Local Planning Using Switching Time Optimization

Elliot R. Johnson and Todd D. Murphey

Abstract— **Switching-time optimization has applications in local motion planning using the geometry of the nonlinear vector fields that govern the control system. In this paper, we present an algorithm for computing the second derivative of a switching-time cost function that enables second-order numerical optimization techniques that often converge quickly compared to first-order only algorithms. The resulting algorithms (for both first and second derivatives) each require only a single integration along the time horizon, yielding excellent computational performance. We present an example that uses this method to do local motion planning for a parallel parking maneuver for a kinematic car using the infinitesimal Lie bracket expansion that is used to demonstrate controllability. This same expansion allows one to construct a sequence of motions and approximate switching times that can then be used in the switching time optimization for a finite (non-infinitesimal) motion.**

I. INTRODUCTION

Switched dynamic systems discontinuously switch from one dynamic function to the next in a known sequence as certain switching times are reached. Each dynamic function (i.e. $\dot{x} = f(x, u, t)$) is itself continuous and differentiable. Hybrid systems can be represented as switched mechanical systems when the ordering of the active states (or modes) are known. Switching time optimization is the problem of determining a set of switching times that minimize a cost function. For example, we can design a cost function whose minimizer will be the best approximation of a desired trajectory.

Geometric planning tools for nonlinear systems often represent local planning tasks as switched systems where the switching arises from turning on and off vector fields that generate the Lie algebra. This paper provides the tools necessary to optimize over the switching times so that the infinitesimal motion provided by the Lie bracket motion can be turned into a finite motion that is feasible for the real system.

In this paper, we consider autonomous dynamic systems which have no control input (i.e. $\dot{x} = f(x, t)$). These optimizations are implemented with standard iterative, numerical algorithms [17][18] (e.g. the Steepest descent algorithm). The focus of research has been algorithms to calculate the derivatives of a cost function with respect to the switching times [12], [4], [2], [1], [11], [20], [5], [14], [13], [16], [23]. In general, these works have focused specifically on calculating first derivatives [12], [5], [14], [13], [16], though some have also discussed (but not computed) the second derivative [11], [20], [23]. Switching time optimization is useful for system identification problems as well. It is used in [7] to determine the slip-state (i.e. which wheels have traction and which do not) of a slip-steered vehicle. In that particular case, the authors demonstrate significant performance benefits by implementing a second-order optimization. Lastly, switching time optimization techniques typically assume that mode order is known, whereas [4], [2], [1], [16], [8] focus on optimizing over mode order as well.

The work in [12], [5], [14], [13], [16] represents the state-of-the-art for calculating the first derivative. In that work, the authors present an elegant algorithm to calculate the derivative with respect to every switching time with a single backwards integration of a differential equation that is independent of the number of switching times. The algorithm is developed with constrained optimization techniques that use Lagrangian multipliers.

This paper uses a different approach to find the same result as [12] that involves fewer steps and relies on fundamental principles in calculus instead of constrained Lagrange multiplier techniques. More importantly, the derivation generalizes to find the second derivative of the cost function using an equally simple single integration strategy.

Optimizations that use only the first derivative are restricted to the steepest descent algorithm and achieve linear convergence. The second derivative enables implementations of Newton's Method that have quadratic convergence. The difference between convergence rates are often significant in practice [7], [8]. We present a local planning example that compares both methods and demonstrates the importance of using the second derivative in the optimization.

Section II establishes the precise problem definition. Sections III and IV derive the first derivatives of the trajectory and cost function, respectively. The same technique is extended to find the second derivatives of the trajectory and cost in Sec. V and VI. Section VIII applies the techniques to a planning example where the Lie bracket is used to determine mode order and approximate switching times.

This work was supported by the National Science Foundation under grant IIS-0940166.

`elliot.r.johnson@u.northwestern.edu`
`t-murphey@northwestern.edu`
Mechanical Engineering, McCormick School of Engineering at Northwestern University, Evanston, IL

A significantly expanded version of this paper is under consideration in the IEEE Transactions on Automatic Control. That paper includes all the proofs of the results as well as additional examples. This work has not been published any place else, nor is it under consideration anywhere else.

II. Problem Definition

Consider an n-dimensional non-linear system governed by a sequence of N dynamic models:

$$\frac{d}{dt}[x(t)] = f(x,t) = \begin{cases} f_1(x,t) & \tau_1 \leq t < \tau_2 \\ f_2(x,t) & \tau_2 \leq t < \tau_3 \\ \vdots & \\ f_N(x,t) & \tau_N \leq t < \tau_{N+1} \end{cases} \tag{1}$$

with $\tau_1 = t_0$ and $\tau_{N+1} = t_f$ defining the time horizon and with initial condition $x(t_0) = x_0$. Each $f_k(x,t)$ is at least C^2 in x.

We seek the $N-1$ switching times[1] $\tau_2 \dots \tau_N$ that optimize a total cost:

$$J(\tau_2, \tau_3 \cdots \tau_N) = \int_{t_0}^{t_f} \ell(x(t),t)\,\mathrm{d}t \tag{2}$$

where $\ell(x,t)$ is an arbitrary C^2 (in x) incremental cost function chosen for a specific problem. For example we might choose

$$\ell(x,t) = (x - x_d(t))^T (x - x_d(t))$$

to find the switching times that result in the best possible tracking of the desired trajectory x_d.

