Optimal Motion Planning for a Class of Hybrid Dynamical Systems with Impacts

Andrew W. Long, Todd D. Murphey, and Kevin M. Lynch

***Abstract*— Hybrid dynamical systems with impacts typically have controls that can influence the time of the impact as well as the result of the impact. The leg angle of a hopping robot is an example of an impact control because it can influence when the impact occurs and the direction of the impulse. This paper provides a method for computing an explicit expression for the first derivative of a cost function encoding a desired trajectory. The first derivative can be used with standard optimization algorithms to find the optimal impact controls for motion planning of hybrid dynamical systems with impacts. The resulting derivation is implemented for a simplified model of a dynamic climbing robot.**

I. Introduction

Running, hopping and juggling robots are examples of hybrid dynamical systems with impacts (HDSI). These systems experience continuous dynamics until they undergo an impact, resulting in a switch in the dynamics and/or a discontinuity in the state. Typically these systems have controls that can influence the time of the impact as well as the result of the impact. An example impact control could be the leg angle prior to impact for a hopping robot: the leg angle determines when the impact occurs and influences the angle of the impulse due to the impact. This paper addresses motion planning with impact controls for a particular class of HDSI.

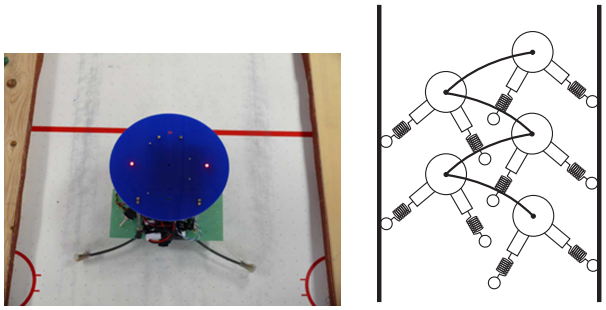
Consider a hybrid dynamical system with impacts, described by the equations:

$$\dot{x} = f(x, u, t) \quad \text{when } (x, u, \xi) \notin \mathcal{S} \tag{1}$$

$$x^+ = \Delta(x, u, \xi) \quad \text{when } (x, u, \xi) \in \mathcal{S} \tag{2}$$

where $x \in \mathcal{X}$ is the state of the system, f represents the continuous dynamics, $\mathcal{S}$ is the impact surface, and Δ is an impact map that instantaneously maps a pre-impact condition to a post-impact state x^+. The controls consist of the continuous-time control $u \in \mathcal{U}$ and the impact control $\xi \in \Xi$. The impact control ξ may affect both the impact map Δ (e.g., control impulses applied at the impact time) as well as the time at which an impact occurs. Impacts occur on the impact surface $\mathcal{S}$, which may be a function of x, u, and/or ξ.

An example of an HDSI is a biped robot. The dynamics f describe the phases where the robot has zero, one, or two feet on the ground, and the impact map Δ corresponds to events when a foot hits the ground. In this case, the impact surface $\mathcal{S}$ depends only on x, i.e., $\mathcal{S} = \mathcal{S}(x)$. Another example is the Monkeybot, a two-link planar robot that dynamically locomotes on a vertical steel wall by using (1) a single motor at the joint between the two links, and (2) passive pivot joints ("hands") at the endpoints of the links that can be electromagnetically connected to or disconnected from the wall. (See, for example, a video of an early prototype of the Monkeybot at http://www.youtube.com/watch?v=0hfwJEVQyeQ.) The dynamics f describe phases when zero, one, or two pivot joints are attached to the wall; the control u is the torque at the joint motor; and impacts occur when the electromagnet at a free-swinging hand is clamped to the wall, creating a pivot joint. In this case, ξ is the set of impact times. Consequently, the impact surface $\mathcal{S}$ is independent of x and u.



(a) (b)

Fig. 1. (a) Image of ParkourBot. (b) A cartoon of the ParkourBot ascending a chute

In this paper, our interest is in systems of the form (1)–(2) where the continuous control is either zero ($u = 0$) or given by a specified feedback law ($u = u(x)$). The impact surface $\mathcal{S}$ may be a function of both x and ξ, i.e.,

$$\Sigma : \begin{cases} \dot{x} = f(x, t) & \text{when } (x, \xi) \notin \mathcal{S} \\ x^+ = \Delta(x, \xi) & \text{when } (x, \xi) \in \mathcal{S}, \end{cases} \tag{3}$$

where f is assumed to be at least once differentiable with respect to each input. A sequence of impact controls is given by $\xi = [\xi_1, \ldots, \xi_N]^T$ for N impacts with impact times $\tau = [\tau_1, \ldots, \tau_N]^T$ and post-impact states $x^+ = [x_1^+, \ldots, x_N^+]^T$. Since the continuous control $u(x)$ has been specified, the HDSI has been converted to a discrete-time problem.

