# Continuous-time optimal control of impacting mechanical systems via a projected Hamilton's principle*

Vlad Seghete, *Student Member, IEEE,* Todd D. Murphey, *Member, IEEE*

***Abstract*— In this paper we present a method of generating an optimal controller for impulsive hybrid mechanical systems, such as those undergoing impact. Our goal is for the optimization procedure to incorporate the dynamics of the impact rather than treating it as a disturbance. To this purpose we make use of a projection operator—obtained from a projected version of Hamilton's principle—to build an equivalent switched system that is expressed throughout the state space, including the infeasible regions. This eliminates the discontinuous jumps in velocity of impulsive systems. The approach allows us to apply continuous-time optimization techniques intended for normed function spaces (rather than generalized function spaces) and concretely produces an optimal controller hybrid mechanical system. We developed a Python package that applies the required transformation to simple mechanical systems undergoing impact and implements optimal control methods. Finally, we apply the projection-based technique described to a simple bouncing ball example.**

## I. INTRODUCTION

A major deterrent in designing controllers for impacting mechanical systems is the fact that the resulting trajectories in state space are discontinuous. For example, in the classic problem of a bouncing ball the velocity experiences jumps at the moment of impact. Moreover, the force required at the time of impact is infinite. These facts result in the standard control systems theory only being useful on time intervals away from impact, and thus only useful if treating the impact as introducing a disturbance that needs to be eliminated. On the other hand the theory of hybrid systems has made great progress in recent years [1]–[6] although it is hardly surprising, due to the difficult nature of the problems being tackled, that it has yet to attain the same degree of maturity as classical control theory.

One such example of the aforementioned limitation is the method of infinite dimensional optimization [7], which has been applied successfully to continuous-time systems [8], [9], discrete-time systems [8] and switched systems [10]. What is lacking, however, is an extension of these methods to impulsive systems. In this paper we propose to fill this gap by means of a projection-based method which transforms an impulsive system into an equivalent switched system with continuous trajectories in state space. We do this while preserving the actuator effects on the original system, such that solving the optimal control problem for the smooth, switched system generates, in fact, controllers for the original impacting system. The mechanism behind designing the needed projection and the dynamics associated with it are based on the idea of allowing the system's trajectory access to the full state space instead of restricting it to the feasible configuration set. In effect, we create a virtual system where objects are allowed to pass through obstacles but with dynamics such that after an appropriate non-smooth projection the behavior of the original system is recovered. This idea is a relatively new approach to hybrid systems [6] and our developments are based directly on work done by Pekarek et.al. [11], [12].

The paper is organized in two main sections. The former is designed to introduce the reader to the concept of unfolded and projected dynamics, and also to the notation used in the rest of the paper. We show how the dynamics of the system change under a coordinate transformation designed to flatten the boundary between the feasible and infeasible regions; we introduce index notation for dealing with tensor products and tensor derivatives; we design a projection operator and obtain a new set of system dynamics; and we implement all of the above concepts in the simulation of a bouncing ball example with a sinusoidal floor.

The second main part of the paper introduces a continuous-time optimization mechanism and the changes necessary in order to use this technique in conjunction with the projected dynamics of Sec. II. We present the results of trajectory optimization for the system simulated in Sec. II-D towards a reference trajectory and we comment on the technique's observed convergence rate.

We summarize our results in the Conclusion, as well as propose several new possible research directions for future work.

## II. SIMULATION

In this section we present a method by which an impacting hybrid system can be transformed into a switched system with continuous velocity and simulated as such. While there is no direct advantage to using this method of simulating mechanical system, the results of this section are important both as a proof-of-concept and as a prerequisite for creating optimal controllers of impacting mechanical systems using infinite dimensional optimization methods [7], [8], [10], as we shall see in Sec. III-A.

We will restrict our scope on simple mechanical systems of the form:

$$L(q,\dot{q}) = \frac{1}{2}\dot{q}^{\mathrm{T}}M(q)\dot{q} - V(q)$$

Here $q \in Q$ and $\dot{q} \in TQ$. We also have a region $C \subset Q$ in which configurations are considered feasible. This region is

Vlad Seghete (vlad.seghete@gmail.com) and Todd. D. Murphey (murphey@northwestern.edu) are with the Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Rd. Evanston, IL 60208

bounded and its boundary $\partial C$ is defined by the roots of a function $\phi(q) = 0$. The common convention is that $\phi(q) > 0 \equiv q \in C$. We assume also that our system behaves elastically through all impacts. As such, every time the configuration encounters the boundary at a time $t^*$, the state is updated according to a reset map of the form:

$$\dot{q}(t_*^+) = \dot{q}(t_*^-)\left(\mathbb{I} - 2\frac{M^{-1}\mathrm{D}\phi^{\mathrm{T}}\mathrm{D}\phi}{\mathrm{D}\phi M^{-1}\mathrm{D}\phi^{\mathrm{T}}}\right),$$

where D represents the derivative of a function with respect to its argument and all instances of $\mathrm{D}\phi$ and $M^{-1}$ are evaluated at $q(t_*)$ and $\mathbb{I}$ signifies the identity matrix. The application of the reset map is undesirable, as it creates a discontinuity in the state trajectory and makes it impossible to apply continuous-time optimization techniques as those described in Sec. I.

