iLQR for Piecewise-Smooth Hybrid Dynamical Systems

Nathan J. Kong[1], George Council[1], and Aaron M. Johnson[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 due to the hybrid nature of the dynamics. The optimal sequence and timing of these modes are 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 with state jumps 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.**

I. 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. 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), [1–3]. 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. [5–7]), 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 [7, 8] which is a simultaneous optimization over each smooth segment with the reset map defining boundary conditions between them [9, 10]. However, the optimal mode sequence is often not known, and so contact-implicit optimization methods have been proposed [11, 12]. 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.

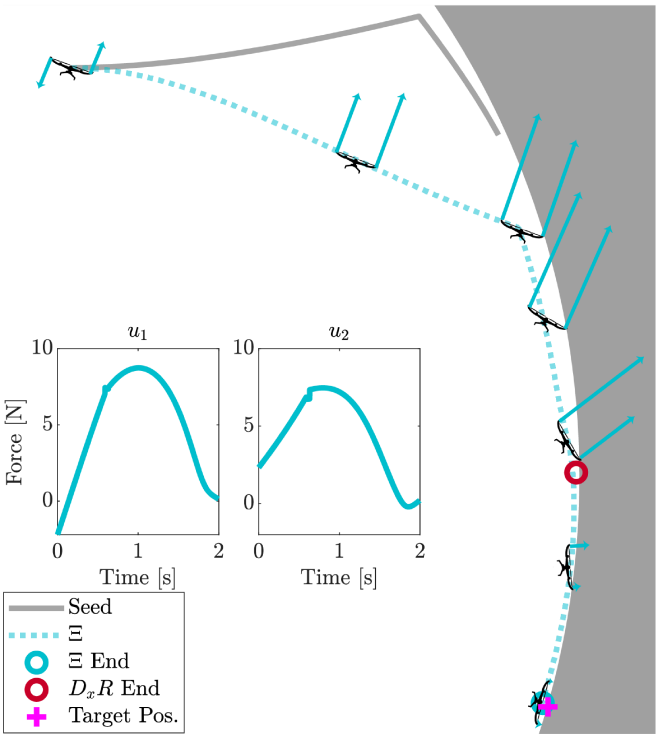

Fig. 1. Demonstrating an example solution using the proposed hybrid iLQR algorithm (labeled with Ξ, the saltation matrix, Def. 2) where the goal is to control a quadcopter to a target final position (shown with a magenta plus) and can make contact with a curved wall with friction. Using a different approximation for the gradient (Jacobian of the reset map, D_xR, [4]) 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.

In this paper, we propose to extend the Iterative Linear Quadratic Regulator (iLQR) method [13, 14] to work for hybrid systems. iLQR (a special case of the Differential Dynamic Programming method, DDP [15]) is a shooting method [5] that utilizes linearization in the search direction (backward pass), but implements the full nonlinear dynamics when obtaining the states of the optimized trajectory

This material is based upon work supported by the U.S. Army Research Office under grant #W911NF-19-1-0080 and the National Science Foundation under grant #ECCS-1924723. 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. Corresponding author N. J. Kong njkong@andrew.cmu.edu.

[1] Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania

(forward pass). One advantage of iLQR, like most shooting methods, is that it can be stopped prematurely to produce a feasible trajectory [11].

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. III-B.
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. III-C.
3) Using reference extensions when there is a mode mismatch to get a valid control input in each mode, Sec. III-D.

In previous hybrid system DDP/iLQR work, [4] took an important first step by extending the approach from [16] 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 enforced through an Augmented Lagrangian method in an outer layer in their algorithm. We instead use the saltation matrix (Def. 2), [17–20], 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. V. 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.

II. Problem Formulation

There are many different formulations of hybrid dynamical systems, with similar but subtly different properties, e.g. [1–3]. For concreteness, in this paper we restrict our attention to the class of systems as given in [21, Def. 2], with the addition of control inputs for each smooth vector field. We elect this class as it includes mechanical systems subject to unilateral and bilateral holonomic constraints, which are of essential interest for robotics.