Examples of such systems include:

- **Robot parkour.** The ParkourBot (see Fig. 1(a)), developed at Carnegie Mellon, is a mobile robot with two springy BowLegs that allow it to dynamically locomote

This work was supported by NSF IIS-0803734 and Andrew Long's NDSEG fellowship

A. W. Long, T. D. Murphey, and K. M. Lynch are with the Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208 USA awlong@u.northwestern.edu, {t-murphey, kmlynch}@northwestern.edu

in planar vertical environments with vertical footholds [1] as shown in the cartoon of Fig. 1(b). A simple model of this system treats the robot as a point mass in ballistic flight with state x, plus impacts where one of the two springy legs bounces the robot off of a foothold. The impact control ξ consists of the leg angle at impact and the kinetic energy added to or subtracted from the system at impact. The impact surface is written $\mathcal{S}(x, \xi)$, as contact with a foothold depends on both the position of the robot and the angle of the leg.

- **Robot juggling.** In single-ball "bat juggling," a robot arm repetitively bats a ball to achieve a desired vertical juggling cycle [2], [3]. Treating the ball's state as x, the dynamics f consist of the uncontrolled ballistic flight of the ball, and the impact control ξ is the configuration and velocity of the batter at impact.

Given a system Σ, the purpose of this paper is to find a finite sequence of impact controls ξ minimizing a cost function J encoding the desired behavior of the system. Standard optimization algorithms utilize first and possibly second derivatives of the cost function to compute a locally optimal solution. The second derivative is typically expensive to compute, so there are many algorithms based entirely on the first derivative. The first derivative can be difficult to calculate if the cost function cannot be expressed analytically. One can imagine approximating the derivative using finite difference approximation, but such an approach is typically undermined by numerical instability unless all the differential equations involved happen to be asymptotically stable.

The primary contributions of this paper are the following:

1) We derive an explicit expression for the derivative of J with respect to the impact controls ξ_k. By deriving explicit expressions for the derivative, we get the advantage of stable numerical methods for evaluating trajectories of differential equations. In addition, the explicit expression for the derivative provides a numerical test of optimality, which is useful for terminating the optimization algorithm.
2) We apply the results to motion planning for the simplified model of the ParkourBot, finding impact controls for a variety of motion goals using a gradient descent method.

In Section II we place this work in the context of previous work. Section III gives the problem statement and the notation used in the paper. Section IV describes the discretized trajectory of a hybrid dynamical system with impacts. The cost function and its first derivative are discussed in Section V. The impact control optimization is preformed for several motion goals of the ParkourBot in Section VI.

II. Related Work

Robotics applications of HDSI include legged robotic locomotion and robotic juggling. Raibert and Brown's pioneering work on legged locomotion showed that a single-legged robot could produce stable limit cycles and even hop over obstacles [4]. Hodgins and Raibert developed several simple controllers to vary the step length of the robot [5], but noted that precise placement of the foot for one step may produce uncontrollable motion after several steps. Real-time planning of a single-legged BowLeg hopper was implemented experimentally by Brown and Zeglin to traverse stepping stones by using a best first search graph method [6]. Additionally, mirror laws and recurrent control theory has been studied as a method for producing asymptotic stable limit cycles of HDSI in the area of juggling robots [2], [3].

Brogliato and Río developed a framework to study the controllability and stability of hybrid mechanical systems with impacts [7]. The controllability of hybrid mechanical systems with time controlled impacts such as the Monkeybot was studied by Bullo and Zefran with a geometric approach using affine connections and linear jump transition maps [8]. The method provided in this paper optimizes impact controls for planning of multiple impacts of HDSI that do not have direct control of the impact time, but this method does not investigate the controllability and stability of these systems.

The motivation for the work in this paper is drawn from switching time optimization for systems with complete control of the switching times [9], [10], [11]. The systems considered in this paper are different in that they do not necessarily have direct control of the impact times. The work in [9] and [11] utilizes the fundamental principles of calculus to calculate the first and second derivative of the cost with respect to the switching times for switched systems and switched systems with impulses, respectively. In this paper, a similar approach is used to derive the first derivative of the cost with respect to a more general class of impact controls. By using this method, we have an explicit expression for the derivative of the cost, resulting in a stable numerical method for standard optimization algorithms.

III. Problem Statement

Given a system Σ specified by (3) with initial conditions $x(\tau_0) = x_{\text{init}}$, we seek a locally optimal sequence of impact controls ξ that minimizes a cost function J encoding the desired trajectory for N impacts. The cost function considered in this paper is a discrete-time function that weights each of the N post-impact states:

$$J(x_{init}, \xi) = \sum_{k=1}^{N} L_k(x_k^+, \tau_k), \tag{4}$$

where x_k^+ is the kth post-impact state. Note that this cost function does not directly include the impact controls; however, the method presented in this paper can be modified easily to to include the controls in the cost function.

For example, the goal could be to track a desired set of post-impact states:

$$L_k(x_k^+, \tau_k) = \frac{1}{2}(x_k^+ - x_{k,d}^+)^T Q_k (x_k^+ - x_{k,d}^+), \tag{5}$$

where $x_{k,d}^+$ is a desired post impact state and Q_k is a positive-definite symmetric weighting matrix.

The numerical optimization in this paper is conducted using a standard first-order steepest descent algorithm [12].

In order to implement this algorithm, the first derivative of the cost with respect to the impact controls is required.

A. Notation

This paper uses operator notation $Dg(x) \circ \partial x$ to represent derivatives, where this notation reads $Dg(x)$ operates on the perturbation of x, ∂x. An example of this operator notation is $A(\cdot) \circ (x_1, x_2)$ for the bilinear map (such as the second derivative) $B(x_1, x_2) = x_1^T A(\cdot) x_2$. Operator notation is used in this paper to simplify the representation of multi-dimensional derivatives without having to specify the matrix multiplication such as in the example above.