In order to avoid the use of the reset map altogether, we turn to the technique derived from a modified Hamilton's principle, also known as the projected Hamilton's principle [11], [12]. Instead of using a reset map, we will allow the configuration access to the infeasible parts of the space under the condition it satisfies a certain set of modified dynamics in that region. This will provide a continuous state space trajectory which we then project back into the feasible region using a specialized operator. Essentially, this technique allows us to move the discontinuity in state into a discontinuity in dynamics

### *A. Boundary Flattening Through a Coordinate Transformation*

The method described in [11] assumes a flat boundary between the feasible and unfeasible areas of the configuration space. However, this is rarely the case in practice and as such [12] describes how the method can be extended to any boundary that is homeomorphic to a flat surface. Doing this involves applying a coordinate transformation to our configuration space. We assume through the rest of the paper that our boundaries are homeomorphism to planes and that this homeomorphism is known. Thus, we can build a coordinate transformation such that, in the new coordinates, the boundary becomes a plane aligned with one of the configuration variables.

We assume that, for our system, such a transformation exists. In other words, there exists a diffeomorphism $\Psi : Q \to \overline{Q}$ such that $\phi(\Psi^{-1}(\bar{q})) = \bar{q}_0$, where $\bar{q}_0$ represents the first element of the vector $\bar{q}$. We chose zero-indexing as it is standard in most programming languages and will make implementation of the rest of the math in this document easier. Also, for ease of notation we will name the inverse of psi:

$$\Omega = \Psi^{-1} : \overline{Q} \to Q.$$

Under the above coordinate transformation we obtain that our Lagrangian becomes:

$$L(\bar{q}, \dot{\bar{q}}) = \frac{1}{2}\dot{\bar{q}}^{\mathrm{T}} \cdot \overline{M}(\bar{q}) \cdot \dot{\bar{q}} - \overline{V}(\bar{q})$$

where we have that

$$\overline{M}(\bar{q}) = \mathrm{D}\Omega(\bar{q})^{\mathrm{T}} \cdot M(\Omega(\bar{q})) \cdot \mathrm{D}\Omega(\bar{q}), \tag{1}$$
$$\overline{V}(\bar{q}) = V(\Omega(\bar{q})), \tag{2}$$
$$\bar{\phi}(\bar{q}) = \bar{q}_0, \tag{3}$$
$$\mathrm{D}\bar{\phi}(\bar{q}) = [1, 0, \ldots], \tag{4}$$

where we dropped the dependence on $\bar{q}$ in the more obvious places. Note that one can show, through the use of the chain rule, that $\mathrm{D}\Omega(\bar{q}) = [\mathrm{D}\Psi(q)]^{-1}$. Thus, from now on, we will use this equivalence in order to write clearer and more compact terms.

### *B. Index Notation Summary*

Before the equations get more complicated, we must introduce the Einstein summation notation for tensor product and tensor derivatives, which we will use in the remaineder of the paper. Given a mapping on $\mathbb{R}^n$:

$$T : \mathbb{R}^n \to Y,$$

we indicate by $T_i$ the $i$th element across the first dimension of $T$, by $T_{ij}$ the element indexed by $i$ in the first dimension and by $j$ in the second dimension, and so on. When two tensors are multiplied that share an index it will stand as shorthand for summation over that index:

$$S_{ik} = T_{ij}U_{jk} = \sum_j T_{ij}U_{jk}.$$

Finally, derivatives with respect to the main argument of the tensor mapping will be indicated by a new index after a comma:

$$T_{ij,k} = \frac{\partial T_{ij}(x)}{\partial x_k}$$

To illustrate this notation, let us use it to write (1), (2) and (4) respectively:

$$\overline{M}_{ij} = \Omega_{i,p}M_{pq}\Omega_{q,j}, \tag{5}$$
$$\overline{V}_{,i} = V_{,j}\Omega_{i,j}, \tag{6}$$
$$\phi_{,i} = \delta_{i1}, \tag{7}$$

where $\delta$ represents the Dirac delta function. Note also that, unless otherwise noted, we assume that superscripts are always to be applied *before* indexing. E.g.

$$\overline{M}_{ij}^{-1} = \left(\overline{M}^{-1}\right)_{ij}.$$

The one notable exception to this rule involves the projection $\mathscr{P}$ and its derivative:

$$\mathscr{P}_{i,j}^{-1} = \left[(\mathrm{D}\mathscr{P})^{-1}\right]_{ij},$$

i.e. we will never refer to $\mathscr{P}$'s inverse function, only to the inverse of its derivative in a matrix sense.

