Second Order Switching Time Optimization for Time-Varying Nonlinear Systems

Elliot R. Johnson and Todd D. Murphey

***Abstract*— This paper presents second-order switching time optimization techniques for nonlinear time-varying systems that are piecewise second-differentiable. The techniques are based on direct analysis of flows rather than constrained optimization methods, making it possible to take higher order derivatives. The resulting algorithms also use the same number of integrations along the trajectory regardless of the number of switching times. Two examples are included–a linear example seen previously in the literature and a nonlinear example that emphasizes the fast convergence rates of second-order optimizations compared to first order techniques (e.g. convergence in 19 iterations compared to over 25,000).**

I. INTRODUCTION

Switching time optimization frequently arises in the analysis of hybrid systems–systems that discretely change their dynamics over time. Currently, techniques for numerically computing the directional derivative of the cost with respect to the switching times rely upon adjoining the cost function with the system dynamics using Lagrange multipliers [3]. This method provides fast computation of the directional derivative by integrating a single backwards differential equation regardless of the number of switching times [5] [4]. The difficulty with this approach is that gradient descent may be quite slow depending on the condition number of the local approximation of the cost function.

Newton's method addresses this issue, but second-order approximations of the cost function are necessary to implement Newton's method directly (although quasi-Newton methods are available [6] which we do not consider here). The adjoint gradient calculation does not provide a clear road map for computing second derivatives. Instead, we compute both the first derivative without adjoining the system dynamics to the cost. This technique then easily extends to derive second derivatives. We then show that the for the first derivative we also only compute a single integration and that for the second derivative we compute only a single integration of a larger dimensional operator.

Section II presents a formal problem statement and introduces conventions used in this work. Section III briefly reviews the standard numeric optimization algorithm. The cost function and its derivatives are considered in Sec. IV. Section. VI derives an algorithm for calculating the first derivative of a trajectory with respect to a switching time. This algorithm is refined in Sec. VII into a computationally efficient form. Likewise, an algorithm for calculating the second derivative of a trajectory is given in Sec. VIII and improved in Sec. IX. Example optimizations for linear and non-linear systems are considered in Sec. X and XI.

Elliot R. Johnson is a graduate student in Mechanical Engineering at Northwestern University, Evanston, IL, USA elliot.r.johnson@u.northwestern.edu

Todd D. Johnson is an Assistant Professor in Mechanical Engineering at the McCormick School of Engineering at Northwestern University, Evanston, IL, USA t-murphey@northwestern.edu

II. PROBLEM STATEMENT

A switching time optimization involves an n-dimensional dynamic system described by different dynamic functions over a time interval:

$$\dot{x} = \begin{cases} f_1(x) & \tau_1 < t \leq \tau_2 \\ f_2(x) & \tau_2 < t \leq \tau_3 \\ \vdots & \\ f_N(x) & \tau_N < t \leq \tau_{N+1} \end{cases} \tag{1}$$

with initial condition $x(\tau_1) = x_{init}$. The goal of the optimization is to find a set of τ_2, τ_3, ... τ_N switching times that minimize a cost function:

$$J(x_{init}, \tau_1, \tau_2, ..., \tau_{N+1}) = \int_{\tau_1}^{\tau_{N+1}} L(x(t), t)\mathrm{d}t \tag{2}$$

where $L(x,t)$ is called the incremental cost function and is chosen depending on the purpose of the optimization. For example, if the goal is to track a desired trajectory $x_d(t)$ as closely as possible, the incremental cost might be

$$L(x,t) = ||x - x_d(t)||$$

We emphasize that the first switching time is τ_2, not τ_1. Instead, τ_1 and τ_{N+1} are the *fixed* initial and final times of the trajectory (typically $\tau_1 = 0$). This choice is awkward, but leads to much cleaner notation throughout the paper.

A. Notation

Consistent notation greatly clarifies the ideas in optimal control. The notation used in this paper, though sometimes unfamiliar, has been carefully chosen and strives to be as consistent as possible.

Whenever possible we try to refer to the specific dynamic system that a quantity is associated with. For example $x_k(t)$ is the system trajectory for $\tau_k < \tau \leq \tau_{k+1}$.

The most important tool for the derivations in this paper is the derivative and we find several different derivative notations necessary. The familiar $\frac{\partial}{\partial t}[x]$ notation implies a derivative that has yet to be applied to the argument, so rules like the chain or product rule have yet to be applied. A derivative written as $D_x F(\cdot) \circ \partial x$ has already been evaluated–the chain rule and product rule have already been applied. Sometimes

instead of writing the name over the derivative variable, we use slot derivative notation: $D_i F(\arg_1, \arg_N, ...) \circ \partial \arg_i$ is the derivative of F with respect to its i-th argument. In both forms, the capital D implies that derivative has already been "applied" (e.g. the chain rule has been applied).

