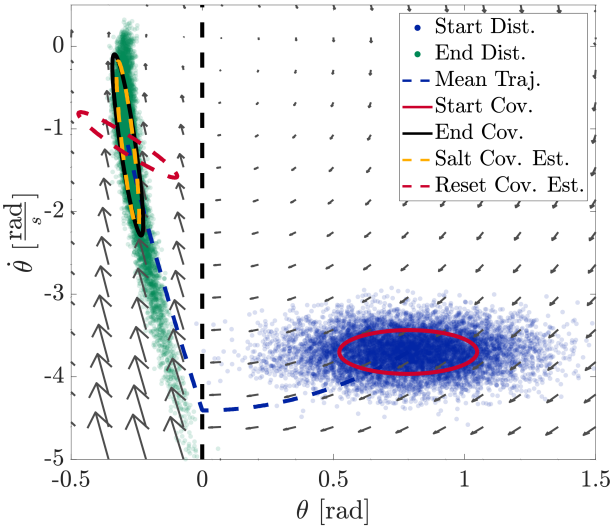

Figure 3.11: Flowing an initial distribution (blue dots) with covariance (red solid line) along a nominal trajectory (blue dashed line) through the pendulum hitting a spring damper hybrid system's dynamics (gray arrows) with guards (black dashed line and labeled). The final distribution (green dots) is overlaid with the actual covariance (black line). Estimated covariance using the Jacobian of the reset map (red dashed line) is compared against our proposed estimate using the saltation matrix (gold dashed line).

$$F_2 = \left[\dot{\theta}, \frac{-(k\theta + c\dot{\theta} + ma_g l \sin(\theta))}{ml^2}\right]^T \tag{3.166}$$

where a_g is the acceleration from gravity, l is the length of the pendulum's center of mass along the arm, and m is the mass of the pendulum. In the experiment, the constants were set to $a_g = 9.8$, $l = 1$, $m = 1$, $k = 10$, and $c = 10$. Both resets are identity transformations because there are no instantaneous changes in state during mode transition. This system is nonlinear but with identity resets.

For the propagation experiment, the starting mean was $x = [\frac{\pi}{4}, -3.7]^T$, the total simulation time was 0.3 seconds, and the time steps were 0.001 seconds as shown in Fig. 3.9. An example run is shown in Fig. 3.11.

The high covariance level was $0.05I$ and the low covariance level was $0.001I$. All samples from the system began in hybrid mode 1 and ended in hybrid mode 2. This example demonstrates the validity of the linear approximations when the higher order terms are small and as expected, using the saltation update decreases KL-divergence when compared against the Jacobian of the reset map

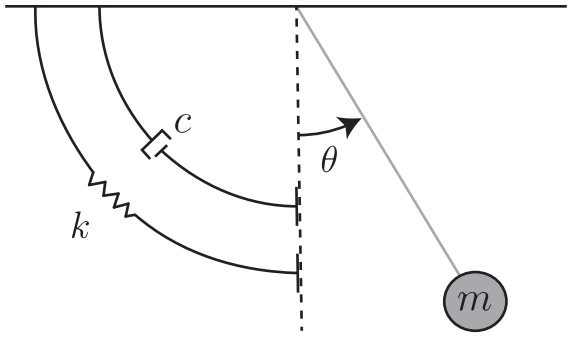

Figure 3.12: Pendulum hitting a spring damper hybrid system where the pendulum engages a spring damper at the $\theta = 0$.

as shown in Table 3.1.

Asymmetric Spring Loaded Inverted Pendulum (ASLIP)

The asymmetric spring loaded inverted pendulum (ASLIP) system consists of a spring leg, torsional spring hip, and a body with inertia in the sagittal plane as shown in Fig. 3.13. This system is similar to the one in [Poulakakis and Grizzle, 2009] and a full derivation for the system dynamics can be found in Appendix B. In this system, the body configuration space is defined to be the position and orientation of the body $q_b := [x_b, y_b, \theta_b]^T \in \mathbb{R} \times \mathbb{R} \times \mathbb{S}^1$. The leg configuration space is defined to be the angle between the toe and the ground, the angle of the hip, and the length of the leg $q_l := [\theta_t, \theta_h, l_l]^T \in \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{R}$, where impact location of the toe defines a pin joint for the body to pivot around. Once the location of the toe, $q_t = [x_t, y_t]^T \in \mathbb{R} \times \mathbb{R}$, is fixed to a ground location, either configuration can be used to define the full configuration space of the system. When the toe position is known, the transformation from the leg configuration to the body configuration $T_{lb} : (q_l, q_t) \mapsto q_b$. The inverse mapping can also be calculated which maps the body configuration to the leg configuration $T_{bl} : (q_b, q_t) \mapsto q_l$.

Hybrid mode 1 is defined to be when the toe is not in contact with the ground. The resulting domain $\mathcal{D}_1$ is chosen to be parameterized by the body's configuration, toe position, and body's velocity.

$$[x_b, y_b, \theta_b, x_t, y_t, \dot{x}_b, \dot{y}_b, \dot{\theta}_b]^T \in \mathcal{D}_1 \tag{3.167}$$

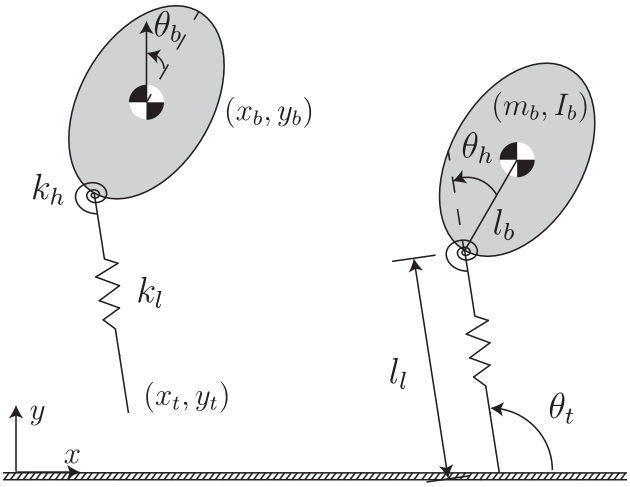

Figure 3.13: Asymmetric Spring Loaded Inverted Pendulum (ASLIP) diagram showing the aerial phase hybrid mode on the left and the stance phase hybrid mode on the right and their corresponding configuration variables.

When the toe is in contact with the ground, the hybrid mode is 2. The domain $\mathcal{D}_2$ is chosen to be parameterized by the toe angle with the ground, hip angle, the leg extension, toe position, the time derivative of the toe angle, hip angle, and leg extension.

$$[\theta_t, \theta_h, l_l, x_t, y_t, \dot{\theta}_t, \dot{\theta}_h, \dot{l}_l]^T \in \mathcal{D}_2 \tag{3.168}$$

Note that the toe position is augmenting the state rather than being treated as an external parameter because variations in the toe placement affect the other configuration states. Because of this, the toe dynamics are constrained relative to the body in domain 1 and relative to the ground contact in domain 2. These dynamics F_1, F_2 are derived using a Lagrangian approach as shown in Appendix 3.8.5.

The system parameters and their experimental values are body mass $m_b = 1$, body inertia $I_b = 1$, leg spring constant $k_l = 1000$, hip spring constant $k_\theta = 400$, body length $l_b = 0.5$, acceleration due to gravity $a_g = 9.8$, resting leg length $l_{l0} = 1$, and resting angle of the hip spring $\theta_{h0} = -\frac{\pi}{8}$.

For the propagation experiment, the starting mean was $x = [0, 1.8, \frac{5pi}{12}, 0.0011, 0.3256, 0-10]^T$, the total simulation time was 0.55 seconds, and the time steps were 0.0001 seconds. The high covariance level was 0.001 and the low covariance level was 0.00005 for the non toe states. There

Table 3.1: KL-divergence comparison between the ground truth and estimated second moment of distributions that undergo hybrid transitions for the 4 hybrid system test cases using the saltation matrix $\Xi_{(I,J)}$ and the Jacobian of the reset map $D_x R_{(I,J)}$.

System	Nonlinear	Identity Reset	Init. Cov.	KL-div $DxR_{(I,J)}$	KL-div $\Xi_{(I,J)}$
1 Constant Flow	No	Yes	Low	1.9193	1.8419×10^{-6}
			High	1.9219	2.2574×10^{-7}
2 Bouncing Ball	No	No	Low	32.2253	0.0035
			High	32.2253	0.0035
3 Pendulum	Yes	Yes	Low	29.2477	0.0011
			High	32.3330	0.2058
4 ASLIP	Yes	No	Low	26.8931	0.0011
			High	24.8121	0.0867

was no additional noise injected into the toe states, because they are constrained to the body states. The system started in hybrid mode 1 and ended in hybrid mode 2. This scenario is the most complex out of all the considered hybrid systems because it includes both nonlinear dynamics and nonidentity resets. Nevertheless, the change in covariance is still captured well using the saltation matrix and is not captured well when using the Jacobian of the reset, Table 3.1. Similar to the pendulum example, as the initial covariance increases, the saltation update approximation gets worse due to the inaccuracies in the linearization.

Overall, using the saltation update estimates the covariance very well for the linear cases as shown by the KL-divergence. For the non-linear cases, the saltation update is a good linear approximation. This is especially apparent in the linear cases where increasing the initial covariance did not have much affect on the KL-divergence while in the nonlinear cases the increase in initial covariance increased the resulting KL-divergence; this is unsurprising, as the linearization is local and will generally increasingly fail to predict the correct dynamics as the domain is enlarged. These results motivate the use of the saltation update in a Kalman filtering framework.

Chapter 4

The Salted Kalman Filter: Kalman Filtering on Hybrid Dynamical System

This chapter previously appeared in [Kong et al., 2021b].

4.1 Abstract

Many state estimation and control algorithms require knowledge of how probability distributions propagate through dynamical systems. However, despite hybrid dynamical systems becoming increasingly important in many fields, there has been little work on utilizing the knowledge of how probability distributions map through hybrid transitions. Here, we make use of a propagation law that employs the saltation matrix (a first-order update to the sensitivity equation) to create the Salted Kalman Filter (SKF), a natural extension of the Kalman Filter and Extended Kalman Filter to hybrid dynamical systems. Away from hybrid events, the SKF is a standard Kalman filter. When a hybrid event occurs, the saltation matrix plays an analogous role to that of the system dynamics, subsequently inducing a discrete modification of both the prediction and update steps. The SKF outperforms a naive variational update – the Jacobian of the reset map – by having a reduced mean squared error in state estimation, especially immediately after a hybrid transition event. Compared against a hybrid particle filter, the particle filter outperforms the SKF in mean squared error only

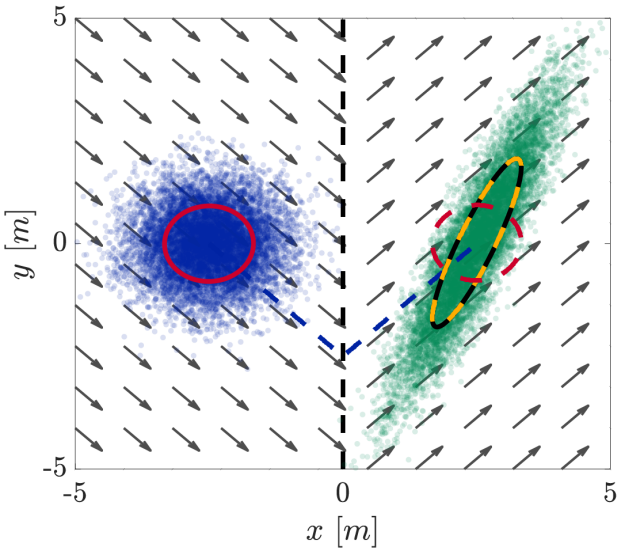

Figure 4.1: Flowing an initial distribution (blue dots) with covariance (red solid line) along a nominal trajectory (blue dashed line) through hybrid systems with dynamics (arrows) and a single guard (black dashed line). The final distribution (green dots) is overlaid with the actual covariance (black line). Estimated covariance using the Jacobian of the reset map (red dashed line) is compared against our proposed estimate using the saltation matrix (gold dashed line).

when a large number of particles are used, likely due to a more accurate accounting of the split distribution near a hybrid transition.

4.2 Introduction

From legged robots to manipulator systems, many important contemporary control problems revolve around systems that make and break contact with their environments. These contact events are often represented as a discrete change to the system dynamics which introduces complexity for state estimation and control, as classic methods assume smoothness [Bloesch et al., 2012; Hartley et al., 2020; Bledt et al., 2018; Varin and Kuindersma, 2018]. These "hybrid systems" [Back et al., 1993; Lygeros et al., 2003; Goebel et al., 2009] are systems with both continuous states (such as the position and velocity of a robot's center of mass and joints) and discrete states (such as whether or not a limb is in contact with the ground). Lacking out-of-the-box solutions, state estimation for these systems is a frontier with novel difficulties [Blom and Bar-Shalom, 1988; Skaff et al., 2005],

including how to deal with nonlinear dynamics on the continuous phases [Barhoumi et al., 2012], discrete jumps in the continuous state [Balluchi et al., 2013], and real time computation [Zhang et al., 2020].

In this work we propose a Kalman-like filter compatible with hybrid dynamical systems while also avoiding the combinatorial effects of considering multiple modes simultaneously [Zhang et al., 2020]. To do this, we apply the saltation matrix (a standard tool from non-smooth analysis [Leine and Nijmeijer, 2013]) to propagate state uncertainty covariance through hybrid transitions [Biggio et al., 2014]. The saltation matrix provides a first order approximation of the effects of a hybrid domain change based on the dynamics in the individual modes, the reset functions, and the location of the reset. It might be assumed that the propagation of uncertainty through hybrid transitions could be approximated by simply examining the first order approximation of the reset map itself, i.e. the Jacobian of the reset map. For example, [Hartley et al., 2020] and [Bloesch et al., 2012] assume that the hybrid transition does not affect the second moment of the distribution; i.e the reset map is identity and therefore the Jacobian would be an identity matrix. However, this approach does not take into account the differing dynamics in the distinct modes. The inaccuracy of the naive approach can be seen in Fig. 4.1, where the system has an identity reset map but keeping the second moment constant through the transition does not capture the effect of the hybrid transition. As such, attempting to use the Jacobian of the reset map, while a "natural" idea, is ultimately incorrect.

The remainder of this paper is organized in the following manner. Section 4.3 provides a brief review of the hybrid system estimation literature. Section 4.4 defines the problem that we seek to solve in this work as well as establishing the notation and conventions used. Section 4.5 introduces the "Salted Kalman Filter" (SKF), which is a Kalman Filter augmented with the capability to propagate the estimated first and second moments through hybrid transitions. Section 4.6 explains the experiments used to validate the performance of the Kalman filter. Section 4.7 compares results from using the SKF to results using the Jacobian of the reset map and to a particle filter. Finally, Section 4.8 provides a discussion of the work presented and potential future work.

4.3 Related Work

There has been a variety of work on the topic of state estimation for systems with differing dynamics and discrete modes, however current approaches either do not consider systems with state-driven mode transitions (i.e. are limited to the "switched system" case) [Blom and Bar-Shalom, 1988; Blom and Bloem, 2004; Balluchi et al., 2002; Skaff et al., 2005; Eras-Herrera et al., 2019; Hwang et al., 2006] or are computationally expensive and difficult to run in an online filtering setting [Koval et al., 2015b; Zhang et al., 2020].

Our work seeks to understand how distributions are propagated through state-driven hybrid dynamical systems by applying knowledge from non-smooth systems literature [Johnson et al., 2016a; Aizerman and Gantmakher, 1958; Hiskens and Pai, 2000] in order to make simplifying assumptions which retain sufficient information for the purposes of online state estimation.

4.3.1 Hybrid System Estimators

One approach to filtering on hybrid systems with linear dynamics is to use a filter bank where a filter is assigned to each discrete mode and the output of the filter with the lowest residual is used as the current state estimate [Balluchi et al., 2002]. Another style of filter bank method mixes the outputs of individual filters by utilizing a probability weight calculated based on measurement residuals and *a posteriori* estimate likelihoods such as the interacting multiple model (IMM) [Blom and Bar-Shalom, 1988]. These filtering methods have been extended to hybrid systems with nonlinear dynamics [Barhoumi et al., 2012] and hybrid systems with non-identity reset maps during hybrid transitions [Balluchi et al., 2013]. However, these filtering bank strategies consider hybrid systems with transitions that do not depend on continuous state and therefore do not account for the effect that the continuous state dependent transitions have on the distribution. This is an issue because the first 2 moments of the distribution are not guaranteed to be captured after a transition.

Particle filtering approaches seek to represent uncertainty distributions directly with a variety of sample points rather than by representing belief as a parametric (e.g. Gaussian) distribution

[Koutsoukos et al., 2002; Koval et al., 2015b]. One of the major drawbacks of particle filters is that they are computationally expensive – they may require ($O(2^n)$) where n is the number of states [Thrun, 2002]. Because of this, it may be difficult to utilize them in a real-time setting.

Some optimization based methods seek to circumvent this issue of computational complexity by simultaneously selecting the continuous and discrete states over all timesteps to minimize the error associated with the measurements and the dynamics[Zhang et al., 2020; Ferrari-Trecate et al., 2002]. The resulting optimization problem requires a much higher computational load compared to causal forward time stepping methods such as Kalman filters and finite impulse response filters whose computational burden is polynomial in the dimension of the state, e.g. methods that rely on a fixed finite number of matrix products per timestep and as such may be limited to offline estimation settings.

Online state estimation methods have been created for complex systems with continuous states and discrete modes, such as the case for legged robots making and breaking contact with the ground [Hartley et al., 2020; Bloesch et al., 2012]. In these settings, an extended Kalman filter is used to estimate the continuous states and the discrete mode is directly measured through contact sensors. The primary focus of these works is on the continuous phases rather than the discrete mode transitions due to the presence of direct mode sensing. Therefore, these estimators do not directly work for general hybrid systems, because there might not be a sensor to determine the hybrid event and there might be discontinuous jumps in the state.

4.3.2 Non-smooth systems and the saltation matrix

This work makes extensive use of the saltation matrix [Aizerman and Gantmakher, 1958; Hiskens and Pai, 2000; Leine and Nijmeijer, 2013; Burden et al., 2018b], which is a discontinuous update to the variational equation solution [Khalil and Grizzle, 2002] and is a key part of linearizing hybrid dynamics around a chosen trajectory. They have previously been used to analyze stability of periodic solutions [Aizerman and Gantmakher, 1958], trajectory sensitivity [Hiskens and Pai, 2000], and infinitesimal contraction [Burden et al., 2018b]. Most importantly for this work, the

saltation matrix has also been used to derive a covariance propagation update law for mapping distributions through hybrid transitions [Biggio et al., 2014].

4.4 Problem Formulation

The problem we address in this work is the estimation of continuous states of a hybrid dynamical system given:

1. A model of the dynamics in each mode.
2. A model of how the state resets between modes.
3. The location of the hybrid guards.
4. Measurements of the system's continuous state.

We are specifically not considering:

1. The probability of the discrete state.
2. Hybrid systems with intersecting guards [Scholtes, 2012, § 3-4] (e.g. in a walking system when multiple feet impact simultaneously).

Hybrid systems of the type given in Def. 1 can exhibit complex behavior including sliding [Jeffrey, 2014], branching [Simić et al., 2000], Zeno, and more. To ensure that the saltation matrix is well defined for all transitions, we accept the assumptions (which are conventional, e.g., [Aizerman and Gantmakher, 1958; Bernardo et al., 2008; Leine and Nijmeijer, 2013; Burden et al., 2016]) enumerated in [Burden et al., 2018b, Assumptions. 1] to limit the class of hybrid dynamic systems under consideration to possess piecewise-smooth trajectories. In particular, a key assumption is that transitions are *transverse*, i.e.,

$$\frac{d}{dt} g_{(I,J)}(t, x(t)) =$$

$$D_t g_{(I,J)}(t,x) + D_x g_{(I,J)}(t,x) \cdot F_I(t,x) < 0, \tag{4.1}$$

Note that (4.1) restricts the definition of the guard from Def. 1 to be both a sublevel set and only exist when the vector field is transverse to it at the boundary. That is, we can write each guard set $G_{(I,J)}$ as the following, where $g := g_{(I,J)}$, and $x(t)$ is a trajectory in D_I

$$G_{(I,J)} := \left\{ x \in D_I \;\middle|\; g(t,x) \leq 0, \frac{d}{dt} g(t, x(t)) < 0 \right\} \tag{4.2}$$

Intuitively, transversality implies that trajectories initialized nearby a given $G_{(I,J)}$ undergo exactly one transition for small times. This assumption also ensures the denominator in (3.2) does not approach zero.

With these definitions and assumptions, we can now apply the saltation matrix to propagate covariance [Biggio et al., 2014, Eq. 17] as part of a dynamic update of a probability distribution at a hybrid transition,

Proposition 3. *When the higher order terms are zero, the mean μ and covariance Σ of a hybrid system at the time of a reset are updated as (where $\mu^* := \mu(\bar{t}_i)$),*

$$\mu(\underline{t}_{i+1}) = R_{(I,J)}(\bar{t}_i, \mu^*) \tag{4.3}$$

$$\Sigma(\underline{t}_{i+1}) = \Xi_{(I,J)}(\bar{t}_i, \mu^*) \Sigma(\bar{t}_i, \mu^*) \Xi_{(I,J)}(\bar{t}_i, \mu^*)^T \tag{4.4}$$

4.5 Kalman filtering for hybrid systems

In this section, we present the Salted Kalman Filter (SKF) by applying Prop. 3 on the mapping of second moments to Kalman filters, which enables their use on hybrid dynamical systems. First, we assume $\forall I,\ F_I(t,x) = A_I(t)x + B_I(t)u(t)$, i.e. each mode's vector field is linear. Note that for a non-linear or linear time varying F_I, A_I and B_I are obtained through sampling. Discretized linear matrices with timestep Δ are denoted with $A_{I,\Delta}$ and $B_{I,\Delta}$. To simplify expressions for discrete

timesteps, we abuse notation and use $a(k) := a(t_k)$ for any relevant function a. Without loss of generality, we assume the case $u(k) = 0 \forall\, k$. To start, the stochastic difference equations considered for the standard Kalman filter [Welch and Bishop, 1995, Eqn. 1.1] on domain I for a hybrid dynamical system with linear dynamics are given by

$$x(k+1) := A_{I,\Delta} x(k) + \omega_{I,\Delta}(k) \tag{4.5}$$

where the process noise, $\omega_{I,\Delta}$, is sampled from a zero mean Gaussian distribution with covariance $W_{I,\Delta}$ at each timestep where the effect of the noise is constant throughout the timestep and is handled by integration.

$$f_{I,\Delta}(x, u, \omega(k)) = \int_{t_k}^{t_k+\Delta} (F_I(t, x, u) + \omega(k))\, dt \tag{4.6}$$

The stochastic measurement equation [Welch and Bishop, 1995, Eqn. 1.2] is defined to be $y(k) := C_I x(k) + v_I(k)$, where C_I is the measurement matrix, and v_I is the measurement noise that is sampled from a zero mean Gaussian distribution with covariance V_I.