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

1) $\mathcal{J} := \{I, J, ..., K\} \subset \mathbb{N}$ is the finite set of discrete **modes**.
2) $\mathit{\Gamma} \subset \mathcal{J} \times \mathcal{J}$ 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 with corners [22].
4) $\mathcal{F} := \amalg_{I \in \mathcal{J}} F_I$ is a collection of C^r time-varying **vector fields** with control inputs, $F_I : \mathbb{R} \times D_I \times U_I \to \mathcal{T}D_I$, where U_I is the space of admissible control inputs in mode I.
5) $\mathcal{G} := \amalg_{(I,J) \in \mathit{\Gamma}} G_{(I,J)}(t)$ is the collection of **guards**, where $G_{(I,J)}(t) \subset D_I \times U_I$ for each $(I, J) \in \mathit{\Gamma}$ is defined as a sublevel set of a C^r function, i.e. $G_{(I,J)}(t) = \{(x, u) \in D_I \times U_I | g_{(I,J)}(t, x, u) \leq 0\}$.
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 $(I, J) \in \mathit{\Gamma}$.

An execution of such a hybrid system [21, Def. 4] begins with an initial condition $x_0 \in D_I$ and input $u_I(t, x)$ and adheres to the dynamics F_I on D_I. If the solution reaches guard $G_{(I,J)}$, the reset map $R_{(I,J)}$ is applied to advance the state to domain D_J and activates controller $u_J(t, x)$. An execution is defined over a *hybrid time domain* [21, Def. 3], a disjoint union of intervals $\amalg_{j \in \mathcal{N}} [\underline{t}_j, \bar{t}_j]$. The dynamics of hybrid systems in this class can exhibit complex behavior, including sliding [23], branching [2], and Zeno [24]. Since we make essential use of local linearization theory, we make several assumptions to eliminate these pathologies:

- Assume isolated transition surfaces with transverse [2, 25] intersections.
- Assume that there are no Zeno executions.
- Assume that all vector fields F_I are fully controllable.

Our essential tool to linearize the dynamics of a hybrid system around a switching event is the *saltation matrix* [17–20], which is the necessary update for the variational equation when a hybrid transition occurs.

Definition 2 ([20, Prop. 2]): The **saltation matrix**,

$$\Xi := D_x R + \frac{(F_J - D_x R F_I - D_t R)\, D_x g}{D_t g + D_x g F_I} \tag{1}$$

where

$$\begin{aligned} &\Xi := \Xi_{(I,J)}(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)),\ g := g_{(I,J)}(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)), \\ &R := R_{(I,J)}(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)), \quad F_I := F_I(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)), \\ &F_J := F_J(\underline{t}_{j+1}, R_{(I,J)}(\bar{t}_j, x(\bar{t}_j), u(\underline{t}_{j+1})) \end{aligned}$$

is the first order approximation of variations at hybrid transitions from mode I to J and maps perturbations to first order from pre-transition $\delta x(\bar{t}_i)$ to post-transition $\delta x(\underline{t}_{i+1})$ during the j^{th} transition in the following way:

$$\delta x(\underline{t}_{j+1}) = \Xi_{(I,J)}\big(\bar{t}_j, x(\bar{t}_j)\big)\delta x(\bar{t}_j) + \text{h.o.t.} \tag{2}$$

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

Our class of systems are those whose executions admit linearizations. When a trajectory is away from the hybrid transition, the linearization around a trajectory $(x(t), u(t))$ is exactly the conventional variational equation $\frac{d}{dt}\delta x = D_x F_I((x(t), u(t))\delta x + D_u F_I((x(t), u(t))\delta u$, a linear time-varying ODE. At times $\bar{t}_j$ where the trajectory $x(t)$ undergoes a discrete transition and is thus discontinuous, the variational equation is updated discontinuously with the saltation matrix. For a detailed description of the saltation matrix and its role in linearization, see [17, Chp. 7].

III. Derivation/implementation

This section introduces an abridged derivation of iLQR [13] following [14], proposes the changes to make iLQR work on hybrid systems, and discusses several important key features of the new algorithm.