The optimization problem is approached using standard iterative numeric algorithms (i.e. Gradient-descent, Newton's Method). The mathematical problem considered in this paper is how to calculate the first and second derivatives of the cost function needed to implement these algorithms in a planning task. Before we find the first derivative, we mention several conventions used throughout this paper.

A. Notation

The most important notational point for this paper is that we abbreviate a trajectory as $x(t)$ when strictly it should be $x(x_0, \tau_1, \tau_2 \cdots \tau_N, t)$. This is a crucial point to remember when taking derivatives of $x(\cdot)$ with respect to a switching time τ_i since a switching time may also be an argument as the time parameter.

For any time-dependent function $y(t)$, we refer to a segment $y_k(t)$ to be $y(t)\,\forall\, t \in [\tau_k, \tau_{k+1})$. Note that, when there is continuity, $y_k(\tau_{k+1}) = y_{k+1}(\tau_k)$.

We prefer to think of linear and bilinear maps, especially derivatives, as operators and write $M \circ U$ to mean "the linear operator M applied to U". For finite dimensional linear operators, we use square brackets $[M]$ to indicate matrix representations (e.g. $M \circ U = [M]U$ and $M \circ (U,V) = U^T[M]V$).

We use $Df(x)$ notation for derivatives. $D_n f(arg_1, arg_2, ...) \circ (\partial arg_N)$ represents the derivative of $f(\cdot)$ with respect to the n-th argument. This is called the slot derivative. Finally, $D_{var} f(arg_1, arg_2, \dots) \circ (\partial var)$ is the derivative of $f(\cdot)$ with respect to the variable var.

[1]Having τ_2 be the first switching time is awkward, but otherwise the first trajectory will be $x_0(t)$, which clashes with the conventional notion that x_0 is a constant initial condition.

III. First Derivative of $x(t)$

The first derivative of the cost function, $D_{\tau_i} x(t) \circ \partial\tau_i$, involves the derivatives of the trajectory $x(t)$ with respect to each switching time $\tau_i \,\forall\, i = 2 \cdots N$.

Lemma 3.1:

$$D_{\tau_i} x(t) \circ \partial\tau_i = \begin{cases} 0 & t < \tau_i \\ \Phi(t,\tau_i) \circ X^i & t \geq \tau_i \end{cases} \tag{3}$$
$$X^i = \big(f_{i-1}(x(\tau_i),\tau_i) - f_i(x(\tau_i),\tau_i)\big)\partial\tau_i$$

where $\Phi(t,\tau)$ is the state transition matrix for the state system $\dot{x} = A(t)x$ with $A(t) = [D_1 f(x(t),t)]$.

Proof: Use the fundamental theorem of calculus with (1) and continuity of $x(t)$ to express each segment of the trajectory in integral form.

$$x_0(t) = x_0$$
$$x_k(t) = x_{k-1}(\tau_k) + \int_{\tau_k}^{t} f_k(x(s),s)\,\mathrm{d}s \tag{4}$$

Derivatives of $x_0(t)$ are clearly zero and will not be explicitly mentioned for the rest of the discussion. Take the derivative of (4) with respect to τ_i.

$$\begin{aligned} D_{\tau_i} x_k(t) \circ \partial\tau_i &= D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \\ &\quad + D_t x_{k-1}(\tau_k) \circ \tfrac{d\tau_k}{d\tau_i} - f_k(x_k(\tau_k),\tau_k)\tfrac{d\tau_k}{d\tau_i} \\ &\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s) \circ D_{\tau_i} x_k(t) \circ \partial\tau_i\,\mathrm{d}s \\ &= D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \\ &\quad + f_{k-1}(x_{k-1}(\tau_k),\tau_k)\tfrac{d\tau_k}{d\tau_i} - f_k(x_k(\tau_k),\tau_k)\tfrac{d\tau_k}{d\tau_i} \\ &\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s) \circ D_{\tau_i} x_k(s) \circ \partial\tau_i\,\mathrm{d}s \end{aligned} \tag{5}$$

where the third term appears from the Leibniz Integral rule and

$$\frac{d\tau_k}{d\tau_i} = \begin{cases} \partial\tau_i & k = i \\ 0 & k \neq i. \end{cases}$$

This is the intuitive notion that the derivative of an independent variable with respect to itself is the identity, and with respect to any other independent variable is zero.

Use the fundamental theorem of calculus to express (5) in differential form.

$$\begin{aligned} D_{\tau_i} x_k(\tau_k) \circ \partial\tau_i &= D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \\ &\quad + f_{k-1}(x_{k-1}(\tau_k),\tau_k)\tfrac{d\tau_k}{d\tau_i} - f_k(x_k(\tau_k),\tau_k)\tfrac{d\tau_k}{d\tau_i} \\ \tfrac{\partial}{\partial t} D_{\tau_i} x_k(t) \circ \partial\tau_i &= D_1 f_k(x_k(t),t) \circ D_{\tau_i} x_k(t) \circ \partial\tau_i \end{aligned}$$

We have a linear differential equation that is the same form for all k and i. Solutions to linear differential equations can be represented by a state transition matrix operating on an initial condition [9]

$$D_{\tau_i} x_k(t) \circ \partial\tau_i = \Phi_k(t,\tau_k) \circ D_{\tau_i} x_k(\tau_k) \circ \partial\tau_i \tag{6}$$

where $\Phi_k(t,\tau)$ is the state transition matrix for the linear system with $A(t) = [D_1 f(x_k(t),t)]$.