The standard Kalman filter consists of two parts: the *a priori* update,

$$\hat{x}(k+1|k) = A_{I,\Delta} \hat{x}(k) \tag{4.7}$$

$$\hat{\Sigma}(k+1|k) = A_{I,\Delta} \hat{\Sigma}(k) A_{I,\Delta}^T + W_{I,\Delta} \tag{4.8}$$

and the *a posteriori* update,

$$K_{k+1} = \hat{\Sigma}(k+1|k) C_I^T \left[C_I \hat{\Sigma}(k+1|k) C_I^T + V_I \right]^{-1} \tag{4.9}$$

$$\begin{aligned} \hat{x}(k+1|k+1) = \hat{x}(k+1|k) \\ + K_{k+1} \left[y(k+1) - C_I \hat{x}(k+1|k) \right] \end{aligned} \tag{4.10}$$

$$\hat{\Sigma}(k+1|k+1) = \hat{\Sigma}(k+1|k) - K_{k+1} C_I \hat{\Sigma}(k+1|k) \tag{4.11}$$

where K_{k+1} is the Kalman gain [Welch and Bishop, 1995, Eqns. 1.9–1.13].

While the standard Kalman filter is adequate when a trajectory is confined to a single domain, we must also account for hybrid events. In this setting, we assume that the true time of impact to the guard $\bar{t}_i$ is unknown to the filter and is estimated by determining when a hybrid transition occurs for the mean. In this filter, we allow both the *a priori* and *a posteriori* update to trigger a hybrid transition. Therefore, both updates are modified such that the mean and covariance are properly transformed during the hybrid transition.

In this section we first show these changes for a Kalman filter on a hybrid dynamical system with linear dynamics (Sec. 4.5.1–4.5.2), then the same changes are similarly applied for the Extended Kalman filter on general hybrid dynamical systems (Sec. 4.5.3).

4.5.1 Hybrid transition during *a priori* update

For the *a priori* update, the state is propagated from the previous estimate for a single timestep Δ. If the guard and transversality conditions (4.2) are not met during the propagation, no hybrid transition is considered and the standard update is used (4.7)–(4.8). If the conditions are met for the estimated mean trajectory, then the forward simulation is stopped and the time of impact $\bar{t}_i = t_k + \Delta_1$ is estimated to be the stopping time – where $\Delta_1 = \bar{t}_i - t_k$ and $\Delta_2 = t_{k+1} - \bar{t}_i$ denote the sub-timesteps such that $\Delta_1 + \Delta_2 = \Delta$. Because we assume that a finite number of isolated transitions occur, this process can be repeated until the entire timestep is simulated. Without loss of generality, in this section we only consider the case where a single transition occurs, but appending additional transitions can be computed in a similar fashion.

If a transition occurs from mode I to mode J, the stochastic dynamics (4.5) are defined to be,

$$x(k+1) := A_{J,\Delta_2}\left(R_{(I,J)}\left(\left[A_{I,\Delta_1}x(k) + \omega_{I,\Delta_1}(k)\right]\right) + \omega_{R_{(I,J)}}(k)\right) + \omega_{J,\Delta_2}(k) \tag{4.12}$$

where $\omega_{R_{(I,J)}}$ is the reset process noise, sampled from a zero mean Gaussian distribution with covariance $W_{R_{(I,J)}}$, ω_{I,Δ_1} is the process noise in domain I with timestep Δ_1, and ω_{J,Δ_2} is the process

noise in domain J with timestep Δ_2. The dynamic update at transition (4.3)–(4.4) augmented with the reset process noise is,

$$x(\underline{t}_{i+1}) = R_{(I,J)} x(\bar{t}_i) \tag{4.13}$$

$$\Sigma(\underline{t}_{i+1}) = \Xi_{(I,J)} \Sigma(\bar{t}_i) \Xi_{(I,J)}^T + W_{R_{(I,J)}} \tag{4.14}$$

where the saltation matrix is evaluated at $\Xi_{(I,J)} = \Xi_{(I,J)}(\bar{t}_i, x(\bar{t}_i))$. Combined with the continuous *a priori* updates before and after transition, (4.7)–(4.8), the *a priori* update over a full timestep is,

$$\hat{x}(k+1|k) = A_{J,\Delta_2} R_{(I,J)} A_{I,\Delta_1} \hat{x}(k) \tag{4.15}$$

$$\hat{\Sigma}(k+1|k) = A_{J,\Delta_2} [\Xi_{(I,J)} (A_{I,\Delta_1} \Sigma(k) A_{I,\Delta_1}^T + W_{I,\Delta_1}) \Xi_{(I,J)}^T + W_{R_{(I,J)}}] A_{J,\Delta_2}^T + W_{J,\Delta_2} \tag{4.16}$$

where the saltation matrix is evaluated at $\Xi_{(I,J)} = \Xi_{(I,J)}(\bar{t}_i, A_{I,\Delta_1} \hat{x}(k))$.

A naive approach to updating the covariance through a hybrid transition is to use the Jacobian of the reset function instead of the saltation matrix in Eq. (4.16). To illustrate the difference between this naive approach and the proposed, we compare using the Jacobian of the reset map instead of the saltation matrix in all experiments.

4.5.2 Hybrid transition during *a posteriori* update

Next, we consider the case where the measurement update pulls the mean estimate into a guard set (4.2), i.e. $\hat{x}(k+1|k+1) \in G_{(I,J)}$ for some J. In that case, the *a posteriori* update is modified by applying the reset to the mean and the saltation update to the covariance after applying the standard update (4.9)–(4.11),

$$\tilde{x}(k+1|k) = R_{(I,J)} \hat{x}(k+1|k) \tag{4.17}$$

$$\tilde{\Sigma}(k+1|k) = \Xi_{(I,J)} \hat{\Sigma}(k+1|k) \Xi_{(I,J)}^T + W_{R_{(I,J)}} \tag{4.18}$$

where the saltation matrix is evaluated at $\Xi_{(I,J)} = \Xi_{(I,J)}(\bar{t}_i, \hat{x}(k+1|k))$. These $\tilde{x}(k+1|k)$ and $\tilde{\Sigma}(k+1|k)$ are the updated *a posteriori* mean and covariance in the new hybrid domain, J. Note that this update is identical to (4.13)–(4.14).

4.5.3 Extended Kalman Filter

Similar to the Kalman filter, the standard Extended Kalman Filter (EKF) [Welch and Bishop, 1995, Eqn. 2.1–2.2] can be directly applied for nonlinear hybrid systems when no transition occurs. The nonlinear stochastic dynamics are given by

$$x(k+1) = f_{I,\Delta}(x(k), u(k), \omega(k)) \tag{4.19}$$

$$\hat{A}_{I,\Delta} = D_x f_{I,\Delta}(x(k), u(k), \omega(k)) \tag{4.20}$$

$$\hat{W}_{I,\Delta} = D_\omega f_{I,\Delta}(x(k), u(k), \omega(k)) \tag{4.21}$$

$$y(k) = h_I(x(k), v_I(k)) \tag{4.22}$$

$$\hat{C}_I = D_x h_I(x(k)) \tag{4.23}$$

where $f_{I,\Delta}$ is the discrete nonlinear update for the continuous dynamics F_I, $\hat{A}_{I,\Delta}$ is the linear approximation of the dynamics, $\hat{W}_{I,\Delta}$ is the linear approximation of the process noise, h_I is the measurement function and $\hat{C}_I$ is the linear approximation of the measurement function.

When there is a hybrid transition during the *a priori* update, the dynamic updates for the nonlinear transition case are substituted in the same manner as the linear case into (4.15)–(4.16). When there is a hybrid transition during the *a posteriori* update, the mean update equation (4.17) is applied with the full nonlinear reset map, while the covariance update (4.18) is the same for both the linear and nonlinear hybrid systems because the saltation matrix is already a linearization. With these updates, the nonlinear extension to the Salted Kalman Filter follows naturally.

4.5.4 Summary and psuedocode

The Salted Kalman Filter (SKF) as presented above is summarized in Algorithm 1. Note that the only difference from the standard Kalman Filter algorithm is applying the proposed moment updates when the estimated state satisfies the guard condition (lines 7–11 and 16–20). The SKF is in many ways similar to the EKF because the saltation matrix is a linearization about the hybrid transition – if the transition is linear or the prediction is close to the actual then the filter performs well. This property holds for the nonlinear Extended SKF as well, and in general this filter suffers from the same pitfalls as the EKF. Furthermore, like the EKF this linearization means that the optimal belief may not remain Gaussian, and thus that the filter may fail to have the optimally properties we obtain in the linear case.

For the measurement update, if a hybrid transition is triggered, the approach presented here simply transforms the already updated estimates. However, a more accurate approach might include breaking up the measurement update into sub-updates over each domain. In this work, we assume the updates are small enough such that this isn't an issue, but as the measurement update magnitude increases, this may be worth investigating. While the extended version of this filter is not optimal, like the EKF, we expect that it will perform well when the covariances and timesteps are relatively small so that the local linearizations hold. Therefore, we expect the performance of the filter to falter when the estimation heavily deviates from the actual trajectory in cases such as initializing the filter far away from the actual starting state, initializing in the wrong mode, or trajectories with grazing impact (when the dynamics are not transverse to the guard).

4.6 Experiments

This section lays out the experimental design (Sec. 4.6.1) and example systems (Sec. 4.6.2) that are used to test the utility of the Salted Kalman Filter.

Algorithm 1 Salted Kalman Filter (SKF)

input $(t_k, x_k, \Sigma_k, m_k, y_{k+1})$
$\hat{t} \leftarrow t_k, \hat{x} \leftarrow x_k, \hat{\Sigma} \leftarrow \Sigma_k, I \leftarrow m_k$
while $(\hat{t} < t_k + \Delta)$ **do**
    $(\hat{t}^+, \hat{x}) \leftarrow$ integrate $F_I(\hat{t}, \hat{x})$
        until $(\hat{t}^+ = t_k + \Delta)$ or $(\exists J$ s.t. $\hat{x} \in G_{(I,J)})$
    $\Delta_1 \leftarrow \hat{t}^+ - \hat{t}, \hat{t} \leftarrow \hat{t}^+$
    $\Sigma_k \leftarrow A_{I,\Delta_1} \hat{\Sigma} A_{I,\Delta_1}^T + W_{I,\Delta_1}$ ▷ (4.8)
    if $\exists J$ s.t. $\hat{x} \in G_{(I,J)}$ **then**
        $\hat{x} \leftarrow R_{(I,J)}(\hat{t}, \hat{x})$ ▷ (4.13)
        $\hat{\Sigma} \leftarrow \Xi_{(I,J)} \hat{\Sigma} \Xi_{(I,J)}^T + W_{R_{(I,J)}}$ ▷ (4.14)
        $I \leftarrow J$
    end if
end while
$K \leftarrow \hat{\Sigma} C_I^T \left[C_I \hat{\Sigma} C_I^T + V_I \right]^{-1}$ ▷ (4.9)
$\hat{x} \leftarrow \hat{x} + K \left[y_{k+1} - C_I \hat{x} \right]$ ▷ (4.10)
$\hat{\Sigma} \leftarrow \hat{\Sigma} - K C_I \hat{\Sigma}$ ▷ (4.11)
if $\exists J$ s.t. $\hat{x} \in G_{(I,J)}$ **then**
    $\hat{x} \leftarrow R_{(I,J)}(\hat{t}, \hat{x})$ ▷ (4.17)
    $\hat{\Sigma} \leftarrow \Xi_{(I,J)} \hat{\Sigma} \Xi_{(I,J)}^T + W_{R_{(I,J)}}$ ▷ (4.18)
    $I \leftarrow J$
end if
$t_{k+1} \leftarrow \hat{t}, x_{k+1} \leftarrow \hat{x}, \Sigma_{k+1} \leftarrow \hat{\Sigma}, m_{k+1} \leftarrow I$
return $(t_{k+1}, x_{k+1}, \Sigma_{k+1}, m_{k+1})$

4.6.1 Experimental Design

In the experiments, three different estimation techniques are used: 1) the proposed Salted Kalman Filtering (SKF) algorithm using the saltation matrix to map covariance, 2) the naive mapping using the Jacobian of the reset map (which we call the Jacobian of the Reset Kalman Filter, JRKF, and which follows Algorithm 1 but with the saltation matrix Ξ replaced by the Jacobian of the reset map $D_x R$), and 3) a hybrid system Particle Filter (PF), following [Koutsoukos et al., 2002]. Experiments are performed in simulation to ensure consistency and accurate model knowledge. These experiments evaluate the SKF by comparing the mean squared error of the 3 filters in a series of Monte Carlo tests.

For the simulation, the stochastic difference equation, (4.19), is calculated for each timestep using MATLAB's `ode45` [Shampine et al., 2003] where the integration follows (4.6). Ode45 is

used to account for the guard zero crossing detection using the MATLAB event location feature.

Tests comparing the Kalman Filters were run with a range of measurement noise, process noise, and time steps. Tests comparing to the particle filter were run with a range of time steps with a single representative process and measurement noise. For simplicity the starting covariance, starting mean, reset covariance, chosen measurements, and simulation time were held constant between trials.

The effectiveness of the filter for each trial is evaluated by calculating the mean squared error (MSE) along a simulated trajectory,

$$\text{MSE} = \frac{1}{K}\sum_{k=1}^{K}\left((x(t_k) - \hat{x}(t_k))^T(x(t_k) - \hat{x}(t_k)\right) \tag{4.24}$$

where K is the number of time steps, $\hat{x}(t_k)$ is the state estimate at time t_k, and $x(t_k)$ is the true state at time t_k. For each measurement noise, process noise, and time step combination, the filter is run on 1000 randomly sampled starting conditions with randomly sampled process noise and randomly sampled measurements. The same random trials are then passed to each filter for comparison. Each set of trials are compared using the *sign test* [Dixon and Mood, 1946]. The null hypothesis is that the median difference between the pairs is zero, $H_0 : MSE_1 - MSE_2 = 0$. The sign test is chosen because the data are not normally distributed, which rules out the paired t-test, and are not necessarily symmetric, which rules out the Wilcoxon Signed Rank test.

4.6.2 Hybrid System Definitions

We present experiments for two different hybrid systems: 1) a simpler system which retains a Guassian distribution, Sec. 4.6.2, and 2) a more complex system with nonlinear non-identity reset maps, nonlinlear dynamics, and a higher dimensional state space, Sec. 4.6.2.

Constant Flow

The simplest hybrid system we examine is the case where there are two hybrid modes that are linearly separated and which have constant, but distinct, dynamics in each mode. The dynamics in the hybrid modes are defined as, $F_1 = [1, -1]^T$, and $F_2 = [1, 1]^T$. The guard sets are defined at $x_1 = 0$, such that the domain of F_1 is the left half plane and the domain of F_2 is the right half plane (Fig. 4.1). The reset is an identity map. The measurements for this system were chosen to be both states, i.e., $h_I(x) = x$.

Asymmetric Spring Loaded Inverted Pendulum (ASLIP)

The asymmetric spring loaded inverted pendulum (ASLIP) system consists of a spring leg, torsional spring hip, and a body with inertia in the sagittal plane as shown in Fig. 4.2. This system is similar to the one in [Poulakakis and Grizzle, 2009] and a full derivation for the system dynamics can be found in Section 3.8.5. This hybrid system is especially useful to analyze because it includes both nonlinear dynamics and non-identity resets.

In this system, the body configuration space is defined to be the position and orientation of the body $q_b := [x_b, y_b, \theta_b]^T \in \mathbb{R}\times\mathbb{R}\times\mathbb{S}^1$. The leg configuration space is defined to be the angle between the toe and the ground, the angle of the hip, and the length of the leg $q_l := [\theta_t, \theta_h, l_l]^T \in \mathbb{S}^1\times\mathbb{S}^1\times\mathbb{R}$, where impact location of the toe defines a pin joint for the body to pivot around. Once the location of the toe, $q_t = [x_t, y_t]^T \in \mathbb{R}\times\mathbb{R}$, is fixed to a ground location, either configuration can be used to define the full configuration space of the system. When the toe position is known, the transformation from the leg configuration to the body configuration is defined as $T_{lb} : (q_l, q_t) \mapsto q_b$, while the inverse mapping is defined as $T_{bl} : (q_b, q_t) \mapsto q_l$.

Hybrid mode 1 is defined to be when the toe is not in contact with the ground. The resulting domain $\mathcal{D}_1$ is chosen to be parameterized by the body's configuration, toe position, and body's velocity.

$$[x_b, y_b, \theta_b, x_t, y_t, \dot{x}_b, \dot{y}_b, \dot{\theta}_b]^T \in \mathcal{D}_1 \tag{4.25}$$

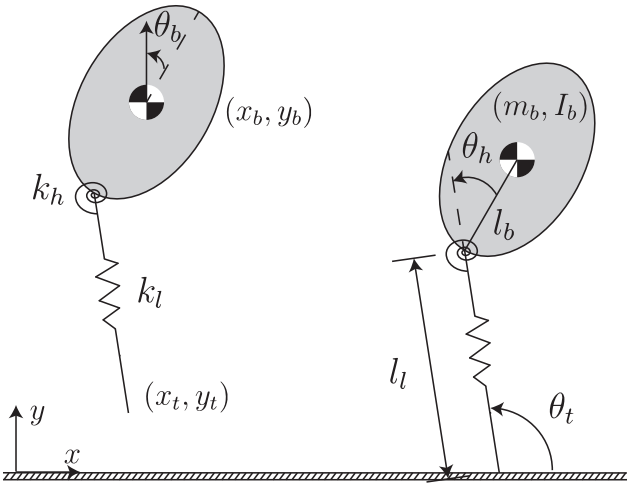

Figure 4.2: Asymmetric Spring Loaded Inverted Pendulum (ASLIP) diagram showing the aerial phase hybrid mode on the left and the stance phase hybrid mode on the right and their corresponding configuration variables.

When the toe is in contact with the ground, the hybrid mode is 2. The domain $\mathcal{D}_2$ is chosen to be parameterized by the toe angle with the ground, hip angle, the leg extension, toe position, the time derivative of the toe angle, hip angle, and leg extension.

$$[\theta_t, \theta_h, l_l, x_t, y_t, \dot{\theta}_t, \dot{\theta}_h, \dot{l}_l]^T \in \mathcal{D}_2 \tag{4.26}$$

Note that the toe position is augmenting the state rather than being treated as an external parameter because variations in the toe placement affect the other configuration states. Because of this, the toe dynamics are constrained relative to the body in domain 1 and relative to the ground contact in domain 2. These dynamics F_1, F_2 are derived using a Lagrangian approach (see Section 3.8.5). The system parameters and their experimental values are body mass $m_b = 1$, body inertia $I_b = 1$, leg spring constant $k_l = 1000$, hip spring constant $k_\theta = 400$, body length $l_b = 0.5$, acceleration due to gravity $a_g = 9.8$, resting leg length $l_{l0} = 1$, and resting angle of the hip spring $\theta_{h0} = -\frac{\pi}{8}$.

The guard for mode 1 is defined to be when the toe touches the ground, $g_{(1,2)}(t, q, \dot{q}) = y_t$, and the guard for mode 2 is defined to be when the normal force of the toe reaches zero, i.e when the leg spring reaches the resting length, $g_{(2,1)}(t, q, \dot{q}) = l_l - l_{l0}$. The reset maps are defined to be the

coordinate changes from the body states to the leg states.

$$R_{1,2} = [T_{bl}(q_b), q_t, D_{q_b} T_{bl}(q_b, q_t)\dot{q}_b]^T \tag{4.27}$$

$$R_{2,1} = [T_{lb}(q_l) q_t D_{q_l} T_{lb}(q_l, q_t)\dot{q}_l]^T \tag{4.28}$$

For this system, only measurements of the body states are given, because it is assumed that the hybrid mode is unknown. Therefore, in the aerial phase, hybrid mode 1, the measurement function is simply, $h_1(x) = [q_b, \dot{q}_b]^T$. However, in the stance phase, hybrid mode 2, the states are the leg states and the toes positions and cannot be directly compared against the body measurements. Therefore, the measurement function in hybrid mode 2 is the transformation from leg states to body states, $h_2(x) = [T_{lb}(q_l), T_{lb}(q_l, q_t, \dot{q}_l)]^T$.

4.7 Results

In this section we present the results of the experiments detailed in the previous section on the example systems.

4.7.1 Constant Flow

The first experiment uses the constant flow system defined in Sec. 4.6.2 and shown in Fig. 4.1. The system was simulated for 5 seconds with 4 different time steps: $\Delta = 5, 1, 0.1$, and 0.05 seconds. The process covariance levels ranged from $\|W_{I,\Delta}\| = 0.1\Delta^2$ to $0.0001\Delta^2$ and the measurement covariance was swept from $\|V_I\| = 1$ to 0.0001, both in powers of 10, for a total of 4 process covariance levels and 5 measurement covariance levels. In total, the Monte Carlo simulations for the 80 parameter sets were tested with 1000 trials each. An example experiment is shown in Fig. 4.3. Note the difference when comparing the error starting at the hybrid transition.

The result of the Monte Carlo Kalman filter tests were that the SKF performed better than the JRKF for 76 of the 80 combinations (to statistical significance $p < 0.05$). In the 4 remaining

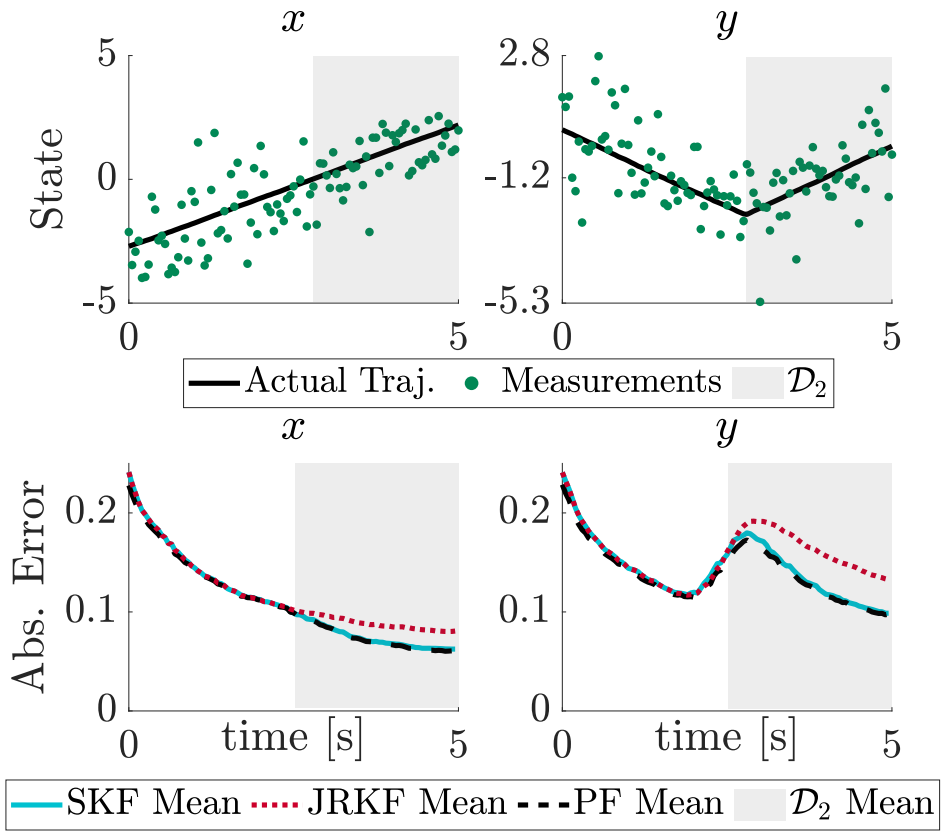

Figure 4.3: Kalman filter results on the constant flow system. Note that the main differences are just after the transition where the methods differ, but because Kalman Filters are stable these differences disappear as time goes on. Testing conditions for this example are timestep $\Delta = 0.05$s, process noise $\|W_{I,\Delta}\| = 0.01\Delta^2$, and measurement noise $\|V_I\| = 1$. Top: For a single trial, the ground truth trajectory (black solid) is shown with the measurements (green dots) and highlighting (gray shaded) when the system is in $\mathcal{D}_2$. Bottom: Absolute mean error is plotted for the SKF (blue solid), JRKF (red dots), and PF (black dashed) while highlighting (gray shaded) the mean transition time to $\mathcal{D}_2$.

cases the filters are statistically indistinguishable, and in none of the experiments did the JRKF outperform the SKF to statistical significance. For each of these cases, the time step is large, the measurement noise is low, the process noise is high, and so both filters depend mostly on the sensors. Therefore, the difference in dynamic update becomes less important.