### C. Regular and Projected Dynamics

Having established the notation to be used in the remainder of the paper we move on to determining the dynamics in this modified coordinate system. We apply the Euler-Lagrange equations:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\bar{q}}} - \frac{\partial L}{\partial \bar{q}} = 0$$

which gives us

$$\ddot{\bar{q}}_l = \overline{M}^{-1}_{lj}\left(-\dot{\bar{q}}_i \overline{M}_{ij,k}\dot{\bar{q}}_k + \frac{1}{2}\dot{\bar{q}}_i \overline{M}_{ik,j}\dot{\bar{q}}_k + \overline{V}_{,j}\right) \tag{8}$$

In order to calculate the derivative of the mass matrix, we used chain rule and (1) to obtain

$$\overline{M}_{ij,k} = \Omega_{p,ik} M_{pq} \Omega_{q,j} + \Omega_{p,i} M_{pq,k} \Omega_{q,j} + \Omega_{p,i} M_{pq} \Omega_{q,jk}$$

where we assume we have access to the first derivative of the original mass matrix $M_{ij,k}$. We also need to calculate the inverse of $\overline{M}$:

$$\overline{M}^{-1}_{ij} = \Psi_{i,p} M^{-1}_{pq} \Psi_{q,j}$$

Next, we want to find a smooth trajectory $\bar{z}(t) \in \overline{Q}$ and a mapping $\mathscr{P} : \overline{Q} \to \overline{C}$ such that: $\bar{q}(t) = \mathscr{P}(\bar{z}(t))$. That is, we want to allow the system to follow trajectories into the unfeasible region of the configuration space but change the dynamics in that region such that, when projected back into the feasible region, the resulting trajectory follows the original dynamics *and* the impact behavior is elastic. As it turns out, previous work done by Pekarek shows that, the following form of a projection does the trick:

$$\mathscr{P}_i(\bar{z}) = \begin{cases} \bar{z}_i & \text{if } \phi(\bar{z}) > 0 \\ \bar{z}_0 & \text{if } \phi(\bar{z}) \le 0 \text{ and } i = 0 \\ \bar{z}_i - \frac{2\bar{z}_0}{1+k\bar{z}_0^2}\Delta_i, & \text{if } \phi(\bar{z}) \le 0 \text{ and } i \neq 0 \end{cases} \tag{9}$$

$$\Delta_i = \overline{M}^{-1}_{i0} / \overline{M}^{-1}_{00}, \tag{10}$$

where $k$ is a scalar that must be larger than a system dependent lower bound [11]. Notice that this projection is the identity function for all feasible configurations, just as expected, and hence won't influence the dynamics in the feasible region of the configuration space. However, in the unfeasible region the dynamics have to change in order for the trajectory to follow the correct dynamics after the projection. We do this by substituting

$$\bar{q} = \mathscr{P}(\bar{z}), \quad \dot{\bar{q}} = \mathrm{D}\mathscr{P}(\bar{z}) \cdot \dot{\bar{z}} \tag{11}$$

into (8). We obtain the following dynamics:

$$\ddot{\bar{z}}_p = (\mathscr{P}_{p,l})^{-1} (\overline{M}^{-1})_{lj} \left( \frac{1}{2}\dot{\bar{q}}_i \overline{M}_{ik,j}\dot{\bar{q}}_k - \dot{\bar{q}}_i \overline{M}_{ij,k}\dot{\bar{q}}_k + \overline{V}_{,j}\right) - (\mathscr{P}_{p,i})^{-1}\dot{\bar{q}}_j \mathscr{P}_{i,jk}\dot{\bar{q}}_k \tag{12}$$

where all the tensors are evaluated at their corresponding arguments: all instances of $\mathscr{P}$ and its derivatives are evaluated at $\bar{z}$ while everything else is evaluated at $\bar{q}$. In the case where $\bar{\phi}(\bar{z}) \ge 0$ we have that $\bar{z} = \bar{q}$, the first derivative of $\mathscr{P}$ is identity while all the higher derivatives of $\mathscr{P}$ are identically zero. Thus, in this case, (12) reduces to the feasible region dynamics from (8).

In evaluating (12) we make use of everything we derived in the previous section, but we also need to find the terms associated with $\mathscr{P}$. The form of $\mathscr{P}_i$ was shown in (9). Its derivative is given by

$$\mathscr{P}_{i,j} = \begin{cases} \delta_{ij} & \text{if } \phi(\bar{z}) > 0 \\ -\delta_{ij} & \text{if } \phi(\bar{z}) \le 0, i = 0 \\ \delta_{ij} - \Delta_{i,j}\frac{2\bar{z}_0}{1+k\bar{z}^2} & \text{if } \phi(\bar{z}) \le 0, i \neq 0, j \neq 0 \\ -\Delta_{i,j}\frac{2\bar{z}_0}{1+k\bar{z}^2} - 2\Delta_i \frac{1-k\bar{z}_0^2}{(1+k\bar{z}_0^2)^2} & \text{otherwise} \end{cases}$$

where

$$\Delta_{i,j} = \frac{\overline{M}^{-1}_{i0,j}\overline{M}^{-1}_{00} - \overline{M}^{-1}_{i0}\overline{M}^{-1}_{00,j}}{\overline{M}^{-2}_{00}} \tag{13}$$

We also require the second derivative of the projection in order to fully specify the dynamics. Since neither accuracy nor speed of computation are the focus of the current paper, we chose to forego generalizing the previous method to the second derivative. This would have produced a rather large mathematical expression, prone to errors and would have derailed us from the main objective, which is to generate an optimal controller for systems such as we have described. Instead, we used, in turn, and with good results, a finite differencing method and a symbolic package in order to obtain $\mathrm{D}^2\mathscr{P}$. The downside of the first method is loss of accuracy and its poor extensibility to higher derivatives, which we we will see later in Sec. III-A that we need in order to calculate the derivative of the dynamics. Thus, we decided on a symbolic package approach, as detailed in Sec. III-C.