The full derivative of $g(\cdot)$ with respect to a variable will be written as $D_{var} g(arg_1, arg_2, \ldots) \circ \partial var$. Note that multiple arguments may be differentiable by var in this notation, therefore, chain rule may be necessary. $D_n g(arg_1, arg_2, \ldots) \circ \partial arg_n$ is known as a slot derivative in which $g(\cdot)$ is differentiated with respect to the nth argument. If the argument is a vector, slot derivatives with respect to the ith element of the argument will be written as $D_{n,i} g(arg_1, arg_2, \ldots) \circ \partial arg_{n,i}$ where n is the argument and i corresponds to the ith element of the nth argument. For example, the full derivative of $f(x, y(x, z))$ with respect to x is given by

$$D_x f \circ \partial x = D_1 f \circ \partial x + D_2 f \circ D_x y \circ \partial x,$$

where the dependencies are not written for the individual functions in order to save space.

Sometimes we will write the state $x(t)$ as $x(\xi, t)$ to be explicit about the dependence on the impact controls.

IV. Discrete Trajectory

Since the continuous-time controller $u(x)$ is assumed to be given, the system can be represented by discrete-time states x_k^- at time τ_k^- and x_k^+ at time τ_k^+ which correspond to the pre-impact and post-impact states, respectively. Note that $\tau_k^- = \tau_k^+$, but this notation will be used to specify just before impact and just after impact.

The initial condition and initial time will be denoted as x_0 and τ_0. Note that any derivative of these quantities will be zero, since they are assumed to be given.

The pre-impact state x_k^- can be calculated by integrating (3) from x_{k-1}^+. This can be written using the fundamental theorem of calculus as

$$x_k^- \equiv x(\xi, \tau_k^-) = \Delta_{k-1}(x_{k-1}^-, \xi_{k-1}) + \int_{\tau_{k-1}^+}^{\tau_k^-} f_{k-1}(x(s), s) ds. \tag{6}$$

The kth post-impact state x_k^+ from (3) is given by

$$x_k^+ = \Delta_k(x_k^-, \xi_k). \tag{7}$$

The kth impact time is given as a function of an impact control, a previous state and the time of the previous state,

$$\tau_k = \tau_k(\xi_k, x_{k-1}^+, \tau_{k-1}), \tag{8}$$

where x_{k-1}^+ and τ_{k-1} are x_0 and τ_0, respectively, if $k = 1$. For this paper, it is assumed that this expression is known analytically. If the time of impact expression is not known explicitly, the time of impact can be computed numerically with a root-finding method. The derivatives of this expression with respect to its inputs can be computed analytically by applying the Implicit Function Theorem and using the integral representation of the dynamics.

The first derivative of the cost function with respect to the impact controls depends on the derivative of the trajectory with respect to each impact control. Since the impact times depend on the previous states and impact controls, the Leibniz Integral rule must be taken into consideration when differentiating the pre-impact state in the integral form as shown in (6).

V. The Cost and Derivative

This section contains the main mathematical result of the paper. The equations provided in this section rely on a result derived in the Appendix.

The numerical optimization requires computing $J(\cdot)$ and $DJ(\cdot) \circ z$, where z is the perturbation. This paper derives an explicit expression for the derivative of the cost, even when the differential equations governing the dynamics cannot be integrated analytically. The derivative of the cost in (4) with respect to the each set of impact controls, ξ_i, at impact i is given by chain rule:

$$\begin{aligned} D_{\xi_i} J(x_{init}, \xi) \circ \partial \xi_i = \sum_{k=1}^{N} \big[& D_1 L_k(x_k^+, \tau_k) \circ D_{\xi_i} x_k^+ \circ \partial \xi_i \\ & + D_2 L_k(x_k^+, \tau_k) \circ D_{\xi_i} \tau_k \circ \partial \xi_i \big]. \end{aligned} \tag{9}$$

By investigating the structure of (9), the derivative for each impact control can be concisely expressed. It will be shown that only a few things need to be computed for each impact in order to compute all the derivatives.

Equation (9) can be expressed in matrix notation as:

$$\begin{aligned} & D_{\xi_i} J(x_{init}, \xi) \circ \partial \xi_i = \\ & \sum_{k=1}^{N} \begin{bmatrix} D_1 L_k(x_k^+, \tau_k) \\ D_2 L_k(x_k^+, \tau_k) \end{bmatrix}^T \begin{bmatrix} D_{\xi_i} x_k^+ \circ \partial \xi_i \\ D_{\xi_i} \tau_k \circ \partial \xi_i \end{bmatrix}. \end{aligned} \tag{10}$$

We need a way to compute these right hand side terms. To do so, note that these terms are simply the derivative of the impact map and the impact time with respect to the impact control:

$$\begin{bmatrix} D_{\xi_i} x_k^+ \circ \partial \xi_i \\ D_{\xi_i} \tau_k \circ \partial \xi_i \end{bmatrix} = \begin{bmatrix} D_{\xi_i} \Delta_k(x_k^-, \xi_k) \circ \partial \xi_i \\ D_{\xi_i} \tau_k(\xi_k, x_{k-1}^+, \tau_{k-1}) \circ \partial \xi_i \end{bmatrix}.$$

Let

$$\Upsilon_k^i = \begin{bmatrix} D_{\xi_i} \Delta_k(\cdot) \circ \partial \xi_i & D_{\xi_i} \tau_k(\cdot) \circ \partial \xi_i \end{bmatrix}^T$$

$$C_k = \begin{bmatrix} D_1 L_k(x_k^+, \tau_k) & D_2 L_k(x_k^+, \tau_k) \end{bmatrix}.$$

The derivative of the impact map and the impact time with respect to the each impact control is derived in the Appendix.