The initial conditions, on the other hand, for the differential are dependent on the relationship between k and i. For $k < i$:

$$\begin{aligned} D_{\tau_i} x_k(\tau_k) \circ \partial\tau_i &= D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \\ &= \Phi(\tau_k, \tau_{k-1}) \circ D_{\tau_i} x_{k-1}(\tau_{k-1}) \circ \partial\tau_i \end{aligned}$$

This is a recursive equation, with k decreasing with each recursion. Since $k < i$, it will terminate with $k = 0$ which is clearly 0 from (6). Therefore the initial condition will be zero for each $k < i$ and so $D_{\tau_i} x(t) = 0$ for $t < \tau_i$.

For $k = i$ we find

$$D_{\tau_i} x_k(\tau_k) \circ \partial\tau_i = f_{k-1}(x_{k-1}(\tau_k), \tau_k)\partial\tau_i - f_k(x(\tau_k), \tau_k)\partial\tau_i$$

For $k > i$ we again find

$$\begin{aligned} D_{\tau_i} x_k(\tau_k) \circ \partial\tau_i &= D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \\ &= \Phi(\tau_k, \tau_{k-1}) \circ D_{\tau_i} x_{k-1}(\tau_k) \circ \partial\tau_i \end{aligned}$$

In this case, the k towards i decreases with each recursion, terminating on $k = i$. As a result, we have a continuous flow along the differential equation from the initial condition at $t = \tau_i$ as stated in the Lemma. This continuity allows us to drop the k subscript and consider the derivative (for $t \geq \tau_i$) as a single trajectory. ■

The state transition matrix of a linear system has several well-known [9] properties:

$$\Phi(t, t) = I \tag{7a}$$

$$\tfrac{d}{dt}\Phi(t, \tau) = A(t) \circ \Phi(t, \tau) \tag{7b}$$

$$\tfrac{d}{d\tau}\Phi(t, \tau) = -\Phi(t, \tau) \circ A(\tau) \tag{7c}$$

$$\Phi(t, \tau) = \Phi(t, s) \circ \Phi(s, \tau) \tag{7d}$$

We use of these identities in the following section to derive the first derivative of the cost function $J(\cdot)$.

IV. First Derivative of $J(\cdot)$

The first derivative of $J(\cdot)$ is calculated using one of two approaches. The first comes from directly differentiating the cost function and integrating forward in time. The second, which is equivalent to the results in [12] arises by trivially modifying the first method to integrate backwards in time rather than forwards. This trivial change produces significant improvements in the computational effort required to calculate the first derivative.

Lemma 4.1: The derivative of the cost (2) with respect to each switching time τ_i is

$$D_{\tau_i} J(\cdot) \circ \partial\tau_i = \psi(t_f, \tau_i) \circ X^i \tag{8}$$

where $\psi(t_f, \tau) : \mathbb{R}^n \to \mathbb{R}$ is found by integrating

$$\psi(t, t) \circ U = 0 \tag{9a}$$

$$\begin{aligned} \tfrac{\partial}{\partial\tau}\psi(t, \tau) \circ U = &-D_1\ell(x(\tau), \tau) \circ U \\ &- \psi(t, \tau) \circ D_1 f(x(\tau), \tau) \circ U \end{aligned} \tag{9b}$$

backwards along τ from t_f to τ_i

Proof: Take the derivative of (2). The resulting integrand from t_0 to τ_i is zero because $D_{\tau_i} x(t) = 0$ for $t < \tau_i$, leaving us with

$$D_{\tau_i} J(\cdot) \circ \partial\tau_i = \int_{\tau_i}^{t_f} D_1\ell(x(s), s) \circ D_{\tau_i} x(s) \circ \partial\tau_i \,\mathrm{d}s \tag{10}$$

Substitute (3) into the above and recognize that X^i is independent of the variable of integration, s and can be pulled outside the integration by linearity of the integral.

$$D_{\tau_i} J(\cdot) \circ \partial\tau_i = \left(\int_{\tau_i}^{t_f} D_1\ell(x(s), s) \circ \Phi(s, \tau_i) \,\mathrm{d}s \right) \circ X^i \tag{11}$$

Define the linear operator $\psi(t, \tau)$ to represent the expression in parentheses.

$$\psi(t, \tau) \circ U = \left(\int_{\tau}^{t} D_1\ell(x(s), s) \circ \Phi(s, \tau) \,\mathrm{d}s \right) \circ U \tag{12}$$

Substituting (12) into (11) results in the first part of the Lemma, (8).

The above provides a complete set of equations to calculate $D_{\tau_i} J(\cdot)$, but we must integrate $\psi(t_f, \tau_i)$ (which also needs $\Phi(t, \tau)$) for each[2] τ_i.

Remember that the purpose of $\psi(t, \tau)$ is to evaluate the cost derivative (8) which requires $\psi(t_f, \tau_2) \cdots \psi(t_f, \tau_N)$. If we think of the integration (12) as a forward differential equation in t, we are finding $\psi(t, \tau)$ for (infinitely) many values of t and a single value of τ. However, we only need $\psi(t, \tau)$ for a single value of t (namely, t_f) and many values of τ (namely, $\tau_2 \cdots \tau_N$). Differentiating (12) with respect to τ and integrating backwards from t_f to τ_2 will find all of our values of $\psi(t_f, \tau_i)$ in a single integration.