### D. Results

The first step in using the derivations shown in this paper is to run simulations of systems undergoing impact. The problem we chose was that of a point mass in the plane, restricted by a sinusoidal floor:

$$\begin{aligned} q(t) &= [x(t), y(t)]^{\mathrm{T}}, \\ \phi(q) &= y - \sin(x), \\ V(q) &= -mgy, \\ M(q) &= mI, \\ \Psi(q) &= [x, \phi(q)]^{\mathrm{T}}, \\ \Omega(\bar{q}) &= [\bar{q}_1, \bar{q}_2 + \sin(\bar{q}_1)]^{\mathrm{T}} = \Psi^{-1}(\bar{q}) = q. \end{aligned}$$

We then must choose $k$ such that $\mathscr{P}$ as defined in (9) is a valid global projection as per [11]. In the case of our example problem the lower bound is $k = 1$ and any $k > 1$ will suffice. In practice, a choice of $k$ that is too close to the bound will render $\mathscr{P}_{i,j}$ close to singular and hence we will not be able to invert it. Thus, for this problem it is good numerical practice to pick something like $k = 5$ or larger. Note also that too large a $k$ renders a very stiff system at the boundary $\partial C$, while a value of $k$ that is too small gives us bad conditioning of the dynamics below the surface. An optimal choice for $k$ could and should be investigated in the future.

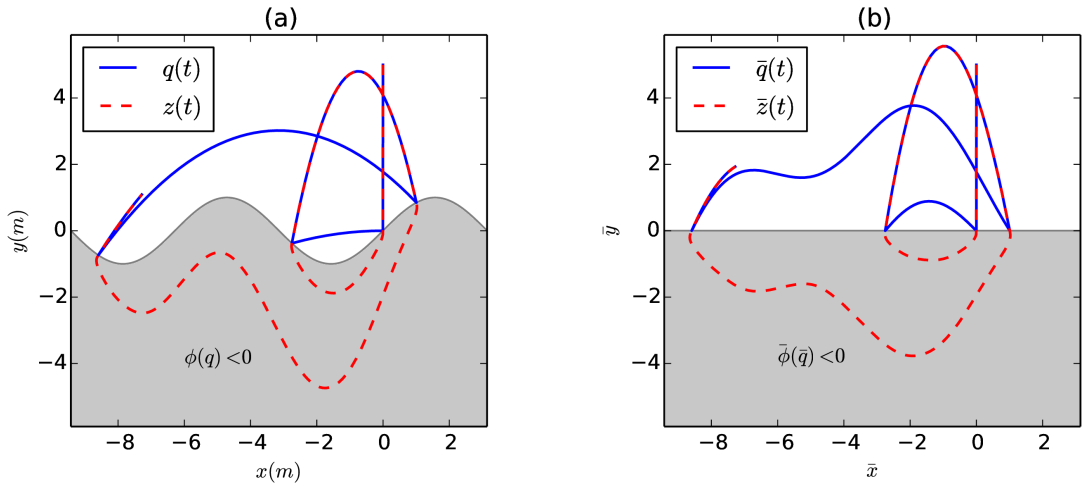


Fig. 1: Trajectory of a point mass interacting elastically with a sinusoidal floor, simulated for 5$s$; (a) shows the trajectory physical configuration space $Q$, as well as the trajectory $z(t)$ in the corresponding unfeasible region; (b) shows the trajectory in the modified space $\overline{Q}$ where the sinusoidal floor has been transformed into a horizontal plane. Notice that in both figures the trajectories before and after the projection overlap when they are in the feasible region. Also note the unusual free flight dynamics in (b)—both in the feasible and in the unfeasible areas—which are the result of applying the coordinate transformation used in obtaining a flat floor.

Given the above information we built a projection $\mathscr{P}$ with parameter $k = 5$ and then numerically solved equation (12) using the libraries provided by the SymPy [13] and NumPy packages in Python. Figure 1 shows the result of the simulation for an example trajectory including several elastic impacts over the course of several seconds. Part (a) of the figure shows the real trajectory $q(t)$ as well as its counterpart in the unfolded space, $z(t) = \Omega(\bar{z}(t))$. Part (b) of the figure shows the transformed space, in which the $\bar{z}(t)$ and $\bar{q}(t) = \mathscr{P}(\bar{z}(t))$ trajectories exist. In this space the boundary is flattened to a line, no longer being sinusoidal. The effect of this is a curvature of the dynamics that did not exist in the original space. A video file of the simulation has also been generated [14] to aid in understanding the concepts presented here.

## III. CONTROL

We propose using the infinite dimensional optimization methods described by Hauser [7] for designing controllers of impacting mechanical system. These methods have already been used for trajectory optimization in continuous mechanical systems [8], [15], [16] and also for a host of hybrid systems [10]. The challenge presented by impulsive systems—which is how impacting mechanisms tend to behave—is that the state trajectory is discontinuous, and thus the methods presented in [7] cannot be used.