The results of the derivation are

$$\Upsilon_k^i = \begin{cases} \left(\prod_{j=i+1}^{k} \Gamma_{k-j+i+1} \right) \Upsilon_i^i & k > i \quad (11a) \\ \Upsilon_i^i & k = i \quad (11b) \\ \mathbf{0} & k < i \quad (11c) \end{cases}$$

where

$$\Gamma_k = \begin{bmatrix} D_1\Delta_k(\cdot) \circ \alpha_k & D_1\Delta_k(\cdot) \circ \beta_k \\ (D_2\tau_k(\cdot))^T & D_3\tau_k(\cdot) \end{bmatrix}$$

$$\alpha_k = \Phi_{k-1}(\tau_k^-, \tau_{k-1}^+) + f_{k-1}(x_k^-, \tau_k)(D_2\tau_k(\cdot))^T$$

$$\begin{aligned} \beta_k =& f_{k-1}(x_k^-, \tau_k) D_3\tau_k(\cdot) \\ & - \Phi_{k-1}(\tau_k^-, \tau_{k-1}^+) f_{k-1}(x_{k-1}^+, \tau_{k-1}), \end{aligned}$$

and $\Phi(t, \tau)$ is the state transition matrix [13] calculated from

$$\frac{d}{dt}\Phi(t, \tau) = A(t) \circ \Phi(t, \tau),$$

with $A(t) = D_1 f_{k-1}(x(t), t)$ and $\Phi(\tau, \tau) = I$, where I is the identity matrix.

The derivative of the cost simplifies with (11c) to

$$D_{\xi_i} J(x_{init}, \xi) \circ \partial\xi_i = \sum_{k=i}^{N} C_k \Upsilon_k^i. \quad (12)$$

since the derivative of the cost's summand is zero for $k < i$.

To compute the derivative of the cost with respect to each impact control, we only need to compute $C_k, \Upsilon_k^k, \Gamma_k \; \forall \; k = 1, \ldots, N$. Each derivative is simply a combination (sum and multiplication) of some or all of these quantities as shown in (12).

The explicit expression of the derivative in (12) can be used with stable numerical methods to optimize the impact control sequence. In addition, the explicit expression for the derivative provides a numerical test of optimality, which is useful for terminating the optimization algorithm.

VI. Example: ParkourBot

The system studied in this example is the two-legged ParkourBot designed at Carnegie Mellon and depicted in Fig. 1(a). The ParkourBot is equipped with two BowLegs and is similar to the one-legged BowLeg hopper in [14]. During flight, the ParkourBot is capable of independently positioning the leg angles and storing energy by compressing the spring-like legs. During stance, the stored energy is converted to kinetic energy. The net amount of energy input to the system is the difference between the amount of energy stored and losses due to the impact. By controlling the net energy input and the leg angles, the robot is capable of climbing up and down as well as bouncing in place. A cartoon of the robot climbing is seen in 1(b). (See videos at http://www.dynaclimb.com/.)

A cartoon of the ParkourBot is shown in Fig. 2. This example assumes a point mass body, massless legs, hips located at the point mass and zero stance phase time (instantaneous impacts). Impulses act along the leg at impact. It is assumed that the robot has complete control of the net energy input as well as the leg angles. To simplify the example, the net energy input will be specified. The motion planning optimization is to determine the locally optimal leg angle sequence to drive the robot from an initial state to a final state in a vertical chute.

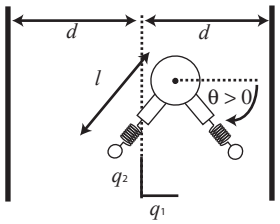


Fig. 2. Definition of variables for climbing robot example.

Three cases are investigated with this example: bouncing in place with specifying only the final state, bouncing in place with trajectory tracking and climbing with trajectory tracking. The purpose of the first two cases is to show the benefit of the trajectory tracking. The third case illustrates that the optimization can be used for climbing with constant non-zero energy input and a constant increase in the vertical impact location at each impact. Gradient descent with an Armijo Line search algorithm was used to determine the locally optimal controls [15].

The ParkourBot's configuration is the horizontal and vertical coordinates of the point mass denoted as $q(t) = [q_1(t), q_2(t)]^T \in Q$. The inertial reference frame is located in the center of the chute with the vertical walls located a distance d away. The state is given as $x(t) = [q(t)^T, \dot{q}(t)^T]^T \in TQ$. The free flight dynamics are governed by:

$$\dot{x}(t) = f(x(t), t) = \left[\dot{q}_1(t), \dot{q}_2(t), 0, -g/m\right]^T,$$

where m is the mass and g is gravity. For this example, $m = 1$ and $g = 1$. The legs are assumed to be the same length, $l = 0.3$. The angles of the legs, θ, are measured positive clockwise from the horizontal axis positive to the right as shown in Fig. 2. This convention was used to have positive leg angles. For this example, the impact controls are the leg angles ($\xi = \theta$). The distance from the center to either wall is given as $d = 1 + l\cos(\pi/4)$. The initial condition was selected to be $x(\tau_0) = [-1, -0.5, 1, 1]^T$, which produces period-1 bouncing in place motion for optimal angles of $\pi/4$ and $3\pi/4$ for the right and left walls, respectively. This initial condition was chosen to verify the results of the optimization.