Evaluate (12) with $\tau = t$ to find the initial condition (9a) for the integration and differentiate (12) with respect to τ:

$$\begin{aligned} \tfrac{\partial}{\partial\tau}\psi(t, \tau) \circ U = &-D_1\ell(x(\tau), \tau) \circ \Phi(\tau, \tau) \circ U \\ &- \int_{\tau}^{t} D_1\ell(x(s), s) \circ \Phi(s, \tau) \circ A(\tau) U \,\mathrm{d}s \\ = &-D_1\ell(x(\tau), \tau) \circ U \\ &- \left(\int_{\tau}^{t} D_1\ell(x(s), s) \circ \Phi(s, \tau) \,\mathrm{d}s \right) \circ A(\tau) U \\ = &-D_1\ell(x(\tau), \tau) \circ U - \psi(t, \tau) \circ D_1 f(x(\tau), \tau) \circ U \end{aligned} \tag{13}$$

where the first term comes from the Leibniz integral rule and we have used the identities from (7). This proves the final statement of the Lemma. ■

Lemma 4.1 is a useful result, particularly because it does not involve the state transition matrix $\Phi(t, \tau)$. This further reduces the computational effort required to calculate the first derivative.

This result has previously been reported by [12], where it was derived using multiplier methods. Here we have only used basic derivative rules. The derivation has fewer

[2]This can be reduced to a single integration by finding each $\psi(t_{\tau_{k+1}}, \tau_k)$ using appropriate linear compositions. However, that method still requires more computational effort (integrating an $n \times n$ matrix) and algorithmic complexity (more linear compositions and addition) than the result from backwards integration.

steps and additionally provides a way to calculate (12) with forward integration if desired. Most importantly, this approach *naturally extends to the second derivative*.

V. Second Derivatives of $x(t)$

The second derivative of $x(t)$ is found using the same strategy from Sec. III. We find a differential equation describing the second derivative and then show that the solutions can be expressed with a state transition matrix and an analogous bilinear operator. We also derive two identities (Corollary 5.3) of the new operator that will be useful for the second derivatives of the cost.

The second derivative of $x(t)$ is symmetric (i.e, mixed partials commute), so *we assume* $\boldsymbol{i \geq j}$ for the remainder of the paper without loss of generality. This is only for brevity; the same strategy will find the full second derivative without assuming symmetry a priori.

Proposition 5.1: With $i \geq j$ (and $t \geq \tau_i$), the second derivative of the trajectory satisfies a differential equation (14a) with initial condition (14b).

$$\begin{aligned}\tfrac{d}{dt} D_{\tau_j} D_{\tau_i} x(t) \circ (\partial\tau_j, \partial\tau_i) = \\ D_1 f(x(t),t) \circ D_{\tau_j} D_{\tau_i} x(t) \circ (\partial\tau_j, \partial\tau_i) \\ + D_1^2 f(x(t),t) \circ (D_{\tau_j} x(t) \circ \partial\tau_j, D_{\tau_i} x(t) \circ \partial\tau_i)\end{aligned} \tag{14a}$$

$$\begin{aligned} D_{\tau_j} D_{\tau_i} x(\tau_i) \circ (\partial\tau_j, \partial\tau_i) = \\ \begin{cases} \textit{For } i=j\textit{:} \\ \quad D_1 f_i(x(\tau_i),\tau_i) \circ f_i(x(\tau_i),\tau_i) \partial\tau_j \partial\tau_i \\ \qquad + D_1 f_{i-1}(x(\tau_i),\tau_i) \circ f_{i-1}(x(\tau_i),\tau_i) \partial\tau_j \partial\tau_i \\ \qquad - 2 D_1 f_i(x(\tau_i),\tau_i) \circ f_{i-1}(x(\tau_i),\tau_i) \partial\tau_j \partial\tau_i \\ \qquad + D_2 f_{i-1}(x(\tau_i),\tau_i) \circ \partial\tau_j \partial\tau_i \\ \qquad - D_2 f_i(x(\tau_i),\tau_i) \circ \partial\tau_j \partial\tau_i \\ \textit{For } i>j\textit{:} \\ \quad \big(D_1 f_{i-1}(x(\tau_i),\tau_i) - D_1 f_i(x(\tau_i),\tau_i)\big) \circ \\ \qquad \Phi(\tau_i,\tau_j) \circ X^j \partial\tau_i \end{cases}\end{aligned} \tag{14b}$$

Proof: Use the same technique as in the proof of Lemma 3.1.[3] Differentiate (5) and apply the fundamental theorem of calculus. Consider each combination of $k \gtreqless i$ and $k \gtreqless j$ to find the individual initial conditions. This is straightforward and not reproduced here. ■

Unlike the first derivative, the ODE for the second derivative (14a) is not linear, but it is *affine*. If we think of the affine term as an input, the ODE can be modeled as a *forced linear system*.

We can use the forced linear system form to express the second derivative in terms of a state transition matrix and a new bilinear operator, $\phi(t,\tau)$ that is analogous to $\Phi(t,\tau)$, the first order state transition matrix.