To get around this obstacle we make use of the results of the previous section. We essentially use a coordinate transformation and the projected Hamilton's principle in order to re-describe an impulsive mechanical system as a hybrid mechanical system. In other words, we trade a finite impulse applied at the moment of collision with a change in dynamics at that same moment. In the new coordinates and using the new dynamics, our state will be a continuous function of time and thus described by a well-defined (piecewise) nonlinear differential equation. This allows us to consider the space of perturbations to the trajectory and find a descent direction in this space. From there, the methods of [7], [8] can be applied.

### A. Infinite Dimensional Optimization: Overview

The problem we desire to solve is finding the trajectory-control pair $(x(t), u(t))$ on a time interval $[t_a, t_b]$ that minimizes a quadratic cost function representing the distance from a reference trajectory $(x_{ref}(t), u_{ref}(t))$:

$$J(x,u) = \int_{t_a}^{t_b} \left[\bar{x}^{\mathrm{T}} Q_J \bar{x} + \bar{u}^{\mathrm{T}} R_J \bar{u}\right] dt + \bar{x}(t_b)^{\mathrm{T}} F_J \bar{x}(t_b), \quad (14)$$

where, for notational convenience, we dropped the time dependence inside the integral and

$$\bar{x}(t) = x(t) - x_{ref}(t), \quad (15)$$
$$\bar{u}(t) = u(t) - u_{ref}(t), \quad (16)$$
$$\dot{x} = f(x,u). \quad (17)$$

We solve this problem numerically by using a gradient descent method analogous to the classic method on finite dimensional spaces [17]. However, since our space is infinite dimensional—the space of differentiable curves on $(t_a, t_b)$—we need to be careful, especially when we calculate the gradient. Notice also that this we need a constrained optimization algorithm, since the pair $(x,u)$ must conform to the dynamics of the system as defined by the function $f$. Fortunately, previous work explains how to solve this constrained optimization problem in much detail, along with higher order optimization methods [7], [8]. For the scope of this paper we will only provide overview for and use the gradient descent method, and leave the details as well as higher order approaches to future publications.

The gradient descent method consists in finding a descent direction based on the gradient of the cost function we are trying to minimize and then following that direction using

a line search until *sufficient decrease* has been achieved. Sufficient decrease refers to the amount of decrease in the cost function that will guarantee convergence for a locally convex cost function upon iteration of the above steps. One way to guarantee sufficient decrease and thus local convergence to a minimum is to apply an Armijo [18] line search algorithm.

In order to find the steepest descent direction of the cost functional in (14) one needs to find the solution of the dynamics linearized around the point at which the descent direction is required. Let us call this point $(x^*,u^*)$ and let $A(t)=\frac{\partial f}{\partial x}(x^*,u^*)$ and $B(t)=\frac{\partial f}{\partial u}(x^*,u^*)$ be the time-varying matrices obtained from linearizing the dynamics around this trajectory. The descent direction is then found by solving an LQ problem [8] and obtain the pair $(z(t),v(t))$:

$$\begin{aligned}\dot{z} &= Az+Bv,\\ v &= -C_d - K_d z,\\ z(t_a) &= z_a,\end{aligned}$$

where

$$\begin{aligned}K_d &= R_d^{-1}B^{\mathrm{T}}P_d,\\ -\dot{P}_d &= A^{\mathrm{T}}P_d + P_dA - P_dBR_d^{-1}B^{\mathrm{T}}P_d + Q_d,\\ P_d(t_b) &= F_d,\end{aligned}$$

and

$$\begin{aligned}C_d &= R_d^{-1}\left(B^{\mathrm{T}}r + R_J\bar{u}^*\right),\\ -\dot{r} &= \left(A - BR_d^{-1}B^{\mathrm{T}}P_d\right)^{\mathrm{T}}r + Q_J\bar{x}^* - P_dBR_d^{-1}R_J\bar{u}^*,\\ r(t_b) &= F_J\bar{x}^*(t).\end{aligned}$$

The pair $(z(t),v(t))$ obtained as outlined above will give the steepest descent direction of the cost function in (14) at the point $(x^*(t),u^*(t))$.

The above steps are sufficient for solving an unconstrained optimization. However our problem is constrained by the system dynamics, as described by

$$\dot{x} = f(x,u). \tag{18}$$

The extension of the unconstrained procedure involves designing a projection that will transform a curve $(\xi,\mu)$ that does not obey the dynamics into a feasible trajectory:

$$(x,u) = \mathbb{P}((\xi,\mu)), \tag{19}$$

$$\dot{x} = f(x,u). \tag{20}$$

It has been shown [8] that one such projection can be obtained by solving a linear system

$$\dot{x} = Ax+Bu, \tag{21}$$

$$u = \mu - K_p(x-\xi), \tag{22}$$

$$x(t_a) = \xi(t_a), \tag{23}$$

where $A(t)$ and $B(t)$ represent the dynamics linearized around $(x^*,u^*)$; $K_p$ is the full state feedback matrix obtained from solving an LQR problem with weighting matrices $Q_p$, $R_p$ and $F_p$. Note that these matrices can and should be chosen independently from those defining the main cost of the problem, hence the distinct subscript. The solution to the LQR problem is given by