A. 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 $U := \{u_0, u_1, ..., u_{N-1}\}$ be the input sequence, J_N the terminal cost, and J the runtime cost, where J and J_N are both differentiable functions into $\mathbb{R}$.

The optimal control problem over N timesteps is

$$\min_U \quad J_N(x_N) + \sum_{i=0}^{N-1} J(x_i, u_i) \tag{3}$$

$$\text{where} \quad x_0 = x(0) \tag{4}$$

$$x_{i+1} = f_\Delta(x_i, u_i) \quad \forall i \in \{0, ..., N-1\} \tag{5}$$

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

$$J_k(x_k, U_k) := J_N(x_N) + \sum_{i=k}^{N-1} J(x_i, u_i) \tag{6}$$

with $\{x_{k+1}, ..., x_N\}$ the sequence of states starting at x_k based on U_k and (5). 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),

$$V(x_k) := \min_{u_k} \quad J(x_k, u_k) + V(f_\Delta(x_k, u_k)) \tag{7}$$

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 (7). 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

$$\begin{aligned} Q_k(\delta x, \delta u) :=& J(x_k + \delta x, u_k + \delta u) - J(x_k, u_k) + \\ & V(f_\Delta(x_k + \delta x, u_k + \delta u)) - V(f_\Delta(x_k, u_k)) \end{aligned} \tag{8}$$

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

$$Q_k(\delta x, \delta u) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{ux}^T \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x \\ \delta u \end{bmatrix} \tag{9}$$

the expansion coefficients are

$$Q_{x,k} = J_x + f_{x,k}^T V_x \tag{10}$$

$$Q_{u,k} = J_u + f_{u,k}^T V_x \tag{11}$$

$$Q_{xx,k} = J_{xx} + f_{x,k}^T V_{xx} f_{x,k} + V_x f_{xx,k} \tag{12}$$

$$Q_{ux,k} = J_{ux} + f_{u,k}^T V_{xx} f_{x,k} + V_x f_{uu,k} \tag{13}$$

$$Q_{uu,k} = J_{uu} + f_{u,k}^T V_{xx} f_{u,k} + V_x f_{ux,k} \tag{14}$$

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 (12)–(14) 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,

$$\delta u^* = \arg\min_{\delta u} Q(\delta x, \delta u) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x) \tag{15}$$

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 (9)

$$V_x = Q_x - Q_u Q_{uu}^{-1} Q_{ux} \tag{16}$$

$$V_{xx} = Q_{xx} - Q_{ux}^T Q_{uu}^{-1} Q_{ux} \tag{17}$$

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.

$$\hat{x}_0 = x_0 \tag{18}$$

$$\hat{u}_k = K_k(\hat{x}_k - x_k) + \alpha u_{ff,k} \tag{19}$$

$$\hat{x}_{k+1} = f_\Delta(\hat{x}_k, \hat{u}_k) \tag{20}$$

where the new trajectory is denoted with hats $(\hat{x}, \hat{u})$ and α is used as a backtracking line-search parameters $0 < \alpha \leq 1$ [14, Eqn. 12]. The backwards and forwards passes are run until convergence. Following [14], convergence is when the magnitude of the total expected reduction δJ is small

$$\delta J(\alpha) = \sum_{i=0}^{N-1} u_{ff,i}^T Q_{u,i} + \frac{1}{2} \sum_{k=0}^{N-1} u_{ff,i}^T Q_{uu,i} u_{ff,i} \tag{21}$$

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 [14].

B. 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,

$$g_{(I_j,I_{j+1})}(\bar{t}_j, x(\bar{t}_j), u(\bar{t}_j)) = 0 \tag{22}$$

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), is,

$$f_{\Delta_j}(\hat{x}_k, \hat{u}_k) := \int_{t_k}^{t_{k+1}} f_{I_j}(x(t), \hat{u}_k)dt + \hat{x}_k \tag{23}$$