The impact time of (8) can be calculated from

$$\tau_k(\theta_k, x_{k-1}^+, \tau_{k-1}) = \frac{\pm d - q_1(\tau_{k-1}) - l\cos(\theta_k)}{\dot{q}_1(\tau_{k-1})}$$

where the sign of d depends on which wall is impacted. Since the legs are identical, no labeling of the length and angle is required.

With the assumptions above, the impact impulse can only affect the velocity in the direction along the leg. This velocity will be referred to as the radial velocity ($\dot{r}(t)$). The normal velocity ($\dot{n}(t)$) will be the velocity orthogonal to the

leg. With the leg angle measured positive in the clockwise direction, the radial and normal velocities can be calculated with a change in coordinates given by

$$\begin{bmatrix} \dot{n}(\tau_k^-) \\ \dot{r}(\tau_k^-) \end{bmatrix} = \begin{bmatrix} \sin(\theta_k) & \cos(\theta_k) \\ -\cos(\theta_k) & \sin(\theta_k) \end{bmatrix} \begin{bmatrix} \dot{q}_1(\tau_k^-) \\ \dot{q}_2(\tau_k^-) \end{bmatrix}.$$

The post-impact velocities for impact k are given by

$$\begin{bmatrix} \dot{n}(\tau_k^+) \\ \dot{r}(\tau_k^+) \end{bmatrix} = \begin{bmatrix} \dot{n}(\tau_k^-) \\ \sqrt{\dot{r}(\tau_k^-)^2 + \frac{2E_k}{m}} \end{bmatrix},$$

where E_k is the net energy input to the system. Note that with this representation, the pre-impact radial velocity is assumed to be negative and the post-impact velocity is assumed to be positive.

The impact map is then given by

$$\begin{aligned} \Delta_k(x(\tau_k^-), \theta_k) &= \left[q_1(\tau_k^+), q_2(\tau_k^+), \dot{q}_1(\tau_k^+), \dot{q}_2(\tau_k^+)\right]^T \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sin(\theta_k) & -\cos(\theta_k) \\ 0 & 0 & \cos(\theta_k) & \sin(\theta_k) \end{bmatrix} \begin{bmatrix} q_1(\tau_k^-) \\ q_2(\tau_k^-) \\ \dot{n}_1(\tau_k^+) \\ \dot{r}_2(\tau_k^+) \end{bmatrix} \end{aligned}$$

The required derivatives and variables defined in Section V as well as the example simulations were carried out in Mathematica. The optimizations were conducted until the norm of the gradient was less than 10^{-5}.

A. Bounce In Place: Terminal Cost

This example case considers the terminal cost function governed by

$$J(x_{\text{init}}, \xi) = \frac{1}{2}(x_N^+ - x_{N,d}^+)^T Q_N (x_N^+ - x_{N,d}),$$

where Q_N is the identity matrix. The goal of the optimization is to find the optimal leg angles that take an initial state to the same state (with appropriate signs on the horizontal position and velocity) after $N = 6$ bounces. The initial leg angle controls were given as $\xi = \theta = [\pi/5, 4\pi/5, \pi/5, 4\pi/5, \pi/5, 4\pi/5]^T$ with initial cost 0.0026. The optimal angles after 1006 iterations were $\xi^* = [0.677, 2.484, 0.571, 2.484, 0.676, 2.356]^T$. The final cost and norm of the gradient were $1.52\text{x}10^{-9}$ and $9.96\text{x}10^{-6}$, respectively. A plot of the configuration (vertical vs. horizontal positions) for the initial and optimal angles can be seen in Fig. 3(a) and Fig. 3(b), respectively, with the impact sequence marked.

It can be seen that the robot bounces in place at a lower potential energy state until the last bounce. This is most likely poor planning because the robot relies mainly on the last bounce to reach the desired state. Also, the robot will be moving faster at a lower potential energy state, which may be an issue for stability. This trajectory is a logical outcome because no weighting was associated with the previous impacts.

B. Bounce In Place: Incremental Cost

This example case considers trajectory tracking with the incremental cost described in (4):

$$J(x_{init}, \xi) = \sum_{k=1}^{N} \frac{1}{2}(x_k^+ - x_{k,d}^+)^T Q_k (x_k^+ - x_{k,d}^+),$$

where Q_k is the identity matrix and $x_{k,d}^+$ is the desired post-impact state at each impact. The number of impacts and the initial angles were the same as those in the previous case. With this cost function, the initial cost was 0.286. The optimal angles after 65 iterations were $\xi^* = [0.785, 2.356, 0.785, 2.356, 0.785, 2.356]^T$, which are close to the expected optimal angles. The final cost and norm of the gradient were $8.6\text{x}10^{-12}$ and $8.5\text{x}10^{-6}$, respectively. A plot of the configuration (vertical vs. horizontal positions) can be seen in Fig. 3(c) with the impact sequence marked. It can be seen that the robot bounces in place for each impact. This is most likely a more stable trajectory than that of the last case.