$$K_p = R_p^{-1}B_p^{\mathrm{T}}P_p(t), \tag{24}$$

and $P_p$ is found by solving the continuous time Riccati differential equation:

$$-\dot{P}_p = A^{\mathrm{T}}P_p + P_pA - P_pB_pR_p^{-1}B_p^{\mathrm{T}}P_p + Q_p \tag{25}$$

$$P_p(t_b) = F_p. \tag{26}$$

The projection $\mathbb{P}$ thus designed is used at each step of the line search in order to ensure that the cost function is always evaluated at a feasibly trajectory.

### *B. Jump Terms*

While our trajectories $x(t)$ are continuous they are not smooth. This is due to the discontinuity in dynamics at the time of contact. Thus, we expect the linearized dynamics to present jump terms at the time of impact. The application of the Leibniz rule during differentiation shows us that this is indeed the case.

Consider a system of the form

$$\dot{x} = \begin{cases} f^+(x,u), & \phi(x) > 0 \\ f^-(x,u), & \phi(x) \le 0 \end{cases},$$

a trajectory $(x^*(t),u^*(t))$ that is a solution of the system on the time interval $(t_a,t_b)$ and the cost function $J$ from (14).

Assume, without loss of generality, that at time $t_1$ the distance function $\phi(x(t))$ changes sign. We can then write the cost function as

$$\begin{aligned}J(x,u) = &\int_{t_a}^{t_1}\left(\|\bar{x}\|_{Q_J}^2 + \|\bar{u}\|_{R_J}^2\right)dt\\ &+\int_{t_1}^{t_b}\left(\|\bar{x}\|_{Q_J}^2 + \|\bar{u}\|_{R_J}^2\right)dt + \|\bar{x}(t_b)\|_{F_J}^2.\end{aligned} \tag{27}$$

In order to generate the LQ problem that gives us the descent direction of the cost function we need to take the derivatives of the expression in (27) with respect to $x$ and $u$, evaluated at $(x^*,u^*)$. This process is similar to the case where no impact occurs, except, since the switch time $t_1$ depends on $x(t)$, an extra term appears due to the application of the Leibniz rule. The associated LQ problem to solve in this case is

$$\dot{z} = (A+\delta f_1)z + Bv \tag{28}$$

$$v = -C_d - K_d z, \tag{29}$$

$$\delta f_1 = 2\frac{(f^+ - f^-)\ \mathrm{D}\phi}{|\mathrm{D}\phi\ (f^+ + f^-)|}\delta(t-t_1), \tag{30}$$

$$z(t_a) = z_a, \tag{31}$$

where $\delta f_1$ represents a Kronecker delta function at time $t_1$, its numerator is an outer product and it's denominator is an inner product (a scalar). The expression generalizes as one would expect to more switches at times $t_i$, by adding corresponding $\delta f_i$ terms. One thing to note is the presence of the jump terms $\delta f_i$ makes it clear that there will be a problem when trying to calculate the descent direction from a trajectory that has a grazing impact, as this would

generate a division by zero in $\delta f_i$. Intuitively, this is due to the fact that any infinitesimal perturbation of a grazing trajectory—a trajectory tangent to the impact surface—will generate an infinitely larger change in the time of impact. Such trajectories are, luckily, extremely rare in practice, as we have yet to accidentally generate one. For the scope of this paper we will assume none of the trajectories we discuss are grazing. A proper investigation of this rare but important corner case is thus left as an open question for future work.

### C. Derivative of The Dynamics and Implementation

In Sec. II we calculated the dynamics of the unprojected trajectory as they depend on the boundary function and on the projection. However, in order to apply the methods of Section III-A we also need to linearize the dynamics around trajectories, hence we must have access to the derivatives of $f(x,u)$. One can see from looking at (13) that even just calculating the derivative of $\Delta$ analytically for a general mechanical system will prove difficult. While it is certainly possible to do this, it is beyond the scope of this paper to develop the mechanisms to do so. Instead we choose to focus on showing that the infinite dimensional optimization method can be applied in conjunction with the projected Hamilton's principle. Hence, we decided to use a symbolic mathematics package to aid with calculating such derivatives. We leave the development of faster numerical approaches, such as extending the results of Sec. II or making use of automatic differentiation [19] as topics for future publications.