Lemma 5.2: The second derivative of the trajectory, $D_{\tau_j} D_{\tau_i} x(t) \circ (\partial\tau_j, \partial\tau_i)$, is

$$\begin{aligned} D_{\tau_j} D_{\tau_i} x(t) \circ (\partial\tau_j, \partial\tau_i) = \\ \Phi(t,\tau_i) \circ X^{i,j} + \phi(t,\tau_i) \circ \big(\Phi(\tau_i,\tau_j) \circ X^j, X^i\big)\end{aligned} \tag{15}$$

[3]See the journal submission *http://robotics.mech.northwestern.edu/~murphey/murphey-TAC2009sub.pdf* for the complete proof.

where $\Phi(t,\tau)$ is the state transition matrix in Lemma 3.1 and $\phi(t,\tau) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is the bilinear operator defined as

$$\begin{aligned}\phi(t,\tau) \circ (U,V) = \\ \int_\tau^t \Phi(t,s) \circ D_1^2 f(x(s),s) \circ (\Phi(s,\tau) \circ U, \Phi(s,\tau) \circ V)\end{aligned} \tag{16}$$

and $X^{i,j}$ is the initial condition from (14b).

Proof: The solution for a forced linear system is

$$x(t) = \Phi(t,t_0) \circ x_0 + \int_{t_0}^t \Phi(t,s) \circ B(s)\, \mathrm{d}s$$

Treating (14a) as a forced linear system, the solutions become

$$\begin{aligned} & D_{\tau_j} D_{\tau_i} x(t) \circ (\partial\tau_j, \partial\tau_i) \\ &= \Phi(t,\tau_i) \circ X^{i,j} + \int_{\tau_i}^t \Phi(t,s) \circ D_1^2 f(x(s),s) \circ \\ &\qquad (D_{\tau_j} x(s) \circ \partial\tau_j, D_{\tau_i} x(s) \circ \partial\tau_i)\, \mathrm{d}s \\ &= \Phi(t,\tau_i) \circ X^{i,j} + \int_{\tau_i}^t \Phi(t,s) \circ D_1^2 f(x(s),s) \circ \\ &\qquad \big(\Phi(s,\tau_j) \circ X^j, \Phi(s,\tau_i) \circ X^i\big)\, \mathrm{d}s \\ &= \Phi(t,\tau_i) \circ X^{i,j} + \int_{\tau_i}^t \Phi(t,s) \circ D_1^2 f(x(s),s) \circ \\ &\qquad \big(\Phi(s,\tau_i) \circ \Phi(\tau_i,\tau_j) \circ X^j, \Phi(s,\tau_i) \circ X^i\big)\, \mathrm{d}s \\ &= \Phi(t,\tau_i) \circ X^{i,j} + \phi(t,\tau_i) \circ \big(\Phi(\tau_i,\tau_j) \circ X^j, X^i\big)\end{aligned}$$

where we've taken advantage of the linearity of the integral with respect to a independent variables to pull out X^i and X^j. ■

As stated earlier, $\phi(t,\tau)$ is a bilinear/second-order analogy to the state transition matrix. The identities (7) for state transition matrix allowed us to simplify the first derivative calculation. We derive two similar properties for $\phi(t,\tau)$.

Corollary 5.3: For $\phi(t,\tau) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ defined in (16), the following identities hold.

$$\phi(t,t) \circ (U,V) = 0 \tag{17a}$$

$$\begin{aligned}\tfrac{\partial}{\partial\tau} \phi(t,\tau) \circ (U,V) = \\ -\Phi(t,\tau) \circ D_1^2 f(x(\tau),\tau) \circ (U,V) \\ -\phi(t,\tau) \circ \big(D_1 f(x(\tau),\tau) \circ U, V\big) \\ -\phi(t,\tau) \circ \big(U, D_1 f(x(\tau),\tau) \circ V\big)\end{aligned} \tag{17b}$$

Proof: Both properties follow directly from (16). ■

Corollary 5.3 enables the same strategy from Sec. IV for the second derivative of the cost.

VI. Second Derivative of $J(\cdot)$

The derivation of the second derivative is similar to the first derivative. We find a forward integration method for the calculation that is sufficient but computationally expensive. The integration is replaced with with a backwards differential equation in time that results in a much improved algorithm. In particular, we avoid computing $\phi(t,\tau)$.

Theorem 6.1: The second derivative with respect to switching times τ_j and $\tau_i \geq \tau_j$ is

$$\begin{aligned} D_{\tau_j} D_{\tau_i} J(\cdot) \circ (\partial\tau_j, \partial\tau_i) = & \qquad (18) \\ - D_1\ell(x(\tau_i), \tau_i) \circ X^i \partial\tau_j \delta_i^j + \psi(t_f, \tau_i) \circ X^{i,j} \\ + \Omega(t_f, \tau_i) \circ (\Phi(\tau_i, \tau_j) \circ X^j, X^i) \end{aligned}$$

where δ_i^j is the Kronecker delta and $\Omega(t,\tau) \circ (U, V) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the bilinear operator found by integrating

$$\Omega(t,t) \circ (U,V) = 0_{n\times n} \qquad (19a)$$

$$\begin{aligned} \tfrac{\partial}{\partial\tau}\Omega(t,\tau) \circ (U,V) = -D_1^2\ell(x(\tau),\tau) \circ (U,V) \qquad (19b) \\ -\psi(t,\tau) \circ D_1^2 f(x(\tau),\tau) \circ (U,V) \\ -\Omega(t,\tau) \circ \big(D_1 f(x(\tau),\tau) \circ U, V\big) \\ -\Omega(t,\tau) \circ \big(U, D_1 f(x(\tau),\tau) \circ V\big) \end{aligned}$$

backwards over τ from t_f to τ_i

Proof: The proof is analogous to that of Lemma 4.1 and is only outlined here.[4] Take the derivative of (10) with respect to another switching time τ_j. The resulting integral is split into a form that matches $\psi(t,\tau)$ and a second term, called $\Omega(t,\tau)$ with the form

$$\begin{aligned} \Omega(t,\tau) \circ (U,V) = \int_\tau^t D_1\ell(x(s),s) \circ \phi(s,\tau) \circ (U,V) \\ + D_1^2\ell(x(s),s) \circ (\Phi(s,\tau) \circ U, \Phi(s,\tau) \circ V)\,\mathrm{d}s \end{aligned} \qquad (20)$$

This results in the first part of the theorem, (18).