C. Climb: Incremental Cost

This case uses the same cost function as the previous case, except that $Q_k = diag(0, 1, 0, 0)$ to only weight the position of the vertical coordinate for each impact. For constant vertical climbing, q_2 should increase by the same amount at each impact. The constant energy input for each impact was selected to add the required gain in potential energy (mgh) for a single bounce. The desired vertical position of the first bounce was selected to be at the same height as the initial condition. All other desired vertical positions were located $(k-1)h$ above the initial condition, where k is the number of impacts from the initial state. For $h = 0.15$ and 6 bounces, the initial angles of $\xi = [\pi/5, 4\pi/5, \pi/5, 4\pi/5, \pi/5, 4\pi/5]^T$ produced an initial cost of 0.0404. The optimal angles after 39 iterations were $\xi^* = [0.697, 2.572, 0.554, 2.588, 0.556, 2.509]^T$. The final cost and norm of the gradient were 0.000157 and $6.3\text{x}10^{-6}$, respectively. A plot of the configuration (vertical vs. horizontal positions) for the initial and optimal angles can be seen in Fig. 4(a) and Fig. 4(b), respectively, with the impact sequence marked. It can be seen that after the first bounce the robot climbs at a constant increase in the vertical direction.

VII. Conclusion and Future Work

The work presented in this paper formulates an expression for the first derivative of the cost with respect to impact controls for hybrid dynamical systems with impacts. This derivative can be utilized in standard gradient descent methods as shown with the ParkourBot example. It has been shown that second-order methods converge faster to optimal solutions [9]; therefore, the next step in this research will be to derive the second derivative of the cost with respect to the impact controls. Another next step for this research is to embed the optimization on an experimental robot such as the ParkourBot to plan impact controls in real time.

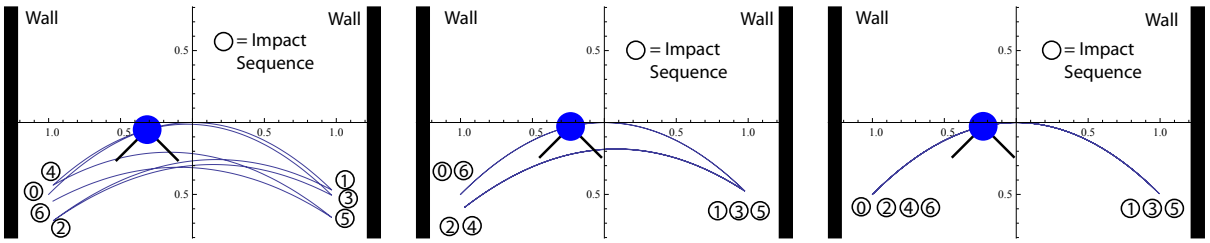


Fig. 3. Bouncing in place with impacts numbered. (a) Trajectory with initial angles. (b) Trajectory with optimal angles for specified final state. (c) Trajectory with optimal angles for trajectory tracking.

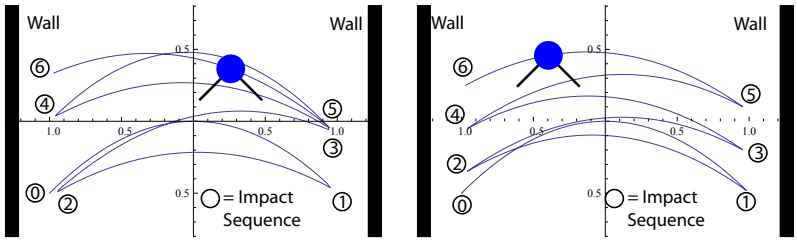


Fig. 4. (a) Climbing with initial angles. (b) Climbing with optimal angles.

VIII. Acknowledgments

The material is based upon work partially supported by the National Science Foundation under Grant IIS-0803734. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The authors gratefully acknowledge Nelson Rosa, Tsahai Phillip, Ben Brown, Amir Degani, and Siyuan Feng for useful discussions about the ParkourBot and hybrid systems with impacts.

Appendix

This Appendix contains the derivation for the derivative of the impact map ($\Delta_k(\cdot)$) and the time of impact ($\tau_k(\cdot)$) given by (11).

Recall

$$\begin{aligned} x_k^- &= x(\xi, \tau_k^-) \\ x_k^+ &= \Delta_k(x_k^-, \xi_k) \\ \tau_k &= \tau_k(\xi_k, x_{k-1}, \tau_{k-1}) \end{aligned}$$

In this derivation, the left hand side of the expressions above will be used as inputs to functions. Since we are using slot derivatives, the right hand side will be used when computing derivatives to explicitly indicate the inputs.