As stated above we use the SymPy [13] symbolic package together with NumPy and SciPy [20] to symbolically define a mechanical system, consisting of $M(q)$, $V(q)$, $\phi(q)$; and associated forcing $u(t)$; boundary function $\phi(q)$, associated coordinate transformations $\Psi(q) = \bar{q}$ and $\Omega(\bar{q}) = q$; and projection $\bar{q} = \mathscr{P}(\bar{z})$ as defined in (9). Based on these elements and their symbolic derivatives we generate a switched dynamical system defined by ordinary differential equations:

$$\dot{x}(t) = f(x(t), u(t)), \tag{32}$$

where

$$\begin{aligned} x &= \begin{bmatrix} \bar{z} \\ \dot{\bar{z}} \end{bmatrix}, \quad f(x,u) = \begin{bmatrix} \dot{\bar{z}} \\ \ddot{\bar{z}} \end{bmatrix} \\ \bar{q} &= \mathscr{P}(\bar{z}), \quad \dot{\bar{q}} = \mathrm{D}\mathscr{P}(\bar{z})\dot{\bar{z}}, \quad q = \Omega(\mathscr{P}(\bar{z})), \\ \ddot{\bar{z}} &= \left[\mathrm{D}\mathscr{P}(\bar{z})\right]^{-1}\left[g(\bar{z}) + \mathrm{D}\Psi(q)M^{-1}(q)u(t)\right], \end{aligned}$$

and

$$g(\bar{z}) = \overline{M}^{-1}(\bar{q})\left[\dot{\bar{q}}_i\left(\frac{1}{2}\overline{M}_{ik,j}(\bar{q}) - \overline{M}_{ij,k}(\bar{q})\right)\dot{\bar{q}}_k + \mathrm{D}\overline{V}(\bar{q})\right] - \dot{\bar{q}}_i\mathscr{P}_{j,ik}(\bar{z})\dot{\bar{q}}_k.$$

The functions $\overline{M}(\bar{q})$ and $\overline{V}(\bar{q})$ are the same as defined in (1) and (2). The index notation is used to indicate tensor derivatives and dot products where ambiguity might exist otherwise. For a very short overview on the use of this notation see Sec. II-B.

In order to implement the methods described in previous sections we developed a software package based on Python, NumPy, SciPy and SymPy. We named it `nlsymb` [21] as its main function is to provide symbolic tools for use in the construction of nonlinear system model and their analysis. In particular we needed to be able to perform Einstein summations and calculate tensor derivatives in a manner that parallels the notation introduced in Sec. II-B. The package has been made available for use and it is our hope it will facilitate symbolic computations in future work on nonlinear systems inside the free Python environment.

### D. Results

Using the `nlsymb` package we implemented the system model described in Sec. II-D. We then generated a reference trajectory $(x_{ref}, u_{ref})$, where $u$ was picked to be identically $[0,0]$ for the whole time interval and $x_{ref}$ was given by the solution of $\dot{x}_{ref} = f(x_{ref}, u_{ref})$ with $x_{ref}(t_a) = [0,1,0,0]$. The trajectory, integrated for two seconds and transformed back into physical space, can be seen in all three panes of Fig. 2. We then built a cost function $J$ based on the reference trajectory, and using the following weighing matrices in (14):

$$\begin{aligned} R_J &= \mathrm{diag}([10,10]), \\ Q_J &= \mathrm{diag}([10,10,1,1]), \\ F_J &= Q_J, \end{aligned}$$

where the diag operator takes vectors into diagonal matrices with the vector elements on the diagonal. All other weighing matrices needed in the equations of Sec. III-A were picked to be identity. The choice for $z_a = 0$ implies that we did not allow changes in the initial condition of our initial guess.

We then picked an initial trajectory which from which to start our optimization, i.e. an initial guess. We already know, from having built $J$ using a feasible trajectory as reference, that the reference trajectory is also the minimum we are looking for. Since our goal is to test our technique, we expect to see linear convergence from a nearby point in the optimization space. We picked the initial guess to be a disturbance of our reference, so that we could make sure we never step out of the convex region of the function around its minimum. The disturbance was achieved by setting the forcing to $u = [1,0]$ for all time, but otherwise using the same parameters and initial conditions as the reference trajectory. The initial guess can be seen in Fig. 2(a), before any iterations took place.

Finally, we applied the trajectory optimization method to the initial trajectory until convergence was achieved. Figures 2(b) and 2(c) show the progress after one and after ten iterations of the optimization, respectively. Figure 3 shows the cost $J$ and also the norm of the gradient $\nabla J$ as a function of iteration. The rate of convergence is linear for the first few iterations after which little extra progress can be made.

## IV. CONCLUSION AND FUTURE WORK

In the first part of this paper we have obtained the differential equations governing a switched system such