If during the integration the hybrid guard condition is met, (22), 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

$$f'_\Delta(\hat{x}_k, \hat{u}_k) = \int_{\underline{t}_{j+1}}^{t_{k+1}} f_{I_{j+1}}(x(t), \hat{u}_k)dt + R_{(I_j,I_{j+1})}\left(\bar{t}_j, \int_{t_k}^{\bar{t}_j} f_{I_j}(x(t), \hat{u}_k)dt + \hat{x}_k\right) \tag{24}$$

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, (6), can be augmented with additional cost terms, J_{N_j}, associated with each hybrid transition between the M hybrid modes, as shown in [16],

$$J_0 = J_N(x_N) + \sum_{i=0}^{N-1} J(x_i, u_i) + \sum_{j=1}^{M-1} J_{N_j}(x_{N_j}) \tag{25}$$

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.

C. 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 the backwards pass occurs at timesteps where a hybrid transition is made. By substituting (24) into (7), and adding the transition cost from (25), the resulting Bellman equation for the timesteps during hybrid transition j over timestep k is

$$V(x_k) = \min_{U_k} J(x_k, u_k) + J_{N_j}(x_{N_j}) + V(f'_\Delta(x_k, u_k)) \tag{26}$$

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 (2).

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

$$\delta x(\bar{t}_j) \approx f_{x,\Delta_j}\delta x(t_k) + f_{u,\Delta_j}\delta u(t_k) \tag{27}$$

$$\delta x(\underline{t}_{j+1}) \approx \Xi \delta x(\bar{t}_j) \tag{28}$$

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 (10)–(14) 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 (9) are,

$$Q_{x,k} = J_x + f_{x,\Delta_j}^T J_{x,N_j} + f_{x,\Delta_j}^T \Xi^T V_x \tag{29}$$

$$Q_{u,k} = J_u + f_{u,\Delta_j}^T J_{x,N_j} + f_{u,\Delta_j}^T \Xi^T V_x \tag{30}$$

$$Q_{xx,k} = J_{xx} + f_{x,\Delta_j}^T J_{xx,N_j} f_{x,\Delta_j} + f_{x,\Delta_j}^T \Xi^T V_{xx} \Xi f_{x,\Delta_j} \tag{31}$$

$$Q_{ux,k} = J_{ux} + f_{u,\Delta_j}^T J_{xx,N_j} f_{x,\Delta_j} + f_{u,\Delta_j}^T \Xi^T V_{xx} \Xi f_{x,\Delta_j} \tag{32}$$

$$Q_{uu,k} = J_{uu} + f_{u,\Delta_j}^T J_{xx,N_j} f_{u,\Delta_j} + f_{u,\Delta_j}^T \Xi^T V_{xx} \Xi f_{u,\Delta_j} \tag{33}$$

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. However, the computation of the second order variation expression may not be easy for systems with large state space.

As an alternative expansion, in [4, Eq. (21)] the authors use an Augmented Lagrangian method to handle variations in impact time and they use the Jacobian of the reset map to propagate perturbations in state through the hybrid transition, instead of the saltation matrix (2). 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 (28)

$$\delta x(\underline{t}_{j+1}) \approx D_x R_{(I_j,I_{j+1})}(\bar{t}_j, x(\bar{t}_j), u(t_k))\delta x(\bar{t}_j) \tag{34}$$

and similarly in the hybrid coefficient expansion equations (29)–(33). 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.

D. 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 [18, 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.

E. Algorithm

With each hybrid modification to iLQR listed in Sections III-B, III-C, and III-D 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 [14]) 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 [14]. 5) Then the backwards pass, hybrid extensions, and forward passes are repeated until convergence.

IV. 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_xR-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 [21]. 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,

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q,\dot{q}) + A(q)^T\lambda = \Upsilon(q,u) \quad (35)$$

$$A(q)\ddot{q} + \dot{A}(q)\dot{q} = 0 \quad (36)$$

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 [21].

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.