For the particle filter experiment, the following parameters were chosen: process noise $\|W_{I,\Delta}\| = 0.01\Delta^2$, measurement noise $\|V_I\| = 1$, initial covariance $\Sigma(0) = 0.1I$, and $\Delta = 5$, 1, 0.1, and 0.05 seconds. There was no noise added to reset because the reset map is an identity transformation. The particle filter was initialized with 50 to 3000 particles sampled from the initial distribution.

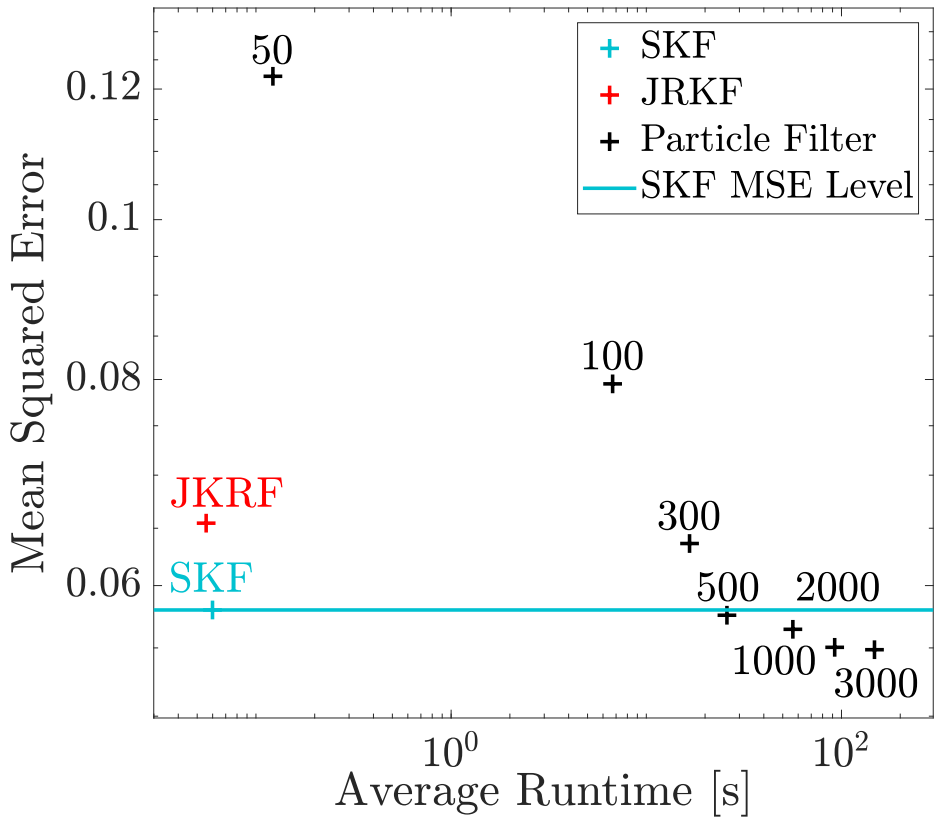

Figure 4.4: Mean Squared Error versus average runtime for constant flow case with the particle filter ranging from 50 to 3000 particles (black pluses) compared against the JRKF (red plus) and SKF (blue plus, with a constant blue line highlighting the SKF MSE level for comparison). The means were taken from a Monte Carlo Simulation with 1000 trials where $\Delta = 0.05$s, process noise $\|W_{I,\Delta}\| = 0.01\Delta^2$, and measurement noise $\|V_I\| = 1$.

The results of the particle filter experiments are shown in Fig. 4.4, where it is clear that the particle filter took significantly higher computation time than the Kalman filters. This is expected, because the Kalman filters is only simulating 1 particle's mean and updating the covariance with matrix computations. Starting only at 1000 particles did the PF perform statistically better than the SKF, with a decrease of 2.7% MSE at the cost of taking 941 times longer to compute. At 2000 particles, the decrease is 5.2% in MSE and the computation required 1736 times the SKF's computation time. Increasing the number of particles to 3000 did not result in a statistically significant improvement over 2000 and so for further comparison with the SKF and JRKF, the number of particles was held constant at 2000.

Considering the effect of the time step on the particle filter experiments, at the largest time step

Table 4.1: The covariance magnitude at the time of transition $\|\Sigma(\bar{t}_i)\|$ and the ratio between the time it takes to transition 99% of the probability mass for each timestep level Δ_T and the current timestep length Δ for the constant flow system with process noise $\|W_{I,\Delta}\| = 0.1\Delta^2$ and measurement noise $\|V_I\| = 1$. Trials where the PF had a statistically lower MSE than the SKF are marked with a *.

Δ	$\|\Sigma(t_i)\|$	Δ_T/Δ
$5s$	0.16	0.38
$1s^*$	0.10	1.5
$0.1s^*$	0.028	7.9
$0.05s^*$	0.015	12

(Δ = 5s) the MSE of the SKF and the PF are statistically indistinguishable. For the smaller time steps (Δ = 1s, 0.1s, and 0.05s), the PF has lower MSE than the SKF ($p < 0.05$). We hypothesize that this is due to the assumption in the SKF that the majority of the probability mass transitions together during a single timestep. The SKF performs comparably worse when the timesteps are small and the distribution is split across a hybrid transition. To test this hypothesis, we compare the time step levels to the time it takes for this system to transition 99% of the probability mass at the transition time as shown in Table 4.1. We find that if the time to transition Δ_T was less than the timestep duration Δ, then no increase in performance was observed with the PF.

4.7.2 ASLIP

The Kalman filtering and particle filtering experiments were also run on the ASLIP system, defined in Sec. 4.6.2. For these tests, we simulated the dynamics for 1.25 seconds which resulted in 2 jumps (4 hybrid transitions). Experiment time steps were set at $\Delta \in \{0.03, 0.01, 0.005, 0.001\}$ seconds. The process noise covariance levels were $\|W_{I,\Delta}\| = 0.01\Delta^2$, $0.001\Delta^2$, and $0.0001\Delta^2$, and the measurement noise covariance levels were $\|V_I\| = 0.005$, 0.001, and 0.0001. The initial covariance was set to be $1 \times 10^{-4} I$, where the noise in the toe position was set to match the constraint between the body configuration and the toe (as the toe position is correlated to the body states). Reset noise is not applied because there is no uncertainty in the coordinate transformation.

In total, the Monte Carlo simulation for the 36 parameter sets that were tested with 100 trials

each. An example experiment is shown in Fig. 4.5. The result of these tests were that the SKF performed better than the JRKF for all 36 combination with statistical significance ($p < 0.001$). While the SKF performs better than the JRKF on average over all states, this does not indicate that the SKF performs better than the JRKF in all coordinates for each timestep. In several of the Monte Carlo simulations, the mean absolute error peaked above the JRKF's mean in $\dot{x}_b$, $\dot{y}_b$, or $\dot{\theta}_b$ for several timesteps – generally after the first touchdown. However, one consistent difference that was seen in all simulations was that SKF had sustained improvements in the vertical body position y_b. The difference between the saltation matrix and the Jacobian of the reset map on impact is in the column associated with the vertical height y_b. Therefore, the improvements in y_b are expected because the dynamics along this axis are accounted for.

For the particle filter experiment, $30,000$ particles were used and the following testing parameters were chosen: process noise $\|W_{I,\Delta}\| = 0.01\Delta^2$, measurement noise $\|V_I\| = 0.005$, and $\Delta = 0.005$ seconds. An example run with these parameters are shown in the top plot of Fig. 4.5 and the filter performance is shown in the lower plot. As with the constant flow system, the filters again perform similarly for each state away from hybrid transitions and the differences are magnified near hybrid transitions.

The result of this experiment was that the SKF has a lower MSE than the particle filter with statistical significance ($p < 0.001$) over the 100 trials. We believe that the particle filters performance can be improved to be better than or equal to the performance of the SKF by increasing the number of particles. However, at $30,000$ particles the computation time is already unwieldy, taking on average 5200 seconds to simulate a 1.25s experiment. Similar to constant flow example, the hybrid particle filter takes significantly longer ($\times 22000$) to run than the SKF.

4.8 Conclusion

In this paper, we created a new Kalman filtering algorithm which allows estimation on hybrid dynamical systems with state-defined transitions, including an extended Kalman filter variant

which can handle nonlinear dynamics with non-identity reset maps. This "Salted Kalman Filter" was validated on both a linear and nonlinear system and compared against both a particle filter and the "Jacobian of the reset map" counterpart.

The results show that using our proposed method is statistically better than or equivalent to the naive method in all tested cases. However, both Kalman filters perform well and have relatively low mean squared error. We believe this is because the naive solution and our proposed method have the same mean update and algorithm structure, the fact that they both perform well highlights the importance of having an accurate update for the mean as well as handling each transition case in the algorithm. When comparing against a hybrid particle filter for the constant flow case, the SKF is statistically indistinguishable when we are able to closely approximate that the probability distribution stays Gaussian and that the majority of the probability mass transitions in a single or several time steps. When the assumption that the probability mass transitions over a small number of time steps is broken, the particle filter outperformed the SKF, but the largest increase in performance was small (5.2%) especially compared to the 1736 times increase in computation time.

For the more complex ASLIP system, the SKF performed statistically better than the $30,000$ particle filter when comparing MSE. However, we believe that with enough particles the particle filter should be better than the SKF, though increasing the number of particles would increase the computation time.

The proposed method, similar to the extended Kalman Filter, suffers when model uncertainty is added to the hybrid dynamical system, when the local approximation is violated, or when the noise is non-Gaussian. Overall, like an extended Kalman filter, if the estimate diverges from the actual trajectory (i.e. the estimate is initialized far away from the actual, the starting mode is incorrect, or if an incorrect transition is made) the performance of the filter will suffer. Incorrect mode transitions are mitigated by the class of hybrid dynamical systems that are considered which require transverse guards (Assumption 4.1). In cases where the non-linearity, non-localness, or non-Gaussianness are significant, a hybrid particle filter or other particle filtering approaches may be more appropriate,

but will be accompanied with a respective increase in computation complexity. For a smooth system, an unscented Kalman filter may be used in place of an extended Kalman filter if the local assumption is not valid. However, using an unscented Kalman filter for a hybrid dynamical system may not transfer well because the sampled sigma points may end up past the guard.

Note that while using the saltation matrix captures the update for the covariance to first order, the saltation matrix is model-dependent, and may require significant effort to obtain in practice in order to use (3.2) directly. However, as the saltation matrix is a linear map relating pre- and post-transition states, regression techniques may be able to approximate it with measured data without the need for complete (and differentiable) models of the hybrid system.

While this is a good start to developing an online hybrid state estimation system, there is still further work needed to improve the estimation. Our method does not explicitly reason about the probability of a state or measurement being in a particular hybrid mode or guard, and an extension that reasons about this probability will be covered in future work. Additionally, future work is required to cover distributions which pass through intersections of hybrid guards, in which case an extension based on the Bouligand derivative [Burden et al., 2016; Scholtes, 2012] could be used to capture the propagation of uncertainty.

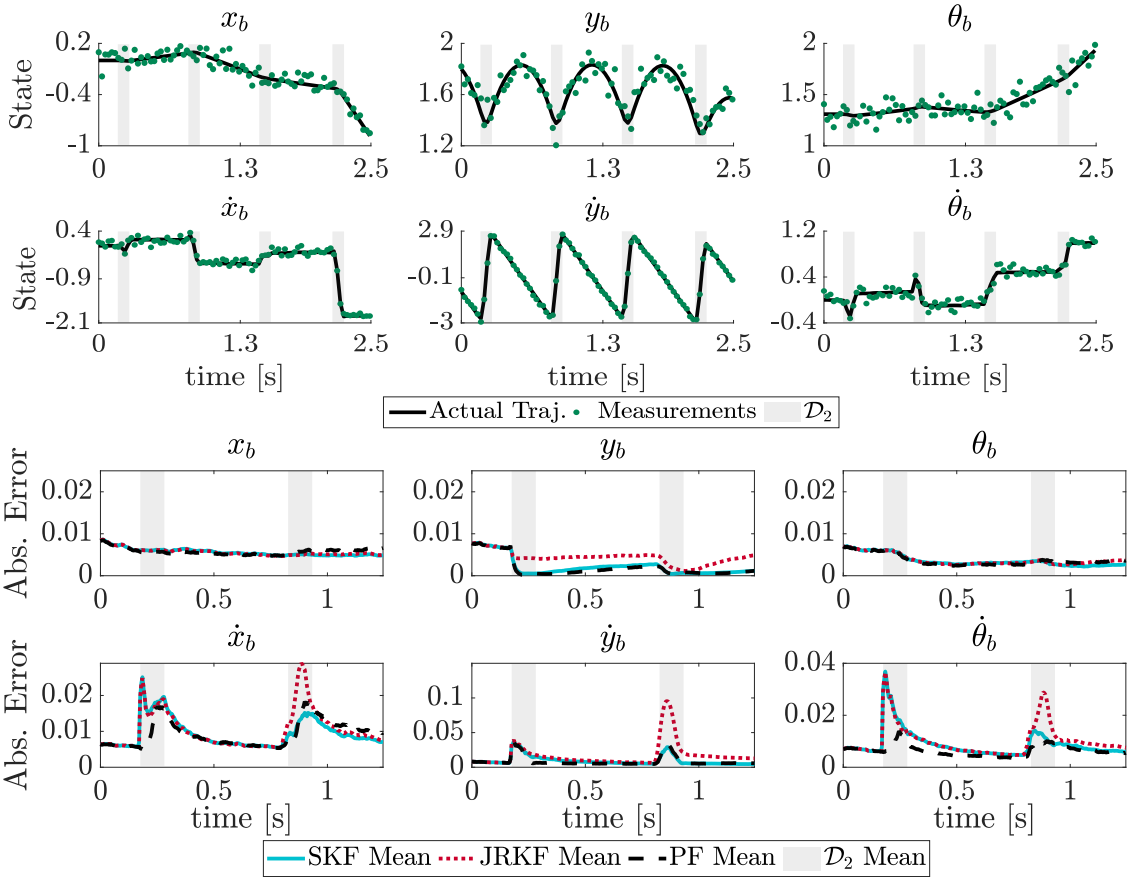

Figure 4.5: ASLIP Kalman filter results comparing the SKF to the JRKF. Note that the main differences between the methods are at the transitions and also that the improvement is in one direction (here, mostly in the vertical body position y_b) because the saltation matrix is different from the Jacobian of the reset map by a rank 1 update. Testing conditions for this example are timestep $\Delta = 0.005$s, measurement noise $\|V_I\| = 0.005$, and process noise $\|W_{I,\Delta}\| = 0.01\Delta^2$. Top: For a single trial, the ground truth trajectory (black solid) is shown with the measurements (green dots) and highlighting (gray shaded) when the system is in $\mathcal{D}_2$. Bottom: Absolute mean error is plotted for the SKF (blue solid), JRKF (red dots), and PF (black dashed) while highlighting (gray shaded) the mean transition times to $\mathcal{D}_2$.

Chapter 5

iLQR for piecewise-smooth hybrid dynamical systems

This chapter previously appeared in [Kong et al., 2021a].

5.1 Abstract

Trajectory optimization is a popular strategy for planning trajectories for robotic systems. However, many robotic tasks require changing contact conditions, which is difficult because of the hybrid nature of the dynamics. The optimal sequence and timing of these modes is typically not known ahead of time. In this work, we extend the Iterative Linear Quadratic Regulator (iLQR) method to a class of piecewise smooth hybrid dynamical systems by allowing for changing hybrid modes in the forward pass, using the saltation matrix to update the gradient information in the backwards pass, and using a reference extension to account for mode mismatch. We demonstrate these changes on a variety of hybrid systems and compare the different strategies for computing the gradients.

5.2 Introduction

For robots to be useful in real world settings, they need to be able to interact efficiently and effectively with their environments. However, systems like the quadcopter perching example shown in Fig. 5.1 often have highly nonlinear dynamics and complex, time-varying environmental interactions that make trajectory planning computationally challenging. These systems are often modeled as mechanical systems with impacts, a type of hybrid dynamical system (Def. 1), [Back et al., 1993; Lygeros et al., 2003; Goebel et al., 2009]. Hybrid dynamical systems differ from smooth dynamical systems in many ways which make planning and control more difficult, including: 1) they contain a discrete component of state (the "hybrid mode") over which the continuous dynamics may differ. 2) These modes are connected by a reset function that applies a discrete (and potentially discontinuous) change to the state. 3) There may be different control authority available in each mode.

While a wide range of trajectory optimization approaches have been proposed for smooth dynamical systems (e.g. [Betts, 1998; Rao, 2009; Kelly, 2017]), most prior methods are not suitable for hybrid dynamical systems. One approach that has been used successfully is direct collocation, which transcribes the trajectory directly into an nonlinear program and optimizes for both the state and control input at discrete points. If the sequence of hybrid modes is fixed and known, the collocation can be solved as a multi-phase method [Von Stryk, 1999; Kelly, 2017] which is a simultaneous optimization over each smooth segment with the reset map defining boundary conditions between them [Schultz and Mombaur, 2009; Posa et al., 2016]. However, the optimal mode sequence is often not known, and so contact-implicit optimization methods have been proposed [Posa et al., 2014; Mordatch et al., 2012]. These methods use complementary constraints to allow for any contact mode sequence, though such constraints are hard to solve in practice and this approach does not extend to generic hybrid systems. Furthermore, for many real-time planning applications direct collocation methods are unfavorable because they scale poorly with time and the trajectories are not feasible until the optimization has finished.

In this paper, we propose to extend the Iterative Linear Quadratic Regulator (iLQR) method

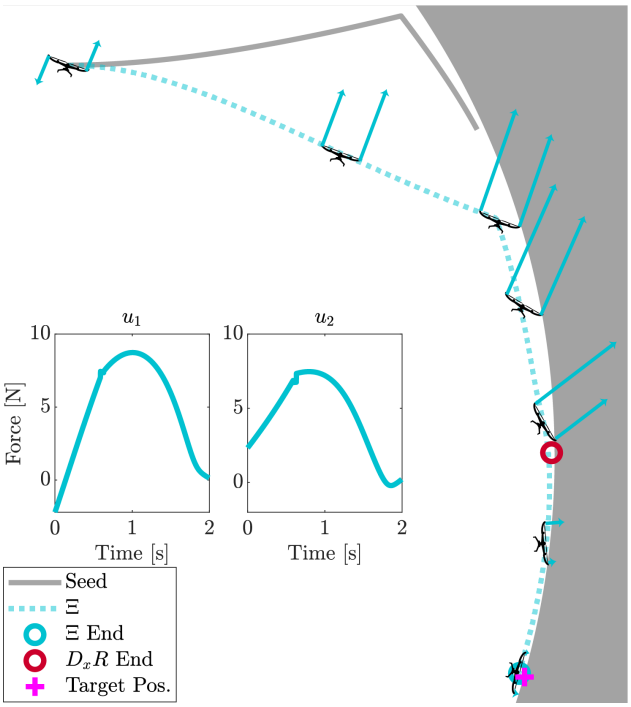

Figure 5.1: Demonstrating an example solution using the proposed hybrid iLQR algorithm (labeled with Ξ, the saltation matrix, Def. 2) where the goal is to control a quadcopter to a target final position (shown with a magenta plus) and can make contact with a curved wall with friction. Using a different approximation for the gradient (Jacobian of the reset map, D_xR, [Li and Wensing, 2020]) leads to poor convergence and significantly higher cost. Note that in the force plots, the optimal input is not smooth because of the hybrid transition.

[Li and Todorov, 2004; Tassa et al., 2012] to work for hybrid systems. iLQR (a special case of the Differential Dynamic Programming method, DDP [Mayne, 1973]) is a shooting method [Betts, 1998] that utilizes linearization in the search direction (backward pass), but implements the full nonlinear dynamics when obtaining the states of the optimized trajectory (forward pass). One advantage of iLQR, like most shooting methods, is that it can be stopped prematurely to produce a feasible trajectory [Posa et al., 2014].

However, traditional iLQR is defined only for smooth systems. Here, we extend iLQR to hybrid systems by:

1. Allowing for varying mode sequences on the forward pass by using event detection to dictate when a transition occurs and enforcing the appropriate dynamics in each mode, Sec. 5.3.2.

2. Applying the reset map on the forward pass and propagating the value function through reset maps in the backwards pass by using a saltation matrix, Sec. 5.3.3.

3. Using reference extensions when there is a mode mismatch to get a valid control input in each mode, Sec. 5.3.4.

In previous hybrid system DDP/iLQR work, [Li and Wensing, 2020] took an important first step by extended the approach from [Lantoine and Russell, 2012] to create an "impact aware" iLQR algorithm which utilizes a prespecified hybrid mode sequence to allow for different dynamics and uses the Jacobian of the reset map to approximate the value function through a hybrid transition. Constrained dynamics and mode sequence are handled by a outer layer in their algorithm. We instead use the saltation matrix (Def. 2), [Leine and Nijmeijer, 2013; Rijnen et al., 2015; Aizerman and Gantmakher, 1958; Burden et al., 2018b], to propagate the value function in the backwards pass. This change makes a significant difference in solution quality and convergence, as we show in Sec. 5.5. Furthermore, to allow use on a more general class of hybrid dynamical systems (not just rigid bodies with contact) without prespecifying the mode sequence, the switching constraints are enforced as part of the dynamics on the forward pass – if the current timestep reaches a hybrid event, the solution jumps to the next hybrid mode using the reset map. These changes enable the algorithm presented here to be run as a standalone algorithm with improved solution quality and convergence properties.