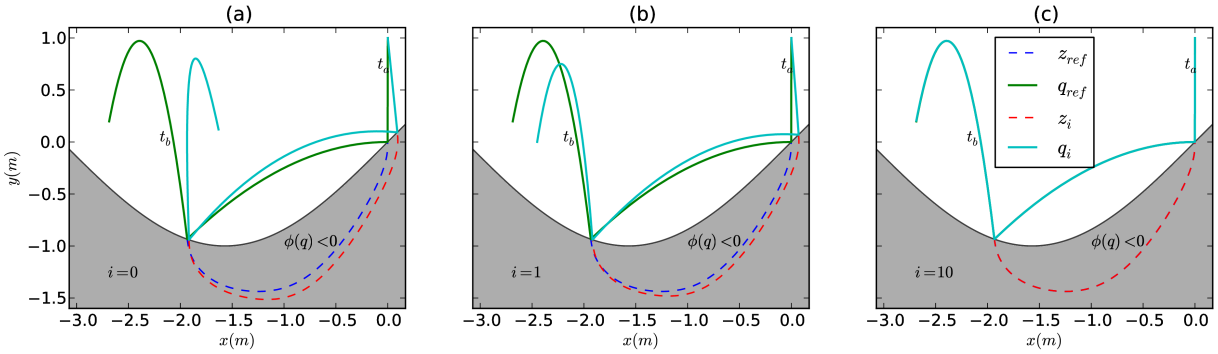


Fig. 2: Reference trajectory together with results at different stages of the optimization loop. Both the reference trajectory and the initial guess share the initial condition at time $t_a$ and end at time $t_b$. The initial guess is shown in (a), before any steps were taken; (b) shows the trajectory after one iteration, only somewhat closer to the reference; in (c) we see that, after tne iterations, the optimized trajectory is visually indistinguishable from the reference trajectory.

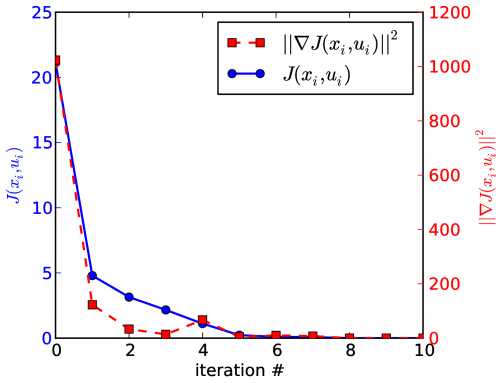


Fig. 3: Plot illustrating the convergence rate of the algorithm. Both the cost function $J$ and the norm of the gradient $\nabla J$ drop at a steady rate with iteration number until convergence to a local minimum is achieved and progress halts.

that, after the application of a proper projection operator on the trajectories that solve it, we obtain the solution to an equivalent impulsive mechanical system—a system undergoing impact. In the latter part of this paper we used the projection method to apply a continuous-time optimal control technique to the modified system and thus obtaining an optimal controller for the original system. We implement our techniques numerically using the Python programming language and present the optimization results as well as the associated convergence plots.

## ACKNOWLEDGEMENT

We wish to acknowledge significant help from David Pekarek without whom this work would not have been possible.

This material is based upon work supported by the National Science Foundation under Grant CMMI 1200321. Any opinions, ndings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reect the views of the National Science Foundation.

## REFERENCES

[1] J. P. Hespanha, D. Liberzon, and A. R. Teel, "Lyapunov conditions for input-to-state stability of impulsive systems," *Automatica*, vol. 44, no. 11, pp. 2735–2744, Nov. 2008.

[2] A. Potini, "Trajectory tracking in switched systems: an internal model principle approach: the elliptical billiard system as a benchmark for theory," Tesi di dottorato, Universit di Roma Tor Vergata, May 2008.

[3] Y. Or and A. D. Ames, "Formal and practical completion of lagrangian hybrid systems," in *Proc. American Control Conference*, June 2009, pp. 3624–3631.

[4] B. Laurent, "Identification of switched linear systems via sparse optimization," *Automatica*, vol. 47, no. 4, pp. 668–677, Apr. 2011.

[5] J. Liu, X. Liu, and W.-C. Xie, "Input-to-state stability of impulsive and switching hybrid systems with time-delay," *Automatica*, vol. 47, no. 5, pp. 899–908, May 2011.

[6] F. Forni, A. Teel, and L. Zaccarian, "Follow the bouncing ball: Global results on tracking and state estimation with impacts," *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1470–1485, 2013.

[7] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *IFAC world congress*, 2002.

[8] E. Johnson, "Trajectory optimization and regulation for constrained discrete mechanical systems," Ph.D., Northwestern University, United States – Illinois, 2012, 00000.

[9] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *American Control Conference (ACC), 2013*, 2013, pp. 4196–4201.

[10] T. Caldwell and T. Murphey, "Switching mode generation and optimal estimation with application to skid-steering," *Automatica*, vol. 47, no. 1, pp. 50–64, Jan. 2011.

[11] D. Pekarek and T. Murphey, "Variational nonsmooth mechanics via a projected hamilton's principle," in *American Control Conference (ACC), 2012*, 2012, pp. 1040–1046.

[12] D. Pekarek and T. D. Murphey, "Global projections for variational nonsmooth mechanics," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2012, p. 55725579.

[13] SymPy Development Team, *SymPy: Python library for symbolic mathematics*, 2013. [Online]. Available: http://www.sympy.org

[14] V. Seghete and T. D. Murphey, "Bouncing ball falls through the floor," 2013. [Online]. Available: http://vimeo.com/75709838

[15] E. Johnson, "trep," Apr. 2012. [Online]. Available: trep.sourceforge.net

[16] J. Schultz and T. Murphey, "Trajectory generation for underactuated control of a suspended mass," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, p. 123129.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[18] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives." *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[19] G. Corliss, *Automatic differentiation of algorithms: from simulation to optimization*. Springer, 2002.

[20] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[21] V. Seghete, "nlsymb: Symbolic nonlinear system tools for Python," 2013–. [Online]. Available: http://github.com/vlsd/nlsymb