A. Bouncing ball elastic impact

We begin with a 1D bouncing ball under elastic impact [3], 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

$$F_1(x,u) = F_2(x,u) := \left[\dot{z}, \frac{u - mg}{m}\right]^T \quad (37)$$

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,

$$D_xR_{(1,2)} = \begin{bmatrix} 1 & 0 \\ 0 & -e \end{bmatrix}, \Xi_{(1,2)} = \begin{bmatrix} -e & 0 \\ \frac{(u-mg)(e+1)}{m\dot{z}} & -e \end{bmatrix} \quad (38)$$

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 problems were seeded with a constant input force to obtain different number of

TABLE I

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

bounces. A suite of bouncing conditions are considered and are summarized in Table I. 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. III-D, 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 $|\delta J| \leq 0.05$.

B. 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. 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 $R = 0.0001$, $Q_N = 100 I_{2\times 2}$ and no input for the seed.

C. 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, (35), are ballistic dynamics with direct inputs on configurations, $M(q) = mI_{2\times 2}$, $N(q, \dot{q}) = [0, -mg]^T$, $C(q, \dot{q}) = 0_{2\times 2}$, 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 $R = 0.0001$, $Q_N = 100 I_{4\times 4}$ 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.

D. Perching quadcopter

We introduce a quadcopter perching example inspired by [26], 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 (35) with the following

$$M(q) := \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}, \quad C(q, \dot{q}) := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (39)$$

$$N(q, \dot{q}) := \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix}, \quad \Upsilon := \begin{bmatrix} -\sin(\theta)(u_1 + u_2) \\ \cos(\theta)(u_1 + u_2) \\ \frac{1}{2}(u_2 w - u_1 w) \end{bmatrix} \quad (40)$$

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,

$$[y_L, z_L]^T = [y - \frac{1}{2} w \cos\theta, z - \frac{1}{2} w \sin\theta]^T \quad (41)$$

$$[y_R, z_R]^T = [y + \frac{1}{2} w \cos\theta, z + \frac{1}{2} w \sin\theta]^T \quad (42)$$

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, $\lambda_t = \mu \lambda_n$, 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

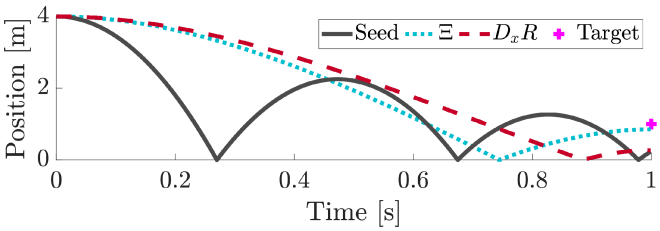

Fig. 2. Bouncing ball with elastic impact where 1 bounce is optimal and 3 bounces are seeded. The target end position is shown in (magenta plus). Both gradient update methods were able to pull away the unnecessary bounces, but the method using D_xR did not converge or get to the target state.

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. 1.

V. 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.

A. Bouncing Ball with Elastic Impacts

The outcomes of the experiment comparing D_xR-iLQR to Ξ-iLQR are shown in Table I. An example run is shown in Fig. 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.

[1]This solution was confirmed as a local minima under a single bounce by comparing it against a trajectory produced using direct collocation [7] constrained to a single bounce, as shown in Table. I.

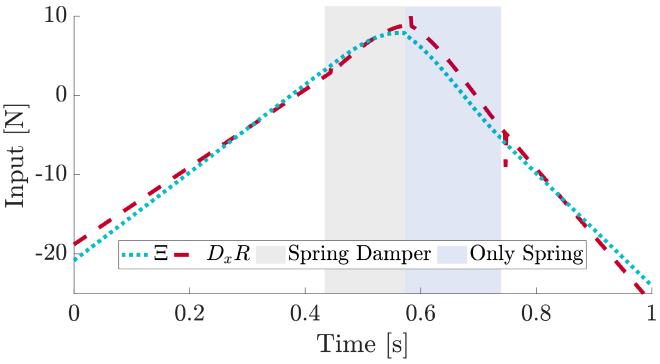

Fig. 3. Bouncing ball on a spring-damper ground where both gradient update methods found similar trajectories but using the Jacobian of the reset map D_xR lead to not being able to fully converge as evident by the residual spikes near hybrid transitions.