5.3 Derivation/implementation

This section introduces an abridged derivation of iLQR [Li and Todorov, 2004] following [Tassa et al., 2012] and then proposes the changes to make iLQR work on hybrid systems and discusses several important key features of the new algorithm.

5.3.1 Smooth iLQR background

Consider a nonlinear dynamical system with states $x \in \mathbb{R}^n$, inputs $u \in \mathbb{R}^m$, and dynamics $\dot{x} = F(x(t), u(t))$. Define a discretization of the continuous dynamics over a timestep Δ such that at time t_k the discrete dynamics are $x_{k+1} = f_\Delta(x_k, u_k)$, where $t_{k+1} = t_k + \Delta$, $x_k = x(t_k)$, and $u_k = u(t_k)$. Let $U := \{u_0, u_1, ..., u_{N-1}\}$ is the input sequence, J_N is the terminal cost and J is the runtime cost, where J and J_N are both differentiable functions into $\mathbb{R}$.

The optimal control problem over N timesteps is

$$\min_U \quad J_N(x_N) + \sum_{i=0}^{N-1} J(x_i, u_i) \tag{5.1}$$

$$\text{where} \quad x_0 = x(0) \tag{5.2}$$

$$x_{i+1} = f_\Delta(x_i, u_i) \quad \forall i \in \{0, ..., N-1\} \tag{5.3}$$

To solve this problem, DDP/iLQR uses Bellman recursion to find the optimal input sequence U, we which briefly review here. Let $U_k := \{u_k, u_{k+1}, ..., u_{N-1}\}$ be the sequence of inputs including and after timestep k. Define the cost-to-go J_k as the cost incurred including and after timestep k

$$J_k(x_k, U_k) := J_N(x_N) + \sum_{i=k}^{N-1} J(x_i, u_i) \tag{5.4}$$

with $\{x_{k+1}, ..., x_N\}$ the sequence of states starting at x_k based on U_k and (5.3). The value function V (Bellman equation) at state x_k is the optimal cost to go $J_k(x_k, U_k)$, which can be rewritten as a recursive function of variables from the current timestep using the dynamics (5.3),

$$V(x_k) := \min_{u_k} \quad J(x_k, u_k) + V(f_\Delta(x_k, u_k)) \tag{5.5}$$

Since there is no input at the last timestep, the boundary condition of the value is the terminal cost, $V_N(x_N) := J_N(x_N)$. Next, define Q_k to be the argument optimized in (5.5). Optimizing the Bellman equation directly is incredibly difficult. DDP/iLQR uses a second order local approximation of Q_k

where perturbations about the state and input (x_k, u_k) are taken. The resulting function is defined to be

$$\begin{aligned} Q_k(\delta x, \delta u) :=& J(x_k + \delta x, u_k + \delta u) - J(x_k, u_k) + \\ & V(f_\Delta(x_k + \delta x, u_k + \delta u)) - V(f_\Delta(x_k, u_k)) \end{aligned} \tag{5.6}$$

where the value function expansion is for timestep $k + 1$ and when expanded to second order

$$Q_k(\delta x, \delta u) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{ux}^T \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix} \tag{5.7}$$

the expansion coefficients are

$$Q_{x,k} = J_x + f_{x,k}^T V_x \tag{5.8}$$

$$Q_{u,k} = J_u + f_{u,k}^T V_x \tag{5.9}$$

$$Q_{xx,k} = J_{xx} + f_{x,k}^T V_{xx} f_{x,k} + V_x f_{xx,k} \tag{5.10}$$

$$Q_{ux,k} = J_{ux} + f_{u,k}^T V_{xx} f_{x,k} + V_x f_{uu,k} \tag{5.11}$$

$$Q_{uu,k} = J_{uu} + f_{u,k}^T V_{xx} f_{u,k} + V_x f_{ux,k} \tag{5.12}$$

where subscripted variables represent derivatives of the function with respect to the variable (e.g. $J_x = D_x J$) and the discretized dynamics are abreviated as $f_k = f_\Delta(x_k, u_k)$. Note that the second derivative terms (where adjacency indicates tensor contraction) with respect to the dynamics ($f_{xx,k}$, $f_{uu,k}$, and $f_{ux,k}$) in (5.10)–(5.12) are used in DDP but ignored in iLQR.

With this value function expansion, the optimal control input, δu^*, can be found by setting the derivative of $Q(\delta x, \delta u)$ with respect to δu to zero and solving for δu,

$$\delta u^* = \arg\min_{\delta u} Q(\delta x, \delta u) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x) \tag{5.13}$$

This optimal control input can be split into a feedforward term $u_{ff} = -Q_{uu}^{-1}Q_u$ and a feedback term $K = -Q_{uu}^{-1}Q_{ux}\delta x$. Therefore, the optimal input for the local approximation at timestep k is the sum of the original input and the optimal control input, $u_k^* = u_k + \delta u^*$.

Once the optimal controller is defined, the expansion coefficients of V for timestep k can be updated by plugging in the optimal controller into (5.7)

$$V_x = Q_x - Q_u Q_{uu}^{-1} Q_{ux} \tag{5.14}$$

$$V_{xx} = Q_{xx} - Q_{ux}^T Q_{uu}^{-1} Q_{ux} \tag{5.15}$$

Now that the expansion terms for the value function at timestep k can be expressed as sole a function of $k+1$ the optimal control input can be calculated recursively and stored $(u_{ff,k}, K_k)$. This process is called the backwards pass.

Once the backwards pass is completed, a forward pass is run by simulating the dynamics given the new gain schedule $(u_{ff,k}, K_k)$ and the previous iterations sequence of states and inputs.

$$\hat{x}_0 = x_0 \tag{5.16}$$

$$\hat{u}_k = K_k(\hat{x}_k - x_k) + \alpha u_{ff,k} \tag{5.17}$$

$$\hat{x}_{k+1} = f_\Delta(\hat{x}_k, \hat{u}_k) \tag{5.18}$$

where the new trajectory is denoted with hats $(\hat{x}, \hat{u})$ and α is used as a backtracking line-search parameters $0 < \alpha \leq 1$ [Tassa et al., 2012, Eqn. 12]. The backwards and forwards passes are run until convergence. Following [Tassa et al., 2012], convergence is when the magnitude of the total expected reduction δJ is small

$$\delta J(\alpha) = \sum_{i=0}^{N-1} u_{ff,i}^T Q_{u,i} + \frac{1}{2} \sum_{k=0}^{N-1} u_{ff,i}^T Q_{uu,i} u_{ff,i} \tag{5.19}$$

Convergence issues may occur when Q_{uu} is not positive-definite or when the second order approximations are inaccurate. Regularization is often added to address these issues and here we

use the same regularization scheme as in [Tassa et al., 2012].

5.3.2 Hybrid system modifications to the forward pass

The first change that is required for iLQR to work on hybrid dynamical systems is that the forward pass must accurately generate the hybrid system execution. The dynamics are integrated for the currently active mode I_j for the duration of the hybrid time period j, i.e. $\forall t \in [\underline{t}_j, \bar{t}_j]$, until a guard condition is met,

$$g_{(I_j, I_{j+1})}(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)) = 0 \tag{5.20}$$

To capture these hybrid dynamics in the discrete forward pass, the discretized dynamics are computed using numerical integration with event detection, so that if no event occurs the dynamic update, (5.3), is,

$$f_{\Delta_j}(\hat{x}_k, \hat{u}_k) := \int_{t_k}^{t_{k+1}} f_{I_j}(x(t), \hat{u}_k)dt + \hat{x}_k \tag{5.21}$$

If during the integration the hybrid guard condition is met, (5.20), the integration halts, the transition state is stored, the reset map is applied, and then the integration is continued with the dynamics of the new mode, I_{j+1}. Suppose that the guard condition is met once (which is ensured for small times by transversality) at time $\bar{t}_j$, such that $t_k \leq \bar{t}_j \leq t_{k+1}$, then

$$\begin{aligned} f'_{\Delta}(\hat{x}_k, \hat{u}_k) = &\int_{\underline{t}_{j+1}}^{t_{k+1}} f_{I_{j+1}}(x(t), \hat{u}_k)dt + \\ &R_{(I_j, I_{j+1})}\left(\bar{t}_j, \int_{t_k}^{\bar{t}_j} f_{I_j}(x(t), \hat{u}_k)dt + \hat{x}_k\right) \end{aligned} \tag{5.22}$$

Note that this process can be repeated for as finitely many times as there are hybrid changes during a single timestep, but there cannot be infinitely many changes during a single timestep (no Zeno). In this work, we use MATLAB's ode45 method for integration and event detection.

Finally, in addition to updating the dynamics the cost function, (5.4), can be augmented with additional cost terms, J_{N_j}, associated with each hybrid transition between the M hybrid modes, as shown in [Lantoine and Russell, 2012],

$$J_0 = J_N(x_N) + \sum_{i=0}^{N-1} J(x_i, u_i) + \sum_{j=1}^{M-1} J_{N_j}(x_{N_j}) \tag{5.23}$$

Such an addition may be desirable if e.g., one wanted to penalize the occurrences of a transition event in the hopes of having a minimal number of hybrid events.

5.3.3 Hybrid system modifications to the backwards pass

The backwards pass must be updated to reflect the discrete jumps that were added through the hybrid transitions. Away from hybrid transitions, the dynamics are smooth and behave the same way as in the smooth iLQR backwards pass, so our modification to backwards pass is occurs at timesteps where a hybrid transition is made. By substituting (5.22) into (5.5), and adding the transition cost from (5.23), the resulting Bellman equation for the timesteps during hybrid transition j over timestep k is

$$V(x_k) = \min_{U_k} J(x_k, u_k) + J_{N_j}(x_{N_j}) + V(f'_\Delta(x_k, u_k)) \tag{5.24}$$

We elect to approximate the hybrid transition timestep to have the hybrid event occur at the end of the timestep in order to maintain smooth control inputs for each hybrid epoch. For the backwards pass to work on the Bellman equation during transition timesteps, we need to find the linearization of $f'_\Delta(x_k, u_k)$. This linearization step is straight forward when using the saltation matrix to map perturbations pre and post hybrid transition (3.10).

The linearization can be broken up into 2 different steps, where each step the linearization is known.

$$\delta x(\bar{t}_j) \approx f_{x,\Delta_j} \delta x(t_k) + f_{u,\Delta_j} \delta u(t_k) \tag{5.25}$$

$$\delta x(\underline{t}_{j+1}) \approx \Xi \delta x(\bar{t}_j) \tag{5.26}$$

where $f_{*,\Delta_j} = D_* f_{\Delta_j}(x, u)$ and the saltation matrix is abbreviated as $\Xi = \Xi_{(I_j, I_{j+1})}(\bar{t}_j, x(\bar{t}_j), u(t_k))$

These linearization steps can be combined and directly substituted in the coefficient expansion equations (5.8)–(5.12) in place of the f_k terms. For the transition cost, J_{N_j}, an expansion is taken about $\delta x(\bar{t}_j)$ which can be mapped back to $(\delta x(t_k), \delta u(t_k))$ and added to the expansion coefficients. When combining all the expansion terms, the hybrid iLQR coefficients in (5.7) are,

$$Q_{x,k} = J_x + f_{x,\Delta_j}^T J_{x,N_j} + f_{x,\Delta_j}^T \Xi^T V_x \tag{5.27}$$

$$Q_{u,k} = J_u + f_{u,\Delta_j}^T J_{x,N_j} + f_{u,\Delta_j}^T \Xi^T V_x \tag{5.28}$$

$$Q_{xx,k} = J_{xx} + f_{x,\Delta_j}^T J_{xx,N_j} f_{x,\Delta_j} + f_{x,\Delta_j}^T \Xi^T V_{xx} \Xi f_{x,\Delta_j} \tag{5.29}$$

$$Q_{ux,k} = J_{ux} + f_{u,\Delta_j}^T J_{xx,N_j} f_{x,\Delta_j} + f_{u,\Delta_j}^T \Xi^T V_{xx} \Xi f_{x,\Delta_j} \tag{5.30}$$

$$Q_{uu,k} = J_{uu} + f_{u,\Delta_j}^T J_{xx,N_j} f_{u,\Delta_j} + f_{u,\Delta_j}^T \Xi^T V_{xx} \Xi f_{u,\Delta_j} \tag{5.31}$$

After this update to the coefficient expansion, the backwards pass continues normally. If the second order variational expression for the saltation matrix is calculated, then these exact changes can be used for a hybrid DDP version of this backwards pass.

As a alternative expansion, in [Li and Wensing, 2020, Eq. (21)] the authors use the Jacobian of the reset map to propagate perturbations in state through the hybrid transition, instead of the saltation matrix (3.10). For the hybrid backwards pass that we define, this change would be the equivalent of substituting the Jacobian of the reset map in place of the saltation matrix in (5.26)

$$\delta x(\underline{t}_{j+1}) \approx D_x R_{(I_j, I_{j+1})}(\bar{t}_j, x(\bar{t}_j), u(t_k)) \delta x(\bar{t}_j) \tag{5.32}$$

and similarly in the hybrid coefficient expansion equations (5.27)–(5.31). We show empirically that using this alternate version with the Jacobian of the reset map does not perform as well as using the saltation matrix and may not converge.

5.3.4 Hybrid extensions for mode mismatches

Since the forward pass can alter the contact sequence, the new trajectory is not confined to the previous trajectory's mode sequence or timing. This feature is intended because the algorithm can now remove, add, or shift mode transitions if cost is reduced. However, this introduces an issue when the reference mode is not the same as the current mode.

In [Rijnen et al., 2015, Eq. 7], the authors consider the problem of mode mismatch for an optimal hybrid trajectory, both of the reference and of the feedback gains – the reference is extended by integration, and the gains are held constant. We employ their strategy, as well as apply this same rule for the input and the feedforward gains – applying the input intended for a different mode can cause destructive results, or be not well-defined. If the number of hybrid transitions exceeds that of the reference, we elected to hold the terminal state and gains constant, though other choices could be made instead.

5.3.5 Algorithm

With each hybrid modification to iLQR listed in Sections 5.3.2, 5.3.3, and 5.3.4 our new algorithm can be summarized as follows: 1) Given some initial state, input sequence, quadratic loss function, number of timesteps, and timestep duration a rollout is simulated to get the initial reference trajectory and mode sequence. 2) A hybrid backwards pass (using the regularization from [Tassa et al., 2012]) computes the optimal control inputs for the reference trajectory. 3) Hybrid reference extensions are computed on the start and end states for each hybrid reference segment. 4) The forward pass simulates the current mode's dynamics until a hybrid guard condition is met or it is the end of the simulation time; if the guard is reached, the corresponding reset map is applied and the simulation is continued. This forward pass is repeated with a different learning rate until the line search conditions are met [Tassa et al., 2012]. 5) Then the backwards pass, hybrid extensions, and forward passes are repeated until convergence.

5.4 Hybrid System Examples and Experiments

In this section, we define a set of hybrid systems – ranging from a simple 1D bouncing ball to a perching quadcopter with constrained dynamics and friction – and a series of experiments which evaluates how our hybrid iLQR algorithm performs in a variety of different settings.

For all of the examples, we assume that there is no desired reference trajectory to track and that there is no hybrid transition cost J_{N_j} – this means the runtime cost is only a function of input. In each experiment, a comparison against using the Jacobian of the reset map instead of the saltation matrix is made by evaluating the expected cost reduction for the entire trajectory and the final cost. The Jacobian of the reset variant is labeled as $D_x R$-iLQR and the main variant which uses the saltation matrix Ξ-iLQR.

For all examples, $m = 1$ is the mass of a rigid body, $g = 9.8$ is the acceleration due to gravity, the number of timesteps simulated is $N = 1000$, and the timestep duration is $\Delta = 0.001$s unless specified.

The dynamics considered here fall into the category of Euler Lagrange dynamics subjected to unilateral holonomic constraints. We use the dynamics, impact law, and complementarity conditions as derived in [Johnson et al., 2016a]. These systems have configuration variables q where the state of the system is the configurations and their time derivatives $x = [q^T, \dot{q}^T]^T$. When the system is in contact with a constrained surface $a(q) = 0$, a constraint force λ is applied to not allow penetration in the direction of the constraint. The accelerations $\ddot{q}$ and constraint forces λ are found by solving the constraint and accelerations simultaneously,

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + N(q, \dot{q}) + A(q)^T \lambda = \Upsilon(q, u) \tag{5.33}$$

$$A(q)\ddot{q} + \dot{A}(q)\dot{q} = 0 \tag{5.34}$$

where $M(q)$ is the manipulator inertia matrix, $C(q, \dot{q})$ are the Coriolis and centrifugal forces, $N(q, \dot{q})$ are nonlinear forces including gravity and damping, $A(q) = D_q a(q)$ is the velocity constraint, and $\Upsilon(u)$ is the input mapping function.

Suppose the constrained surface $a_J(q)$ is the Jth possible hybrid mode, and the current mode is the unconstrained mode. $a_J(q)$ acts as the guard surface for impacts $g_{(1,J)} = a_J(q)$. When the system hits the impact guard, the velocity is reset using a plastic or elastic impact law [Johnson et al., 2016a].

Releasing a constrained mode (liftoff) occurs when a constraint force becomes attractive rather than repulsive; thus we define hybrid guard $g(t, x, u) := \lambda$ and the reset map at these events are identity transforms because no additional constraints are being added.

5.4.1 Bouncing ball elastic impact

We begin with a 1D bouncing ball under elastic impact [Goebel et al., 2009], where the state $x = [z, \dot{z}]^T$ is the vertical position z and velocity $\dot{z}$. The input u is a force applied directly to the ball. The two hybrid modes, 1 and 2, are defined when the ball has negative velocity $\dot{z} < 0$ and when the ball has non-negative velocity $\dot{z} \geq 0$, respectively. The dynamics on each mode are ballistic dynamics plus the input

$$F_1(x, u) = F_2(x, u) := \left[\dot{z}, \frac{u - mg}{m}\right]^T \tag{5.35}$$

Hybrid mode 1 transitions to 2 when the ball hits the ground, $g_{(1,2)}(x) := z$, and mode 2 transitions to 1 at apex $g_{(2,1)}(x) := \dot{z}$. When mode 1 transitions to 2, an elastic impact is applied, $R_{(1,2)}(x) = [z, -e\dot{z}]^T$ where e is the coefficient of restitution. The reset map from 2 to 1 is identity.

The Jacobian of the reset map and saltation matrix are,

$$D_x R_{(1,2)} = \begin{bmatrix} 1 & 0 \\ 0 & -e \end{bmatrix}, \; \Xi_{(1,2)} = \begin{bmatrix} -e & 0 \\ \frac{(u-mg)(e+1)}{m\dot{z}} & -e \end{bmatrix} \tag{5.36}$$

When transitioning from 2 to 1, both Jacobian of the reset map and saltation matrix are identity.

The problem data is to have the ball fall from an initial height with no velocity, $x_0 = [4, 0]^T$, and end up at a final height x_{des} with no velocity with penalties $R = 5 \times 10^{-7}/\Delta$, $Q_N = 100 I_{2\times 2}$ and the

Table 5.1: Bouncing ball with elastic impacts. Trials vary in optimal number of bounces, number of seeded bounces, which method was used, total cost, and convergence $|\delta J| < 0.05$

Optimal #	Seed #	Method	Actual #	Cost	Converged
0	0	Ξ	0	53.1	Yes
0	0	$D_x R$	0	53.1	Yes
0	1	Ξ	1	114	Yes
0	1	$D_x R$	0	53.1	Yes
0	1	Direct	1	114	Yes
1	0	Ξ	0	97.3	Yes
1	0	$D_x R$	0	97.3	Yes
1	1	Ξ	1	42.5	Yes
1	1	$D_x R$	0	97.3	Yes
1	3	Ξ	1	42.5	Yes
1	3	$D_x R$	1	125	No
3	1	Ξ	1	105	Yes
3	1	$D_x R$	0	114	Yes
3	3	Ξ	3	0.536	Yes
3	3	$D_x R$	3	19.6	No
3	3	No Ext.	3	53.3	No

problems were seeded with a constant input force to obtain different number of bounces. A suite of bouncing conditions are considered and are summarized in Table 5.1. In the case where 0 bounces are optimal $x_{des} = [3, 0]^T$ while where 1 or 3 bounces are optimal $x_{des} = [1, 0]^T$. For 3 bounces the timestep is set to $\Delta = 0.004$. To evaluate the effectiveness of the hybrid extensions, Sec. 5.3.4, an additional comparison using our hybrid iLQR algorithm where we do not apply any hybrid extensions is made. For all cases, a convergence cutoff for this problem is set to be if $|\delta J| \leq 0.05$.

5.4.2 Ball dropping on a spring-damper

Hard contacts are sometimes relaxed using springs and dampers, so we consider the 1D bouncing ball case, but instead of having a discontinuous event at impact, the impact event is extended by assuming the ground is a spring damper (i.e., a force law $f_{sd}(z, \dot{z}) := kz + d\dot{z}$) when being penetrated and a spring when releasing. With this change the system now has an identity reset, but since the saltation matrix is not identity, the hybrid transition still produces a jump in the linearization.

The hybrid modes are defined as: the aerial phase 1, the spring-damper phase 2 and the spring phase 3. The spring and dampening coefficients are chosen to be $k = 100$ and $d = 5$. The guards are when the ball hits the ground $g_{(1,2)} = z$, when the ball no longer has any penetrating velocity $g_{(2,3)} = \dot{z}$, and when the ball is released from the ground $g_{(3,1)} = z$. For all of these transitions, the reset map is an identity transformation and the states do not change.

The example is setup to have the ball fall an initial height with an initial downwards velocity $x_0 = [3, -2]$, end up at a height with no velocity $x_{des} = [1, 0]$, with penalties $R = 0.0001$, $Q_N = 100 I_{2\times2}$ and no input for the seed.

5.4.3 Ball drop on a curved surface with plastic impacts

To test our algorithm with a nonlinear constraint surface, we designed a system where an actuated ball in 2D space is dropped inside a hollow tube and is tasked to end in a goal location on the tube surface.

The configuration states of the system are the horizontal and vertical positions $q = [y, z]^T$. This system consists of two different hybrid modes: the unconstrained mode 1 and in the constrained mode 2. The constrained surface is defined to be a circle with radius 2, $a(q) = 4 - y^2 - z^2$. The dynamics of the system, (5.33), are ballistic dynamics with direct inputs on configurations, $M(q) = mI_{2\times2}$, $N(q, \dot{q}) = [0, -mg]^T$, $C(q, \dot{q}) = 0_{2\times2}$, and $\Upsilon = [u_y, u_z]^T$. The impact guard from (1,2) is defined by the circle's constrained surface and the liftoff guard from (2,1) is the constraint force λ.

The example is setup to have the ball fall from an initial height with velocity pointing down and to the right $x_0 = [1, 0]$, end up at a specific location on circle with no velocity $x_{des} = [-\sqrt{3}, -1, 0, 0]$, with penalties $R = 0.0001$, $Q_N = 100 I_{4\times4}$ and no input for the initial seed except for a vertical force $2mg$ applied for a small duration to cause the ball to momentarily leave the constraint.