The initial condition (19a) is seen to be zero directly from the definition of $\Omega(t,\tau)$, giving (19a) of the Lemma.

Equation (19b) of the Lemma is found by differentiating (20) with respect to τ and applying the identities for $\Phi(t,\tau)$ from (7) and the identities for $\psi(t,\tau)$ found in (17). ■

Theorem 6.1 is the natural extension of Lemma 4.1 to the second derivative. It has the same property that $\phi(t,\tau)$, the analogous second order state transition matrix, is no longer required and provides an algorithm for calculating every second derivative from a single integration. The first order operator $\psi(t,\tau)$ and the second derivatives of $\ell(x,t)$ and $f(x,t)$ both appear, as one would expect.

It is useful to write (19b) in matrix form to see how it is calculated in practice:

$$\begin{aligned} \tfrac{\partial}{\partial\tau}[\Omega(t,\tau)] = -[D_1^2\ell(x(\tau),\tau)] - [\psi(t,\tau) \circ D_1^2 f(x(\tau),\tau)] \\ - [D_1 f(x(\tau),\tau)]^T[\Omega(t,\tau)] - [\Omega(t,\tau)][D_1 f(x(\tau),\tau)] \end{aligned}$$

While Theorem 6.1 avoids $\phi(t,\tau)$, it does rely on $\Phi(t,\tau)$. The state transition matrix for the first derivative has re-appeared in the initial condition for the second derivative (14b) and the second derivative of the cost (18). There does not seem to be a computationally beneficial way to eliminate this requirement.

We can, however, calculate every value of $\Phi(t,\tau)$ that we need in a single integration along the trajectory by taking advantage of (7d). It allows us to calculate the state transition matrix of each segment $\Phi(\tau_{k+1}, \tau_k)$. These are composed to find $\Phi(\tau_i, \tau_j)$ for any i,j pair.

[4]See the journal submission *http://robotics.mech.northwestern.edu/∼murphey/murphey-TAC2009sub.pdf* for the complete proof.

VII. Optimization Algorithm

We optimize (2) with a standard numeric iterative approach [18] that relies on the derivatives that we have found. In each iteration, we choose a descent direction $z = -[H]^{-1}[D_\tau J(\cdot)]^T$ where H is a positive semidefinite matrix.

Choosing $H = I$ gives the Steepest Descent algorithm. This is a first order optimization that has linear convergence. $H = D_\tau D_\tau J(\cdot)$ (i.e. the Hessian of $J\cdot$) results in Newton's Method, a second order optimization with quadratic convergence. The Hessian must be checked to be positive definite or the cost might increase. When this test fails, implementations typically fall back to a first-order or modified second-order iteration.

Both first- and second-order algorithms benefit from the *Armijo Line Search* [3] algorithm. This is a simple algorithm that reduces the magnitude of the step size until there is a *sufficient decrease*. Satisfying the sufficient decrease condition guarantees that the optimization will eventually converge.

We must also keep the switching times ordered properly (i.e. $\tau_{k+1} \geq tk$). In this work, after calculating the descent direction z, we find the largest $\epsilon \in (0,1]$ such that $x + \epsilon z$ is ordered. This is an improvised method that has worked in practice; better and formal techniques are subjects for future research.

VIII. Example: The Kinematic Car

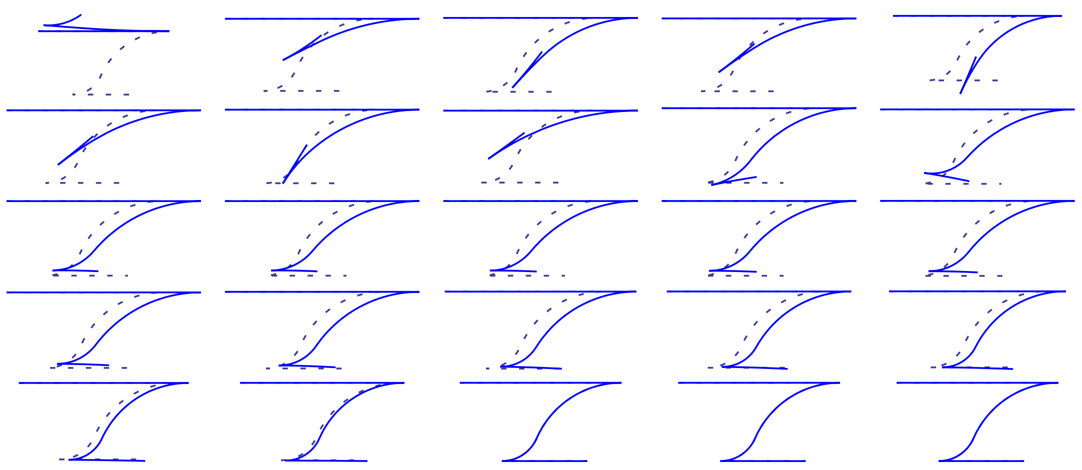
We consider the kinematic car as an example system for switching time optimization. First, the switching times needed to follow a (known to be admissible) path are found by optimization. Second, we use switching time optimization to find a parallel parking trajectory based on the Lie bracket that indicates such a motion should be possible. That is, we push the infinitesimal Lie bracket generator into an actual non-infinitesimal trajectory.