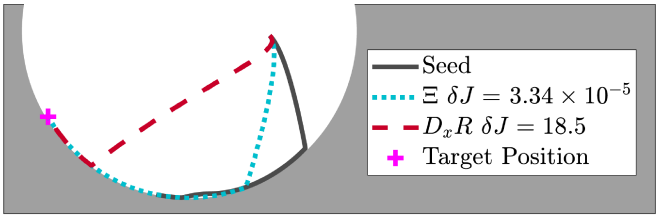

Fig. 4. Ball drop on a curved surface with plastic impacts where both gradient methods produced trajectories that got to the end goal, but using D_xR did not converge and had a significantly higher cost.

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.

B. 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. 3 where D_xR-iLQR was not able to smooth out the spikes near mode changes. This is also 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.

C. 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. 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 a 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.

D. Perching quadcopter

In this example, the final position trajectories are shown in Fig. 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.

VI. Discussion

In this work, we extended iLQR to hybrid dynamical systems with piecewise smooth solutions with state jumps. 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.

In future work, we will investigate systems with intersecting hybrid guards where the Bouligand derivative [27, 28] will play an analogous role as the saltation matrix.

References

[1] A. Back, J. M. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," in *Hybrid Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1993, vol. 736.

[2] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[3] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems," *IEEE control systems magazine*, vol. 29, no. 2, pp. 28–93, 2009.

[4] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448–5455, 2020.

[5] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, 1998.

[6] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, 2009.

[7] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[8] O. Von Stryk, "User's guide for DIRCOL–a direct collocation method for the numerical solution of optimal control problems," Technische Universitat Darmstadt, Tech. Rep., 1999.

[9] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on mechatronics*, vol. 15, no. 5, pp. 783–792, 2009.

[10] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 1366–1373.

[11] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[12] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, p. 43, 2012.

[13] 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*, 2004, pp. 222–229.

[14] 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.

[15] D. Q. Mayne, "Differential dynamic programming–a unified approach to the optimization of dynamic systems," in *Control and Dynamic Systems*. Elsevier, 1973, vol. 10, pp. 179–254.

[16] 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*, vol. 154, no. 2, pp. 382–417, 2012.

[17] R. I. Leine and H. Nijmeijer, *Dynamics and bifurcations of non-smooth mechanical systems*. Springer Science & Business Media, 2013, vol. 18.

[18] M. Rijnen, A. Saccon, and H. Nijmeijer, "On optimal trajectory tracking for mechanical systems with unilateral constraints," in *IEEE Conference on Decision and Control*, 2015, pp. 2561–2566.

[19] M. A. Aizerman and F. R. Gantmacher, "Determination of stability by linear approximation of a periodic solution of a system of differential equations with discontinuous right-hand sides," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 11, no. 4, 1958.

[20] S. A. Burden, T. Libby, and S. D. Coogan, "On contraction analysis for hybrid systems," 2018, arXiv:1811.03956.

[21] 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*, vol. 35, no. 11, 2016.

[22] D. Joyce, "On manifolds with corners," in *Advances in Geometric Analysis*, ser. Advanced Lectures in Mathematics. International Press of Boston, Inc., 2012, vol. 21, pp. 225–258.

[23] M. R. Jeffrey, "Dynamics at a switching intersection: Hierarchy, isonomy, and multiple sliding," *SIAM Journal on Applied Dynamical Systems*, vol. 13, no. 3, pp. 1082–1105, 2014.

[24] J. Zhang, K. H. Johansson, J. Lygeros, and S. Sastry, "Dynamical systems revisited: Hybrid systems with zeno executions," in *Workshop on Hybrid Systems: Computation and Control*, 2000, pp. 451–464.

[25] M. W. Hirsch and S. Smale, "Differential equations, dynamical," *Systems and Linear Algebra*, 1974.

[26] A. Lussier Desbiens, A. T. Asbeck, and M. R. Cutkosky, "Landing, perching and taking off from vertical surfaces," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 355–370, 2011.

[27] 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*, vol. 15, no. 2, pp. 1227–1267, 2016.

[28] S. Scholtes, *Introduction to piecewise differentiable equations*. Springer Science & Business Media, 2012.