5.4.4 Perching quadcopter

We introduce a quadcopter perching example inspired by [Lussier Desbiens et al., 2011], where we consider a planar quadcopter which can make contact with sliding friction on a surface. When both edges of the quadcopter are touching the constraint, we assume some latching mechanism engages and fully constrains the quadcopter in place with no way to release. This problem explores planning with an underactuated system, friction, constraint surfaces, nonlinear dynamics, nonlinear guards, and nonlinear resets.

The configurations of the system are the vertical, horizontal, and angular position $q = [y, z, \theta]^T$ and the inputs are the left and right thrusters, u_1 and u_2. The dynamics are defined by (5.33) with the following

$$M(q) := \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \quad C(q, \dot{q}) := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{5.37}$$

$$N(q, \dot{q}) := \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix}, \quad \Upsilon := \begin{bmatrix} -\sin(\theta)(u_1 + u_2) \\ \cos(\theta)(u_1 + u_2) \\ \frac{1}{2}(u_2 w - u_1 w) \end{bmatrix} \tag{5.38}$$

where $w = 0.25$ is the width and $I = 1$ is the inertia of the quadcopter.

To add more complex geometry, the constrained surface is a circle centered about the origin with radius 5. Since the edges of the quadcopter make contact with the surface, the left and right edges of the quadcopter are located at,

$$[y_L, z_L]^T = [y - \frac{1}{2} w \cos\theta, z - \frac{1}{2} w \sin\theta]^T \tag{5.39}$$

$$[y_R, z_R]^T = [y + \frac{1}{2} w \cos\theta, z + \frac{1}{2} w \sin\theta]^T \tag{5.40}$$

The constraints are then $a_1 = 25 - y_L^2 - z_L^2$ and $a_2 = 25 - y_R^2 - z_R^2$. Frictional force λ_t is defined to be tangential to the constraint with magnitude proportional to the constraint force λ_n, $\lambda_t = \mu \lambda_n$,

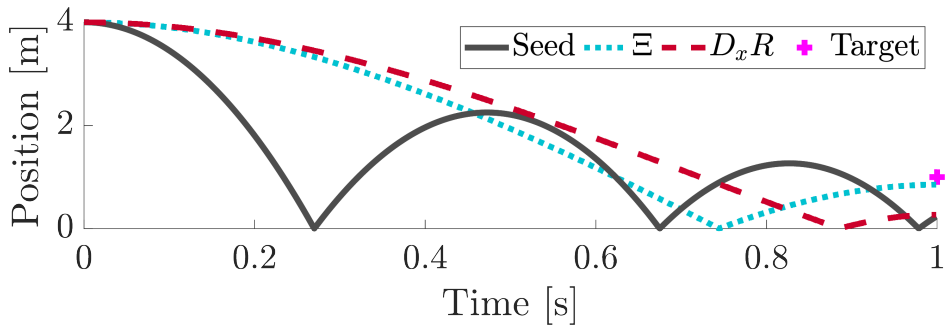

Figure 5.2: Bouncing ball with elastic impact where 1 bounce is optimal and 3 bounces are seeded. The target end position is shown in (magenta plus). Both gradient update methods were able to pull away the unnecessary bounces, but the method using D_xR did not converge or get to the target state.

where μ is the coefficient of friction.

The example is setup to have the quadcopter start some distance away from the constraint with a horizontal velocity, $x_0 = [2, 2.5, -\pi/8, 4, 0, 0]^T$, end up oriented with the constraint with no velocity $x_{des} = [5\cos(-\pi/12), 5\cos(-\pi/12), -7/12\pi, 0, 0, 0]^T$, timesteps $\Delta = 0.002$, with penalties $R = 0.01_{2\times2}$, and $Q_N = [1000I_{3\times3}, 0_{3\times3}; 0_{3\times3}, 0.1I_{3\times3}]$. The position portion is weighted more heavily than velocity because the goal is to get close enough to the desired location to perch. For the seed, a combined thrust of equal to $1.5mg$ was applied constantly and if both edges made contact with the constraint, the thrust force was dropped to $0.1mg$. This initial input resulted in a trajectory which makes contact with the right edge and then shortly after makes double contact with the constraint as shown in Fig. 5.1.

5.5 Results

In this section, the results of the experiments on each system are presented. Overall, the Jacobian of the reset map method D_xR-iLQR has trouble converging and has worse cost compared to our proposed algorithm Ξ-iLQR which uses the saltation matrix.

5.5.1 Bouncing Ball with Elastic Impacts

The outcomes of the experiment comparing D_xR-iLQR to Ξ-iLQR are shown in Table 5.1. An example run is shown in Fig. 5.2. D_xR-iLQR did not converge ($|\delta J| > 0.05$) on any example if a hybrid transition was maintained, while Ξ-iLQR converged on every example. The only cases where D_xR-iLQR converged were when the algorithm removed all of the bounces – which becomes equivalent to smooth iLQR. Ξ-iLQR has lower cost compared to D_xR-iLQR for every example except for when the problem is seeded with no bounces (they obtain the same smooth solution) and when no bounces was the optimal solution but the problem was seeded with a single bounce. In this case, Ξ-iLQR did converge to a different local minima[1], which is not surprising as it is not a global optimization.

The value of the hybrid extension was tested on the three bounce optimal three bounce seeded case. Without the hybrid extension, the optimizer did not converge and did significantly worse than D_xR-iLQR. This highlights the importance of the hybrid trajectory extensions: even though the backwards pass is correct, having mode mismatches will lead to unfavorable convergence and trajectory quality.

Overall, Ξ-iLQR produced locally optimal solutions for each variation and was able to remove unnecessary bounces in some cases, though it never added any. This result is expected because there is no gradient information on the backwards pass being provided to give knowledge about adding additional bounces. Furthermore, as discussed above, there may not be an appropriate controller available when a novel hybrid mode is encountered.

5.5.2 Ball dropping on a spring-damper

For this experiment, Ξ-iLQR and D_xR-iLQR came up with similar solutions where the cost of Ξ-iLQR $J = 13.21$ is slightly lower than D_xR-iLQR $J = 13.29$. This difference is highlighted in Fig. 5.3 where D_xR-iLQR was not able to smooth out the spikes near mode changes. This is also

[1]This solution was confirmed as a local minima under a single bounce by comparing it against a trajectory produced using direct collocation [Kelly, 2017] constrained to a single bounce, as shown in Table. 5.1.

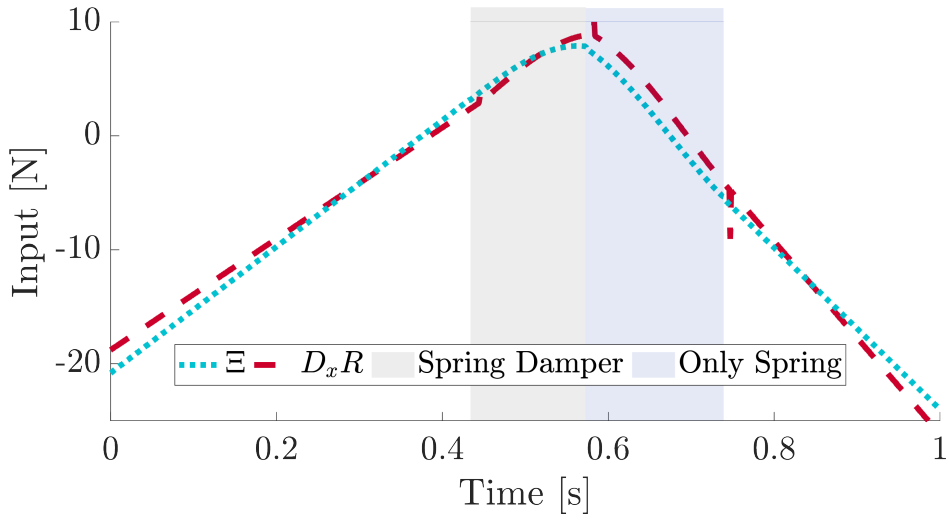

Figure 5.3: Bouncing ball on a spring-damper ground where both gradient update methods found similar trajectories but using the Jacobian of the reset map D_xR lead to not being able to fully converge as evident by the residual spikes near hybrid transitions.

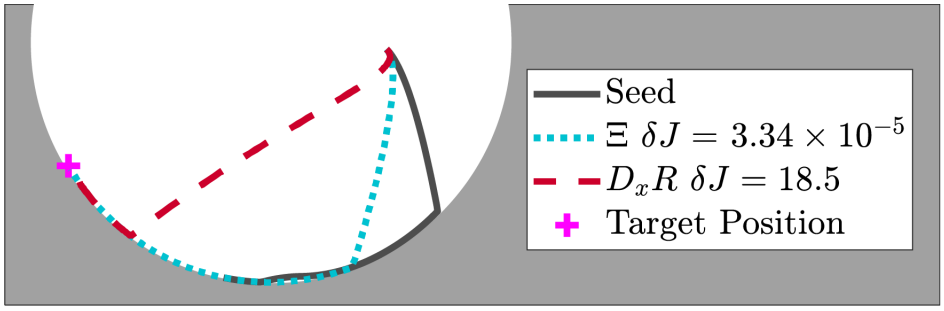

Figure 5.4: Ball drop on a curved surface with plastic impacts where both gradient methods produced trajectories that got to the end goal, but using D_xR did not converge and had a significantly higher cost.

reflected in D_xR-iLQR having a higher expected cost reduction as well $\delta J = 0.001$ where Ξ-iLQR is a magnitude lower $\delta J = 0.00017$. This difference in convergence can most likely be attributed to D_xR providing gradient information that does not adjust the input pre-impact accordingly to allow for adjustments on the spikes post-impact without destructively changing the resulting end state.

5.5.3 Ball drop on a curved surface with plastic impacts

The trajectory produced by Ξ-iLQR has a cost of $J = 10.7$ and D_xR-iLQR a cost of $J = 50.5$. The generated position trajectories along with the initial seeded trajectory are shown in Fig. 5.4 where

both methods ended up at the goal state but D_xR-iLQR converged significantly less than Ξ-iLQR.

In this example, we purposely seeded a sub-optimal trajectory which releases the contact for small duration and returns back to the constraint to evaluate if the algorithms would modify the contact sequence. Ξ-iLQR ended up removing this erroneous contact change and whereas D_xR-iLQR ended up not going back to the constraint surface and ended in the unconstrained mode. We speculate that because D_xR has the wrong gradient information about contacts, it ended up staying in the unconstrained mode for a longer duration and ultimately could not converge.

5.5.4 Perching quadcopter

In this example, the final position trajectories are shown in Fig. 5.1 where Ξ-iLQR converged $\delta J = 0.170$ with a cost of $J = 4.76$ whereas D_xR-iLQR did not converge $\delta J = 3 \times 10^5$ and produced an erratic solution with very high cost of $J = 2.66 \times 10^3$.

Ξ-iLQR seemed to make the natural extension of the seed and followed the constraint until the target position was achieved, but removed the double constrained mode at the end. We postulate that the fully constrained mode was removed in order to better fine tune the final position because position error is weighted significantly more than velocity. However, the true optimal solution should include the fully constrained mode to eliminate any velocity for free.

5.6 Discussion

In this work, we extended iLQR to hybrid dynamical systems with piecewise smooth solutions. We compared our algorithm (Ξ-iLQR) against using the incorrect hybrid backwards pass update (D_xR-iLQR) over a variety of hybrid systems. For each example, Ξ-iLQR outperformed D_xR-iLQR in terms of cost and convergence when there was a hybrid transition in the final trajectory. This result is expected because the saltation matrix is the correct linearization about a hybrid transition.

We believe that our algorithm excels at refining a trajectory which has an initial hybrid mode sequence that needs the timing to be refined. This is similar to other shooting methods, where they

are sensitive to initialization. However, this issue of locality is accentuated in our algorithm by only giving gradient information and control reference for transitions it has seen.

An interesting phenomenon occurs in hybrid systems where the controllability between different hybrid modes can vary significantly. For example, in a jumping robot, there is not much control authority in the air than compared to against on the ground. This poses an issue, as the system may diverge or otherwise be extremely sensitive. In future work, we want to investigate this issue of varying controllabilty through different hybrid modes as well as introducing systems with intersecting hybrid guard where the Bouligand derivative [Burden et al., 2016; Scholtes, 2012] will play an analogous role as the saltation matrix does in this work.

Chapter 6

Hybrid iLQR MPC

The content in this chapter has been submitted to T-RO [Kong et al., 2022a].

6.1 Abstract

Model Predictive Control (MPC) is a popular strategy for controlling robotic systems but is difficult for systems with contact due to the complex nature of hybrid dynamics. To implement MPC for systems with contact, dynamic models are often simplified, or contact sequences are fixed in time, in order to plan trajectories efficiently. In this work, we extend Hybrid iterative Linear Quadratic Regulator to work in a MPC fashion (HiLQR MPC) by modifying how the cost function is computed when contact modes don't align, utilizing parallelizations when simulating rigid body dynamics, and using efficient analytical derivative computations of the rigid body dynamics. The result is a system that can modify the contact sequence of the reference behavior and plan whole body motions cohesively – which is crucial when dealing with large perturbations. HiLQR MPC is tested on two systems: first, the hybrid cost modification is validated on a simple actuated bouncing ball hybrid system. Then HiLQR MPC is compared against methods that utilize centroidal dynamic assumptions on a quadruped robot (Unitree A1). HiLQR MPC outperforms the centroidal methods in both simulation and hardware tests.

6.2 Introduction

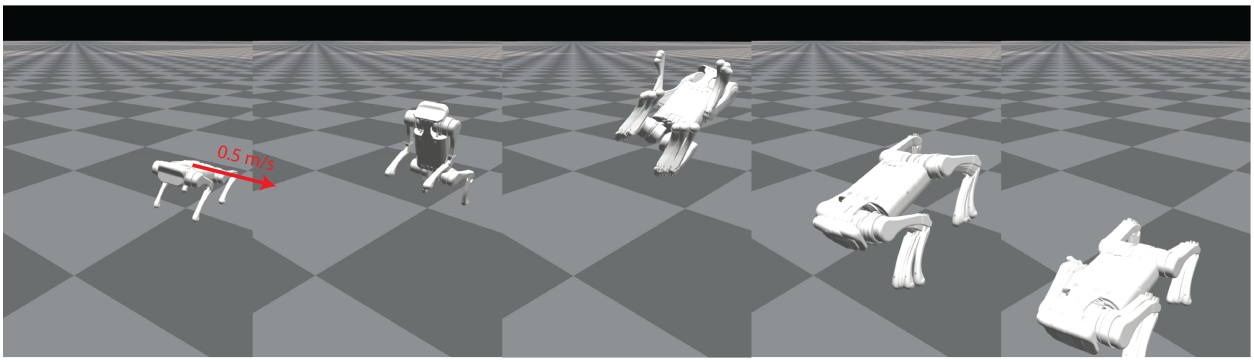

Figure 6.1: HiLQR MPC forward pass for tracking a backflip with an initial 0.5 m/s lateral perturbation on the body. 9 robot models are used on the forward pass to solve the line search in parallel.

In order for robots to reliably move and interact within our unstructured world, they need to be able to replan motions to handle unexpected perturbations or changes in the environment. However, replanning is difficult for robotic systems that have changing contact with the world because of the complexity of the discontinuous dynamics and combinatoric issues that arise.

There are many methods for planning contact-rich behaviors offline [Posa et al., 2014; Mordatch et al., 2012; Mombaur, 2009; Diehl et al., 2006], but these methods generally suffer from poor time complexity and cannot be used directly in real-time applications. Direct methods for contact implicit trajectory optimization [Posa et al., 2014; Mordatch et al., 2012] simultaneously solve for the states, inputs, and contact forces of an optimal trajectory while encoding the contact conditions through complementarity constraints – which are notoriously difficult and slow to solve. A relaxation of contact implicit trajectory optimization is to fix the contact sequence for each timestep [Von Stryk, 1999; Posa et al., 2016; Kelly, 2017; Pardo et al., 2017; Winkler et al., 2018].

Other relaxations have been made for the planning problem to achieve real-time planning for Model Predictive Control (MPC). Centroidal motion planning methods [Di Carlo et al., 2018; Kim et al., 2019; Da et al., 2020; Xie et al., 2021; Gehring et al., 2013] have had a lot of success in planning gaits in real-time by making large simplifications on the robot dynamics and also assuming a fixed contact sequence a priori. Swing legs are often controlled separately using Raibert heuristics

[Raibert, 1986] and capture point methods [Pratt et al., 2006] to regulate body velocities. However, simplifications to the robot dynamics can lead to the controller being less robust to perturbations which require reasoning about the full dynamics, such as nonlinear changes in lever arm for leg extension, varying inertia when the leg changes shape, or not accounting for unexpected changes in contact.

Shooting methods which utilize Differential Dynamic Programming (DDP) [Mayne, 1966] or iterative Linear Quadratic Regulator (iLQR) [Li and Todorov, 2004; Tassa et al., 2012] are good candidates for model predictive control because they are fast, can utilize the full nonlinear dynamics, and solutions are always dynamically feasible. Methods that utilize the full nonlinear dynamics [Li and Wensing, 2020; Mastalli et al., 2020] generally come at the cost of enforcing a fixed contact sequence. [Li et al., 2021] utilizes the full nonlinear dynamics for timesteps closer to the current horizon and then uses simplified dynamics for timesteps later in the future, but also uses a fixed contact sequence. Similar to the fixed contact sequence issue of [Di Carlo et al., 2018], it is less robust due to constraining the solution to maintain the original contact sequence in scenarios where it would be much better to change them.

To allow efficient updates of the contact sequence, [Le Cleac'h et al., 2021] speeds up contact implicit trajectory optimization through strategic linearizations about a target trajectory and focuses on the tracking problem, which allows the possibility of running in real time and can easily change the contact timing and sequence to stabilize a trajectory. However, the basin of attraction may be smaller because it is linearized about a single nominal trajectory. If the robot needs to drastically change the trajectory, the controller will not use a good model given the linearization of the target trajectory.

In this work, we make use of Hybrid iLQR (HiLQR) [Kong et al., 2021a], a full-order contact implicit trajectory optimization approach, in order to create a receding horizon MPC that utilizes nonlinear dynamics and is not constrained to the reference trajectory's gait sequence. By using Hybrid iLQR as an MPC, the contact sequence can be greatly modified when stabilizing large perturbations, e.g. as shown in Fig. 6.1. We show that HiLQR MPC can reject bigger disturbances

than centroidal methods when perturbed along a walking trajectory. We also show that HiLQR MPC working on a real robot in real-time can reject disturbances more reliably than centroidal methods.

6.3 Hybrid systems background

In this section, we two different hybrid simulation techniques are reviewed for rigid body systems with unilateral constraints.

6.3.1 Hybrid Simulators

There are 2 main hybrid simulation techniques for rigid bodies with unilateral constraints – event-driven and timestepping. HiLQR MPC uses a hybrid simulator and can use either method. But different modifications need to be made depending on which simulation type is used. Its important to have a high level understanding of each of these simulation types to understand that modifications discussed in this work.

Event-driven hybrid simulators [Wehage and Haug, 1982; Pfeiffer and Glocker, 1996; Brogliato et al., 2002] follow very closely to the example shown in the definition of hybrid dynamical systems Def. 1. Event-driven simulations are convenient because the dynamics have a well defined structure and contacts are persistently maintained. However, event-driven simulations have problems with behaviors like Zeno, where an infinite number of hybrid transitions are made in a finite amount of time, as they must stop integration and apply a reset map for each individual event.

Time-stepping [Stewart and Trinkle, 1996; Anitescu and Potra, 1997; Brogliato et al., 2002] schemes circumvent issues like Zeno by integrating impulses over small timesteps at a time and are numerically efficient, especially for systems with large numbers of constraints. These methods allow contact constraints to be added or removed at any time step, but only once per time step. Furthermore, no distinction is made between continuous contact forces and discontinuous impulses. However, they are limited to first-order (Euler) integration of the dynamics.

6.4 HiLQR MPC Implementation

In this section, the tracking problem is defined, and we show how to adapt Hybrid iLQR to be a model predictive controller.

6.4.1 Hybrid Cost Update

The goal is now to minimize the difference in state and input with respect to a reference state and input

$$J(x_i, u_i) = (x_i - \hat{x}_i)^T Q_i (x_i - \hat{x}_i) + (u_i - \hat{u}_i)^T R_i (x_i - \hat{u}_i) \tag{6.1}$$

where Q_i is the quadratic penalty matrix on state, and R_i is the quadratic penalty matrix on input, and $(\hat{x}, \hat{u})$ denotes the reference.

However, because Hybrid iLQR is contact implicit (the hybrid mode sequence can differ from the target's mode sequence), the runtime cost (6.1) can be ill defined when the candidate trajectory's mode does not match the target's. For example, if there is an early or late contact in a rigid body system with unilateral constraints, the velocities will be heavily penalized for having a mismatched timing. This issue is further propagated to the backward pass, where the gradient information relies on these differences and can ultimately lead to the algorithm not converging. To mitigate these mode mismatch issues, we propose 2 different solutions for event-driven and timestepping simulations.

For event-driven hybrid simulators, the same hybrid extensions used in reference tracking on the forward pass in Hybrid iLQR can be used when comparing error during mode mismatches. Suppose a hybrid transition occurs at time t. The reference state at pre-transition $\hat{x}(t^-)$ is extended beyond the hybrid guard by flowing the pre-transition dynamics forwards while holding the pre-transition input constant. The post-transition reference state $\hat{x}(t^+)$ is extended backward by flowing the dynamics backward in time while again holding the input constant. With these hybrid extensions, when there is a mode mismatch induced by a transition timing error, the reference is switched to the extension

with the same hybrid mode.

In timestepping simulations, the effect of the hybrid transition is applied over several timesteps rather than instantaneously as in event-driven hybrid simulations. For example, when a contact is made, the penetrating velocities do not immediately go to zero and actually take several timesteps to go to zero. During these timesteps, the hybrid mode is not well defined. Because of this, the hybrid extension method does not work due to the timesteps that are "in between" hybrid modes. Instead, we propose to use a different approach for legged robots, where the constraint forces λ_j are used to scale the penalty on input from R_{min} to R_{max}

$$w_j = \frac{\lambda_j}{\sum_i \lambda_i} \tag{6.2}$$

$$R_j = R_{max} - w_j(R_{max} - R_{min}) \tag{6.3}$$

where j corresponds to the leg index. This modification penalizes changes in input less when a leg applies more ground reaction force and penalizes changes in input more when the leg applies less force to the ground. This is intuitive because when a leg is not supporting much weight, we want that leg to have lower gains because it has less control authority on the robot body.