We consider the kinematic car with the dynamic function:

$$\dot{x} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = f_{car}(x,u) = \begin{bmatrix} u_1\cos(\theta) \\ u_1\sin(\theta) \\ u_1\tan(\phi) \\ u_2 \end{bmatrix} \qquad (21)$$

Distinct dynamic models are derived from the kinematic car by applying piecewise-constant inputs:

$$\begin{aligned} f_1(x) &= f_{car}(x,u_1) \\ f_2(x) &= f_{car}(x,u_2) \\ &\dots \\ f_N(x) &= f_{car}(x,u_N) \end{aligned}$$



1: Trajectories for the kinematic car at each iteration of the optimization. The solid line is the kinematic car and the dashed line is the desired trajectory.

A. Following a Valid Trajectory

For the first example, we consider a parallel parking maneuver made up of 7 sequential inputs:

Move Forward:	$u = [\ 0.3,\ 0]$
Turn Steering Clockwise:	$u = [\ 0,\ -2.8]$
Move Backward	$u = [\ -0.2,\ 0]$
Turn Steering Counterclockwise	$u = [\ 0,\ 2.9]$
Move Backward	$u = [\ -0.09,\ 0]$
Turn Steering Clockwise	$u = [\ 0,\ -1.8]$
Move Forward	$u = [\ 0.09,\ 0]$

For the optimization, we consider the magnitude and sequence of the inputs to be known but the switching times are unknown.

The Armijo optimization parameters [3] are $\alpha = 0.6$, $\beta = 0.0001$, and *zero tolerance* $= 10^{-4}$. The initial switching times were equal intervals from 0 to 7 seconds. Both optimizations converged to the correct switching times:

$$(\tau_1 \cdots \tau_{N+1}) = (0,\ 1,\ 1.5,\ 2.5,\ 3.5,\ 4.414,\ 5.248,\ 7)$$

The first order-only optimization took over 30,000 steps to converge. The second order optimization, which initially took 10 first order steps, converges in 24 iterations. Fig. 1 shows the trajectories for each iteration of the optimization.

We would likely reject 30,000 iterations as impractical for actual applications (certainly for real-time). The second order optimization reduces this by *three orders of magnitude* to where even real-time optimizations could be possible. The next example will use the same system for a more practical application of switching time optimization.

B. Lie Bracket Trajectory

Lie brackets[6], [10], [19] are infinitesimal operations that take advantage of two vector field not commuting to locally produce motion in a direction outside the linear span of the vector fields. The above parallel parking maneuver is a common example of Lie bracket motion. Even though the car only moves forward/backward and rotates the steering, it is able to achieve a net sideways movement by using the correct sequence of inputs.

The Lie bracket is formally based on infinitesimal motion, but its derivation suggests that we could determine a sequence of inputs for the above example from the Lie bracket instead of arbitrary design. For complex systems, this could be the basis for a local motion planning algorithm.

Suppose we seek a trajectory from $q = (0, 0, 0, 0)$ to $q = (0, -1, 0, 0)$. The Lie bracket to produce sideways motion for the kinematic car (21) is the nested bracket

$$\left[\frac{\partial f_{car}}{\partial u_1}, \left[\frac{\partial f_{car}}{\partial u_2}, \frac{\partial f_{car}}{\partial u_1}\right]\right] = (0, -1, 0, 0) \qquad (22)$$

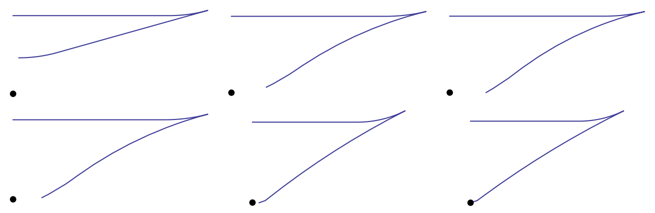
which corresponds to infinitesimal movements with the following inputs:

$u = [\ 1,\ 0]$
$u = [\ 0,\ 1]$
$u = [\ 1,\ 0]$
$u = [\ 0,\ -1]$
$u = [\ -1,\ 0]$
$u = [\ -1,\ 0]$
$u = [\ 1,\ 0]$
$u = [\ 0,\ 1]$
$u = [\ -1,\ 0]$
$u = [\ 0,\ -1]$

We setup an optimization using the above inputs[5] and add a final "constant" function (i.e. $u = [0, 0]$) to give the optimization flexibility in the duration of the maneuver.

Since any trajectory is acceptable, we choose a zero

[5]Note that the three consecutive movements $[1, 0]$, $[1, 0]$, and $[-1, 0]$ were collapsed into a single movement.