The derivative of the impact map requires chain rule of multiple arguments:

$$\begin{aligned} &D_{\xi_i}\Delta_k(x_k^-, \xi_k) \circ \partial\xi_i = \\ &\quad D_1\Delta_k \circ D_{\xi_i}x(\xi, \tau_k^-) \circ \partial\xi_i + D_2\Delta_k \circ \frac{d\xi_k}{d\xi_i} \end{aligned} \tag{13}$$

The derivative of an impact time with respect to ξ_i is

$$\begin{aligned} D_{\xi_i}\tau_k(\xi_k, x_{k-1}^+, \tau_{k-1}) \circ \partial\xi_i = D_1\tau_k \circ \frac{d\xi_k}{d\xi_i} \\ +D_2\tau_k \circ D_{\xi_i}\Delta_{k-1} \circ \partial\xi_i + D_3\tau_k \circ D_{\xi_i}\tau_{k-1} \circ \partial\xi_i \end{aligned} \tag{14}$$

Each impact control is assumed to be independent of all other impact controls:

$$\frac{d\xi_k}{d\xi_i} = \begin{cases} \partial\xi_i & \text{if } k = i; \\ 0 & \text{if } k \neq i. \end{cases}$$

The derivatives in (13) and (14) depend on the relationship between k and i.

For $k \neq i$, (13) and (14) reduce to

$$D_{\xi_i}\Delta_k(x_k^-, \xi_k) \circ \partial\xi_i = D_1\Delta_k \circ D_{\xi_i}x(\xi, \tau_k^-) \circ \partial\xi_i \tag{15}$$

$$\begin{aligned} D_{\xi_i}\tau_k \circ \partial\xi_i = &D_2\tau_k \circ D_{\xi_i}\Delta_{k-1} \circ \partial\xi_i \\ &+ D_3\tau_k \circ D_{\xi_i}\tau_{k-1} \circ \partial\xi_i. \end{aligned}$$

The above equation can be rewritten in matrix form as

$$D_{\xi_i}\tau_k \circ \partial\xi_i = \begin{bmatrix} (D_2\tau_k)^T & D_3\tau_k \end{bmatrix} \Upsilon_{k-1}^i. \tag{16}$$

The first derivative of x_k^- with respect to each impact control ξ_i is given by

$$\begin{aligned} D_{\xi_i}x(\xi, \tau_k^-) \circ \partial\xi_i = D_\xi x(\xi, \tau_k^-) \circ D_{\xi_i}\xi \circ \partial\xi_i \\ +D_2x(\xi, \tau_k^-) \circ D_{\xi_i}\tau_k \circ \partial\xi_i. \end{aligned} \tag{17}$$

where $D_2x(\xi, \tau) = f(x(\tau), \tau)$.

By considering the independence of each impact control, the first term in (17) reduces to the first slot derivative corresponding to the ith element of ξ, $D_{1,i}x(\xi, \tau) \circ \partial\xi_i$.

$$\begin{aligned} D_{\xi_i}x(\xi, \tau_k^-) \circ \partial\xi_i = D_{1,i}x(\xi, \tau_k^-) \circ \partial\xi_i \\ +f_{k-1}(x_k^-, \tau_k)D_{\xi_i}\tau_k \circ \partial\xi_i, \end{aligned} \tag{18}$$

The first term in (18) can be found by differentiating (6) with respect to the impact controls ξ_i:

$$\begin{aligned} D_{1,i}x(\xi,\tau_k^-)\circ\partial\xi_i &= D_{\xi_i}\Delta_{k-1}(\cdot)\circ\partial\xi_i \\ &\quad -f_{k-1}(x_{k-1}^+,\tau_{k-1})D_{\xi_i}\tau_{k-1}(\cdot)\circ\xi_i \\ &\quad +\int_{\tau_{k-1}^+}^{\tau_k^-}(D_1 f_{k-1}(x(\xi,s),s)\circ D_{1,i}x(\xi,s)\circ\partial\xi_i)ds, \end{aligned}$$

where the second term is from Leibniz Integral rule. No Leibniz term is required for the upper limit (τ_k^-) since the upper limit is taken as a constant in the first slot derivative. By the fundamental theorem of calculus, this can be rewritten in differential form:

$$\begin{aligned} D_{1,i}x(\xi,\tau_{k-1}^+)\circ\partial\xi_i =& D_{\xi_i}\Delta_{k-1}\circ\partial\xi_i \\ &-f_{k-1}(x_{k-1}^+,\tau_{k-1})\circ D_{\xi_i}\tau_{k-1}\circ\partial\xi_i \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial t}D_{1,i}x(\xi,t)\circ\partial\xi_i =& \\ & D_1 f_{k-1}(x(\xi,t),t)\circ D_{1,i}x(\xi,t)\circ\partial\xi_i. \end{aligned}$$

This linear differential equation can be expressed with a state transition matrix operating on an initial condition [13]:

$$D_{1,i}x(\xi,t)\circ\partial\xi_i = \Phi_{k-1}(t,\tau_{k-1}^+)\circ D_{1,i}x(\xi,\tau_{k-1}^+)\circ\partial\xi_i \quad (19)$$

where $\Phi_{k-1}(t,\tau_{k-1})$ is calculated with $A(t) = D_1 f_{k-1}(x(\xi,t),t)$. The initial condition can be written in matrix form as

$$D_{1,i}x(\xi,\tau_{k-1}^+)\circ\partial\xi_i = \begin{bmatrix} I & -f_{k-1}(x_{k-1}^+,\tau_{k-1}) \end{bmatrix}\Upsilon_{k-1}^i \quad (20)$$

where I is the appropriately sized identity matrix.