6.4.2 Rollout and Forward Pass

Depending on the hybrid system, HiLQR MPC uses either an event-driven or timestepping simulation for its rollouts and forward passes. In this work, we demonstrate the cost mismatch update for an event-driven simulation on a bouncing ball. However, when multiple contacts are involved, as in the case for a quadruped robot, simulating an event-driven system is significantly more difficult than using an out-of-the-box timestepping rigid body dynamics simulator. Many rigid body contact simulators utilize timestepping simulation methods. In this work, we use "Isaac Gym" (a high performance GPU-based physics simulation) [Makoviychuk et al., 2021], because the simulator has a unique feature where it can simulate multiple robots at once in an optimized fashion. We utilize parallel computations to parallelize the linesearch in the forward pass. An example line-

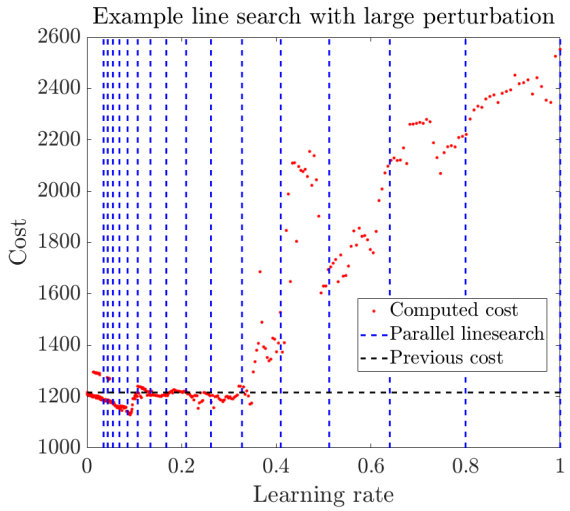

Figure 6.2: Linesearch for the first HiLQR MPC forward pass iteration after applying a $1.5\frac{m}{s}$ lateral perturbation while walking as shown in Fig 6.9. If computed sequentially, the linesearch would terminate after 12 steps.

search is shown in Fig. 6.2, which shows the cost for different learning rates. Note that the cost is discontinuous with respect to the learning rate because the line search explores different contact sequences. In order for cost to be reduced in this case, the linesearch needs to take 12 steps if done sequentially. Due to the efficiency of parallel computations on the forward passes, parallelizing is on average twice as fast as computing the linesearch sequentially when comparing the computation times for the solutions in Fig. 6.9.

Several key implementation features consist of precomputing the gain schedule for the reference trajectory, reusing the valid portions of previous solutions, and always seeding the reference trajectory as one of the parallel solves in the linesearch.

Lastly, quaternion differences [Jackson et al., 2021] are used instead of Euler angles when computing the orientation cost and linear feedback. This change allows for better convergence properties, as well as allowing for tracking more dynamic behaviors like the backflip in Fig. 6.1.

6.4.3 Backward Pass

The main challenge for the backward pass is how to compute the derivatives of the dynamics. For simple hybrid systems like the bouncing ball, the derivatives of the dynamics and saltation

matrix are trivial to find and compute [Kong et al., 2021a]. However, computing the derivatives for the full order rigid body dynamics with unilateral constraints is not trivial – if done naively, the computations are incredibly slow. This is the same for the saltation matrix because it relies on computing the derivative of the impact map. In this work, we utilize a rigid body dynamics library called Pinocchio [Carpentier et al., 2015–2021] (which computes these derivatives in an optimized fashion) for all full order contact rigid body dynamics derivatives.

For the backward pass, HiLQR MPC assumes the trajectory is produced by an event-driven simulation. If the timesteps are small enough, then approximating a timestepping simulation as an event-driven simulation on the backward pass is reasonable. Another approximation HiLQR MPC makes is that when simultaneous contacts are made during a timestep (i.e., 2 feet making contact at the same time), the contact sequencing is assumed to always follow the same contact order and to have happened at the end of the timestep. The chosen order is in increasing order of the indexing of the limbs. These approximations are validated through experimentation, where HiLQR MPC is still able to converge with these approximations in the presence of perturbations.

6.4.4 General Robot Implementation

For all robot experiments using HiLQR MPC, a 50 timestep MPC horizon is used with timesteps of 0.01 seconds. When running HiLQR MPC in simulation, the algorithm is able to pause the simulation in order to compute a new trajectory. Once a trajectory is generated, the first input of the planned trajectory is used as the control input for that timestep. Allowing HiLQR MPC to pause the simulation ensures that we can analyze how well the controller can perform independent of the computation time available. We also run the controller in real-time, because on hardware the dynamics cannot be paused.

To run HiLQR MPC in real-time for the physical robot implementation, several changes are made and hyper parameters are tuned to speed up the algorithm at the cost of performance. The first change is to run a hierarchy of controllers, as shown in Fig. 6.3, where a fast low level Hybrid LQR controller is run asynchronously from the trajectory generator (HiLQR MPC). HiLQR MPC

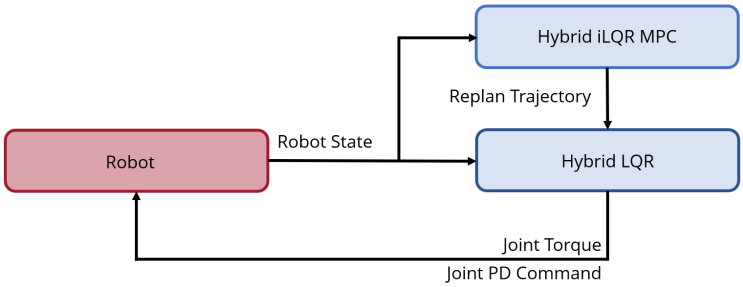

Figure 6.3: Hierarchy of controllers where HiLQR MPC is replanning trajectories as fast as possible while Hybrid LQR is tracking the most recent trajectory that was sent by HiLQR MPC.

runs separately as fast as possible and always using the latest robot state. When solving for a new trajectory, sub-optimal trajectories are sent out at each forward pass iteration in order to send the low level controller the most recent trajectory modifications. If the current solve exceeds the maximum allotted time, the current solve is terminated and a new solve is started for the most recent robot state information. Several hyper-parameters are modified to reduce computation time, from reducing the number of robots running in parallel in the rollouts and forward passes to relaxing the optimality condition. Lastly, joint PD terms from the gain matrix are sent directly to the motor controller, which runs at 10KHz rather than computing the feedback at the Hybrid LQR level.

6.5 Experiments

In this section, the experimental setups for HiLQR MPC are presented, with results given in Section 6.6. To validate the event-driven mode mismatch cost update, we first compare using the proposed update with not using any hybrid cost updates on a simple actuated bouncing ball hybrid system. Then, to show how this approach can scale up to a real system, simulated and physical robot experiments are carried out on a quadrupedal robot (Unitree A1) to compare HiLQR MPC with methods that use centroidal simplifications and Raibert heuristics for swing leg control: "Convex MPC" [Di Carlo et al., 2018] and "Instant QP" [Da et al., 2020; Xie et al., 2021; Gehring et al., 2013]. Convex MPC returns ground reaction forces for the feet that are in contact with the ground and are subjected to friction constraints for a set horizon length. The dynamic model is a linearized

floating base model and the optimization is formulated as a quadratic program. Instant QP solves the same problem, but for a single timestep. Because only one timestep is solved, Instant QP can update the solver with the actual contact condition of the feet and can provide more stability with respect to contact mismatches, but lacks the robustness that is gained from looking ahead.

6.5.1 Bouncing Ball

In this experiment, the same 1D bouncing ball hybrid system from [Kong et al., 2021a] is used. The states of the system $x = [z, \dot{z}]^T$ are the vertical position z and the velocity $\dot{z}$ and the input u is a force applied directly to the ball. The two hybrid modes, 1 and 2, are defined when the ball has a negative velocity $\dot{z} < 0$ and when the ball has a non-negative velocity $\dot{z} \geq 0$, respectively. The dynamics on each mode is ballistic dynamics plus the input

$$F_1(x, u) = F_2(x, u) := \left[\dot{z}, \frac{u}{m} - g\right]^T \tag{6.4}$$

Hybrid mode 1 transitions to 2 when the ball hits the ground, $g_{(1,2)}(x) := z$, and mode 2 transitions to 1 at the apex $g_{(2,1)}(x) := \dot{z}$. When mode 1 transitions to 2, an elastic impact is applied, $R_{(1,2)}(x) = [z, -e\dot{z}]^T$ where e is the coefficient of restitution. The reset map from 2 to 1 is identity. The event-driven simulation is implemented with MATLAB ODE 45 [Shampine et al., 2003] with event detection.

To validate that updating the cost on mode mismatches improves convergence for HiLQR MPC, we first generate a reference trajectory using Hybrid iLQR to create an optimal single bounce trajectory. HiLQR MPC is used to stabilize an initial large perturbation and is run with and without the hybrid cost update for event-driven simulations. For both cases, HiLQR MPC is applied at every timestep. At each timestep, convergence is recorded where convergence is determined by the expected reduction (5.19). For this test, the convergence cut-off is set to be $\delta J < 1e^{-4}$. It is expected that, by utilizing the mode extensions, convergence will improve because the algorithm will not spend unnecessary computation and effort in flipping the velocity of the ball if there is a

mismatch in impact timing, rather it will wait for when the impact applies the flip.

6.5.2 Simulated Robot Controller Comparison

To demonstrate the robustness of cohesively planning whole body motions and allowing contact schedules to change, we compare HiLQR MPC to Convex MPC and Instant QP by applying perturbations to A1 while implementing a walking gait in simulation. To make the comparison fair, the walking gait that HiLQR MPC is tracking is the same one generated from Convex MPC in the absence of perturbations. Similar gait parameters are chosen for Instant QP to produce a similar gait. All controllers are run at each timestep and use the first control input of the new trajectory as the control input for that timestep.

The walking gait starts from a standing pose and then attempts to reach a desired forward velocity of $0.2\frac{m}{s}$. Lateral velocity perturbations are applied to the robot's body at two different magnitudes and eight different times along the gait cycle: four when each foot is in swing when getting up to speed and the other four when the gait is in steady state. The number of times the robot falls and the maximum perturbed lateral position are recorded for each push.

It is expected that HiLQR MPC should be able to recover from a wider variety of perturbations and have less deviation when the perturbations are large when compared against the centroidal methods because it can utilize the nonlinear contact dynamics of the swing and stance legs cohesively.

6.5.3 Physical Robot Controller Comparison

The bulk of the analysis for comparing the controllers is done in simulation because the perturbations can be consistently applied in both cases with a variety of different perturbations. To reliably apply the same perturbation on hardware, we opt for a consistent motor command block for a short period of time while the robot is walking. Once the motor commands are unblocked, the controller must react to the robot falling over, catch itself, and then continue walking.

In this experiment, we compare HiLQR MPC against Instant QP, where both controllers are

able to handle the perturbation in simulation but come up with different solutions. HiLQR MPC tends to replan a stand trajectory after it realizes that it is falling to catch itself, while Instant QP tries to continue the walking gait and recirculates the legs in order to catch itself. The perturbation is applied shortly after walking has started, and the torque commands are blocked for 0.15 seconds. The experiment is run 5 times for each controller and failure is determined by if the robot's body hits the floor and if the controller is able to continue walking after the perturbation. For state estimation, we use the Kalman filter from [Bledt et al., 2018]. Because HiLQR MPC creates a new plan to track in order to handle the perturbation, it is expected to outperform Instant QP which is trying its best to continue walking.

6.6 Results

In this section, we review the results for each experiment. Overall, utilizing the cost mismatch updates is crucial for obtaining good solutions, and HiLQR MPC can withstand large perturbations by modifying the contact sequence in an optimal manner.

6.6.1 Bouncing ball HiLQR MPC

The task for the bouncing ball experiment, detailed in Sec. 6.5.1, is to track a predefined trajectory using HiLQR MPC for a fully actuated bouncing ball. The target trajectory is 1 second long, where the ball starts at 4 meters above the ground with no velocity and ends at 2.5 meters above the ground with no velocity. We compare using the event-driven hybrid cost update (Sec. 6.4.1) to not using this update, and the results of this experiment are shown in Fig. 6.4.

As expected, both methods converge and track well before the impact event is within the horizon of the HiLQR MPC. The approaches differ once the hybrid event is within the horizon, as can be seen by the high control effort and unnatural kink in state space that is produced when not using the cost update. Furthermore, of the 1001 time steps, 8 did not converge when the cost update was not used. Although the number of unconverged timesteps is small, the quality of the trajectory suffered

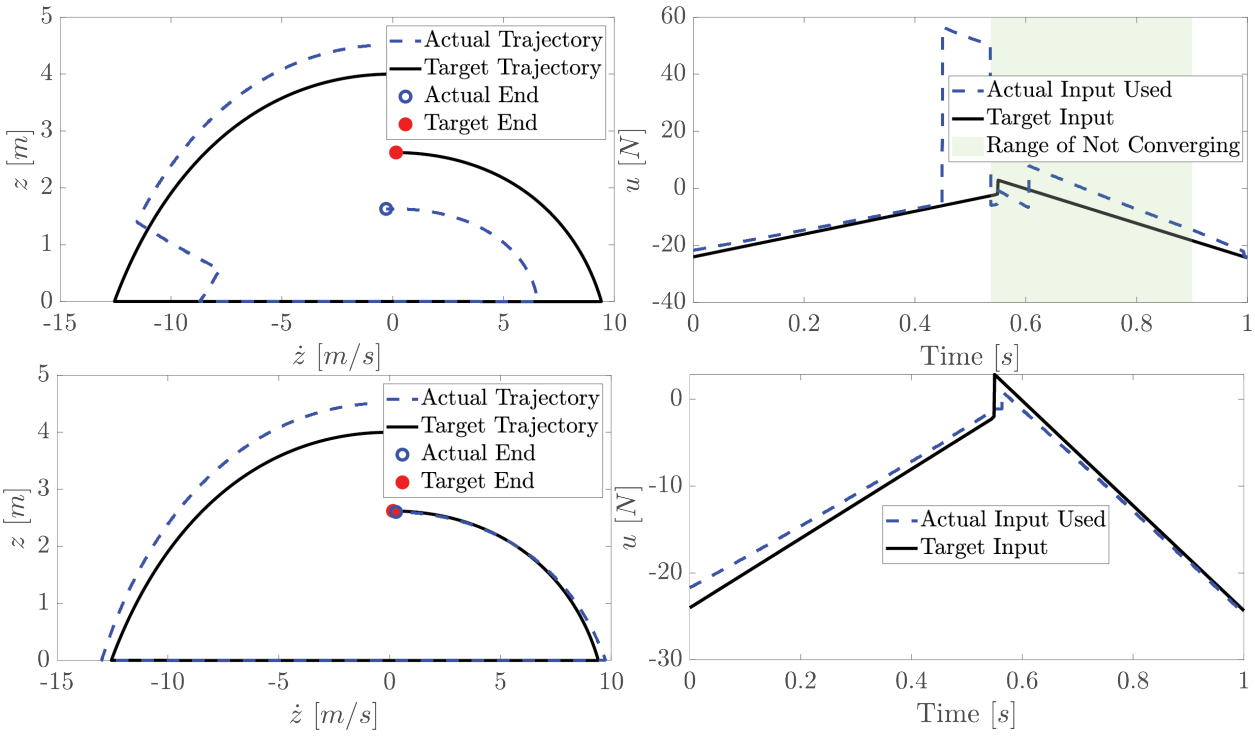

Figure 6.4: Comparing HiLQR MPC not using the event-driven hybrid cost update (top row) and using the event-driven hybrid cost update (bottom row) where the state space trajectory tracking is shown in (left column) and input usage in time series is shown in (right column). HiLQR solutions are shown in (blue dashed) and the target trajectory is shown in (black solid). The end of trajectories are denoted with (circle). When not using the event-driven hybrid cost update the trajectory tracking suffered, as evident by the high input effort and sharp deviations in trajectory that attempt to track the post-impact velocity before the impact occurs. Several solutions did not converge as shown in with (green highlight). Whereas, using the event-driven hybrid cost update led to altogether better convergence and tracking.

greatly, as shown in Fig. 6.4, top row. This is because without updating the cost to account for hybrid mode mismatches, the gradient information biases the solution towards flipping the velocity before impact.

Using the cost update for hybrid mode mismatches, HiLQR MPC can correctly utilize the impact to reduce tracking error, as shown in Fig. 6.4, bottom row. The cost update allows HiLQR MPC to create plans that are closer to the target trajectory by shifting contact times rather than making large modifications to the input to match the contact schedule, which results in significantly better convergence. In addition to having better tracking performance, when using trajectory optimization for MPC, it is desirable to always converge and to not make drastic changes from the

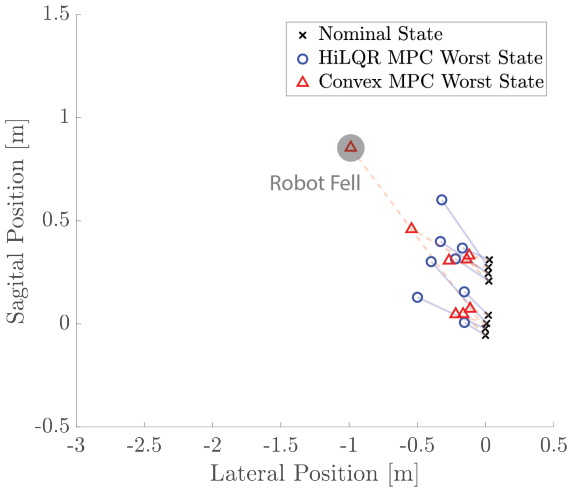

Figure 6.5: Medium perturbation (1.0 m/s lateral perturbation). Plots the nominal trajectory and worst case error in lateral position for both controllers.

planned trajectory unless necessary.

6.6.2 Simulated Robot Controller Comparison

The robustness of HiLQR MPC is compared with Convex MPC and Instant QP for a walking trajectory at eight different perturbations in simulation as discussed in Sec. 6.5.2. The results are summarized in Table 6.1, farthest perturbed position is visualized for each experiment in Figs. 6.5 and 6.6, change in contact sequence in Figs. 6.7 and 6.8, and the resuling behavior shown in Fig 6.9.

As expected, deviations from the smaller perturbation lead to similar results and high success for all controllers. This is most likely because the perturbations do not require the controller to heavily modify the trajectory while stabilizing less stable robot states, such as in the case of the larger perturbations. In the medium and large perturbation experiments, HiLQR MPC had a higher success rate of 100% and 88% compared to the centroidal methods – Convex MPC 88% and 50% and Instant QP 50% and 25%. Failure for the controllers tended to occur when a right leg was in swing (both front right and back). This failure mode is most likely due to the lateral perturbation being applied in the left direction causing the stabilizing maneuvers to be more complicated and less stable. Because HiLQR MPC is able to plan the body and swing legs more cohesively, it can handle these complex maneuvers better than the centroidal methods, where the stance and swing

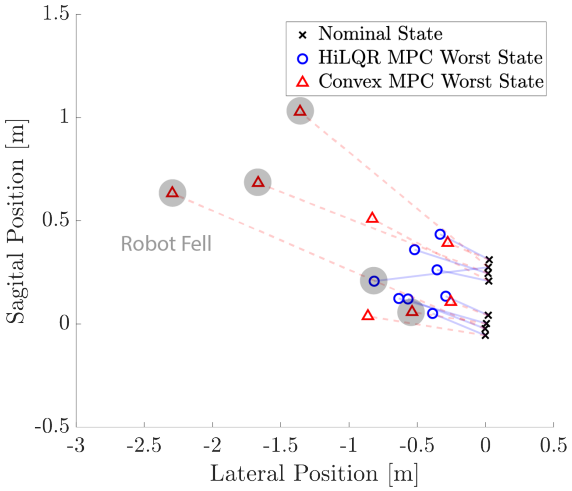

Figure 6.6: Large perturbation (1.5 m/s lateral perturbation). Plots the nominal trajectory and worst case error in lateral position for both controllers.

Table 6.1: Lateral perturbation success rates for a medium perturbation $1.0m/s$, a large perturbation $1.5m/s$, and the average max deviation for the large perturbation over 8 trials.

Controller	$1.0m/s$ Succ. [%]	$1.5m/s$ Succ. [%]	Avg. Dev. $[m]$
HiLQR MPC	100%	88%	$0.512m$
Convex MPC	88%	50%	$1.032m$
Instant QP	50%	25%	$3.729m$

legs are planned separately. This difference is mostly highlighted when the perturbations are larger. Since Instant QP performed worse than Convex MPC, further comparisons are made only between HiLQR MPC and Convex MPC.

In the large perturbation experiments, HiLQR MPC deviated half as much as Convex MPC when comparing max lateral deviations in body position, as shown in Table 6.1. An example trial (large perturbation during the first step) is shown in Fig. 6.9, where HiLQR MPC used less steps to stabilize the perturbation, which ultimately led to the body deviating less than half of the deviation from Convex MPC. The contact sequence for the reference and the initial solution after applying the perturbation are shown in Figures 6.7 and 6.8. Note that HiLQR MPC is solving for new trajectories that modify the contact sequence in order to better stabilize the behavior rather than adhering to the original plan's contact sequence. This is crucial because HiLQR MPC can add or remove contacts to help catch itself, as well as optimize the new contact locations.

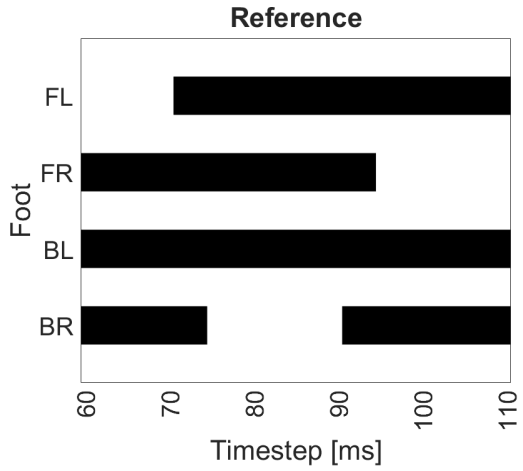

Figure 6.7: Hildebrand diagram for the nominal walking gait where black means the foot is in contact.

Overall, HiLQR MPC performed similarly to or better than Convex MPC when stabilizing perturbations along a walking trajectory. When the perturbations are large, HiLQR MPC outperforms Convex MPC because it is able to replan a new contact sequence to stabilize about and it can fully utilize the nonlinear dynamics for the more aggressive maneuvers.

6.6.3 Physical Robot Controller Comparison

The results of the motor-blocking physical robot experiment from Sec. 6.5.3 – where the motor commands were blocked for 150 milliseconds shortly after the walk started and HiLQR is compared to Instant QP over five trials – are shown in Fig. 6.10 and Table 6.2. Over five trials, HiLQR MPC was able to stabilize successfully after the motor block was released every time, while Instant QP was completely unstable 60% of the time and 40% of the time was able to stand up and walk after the body hit the ground. Two unintended additional perturbations occurred in this experiment – there is a consistent 10 millisecond input delay on A1 and another perturbation caused by the state estimator. The estimator relies on the kinematic information from the legs that are in contact to get a better estimate of the robot body. However, when the motor commands were blocked, all the contact forces went close to zero, which resulted in a degraded estimate of the robot body until the legs made sufficient contact with the ground again.