2: The trajectory at the 1st, 5th, 10th, 15th, 20th, and 22nd (final) iteration. The dot shows the desired final position.

incremental cost and non-zero terminal cost[6]:

$$\ell(x,t) = 0$$
$$m(x,t) = ||x - (0,-1,0,0)||^2$$

The optimization was run with $\alpha = 0.001$, $\beta = 0.6$, $zero\ tolerance = 10^{-4}$ and the initial condition[7]

$$(\tau_1 \cdots \tau_{N+1}) = (0,\ 2,\ 2.5,\ 3,\ 3.5,\ 5.5,\ 6,\ 6.5,\ 7,\ 12.5)$$

The optimization took 10 steepest descent steps followed by 12 second-order steps for a total of 22 iterations. A steepest descent optimization did not converge after 30,000 iterations. Figure 2 shows the final trajectory with several intermediate trajectories. The final switching times were

$$(\tau_1 \cdots \tau_{N+1}) = \\ (0,\ 1.33,\ 2.01,\ 2.58,\ 3.34,\ 5.45,\ 6.97,\ 7.06,\ 8.50,\ 12.5)$$

To generalize to arbitrary end points in a neighborhood of the initial condition, one need only change the terminal condition of the optimization (and potentially compute the infinitesimal approximation of an end point using the Lie bracket, typically using the Campbell-Baker-Hausdorff expansion [19]. If the distance to be traversed is larger than what a single motion can provide, then computing multiple Lie bracket motions—and therefore optimizing over more switching times—may be desirable.

IX. Conclusions and Future Work

We have presented derivations for the first and second derivatives of the cost function. While the first derivative has been reported previously, our derivation is more direct and uses basic calculus tools. More importantly it generalizes to second derivative easily. The local motion planning examples demonstrate the value of the second derivative for fast convergence even for very nonlinear systems.

[6]Terminal costs will be covered in detail in an expanded version of this work. They are a straight-forward modification that only affect the boundary conditions of the ϕ and Ω operators. See *http://robotics.mech.northwestern.edu/∼murphey/murphey-TAC2009sub.pdf* for details of how to treat terminal conditions.

[7]The Lie bracket also suggests the initial (relative) timing: the two fields in each bracket should be about the same length. For example, consider the Lie bracket $[f,[g,h]]$. If we move along f for ϵ seconds, the $[g,h]$ movement should also be ϵ seconds

References

[1] M. Alamir and S. A. Attia. Discussion on the paper "an optimal control approach for hybrid systems" by P. Riefinger, C. Iung and F. Kratz. *European Journal of Control*, pages 459–460, 2003.

[2] M. Alamir and S. A. Attia. On solving optimal control problems for switched hybrid nonlinear systems by strong variations algorithms. *In Proceedings of 6th IFAC Symposium on Nonlinear Control Systems*, pages 558–563, 2004.

[3] L. Armijo. Minimization of functions having lipschitz continuous first-partial derivatives. *Pacific Journal of Mathematics*, Vol. 16, 1966.

[4] S. A. Attia, M. Alamir, and C. Canudas de Wit. Sub optimal control of switched nonlinear systems under location and switching constraints. *In Proceedings of the 16th IFAC World Congress*, 2005.

[5] H. Axelsson, Y. Wardi, M. Egerstedt, and E.I. Verriest. Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. *Journal of Optimization Theory and Applications*, 136:167–186, 2008.

[6] F. Bullo and A.D. Lewis. *Geometric Control of Mechanical Systems*. Number 49 in Texts in Applied Mathematics. Springer-Verlag, 2004.

[7] T. Caldwell and T. D. Murphey. Second-order optimal estimation of slip state for a simple slip-steered vehicle. In *IEEE Int. Conf. on Automation Science and Engineering (CASE)*, pages 133–139, 2009.

[8] T. Caldwell and T. D. Murphey. Switching mode generation and optimal estimation with application to skid-steering. *Automatica*, 2010. Accepted for Publication.

[9] C.T. Chen. *Linear System Theory and Design*. Saunders College Publishing, 1984.

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. The MIT Press, 2005.

[11] X.C. Ding, Y. Wardi, and M. Egerstedt. On-line optimization of switched-mode dynamical systems. *IEEE Transactions on Automatic Control*, 54:2266–2271, 2009.

[12] M. Egerstedt, Y. Wardi, and F. Delmotte. Optimal control of switching times in switched dynamical systems. In *IEEE Conference on Decision and Control*, 2003.

[13] Magnus Egerstedt, Shun ichi Azuma, and Henrik Axelsson. Transition-time optimization for switched-mode dynamical systems. *IEEE Transactions on Automatic Control*, 51:110–115, 2006.

[14] Magnus Egerstedt, Shun ichi Azuma, and Yorai Wardi. Optimal timing control of switched linear systems based on partial information. *Nonlinear Analysis: Theory, Methods and Applications*, 65:1736–1750, 2006.

[15] J. Hauser and A. Saccon. A barrier function method for the optimization of trajectory functionals with constraints. In *45th IEEE Conference on Decision and Control*, 2006.

[16] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Transactions on Automatic Control*, 47:1536–1540, 2002.

[17] C.T. Kelly. *Iterative Methods of Linear and Nonlinear Equations*. SIAM, 1995.

[18] C.T. Kelly. *Iterative Methods for Optimization*. SIAM, 1999.

[19] S. S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, New York, 1999.

[20] Axel Schild, Xu Chu Ding, Magnus Egerstedt, and Jan Lunze. Design of optimal switching surfaces for switched autonomous systems. *IEEE Conference on Decision and Control*, 2009. (submitted).

[21] X. Xu and P.J. Antsaklis. Optimal control of switched autonomous systems. In *IEEE Conference on Decision and Control*, 2002.

[22] X. Xu and P.J. Antsaklis. Optimal control of switched autonomous systems with a prespecified sequence of active subsystems. *ISIS Technical Report*, 2002.

[23] Xuping Xu and Panos J. Antsaklis. Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. *International Journal of Control*, 75:1406 – 1426, 2002.