For $k \neq i$, equation (18) can be written with (16), (19) and (20) as

$$\begin{aligned} D_{\xi_i}x(\xi,\tau_k^-)\circ\partial\xi_i =& \\ \Phi_{k-1}(\tau_k^-,\tau_{k-1}^+)&\begin{bmatrix} I & -f_{k-1}(x_{k-1}^+,\tau_{k-1}) \end{bmatrix}\Upsilon_{k-1}^i \\ +f_{k-1}(x_k^-,\tau_k)&\begin{bmatrix} (D_2\tau_k)^T & D_3\tau_k \end{bmatrix}^T\Upsilon_{k-1}^i. \end{aligned} \quad (21)$$

Equation (21) can be written concisely with α_k and β_k as

$$D_{\xi_i}x(\xi,\tau_k^-)\circ\partial\xi_i = \begin{bmatrix}\alpha_k & \beta_k\end{bmatrix}\Upsilon_{k-1}^i. \quad (22)$$

The derivative of the impact map and the impact time can be expressed in matrix notation by combining (15), (16), and (22) with Γ_k:

$$\Upsilon_k^i = \Gamma_k\Upsilon_{k-1}^i,$$

This is a recursive equation with k monotonically decreasing with each impact. If $k < i$, this equation terminates at $k = 0$. Since the initial condition (Δ_0) and time (τ_0) are given, $D_{\xi_i}\Delta_0\circ\partial\xi_i = 0$ and $D_{\xi_i}\tau_0\circ\partial\xi_i = 0$. This means that $\Upsilon_k^i = [0,0]^T$ for $k < i$, proving (11c).

Now let's consider $k = i$. Equation (11b) is trival. Note that $D_{\xi_i}\tau_{k-1}\circ\partial\xi_i = 0$ and $D_{\xi_i}\Delta_{k-1}\circ\partial\xi_i = 0$ since $k-1 < i$. Equations (13) , (14) and (18) reduce to

$$\begin{aligned} D_{\xi_i}\Delta_k(x_k^-,\xi_k)\circ\partial\xi_i =& \\ D_1\Delta_k\circ D_{\xi_i}x(\xi,\tau_k^-)\circ\partial\xi_i &+ D_2\Delta_k\circ\partial\xi_i \end{aligned} \quad (23)$$

$$D_{\xi_i}\tau_k\circ\partial\xi_i = D_1\tau_k\circ\partial\xi_i \quad (24)$$

$$D_{\xi_i}x(\xi,\tau_k^-)\circ\partial\xi_i = f_{k-1}(x_k^-,\tau_k)D_1\tau_k\circ\partial\xi_i \quad (25)$$

It assumed that we have the explicit forms of Δ_i and τ_i, which we can differentiate with respect to ξ_i. This means that we can calculate Υ_i^i. This means that if $k > i$, Υ_k^i is non-zero and is given by

$$\Upsilon_k^i = \left(\prod_{j=i+1}^{k}\Gamma_{k-j+i+1}\right)\Upsilon_i^i.$$

This proves (11a) and completes the proof.

REFERENCES

[1] A. Degani, S. Feng, H.B. Brown Jr., K. Lynch, H. Choset, and M. Mason, "The ParkourBot - A Dynamic BowLeg Climbing Robot," in *IEEE International Conference on Robotics and Automation*, 2011.

[2] M. Buhler, D. Koditschek, and P. Kindlmann, "Planning and control of a juggling robot," *International Journal of Robotics Research*, vol. 13, no. 2, pp. 101–118, 1994.

[3] K. Lynch and C. Black, "Recurrence, controllability, and stabilization of juggling," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 113–124, 2001.

[4] M. Raibert, H. Brown, and M. Chepponis, "Experiments in balance with a 3D one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, p. 75, 1984.

[5] J. Hodgins and M. Raibert, "Adjusting step length for rough terrain locomotion," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 289–298, 1991.

[6] G. Zeglin and B. Brown, "Control of a bow leg hopping robot," in *IEEE International Conference on Robotics and Automation*. Citeseer, 1998, pp. 793–798.

[7] B. Brogliato and A. Río, "On the control of complementary-slackness juggling mechanical systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 2, pp. 235–246, 2000.

[8] F. Bullo and M. Zefran, "On modeling and locomotion of hybrid mechanical systems with impacts," in *IEEE Conference on Decision and Control*, vol. 3. Citeseer, 1998, pp. 2633–2638.

[9] E. Johnson and T. D. Murphey, "Second-Order Switching Time Optimization for Nonlinear Time-Varying Dynamic Systems," *IEEE Transactions on Automatic Control*, 2010, accepted for Publication.

[10] M. Egerstedt, Y. Wardi, and F. Delmotte, "Optimal control of switching times in switched dynamical systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 110–115, 2006.

[11] M. Travers, L. Pao, and T.D. Murphey, "Data association with impulse optimization," in *IEEE Conference on Decision and Control*. Citeseer, 2010.

[12] C. Kelley, *Iterative methods for optimization*. Society for Industrial Mathematics, 1999.

[13] C. Chen, *Linear system theory and design*. Saunders College Publishing Philadelphia, PA, USA, 1984.

[14] B. Brown and G. Zeglin, "The bow leg hopping robot," in *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings*, vol. 1, 1998.

[15] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific J. Math*, vol. 16, no. 1, pp. 1–3, 1966.