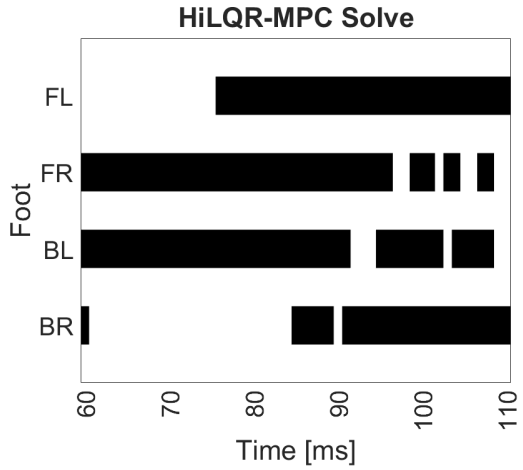

Figure 6.8: Hildebrand diagram for a single solve of HiLQR MPC rejecting the large perturbation at 60 ms as shown in the top of Fig. 6.9 where black means the foot is in contact. See that HiLQR MPC is removing and adding back contacts when advantageous to help stabilize the behavior.

HiLQR MPC was able to replan a stand trajectory in order to catch itself rather than sticking to the original plan of walking as Instant QP. In the times that Instant QP successfully rejected the perturbation, the robot body actually hits the ground and the legs that are planned to be in stance apply enough standing force to get back up while the back right leg recirculates in order to counteract the backward velocity induced by getting back up. Since this relies on the body hitting the ground correctly and the back leg perfectly stabilizing the motion, it is a lot less reliable but is able to catch itself occasionally.

Similarly to the simulated experiments, HiLQR MPC outperforms the centroidal method (Instant QP) because HiLQR MPC does not have to adhere to a rigid gait schedule and can fluidly replan a new contact sequence to help stabilize the perturbation. Although Instant QP utilizes the current contact information to inform which legs are in contact, the controller is trying its best to follow the scheduled gait sequence. In this case, modifying the gait sequence from a walk to a stand is much more reliable. HiLQR MPC is able to track the walking gait when appropriate but modify it to a stand if needed to catch the robot and seamlessly return to walking once the perturbation has been stabilized. Having the ability to automatically modify the gait schedule to generate these stabilizing behaviors is important for a controller because the initial plan might not always be the best in the presence of disturbances.

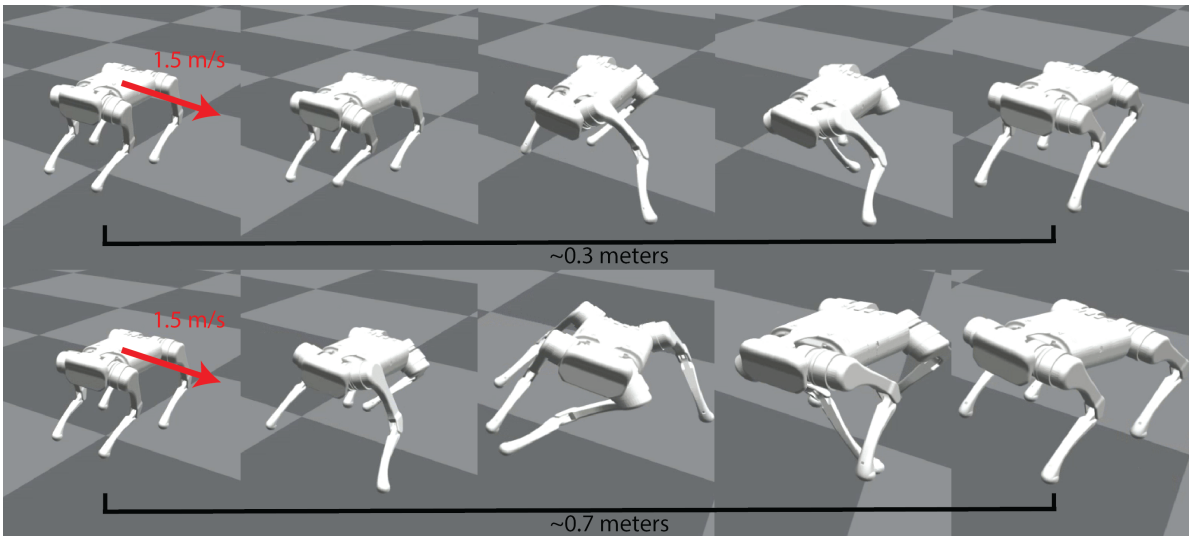

Figure 6.9: Applying 1.5 m/s lateral perturbation during the first step of the walking gait. (Top row) shows HiLQR MPC recovering from the perturbation in one step and accruing a lateral deviation of 0.3 meters while (bottom row) shows Convex MPC taking several steps to handle the perturbation and is perturbed 0.7 meters away from the nominal.

Table 6.2: Motor blocking perturbation results over 5 trials.

Controller	Success	Hit Ground	Uncontrolled
HiLQR MPC	100%	0%	0%
Instant QP	0%	40%	60%

6.7 Discussion

Allowing for varying contact sequences while planning for the full nonlinear dynamics of a robotic system is very difficult, but leads to more robust control. In this work, we extend Hybrid iLQR to work as a model predictive controller, which can vary the contact sequence of the target trajectory as well as plan with the nonlinear dynamics. This extension is made possible by fixing gradient issues that occur when there are hybrid mode mismatches, using fast analytical derivatives of the contact dynamics, and parallelizing the line search in the forward pass.

In simulation, HiLQR MPC outperforms the state of the art centroidal motion planning technique (Convex MPC) for stabilizing perturbations 88% success rate vs 50% for large perturbations and 100% vs 88% for medium perturbations. This is because HiLQR MPC is able to fully utilize the

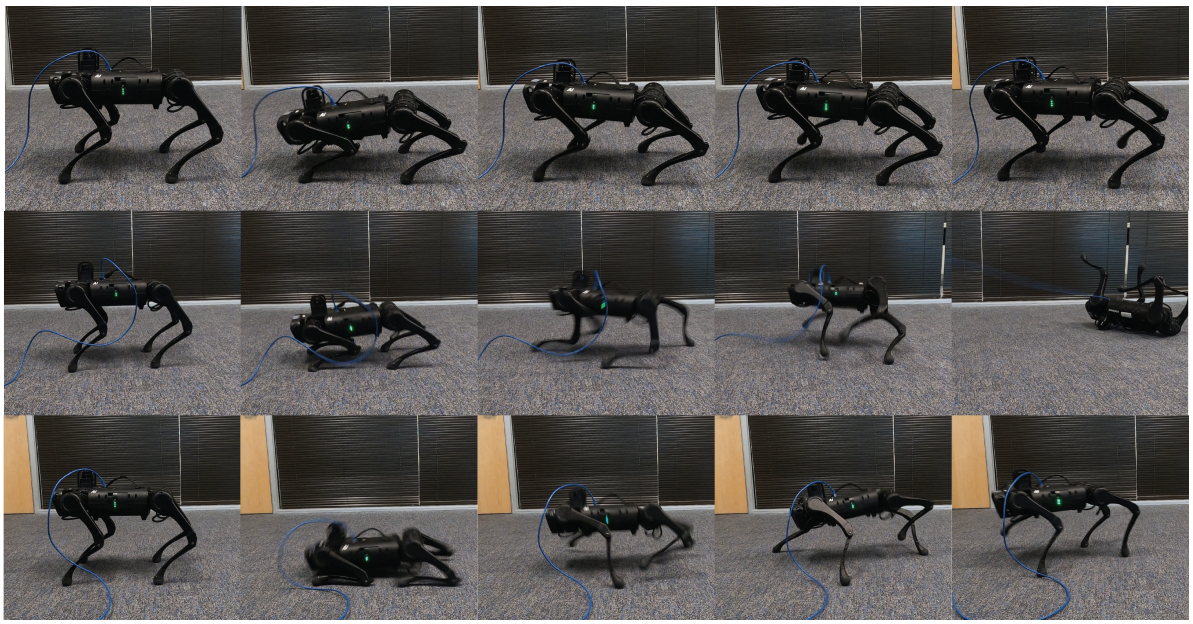

Figure 6.10: Turning off motor commands for 150 ms during the first step. HiLQR MPC (top row) creates a catching behavior and then goes back into the scheduled walk. Instant QP sometimes tries to step to regulate velocity which destabilizes the robot (middle row). Other times, Instant QP hits the ground (bottom row), which stabilizes the body velocities and the robot is able to shoot its legs out in order to get back into the walk.

legs of the robot to help catch itself and can create more efficient and elegant solutions, needing fewer steps to recover.

HiLQR MPC is also able to run in real-time with some modifications to the hyperparameters and utilizing a hierarchical control structure where trajectories are sent to a lower level Hybrid LQR controller to track the hybrid trajectories planned by HiLQR MPC. We are able to show for a motor blocking perturbation that HiLQR MPC is able to withstand this better than Instant QP, where HiLQR MPC succeeded for all trials and Instant QP could only stabilize 40% of the time (and even then only after the robot body hit the ground). The high success rate for HiLQR MPC is due to planning a reliable catching behavior, while Instant QP is continuously attempting to walk as best it can.

The code is currently implemented in Python while using several C++ libraries that utilize Python wrappers. Improvements in the real-time application will be seen by further optimizing the code and implementing it in C++. Overall, HiLQR MPC is a very modular model predictive

controller which can be run for any hybrid dynamical system of type Def. 1. Besides the hybrid dynamical systems definition, there are no restrictive simplifications that are made, which makes the controller generalizable to many different behaviors. Future work will add additional constraints through Augmented Lagrangian [Howell et al., 2019] for obstacle avoidance and actuator constraints.

Chapter 7

Conclusion

This work addresses the issue of state estimation, planning, and control for legged robots. In Chapter 2, we present a trajectory optimization framework to create optimally convergent trajectories that utilize the natural dynamics of a system to funnel trajectories to the nominal. We demonstrate the effectiveness of this framework on both an undersensed GPS denied hill navigation system and an underactuated rotary cart pole system.

Because of the hybrid nature of legged robots, this thesis frequently utilizes the saltation matrix to extend traditionally smooth methods to hybrid. In Chapter 5 we provide a tutorial for using the saltation matrix.

In Chapter 4, we present the Salted Kalman Filter, which has performance comparable to that of a hybrid particle filter, but requires only a fraction of the computation time. For planning in hybrid systems, we present hybrid iLQR in Chapter 5, which plans optimal trajectories through hybrid events, varies contact timings/placements, and provides a jump linear feedback law (the gains can jump at hybrid transitions to match the discontinuous nature of hybrid dynamics).

Reliable control for hybrid systems is complex because large perturbations will disrupt local strategies such as utilizing the jump linear feedback law given by Hybrid iLQR. Large perturbations can also lead to contact mismatches when tracking a reference trajectory and can quickly become unstable. In Chapter 6, we address these issues by creating a model predictive controller which can

replan trajectories in order to handle large perturbations as well as contact mismatches. Replanning also allows for the controller to select a new contact sequence if it is beneficial for tracking the behavior as a whole if it leads to better stabilization to the reference. We compare Hybrid iLQR MPC against popular methods which use heavy linearizations and simplifications as well as a rigid gait schedule. Hybrid iLQR MPC plans new strategies in order to handle a variety of perturbations more robustly than the popular methods.

7.1 Possible Future directions for convergent planning

We showed that planning convergent trajectories increases reliability in both undersensed and underactuated systems in Chapter 2. A possible future direction is to simultaneously design a controller that can converge the directions in which the dynamics naturally diverge. Another direction includes reasoning about uncertainty in modeling parameters and dynamics. This can possibly be done by extending contraction analysis with the ultimate boundedness analysis. However, contraction analysis is already very conservative, and adding additional ultimate boundedness would give an even more conservative bound (which might not be too helpful). A more useful direction may be to create a new divergence metric that captures the average divergence due to parameter uncertainty.

In this work, convergent planning was only considered for smooth underactuated and undersensed systems, but it would be beneficial to design convergent dynamic legged behaviors by extending convergent planning to hybrid systems. Work has been done to extend contraction analysis to hybrid systems [Burden et al., 2018b] by using the saltation matrix, but its even harder to find contraction regions on hybrid systems than on smooth systems. This will lead to even poorer convergence if we use pure contraction. However, the expected divergence trick does not seem to work well because the directions and magnitude of convergence can discontinuously change on hybrid transitions, whereas for smooth systems the divergence smoothly changes. In the future, we are looking at using Hybrid iLQR to create convergent plans for hybrid systems [Zhu and Johnson,

2022].

7.2 Possible Future directions for state estimation

We validated the performance of the SKF by comparing it with a hybrid particle filter and found that the estimation of the SKF suffered when assuming the entire distribution transitions in a single time step in Chapter 4. The performance of the filter can possibly be improved by utilizing multiple SKF's in parallel to better capture the split distributions. Similarly to an interacting multiple model, a bank of SKF filters can be run simultaneously, and the filters with the most likely residual with respect to the posterior covariance are weighted the most.

Another direction of improvement that we have already addressed is to add uncertainty in the guard location, where we augment the saltation matrix with additional terms to account for the uncertainty in the guard [Payne et al., 2022a]. We are also currently addressing the uncertainty in contact timing by solving the moving horizon estimation problem, which is the dual problem to Hybrid iLQR MPC [Payne et al., 2022b]. This will allow us to find the contact timing that optimally represents the data.

In addition to moving horizon estimation, it is also possible to utilize the guard information to determine the likelihood that a transition will be made during the next timestep. For contact systems, the probability of transition can also be calculated using contact sensors or using a generalized-momentum-based disturbance observer [De Luca et al., 2006]. Utilizing the probability of transition will make the filter more robust to missing or accidentally making a contact.

Lastly, we have not run hybrid state estimation on a real system yet, but before that, the probability of transition should be integrated into the algorithm.

7.3 Future directions for Hybrid iLQR

Hybrid iLQR can solve a wide variety of problems because it works for a broad definition of hybrid dynamical systems, as shown in Chapter 5. In Chapter 6, this broad definition includes a

generalizable robot model which allows us to utilize Hybrid iLQR as a model predictive controller to stabilize a quadruped robot while solving the contact implicit optimization problem efficiently.

In this work, we created a foundation from which to easily extend. The first example is planning hybrid convergent trajectories by biasing the trajectory towards a convergent reference point [Zhu and Johnson, 2022]. Separately, any smooth methods that have been applied to iLQR can be extended to hybrid methods, such as using the Augmented Lagrangian to add constraints to optimization [Howell et al., 2019].

For Hybrid iLQR MPC, an interesting next step would be to incorporate terrain estimation data into the rollout and forward pass simulation. By having a good estimate of the terrain and contact geometry, we can efficiently replan if the environment changes or is different than expected. Overall, for a more robust implementation of Hybrid iLQR MPC, the code should be further optimized and implemented in C++ to decrease any effects on delay.

Bibliography

A. Abusorrah, K. Mandal, D. Giaouris, A. El Aroudi, M. M. Al-Hindawi, Y. Al-Turki, and S. Banerjee. Avoiding instabilities in power electronic systems: Toward an on-chip implementation. *IET Power Electronics*, 10(13):1778–1787, 2017.

A. Ageno and A. Sinopoli. Lyapunov's Exponents for Non-Smooth Dynamical Systems: Behavior Identification Across Stability Boundaries With Bifurcations. In *Volume 6: 5th International Conference on Multibody Systems, Nonlinear Dynamics, and Control, Parts A, B, and C*, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pages 1999–2010, 09 2005. doi: 10.1115/DETC2005-84574.

M. Aizerman and F. Gantmakher. On the stability of periodic motions. *Journal of Applied Mathematics and Mechanics*, 22(6):1065–1078, 1958. ISSN 0021-8928. doi: https://doi.org/10.1016/0021-8928(58)90033-9.

M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

R. Arkin. Behavior-based robot navigation for extended domains. *Adaptive Behavior*, 1(2):201–225, 1992.

H. Asahara and T. Kousaka. Stability analysis using monodromy matrix for impacting systems. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 101(6):904–914, 2018.

A. Back, J. M. Guckenheimer, and M. Myers. A dynamical simulation facility for hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1993.

A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 76–89. Springer, 2002.

A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. Sangiovanni-Vincentelli. The design of dynamical observers for hybrid systems: Theory and application to an automotive control problem. *Automatica*, 49(4):915–925, 2013.

S. Banerjee, J. Ing, E. Pavlovskaia, M. Wiercigroch, and R. K. Reddy. Invisible grazings and dangerous bifurcations in impacting systems: The problem of narrow-band chaos. *Phys. Rev. E*, 79:037201, Mar 2009. doi: 10.1103/PhysRevE.79.037201.

S. Banerjee, D. Giaouris, O. Imrayed, P. Missailidis, B. Zahawi, and V. Pickert. Nonsmooth dynamics of electrical systems. In *IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2709–2712, 2011. doi: 10.1109/ISCAS.2011.5938164.

N. Barhoumi, F. Msahli, M. Djemaï, and K. Busawon. Observer design for some classes of uniformly observable nonlinear hybrid systems. *Nonlinear Analysis: Hybrid Systems*, 6(4): 917–929, 2012.

M. Bernardo, C. Budd, A. R. Champneys, and P. Kowalczyk. *Piecewise-smooth dynamical systems: Theory and applications*, volume 163. Springer Science & Business Media, 2008.

D. Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2), 1998.

J. B. Biemond, N. van de Wouw, W. M. H. Heemels, and H. Nijmeijer. Tracking control for hybrid systems with state-triggered jumps. *IEEE Transactions on Automatic Control*, 58(4):876–890, 2012.

M. Biggio, F. Bizzarri, A. Brambilla, G. Carlini, and M. Storace. Reliable and efficient phase noise simulation of mixed-mode integer-N phase-locked loops. In *European Conference on Circuit Theory and Design*, pages 1–4. IEEE, 2013.

M. Biggio, F. Bizzarri, A. Brambilla, and M. Storace. Accurate and efficient PSD computation in mixed-signal circuits: A time-domain approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):905–909, 2014.

F. Bizzarri, A. Brambilla, and G. S. Gajani. Steady state computation and noise analysis of analog mixed signal circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(3): 541–554, 2011a.

F. Bizzarri, A. Brambilla, S. Perticaroli, and G. S. Gajani. Noise in a phase-quadrature pulsed energy restore oscillator. In *20th European Conference on Circuit Theory and Design (ECCTD)*, pages 465–468. IEEE, 2011b.

F. Bizzarri, A. Brambilla, and G. S. Gajani. Periodic small signal analysis of a wide class of type-ii phase locked loops through an exhaustive variational model. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(10):2221–2231, 2012.

F. Bizzarri, A. Brambilla, and G. Storti Gajani. Extension of the variational equation to analog/digital circuits: Numerical and experimental validation. *International Journal of Circuit Theory and Applications*, 41(7):743–752, 2013a.

F. Bizzarri, A. Brambilla, and G. Storti Gajani. Lyapunov exponents computation for hybrid neurons. *Journal of Computational Neuroscience*, 35(2):201–212, 2013b.

F. Bizzarri, A. Brambilla, G. S. Gajani, and S. Banerjee. Simulation of real world circuits: Extending conventional analysis methods to circuits described by heterogeneous languages. *IEEE Circuits and Systems Magazine*, 14(4):51–70, 2014.

F. Bizzarri, A. Colombo, F. Dercole, and G. S. Gajani. Necessary and sufficient conditions for the noninvertibility of fundamental solution matrices of a discontinuous system. *SIAM Journal on Applied Dynamical Systems*, 15(1):84–105, 2016. doi: 10.1137/140959031.

M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros, and M. Hutter. Offline motion libraries and online mpc for advanced mobility skills. *The International Journal of Robotics Research*, page 02783649221102473, 2022.

G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *IEEE International Conference on Robotics and Automation*, pages 4399–4406, 2018.

M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and IMU. In *Robotics: Science and Systems*, pages 17–24, 2012.

H. A. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, 1988.

H. A. Blom and E. A. Bloem. Particle filtering for stochastic hybrid systems. In *IEEE Conference on Decision and Control*, volume 3, pages 3221–3226, 2004.

S. F. Bockman. Lyapunov exponents for systems described by differential equations with discontinuous right-hand sides. In *American Control Conference*, pages 1673–1678. IEEE, 1991.

B. Brogliato, A. Ten Dam, L. Paoli, F. Ge´ not, and M. Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *Appl. Mech. Rev.*, 55(2):107–150, 2002.

S. A. Burden, S. S. Sastry, D. E. Koditschek, and S. Revzen. Event–selected vector field discontinuities yield piecewise–differentiable flows. *SIAM Journal on Applied Dynamical Systems*, 15 (2):1227–1267, 2016.

S. A. Burden, T. Libby, and S. D. Coogan. On contraction analysis for hybrid systems. *CoRR*, abs/1811.03956, 2018a.

S. A. Burden, T. Libby, and S. D. Coogan. On contraction analysis for hybrid systems, 2018b. arXiv:1811.03956.

J. Carpentier, F. Valenza, N. Mansard, et al. Pinocchio: fast forward and inverse dynamics for poly-articulated systems. https://stack-of-tasks.github.io/pinocchio, 2015–2021.

K. Chakrabarty and U. Kar. Control of bifurcation of PWM controlled DC drives. In *IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, pages 1–8, 2012.

K. Chakrabarty and U. Kar. Dynamic behavior of PWM controlled DC drive. *Scientific Voyage*, 1 (1):1–12, 2020.

R. Chawla, A. Rounak, and V. Pakrashi. Stability analysis of hybrid systems with higher order transverse discontinuity mapping. *arXiv preprint arXiv:2203.13222*, 2022.

D.-h. Chen, S.-x. Xie, X.-c. Huang, and Y.-m. Chen. Calculating Floquet multipliers for periodic solution of non-smooth dynamical system. In *International Conference on Applied Mathematics, Modeling, Simulation, and Optimization (AMMSO 2019)*, pages 27–33, 2019.

S. Coombes, Y. M. Lai, M. Şayli, and R. Thul. Networks of piecewise linear neural mass models. *European Journal of Applied Mathematics*, 29(5):869–890, 2018.

J. Cortés, V. Šviković, P. Alou, J. A. Oliver, and J. A. Cobos. Design and analysis of ripple-based controllers for buck converters based on discrete modeling and Floquet theory. In *IEEE 14th*

Workshop on Control and Modeling for Power Electronics (COMPEL), pages 1–9, 2013. doi: 10.1109/COMPEL.2013.6626474.

G. Council, S. Yang, and S. Revzen. Deadbeat control with (almost) no sensing in a hybrid model of legged locomotion. In *International Conference on Advanced Mechatronic Systems*, pages 475–480, 2014.

X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion. In *Conference on Robot Learning*, pages 883–894, 16–18 Nov 2020.

I. Daho. On the co-occurrence of slaw-scale bifurcation in period-doubled orbits in a high order system. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 2, 2012.

I. Daho, D. Giaouris, B. Zahawi, V. Picker, and S. Banerjee. Stability analysis and bifurcation control of hysteresis current controlled ćuk converter using Filippov's method. In *4th IET Conference on Power Electronics, Machines and Drives*, pages 381–385, 2008.

A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630. IEEE, 2006.

M. Di Bernardo, C. J. Budd, A. R. Champneys, P. Kowalczyk, A. B. Nordmark, G. O. Tost, and P. T. Piiroinen. Bifurcations in nonsmooth dynamical systems. *SIAM Review*, 50(4):629–701, 2008.

J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.

L. Dieci and C. Elia. Master stability function for piecewise smooth networks, 2021.

M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

W. J. Dixon and A. M. Mood. The statistical sign test. *Journal of the American Statistical Association*, 41(236):557–566, 1946.

A. El Aroudi, D. Giaouris, H. H.-C. Iu, and I. A. Hiskens. A review on stability analysis methods for switching mode power converters. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(3):302–315, 2015. doi: 10.1109/JETCAS.2015.2462013.

A. El Aroudi, M. S. Al-Numay, W. G. Lu, J. M. Bosque-Moncusí, and H. H.-C. Iu. A combined analytical-numerical methodology for predicting subharmonic oscillation in H-bridge inverters under double edge modulation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(7):2341–2351, 2017.

A. El Aroudi, L. Benadero, E. Ponce, C. Olalla, F. Torres, and L. Martinez-Salamero. Nonlinear dynamic modeling and analysis of self-oscillating H-bridge parallel resonant converter under zero current switching control: Unveiling coexistence of attractors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(4):1657–1667, 2018.

A. El Aroudi, K. Mandal, M. S. Al-Numay, D. Giaouris, and S. Banerjee. Piecewise quadratic slope compensation technique for DC-DC switching converters. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(12):5574–5585, 2020.

A. Elbkosh, D. Giaouris, V. Pickert, B. Zahawi, and S. Banerjee. Stability analysis and control of bifurcations of parallel connected DC/DC converters using the monodromy matrix. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 556–559, 2008a.

A. Elbkosh, D. Giaouris, B. Zahawi, V. Pickert, and S. Banerjee. Control of bifurcation of DC/DC buck converters controlled by double-edged PWM waveform. In *Proc. ENOC*, pages 1–5. Citeseer, 2008b.

W. Y. Eras-Herrera, A. R. Mesquita, and B. O. Teixeira. Equality-constrained state estimation for hybrid systems. *IET Control Theory & Applications*, 13(13):2018–2028, 2019.

M. Fečkan and M. Pospíšil. On the bifurcation of periodic orbits in discontinuous systems. *Communications in Mathematical Analysis*, 8(1):87–108, 2010.

G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Transactions on Automatic Control*, 47(10):1663–1676, 2002.

A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Springer, 1988.

F. Forni, A. R. Teel, and L. Zaccarian. Follow the bouncing ball: Global results on tracking and state estimation with impacts. *IEEE Transactions on Automatic Control*, 58(6):1470–1485, 2013.

C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart. Control of dynamic gaits for a quadrupedal robot. In *IEEE international conference on Robotics and automation*, pages 3287–3292, 2013.

D. Giaouris, A. Elbkosh, S. Banerjee, B. Zahawi, and V. Pickert. Control of switching circuits using complete-cycle solution matrices. In *IEEE International Conference on Industrial Technology*, pages 1960–1965, 2006.

D. Giaouris, S. Banerjee, B. Zahawi, and V. Pickert. Stability analysis of the continuous-conduction-mode buck converter via Filippov's method. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(4):1084–1096, 2008. doi: 10.1109/TCSI.2008.916443.

D. Giaouris, S. Maity, S. Banerjee, V. Pickert, and B. Zahawi. Application of Filippov method for the analysis of subharmonic instability in DC–DC converters. *International Journal of Circuit Theory and Applications*, 37(8):899–919, 2009.

D. Giaouris, S. Banerjee, O. Imrayed, K. Mandal, B. Zahawi, and V. Pickert. Complex interaction between tori and onset of three-frequency quasi-periodicity in a current mode controlled boost converter. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(1):207–214, 2011.

D. Giaouris, C. Yfoulis, S. Voutetakis, and S. Papadopoulou. Stability analysis of digital state feedback controlled boost converters. In *IECON 39th Annual Conference of the IEEE Industrial Electronics Society*, pages 8391–8396, 2013.

T. L. Gibo, L. N. Verner, D. D. Yuh, and A. M. Okamura. Design considerations and human-machine performance of moving virtual fixtures. In *IEEE International Conference on Robotics and Automation*, pages 671–676. IEEE, 2009.

G. Gkizas. Border collisions in interleaved multi-output DC-DC boost converters. In *International Symposium on Nonlinear Theory and Its Applications*, 08 2018.

R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39 (4):402–430, 2020. doi: 10.1177/0278364919894385. URL https://doi.org/10.1177/0278364919894385.

I. A. Hiskens and M. A. Pai. Trajectory sensitivity analysis of hybrid systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(2):204–220, 2000.

T. A. Howell, B. E. Jackson, and Z. Manchester. Altro: A fast solver for constrained trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7674–7679, 2019.

I. Hwang, H. Balakrishnan, and C. Tomlin. State estimation for hybrid systems: applications to aircraft tracking. *IET Proceedings-Control Theory and Applications*, 153(5):556–566, 2006.

N. Hyafil and F. Bacchus. Conformant probabilistic planning via csps. In *ICAPS*, volume 98, pages 205–214, 2003.

B. D. Ilhan, A. M. Johnson, and D. E. Koditschek. Autonomous legged hill ascent. *Journal of Field Robotics*, 35(5):802–832, August 2018.

O. M. Imrayed. *Analysis and control of nonlinear characteristics in DC/DC converters*. PhD thesis, University of Newcastle Upon Tyne, 2012.

A. P. Ivanov. The stability of periodic solutions of discontinuous systems that intersect several surfaces of discontinuity. *Journal of Applied Mathematics and Mechanics*, 62(5):677–685, 1998. doi: 10.1016/S0021-8928(98)00087-2.

A. P. Ivanov. Stability of periodic motions with impacts. In *Impacts in Mechanical Systems*, pages 145–187. Springer, 2000.

B. E. Jackson, K. Tracy, and Z. Manchester. Planning with attitude. *IEEE Robotics and Automation Letters*, 6(3):5658–5664, 2021. doi: 10.1109/LRA.2021.3052431.

M. R. Jeffrey. Dynamics at a switching intersection: Hierarchy, isonomy, and multiple sliding. *SIAM Journal on Applied Dynamical Systems*, 13(3):1082–1105, 2014.

H. Jiang, A. S. Chong, Y. Ueda, and M. Wiercigroch. Grazing-induced bifurcations in impact oscillators with elastic and rigid constraints. *International Journal of Mechanical Sciences*, 127: 204–214, 2017.

A. M. Johnson. Impacts. In *Advanced Robot Dynamics*. Unpublished Notes, 2021. URL `https://www.andrew.cmu.edu/user/amj1/book/`.

A. M. Johnson and D. E. Koditschek. Legged self-manipulation. *IEEE Access*, 1:310–334, 2013.

A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek. Autonomous legged hill and stairwell ascent. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 134–142. IEEE, 2011.

A. M. Johnson, S. A. Burden, and D. E. Koditschek. A hybrid systems model for simple manipulation and self-manipulation systems. *The International Journal of Robotics Research*, 35(11), 2016a.

A. M. Johnson, J. E. King, and S. Srinivasa. Convergent planning. *IEEE Robotics and Automation Letters*, 1(2):1044–1051, 2016b.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *arXiv preprint arXiv:1005.0416*, 2010.

M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

I. Ko, B. Kim, and F. C. Park. Vf-rrt: Introducing optimization into randomized motion planning. In *2013 9th Asian Control Conference (ASCC)*, pages 1–5. IEEE, 2013.

N. Kong and A. M. Johnson. Optimally convergent trajectories for navigation. In *International Symposium on Robotics Research*, October 2019.

N. Kong, C. Li, and A. M. Johnson. Hybrid iLQR model predictive control for contact implicit stabilization on legged robots. *arXiv:2207.04591 [cs.RO]*, 2022a.

N. J. Kong, G. Council, and A. M. Johnson. iLQR for piecewise-smooth hybrid dynamical systems. In *IEEE Conference on Decision and Control*, pages 5374–5381, December 2021a.

N. J. Kong, J. J. Payne, G. Council, and A. M. Johnson. The salted kalman filter: Kalman filtering on hybrid dynamical systems. *Automatica*, 131:109752, 2021b. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2021.109752. URL https://www.sciencedirect.com/science/article/pii/S0005109821002727.

N. J. Kong, J. J. Payne, G. Council, and A. M. Johnson. The Salted Kalman Filter: Kalman filtering on hybrid dynamical systems. *Automatica*, 131:109752, 2021c.

N. J. Kong, J. J. Payne, J. Zhu, S. A. Burden, and A. M. Johnson. Saltation matrices: The essential tool for linearizing hybrid systems, 2022b. In prep.

X. Koutsoukos, J. Kurien, and F. Zhao. Monitoring and diagnosis of hybrid systems using particle filtering methods. In *International Symposium on Mathematical Theory of Networks and Systems*, 2002.

M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa. Robust trajectory selection for rearrangement planning as a multi-armed bandit problem. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2678–2685. IEEE, 2015a.

M. C. Koval, N. S. Pollard, and S. S. Srinivasa. Pose estimation for planar contact manipulation with manifold particle filters. *The International Journal of Robotics Research*, 34(7):922–945, 2015b. doi: 10.1177/0278364915571007. URL https://doi.org/10.1177/0278364915571007.

P. Kowalczyk and P. Glendinning. Micro-chaos in relay feedback systems with bang-bang control and digital sampling. *IFAC Proceedings Volumes*, 44(1):13305–13310, 2011.

O. Kuznyetsov. Calculation of stable and unstable periodic orbits in a chopper-fed DC drive. *Mathematical Modeling and Computing*, 8(1):43–57, 2021.

Y. M. Lai, R. Thul, and S. Coombes. Analysis of networks where discontinuities and nonsmooth dynamics collide: Understanding synchrony. *The European Physical Journal Special Topics*, 227(10):1251–1265, 2018.

G. Lantoine and R. P. Russell. A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory. *Journal of Optimization Theory and Applications*, 154(2):382–417, 2012.

S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

S. Le Cleac'h, T. A. Howell, M. Schwager, and Z. Manchester. Fast contact-implicit model-predictive control. *arXiv preprint arXiv:2107.0561*, 2021.

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

R. Leine and H. Nijmeijer. *Dynamics and bifurcations of non-smooth mechanical systems*. Springer, 2004.

R. Leine and D. Van Campen. Discontinuous bifurcations of periodic solutions. *Mathematical and Computer Modelling*, 36(3):259–273, 2002.

R. Leine and D. Van Campen. Bifurcation phenomena in non-smooth dynamical systems. *European Journal of Mechanics-A/Solids*, 25(4):595–616, 2006.

R. I. Leine and H. Nijmeijer. *Dynamics and bifurcations of non-smooth mechanical systems*, volume 18. Springer Science & Business Media, 2013.

R. I. Leine and D. H. van Campen. Fold bifurcations in discontinuous systems. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 19777, pages 1423–1429. American Society of Mechanical Engineers, 1999.

H. Li and P. M. Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020.

H. Li, R. J. Frei, and P. M. Wensing. Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, 6(2):3373–3380, 2021.

W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics*, pages 222–229, 2004.

Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.

K. Liu, Y. Zhang, A. Dobson, and D. Berenson. Asymptotically near-optimal methods for kinodynamic planning with initial state uncertainty. *IEEE Robotics and Automation Letters*, 2019.

W. Lohmiller and J.-J. E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34 (6):683–696, 1998.

I. Lopez, J. Busturia, and H. Nijmeijer. Energy dissipation of a friction damper. *Journal of Sound and Vibration*, 278(3):539–561, 2004.

T. Lozano-Perez, M. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3, 03 1984. doi: 10.1177/027836498400300101.

A. Lussier Desbiens, A. T. Asbeck, and M. R. Cutkosky. Landing, perching and taking off from vertical surfaces. *The International Journal of Robotics Research*, 30(3):355–370, 2011.

J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

S. Maity and P. K. Sahu. Modeling and analysis of a fast and robust module-integrated analog photovoltaic MPP tracker. *IEEE Transactions on Power Electronics*, 31(1):280–291, 2015.

S. Maity, D. Giaouris, S. Banerjee, T. K. Bhattacharya, B. Zahawi, and V. Pickert. Control of bifurcations in power electronic DC-DC converters through manipulation of the saltation matrix. In *Proc. PhysCon*, pages 1–5, 2007.

V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac Gym: High performance GPU based physics simulation for robot learning. In *Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

R. Mallik, A. M. Pace, S. A. Burden, and B. Johnson. Accurate small–signal discrete–time model of dual active bridge using saltation matrices. In *IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 6312–6317, 2020.

K. Mandal. *Dynamical Analysis of Resonant DC-DC Converters*. PhD thesis, IIT Kharagpur, 2013.

K. Mandal and S. Banerjee. A new software for dynamical analysis of nonsmooth systems. In *Proc. ENOC*, 09 2014.

K. Mandal, S. Banerjee, and C. Chakraborty. A new algorithm for small-signal analysis of DC–DC converters. *IEEE Transactions on Industrial Informatics*, 10(1):628–636, 2013.

K. Mandal, A. Abusorrah, M. M. Al-Hindawi, Y. Al-Turki, A. E. Aroudi, D. Giaouris, and S. Banerjee. Control-oriented design guidelines to extend the stability margin of switching converters. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017. doi: 10.1109/ISCAS.2017.8050578.

S. Mason, N. Rotella, S. Schaal, and L. Righetti. Balancing and walking using full dynamics lqr control with contact constraints. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 63–68, 2016. doi: 10.1109/HUMANOIDS.2016.7803255.

C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *IEEE International Conference on Robotics and Automation*, pages 2536–2542, 2020.

MATLAB. Matlab, 2018. The MathWorks, Natick, MA, USA.

D. Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

D. Q. Mayne. Differential dynamic programming–a unified approach to the optimization of dynamic systems. In *Control and Dynamic Systems*, volume 10, pages 179–254. Elsevier, 1973.

T. McGeer. Passive dynamic walking. *Int. J. Robotics Res.*, 9(2):62–82, 1990.

K. Mombaur. Using optimization to create self-stable human-like running. *Robotica*, 27(3): 321–330, 2009.

I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):43, 2012.

C. Morel, D. Petreus, and A. Rusu. Application of the Filippov method for the stability analysis of a photovoltaic system. *Advances in Electrical and Computer Engineeging*, 11:93–98, 2011.

C. Morel, A. Akrad, R. Sehab, T. Azib, and C. Larouci. Open-circuit fault-tolerant strategy for interleaved boost converters via Filippov method. *Energies*, 15(1):352, 2022.

P. C. Müller. Calculation of Lyapunov exponents for dynamic systems with discontinuities. *Chaos, Solitons & Fractals*, 5(9):1671–1681, 1995.

J.-G. Muñoz, A. Pérez, and F. Angulo. Enhancing the stability of the switched systems using the saltation matrix. *International Journal of Structural Stability and Dynamics*, 19(05):1941004, 2019.

J.-G. Muñoz, F. Angulo, and D. Angulo-Garcia. Designing a hysteresis band in a boost flyback converter. *Mechanical Systems and Signal Processing*, 147:107080, 2021.

R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

R. Nicks, L. Chambon, and S. Coombes. Clusters in nonsmooth oscillator networks. *Physical Review E*, 97(3):032213, 2018.

S. Nobukawa, H. Nishimura, T. Yamanishi, and J.-Q. Liu. Chaotic states induced by resetting process in Izhikevich neuron model. *Journal of Artificial Intelligence and Soft Computing Research*, 5, 2015.

S. Nobukawa, H. Nishimura, and T. Yamanishi. Chaotic resonance in typical routes to chaos in the izhikevich neuron model. *Scientific Reports*, 7(1):1–9, 2017.

N. Okafor, D. Giaouris, B. Zahawi, and S. Banerjee. Analysis of fast-scale instability in DC drives with full-bridge converter using Filippovs method. *IET Conference Proceedings*, pages 235–235(1), 2010a.

N. Okafor, B. Zahawi, D. Giaouris, and S. Banerjee. Chaos, coexisting attractors, and fractal basin boundaries in DC drives with full-bridge converter. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 129–132. IEEE, 2010b.

P. R. Owen. Saltation of uniform grains in air. *Journal of Fluid Mechanics*, 20(2):225–242, 1964.

A. M. Pace and S. A. Burden. Piecewise-differentiable trajectory outcomes in mechanical systems subject to unilateral constraints. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 243–252, 2017.

P. R. Pagilla and B. Yu. An experimental study of planar impact of a robot manipulator. *IEEE/ASME Transactions on Mechatronics*, 9(1):123–128, 2004.

D. Pardo, M. Neunert, A. W. Winkler, R. Grandia, and J. Buchli. Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion. In *Robotics: Science and Systems*, volume 10, 2017.

Y. Park, K. M. Shaw, H. J. Chiel, and P. J. Thomas. The infinitesimal phase response curves of oscillators in piecewise smooth dynamical systems. *European Journal of Applied Mathematics*, 29(5):905–940, 2018.

J. J. Payne, N. J. Kong, and A. M. Johnson. The uncertainty aware Salted Kalman Filter: State estimation for hybrid systems with uncertain guards. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, October 2022a. To appear.

J. J. Payne, N. J. Kong, Z. Manchester, and A. M. Johnson. Hybrid moving horizon estimation for contact systems, 2022b. In prep.

F. Pfeiffer and C. Glocker. *Multibody dynamics with unilateral contacts*. John Wiley & Sons, 1996.

M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation*, pages 1366–1373, May 2016.

I. Poulakakis and J. W. Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009.

J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *IEEE-RAS International Conference on Humanoid Robots*, pages 200–207, 2006.

Quanser Inc. User manual qube-servo experiment, set up and configuration, 2016. Quanser Inc.

M. H. Raibert. *Legged robots that balance*. MIT press, 1986.

M. H. Raibert, H. B. Brown Jr, M. Chepponis, J. Koechling, and J. K. Hodgins. Dynamically stable legged locomotion. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1989.

A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1), 2009.

S. Revzen and M. Kvalheim. Data driven models of legged locomotion. In *Micro-and Nanotechnology Sensors, Systems, and Applications VII*, volume 9467, pages 315–322. SPIE, 2015.

M. Rijnen, A. Saccon, and H. Nijmeijer. On optimal trajectory tracking for mechanical systems with unilateral constraints. In *IEEE Conference on Decision and Control*, pages 2561–2566, 2015.

M. Rijnen, E. De Mooij, S. Traversaro, F. Nori, N. Van De Wouw, A. Saccon, and H. Nijmeijer. Control of humanoid robot motions with impacts: Numerical experiments with reference spreading control. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4102–4107. IEEE, 2017a.

M. Rijnen, A. Saccon, and H. Nijmeijer. Reference spreading trajectory tracking control: Experimental analysis on a one-degreeof-freedom setup. In *9th European Nonlinear Dynamics Conference (ENOC), Budapest, Hungary*, pages 1–2, 2017b.

M. Rijnen, J. B. Biemond, N. Van De Wouw, A. Saccon, and H. Nijmeijer. Hybrid systems with state-triggered jumps: Sensitivity-based stability analysis with application to trajectory tracking. *IEEE Transactions on Automatic Control*, 65(11):4568–4583, 2019.

A. Saccon, N. van de Wouw, and H. Nijmeijer. Sensitivity analysis of hybrid systems with state jumps with application to trajectory tracking. In *53rd IEEE Conference on Decision and Control*, pages 3065–3070. IEEE, 2014.

S. Scholtes. *Introduction to piecewise differentiable equations*. Springer Science & Business Media, 2012.

G. Schultz and K. Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on mechatronics*, 15(5):783–792, 2009.

R. v. Schwerin, M. Winckler, and V. Schulz. Parameter estimation in discontinuous descriptor models. In *IUTAM Symposium on Optimization of Mechanical Systems*, pages 269–276. Springer, 1996.

L. F. Shampine, I. Gladwell, L. Shampine, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge university press, 2003.

S. N. Simić, K. H. Johansson, S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 421–436. Springer, 2000.

S. Skaff, A. Rizzi, H. Choset, and P.-C. Lin. A context-based state estimation technique for hybrid systems. In *IEEE International Conference on Robotics and Automation*, pages 3935–3940, April 2005.

D. Sternad, M. Duarte, H. Katsumata, and S. Schaal. Bouncing a ball: tuning into dynamic stability. *Journal of Experimental Psychology: Human Perception and Performance*, 27(5):1163, 2001.

D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

N. Suda and S. Banerjee. Why does narrow band chaos in impact oscillators disappear over a range of frequencies? *Nonlinear Dynamics*, 86(3):2017–2022, Nov 2016. ISSN 1573-269X. doi: 10.1007/s11071-016-3011-y.

Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

R. Tedrake. *Underactuated Robotics*. Course Notes for MIT 6.832, 2022. URL `http://underactuated.mit.edu`.

S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

T. Van Erven and P. Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

P. Varin and S. Kuindersma. A constrained kalman filter for rigid body systems with frictional contact. In *Workshop on the Algorithmic Foundations of Robotics*, 2018.

O. Von Stryk. User's guide for DIRCOL–a direct collocation method for the numerical solution of optimal control problems. Technical report, Technische Universitat Darmstadt, 1999.

X. Wang and J. K. Hale. On monodromy matrix computation. *Computer Methods in Applied Mechanics and Engineering*, 190(18-19):2263–2275, 2001.

R. A. Wehage and E. J. Haug. Dynamic Analysis of Mechanical Systems With Intermittent Motion. *Journal of Mechanical Design*, 104(4):778–784, 10 1982. ISSN 0161-8458.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95–041, University of North Carolina at Chapel Hill, 1995. Updated: July 24, 2006.

A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

H. Wu and V. Pickert. Stability analysis and control of nonlinear phenomena in bidirectional boost converter based on the monodromy matrix. In *IEEE Applied Power Electronics Conference and Exposition-APEC 2014*, pages 2822–2827, 2014.

H. Wu, V. Pickert, X. Deng, D. Giaouris, W. Li, and X. He. Polynomial curve slope compensation for peak-current-mode-controlled power converters. *IEEE Transactions on Industrial Electronics*, 66(1):470–481, 2018.

Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018.

Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. *arXiv preprint arXiv:2104.09771*, 2021.

W. Yang and M. Posa. Impact invariant control with applications to bipedal locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5151–5158. IEEE, 2021a.

W. Yang and M. Posa. Impact invariant control with applications to bipedal locomotion. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5151–5158, 2021b. doi: 10.1109/IROS51168.2021.9636094.

J. Zhang, A. M. Pace, S. A. Burden, and A. Aravkin. Offline state estimation for hybrid systems via nonsmooth variable projection. *Automatica*, 115:108871, 2020. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2020.108871. URL `http://www.sciencedirect.com/science/article/pii/S0005109820300698`.

J. Zhu and A. M. Johnson. Convergent iLQR for underactuated hybrid dynamical systems. In *RSS Workshop on Risk Aware Decision Making*, June 2022.

J. Zhu, N. J. Kong, G. Council, and A. M. Johnson. Hybrid event shaping to stabilize periodic hybrid orbits. In *IEEE Intl. Conference on Robotics and Automation*, pages 6600–6606, Philadelphia, PA, May 2022.