We are careful to represent derivatives as operators rather than matrices as this avoids dealing with rank-3 tensors and other unfamiliar objects.

Finally, there are some quantities that are represented without explicitly listing all their derivatives. For example, each trajectory segment x_k depends on the initial time and one or more switching times, but rather than write $x_k(x_0, \tau_1, \tau_2, ..., t)$, is much clearer to write abbreviate $x_k(t)$. This seems obvious, but it is very easy to forget and miss a term when applying the chain rule.

III. Numeric Optimization

Switching time optimization is based on standard iterative-descent algorithms [6]. The problem of implementation, then, is how to calculate the necessary derivatives of the cost function needed by these algorithms. The steepest descent method (a first-order method) requires the gradient of the cost function. The second-order Newton's method requires both the gradient and Hessian of the cost function.

Though not discussed in this paper, both algorithms are improved by including an Armijo line-search [1] to determine the actual step-size from the descent direction found in each descent iteration.

A. Maintaining Switching Time Orders

A switching time optimization must take care that switching times remain properly ordered: $\tau_1 \leq \tau_2 \leq ... \leq \tau_{N+1}$.

In this work, the step-size z is reduced by a factor $\nu \in (0, 1]$ before the Armijo line search:

$$z' = \nu z$$

where ν is chosen to be the largest scalar such that $x_i + \nu z$ will maintain the correct ordering. The Armijo line search can then further reduce the step size as needed and will not modify the switching time order.

IV. The Cost and Derivatives

The values $J(\cdot)$ and $DJ(\cdot) \circ z$ are required by the descent algorithm. The integral in (2) is $J(\cdot)$ split over the switching times:

$$J(\cdot) = \int_{\tau_1}^{\tau_{N+1}} L(x(t), t) \mathrm{d}t = \sum_{k=1}^{N} \int_{\tau_k}^{\tau_{k+1}} L(x_k(t), t) \mathrm{d}t$$

Taking the derivative with respect to a switching time τ_i:

$$\begin{aligned} D_{\tau_i} J(\cdot) \circ \partial\tau_i &= \tfrac{\partial}{\partial \tau_i} \left[\sum_{k=1}^{N} \int_{\tau_k}^{\tau_{k+1}} L(x_k(t), t) \mathrm{d}t \right] \\ &= \sum_{k=1}^{N} \Bigg[\int_{\tau_k}^{\tau_{k+1}} D_1 L(x_k(t), t) \circ D_{\tau_i} x_k(t) \circ \partial\tau_i \mathrm{d}t \\ &\quad + L(x_k(\tau_{k+1}), \tau_{k+1}) \tfrac{\partial \tau_{k+1}}{\partial \tau_i} - L(x_k(\tau_k), \tau_k) \tfrac{\partial \tau_k}{\partial \tau_i} \Bigg] \\ &= \sum_{k=1}^{N} \int_{\tau_k}^{\tau_{k+1}} D_1 L(x_k(t), t) \circ D_{\tau_i} x_k(t) \circ \partial\tau_i \mathrm{d}t \end{aligned}$$

The Leibniz terms disappear because of continuity along $x(t)$ and the fact that τ_i is never τ_1 or τ_{N+1}. The second derivative is found by differentiating the above with respect to another switching time τ_j:

$$\begin{aligned} &D_{\tau_j} D_{\tau_i} J(\cdot) \circ (\partial\tau_j, \partial\tau_i) = \\ &\quad \tfrac{\partial}{\partial \tau_j} \left[\sum_{k=1}^{N} \int_{\tau_k}^{\tau_{k+1}} D_1 L(x_k(t), t) \circ D_{\tau_i} x_k(t) \circ \partial\tau_i \mathrm{d}t \right] \\ &= D_1 L(x_{j-1}(\tau_j), \tau_j) \circ D_{\tau_i} x_{j-1}(\tau_j) \circ \partial\tau_i \partial\tau_j \\ &\quad - D_1 L(x_j(\tau_j), \tau_j) \circ D_{\tau_i} x_j(\tau_j) \circ \partial\tau_i \partial\tau_j \\ &\quad \sum_{k=1}^{N} \int_{\tau_k}^{\tau_{k+1}} D_1^2 L(x_k(t), t) \circ (D_{\tau_j} x_k(t) \circ \partial\tau_j, D_{\tau_i} x_k(t) \circ \partial\tau_i) \\ &\qquad + D_1 L(x_k(t), t) \circ D_{\tau_j} D_{\tau_i} x_k(t) \circ (\partial\tau_j, \partial\tau_i) \mathrm{d}t \end{aligned}$$

The above derivatives establish the need for the first and second derivatives of the trajectory, $D_{\tau_i} x_k(t) \circ \partial\tau_i$ and $D_{\tau_j} D_{\tau_i} x_k(t) \circ (\partial\tau_j, \partial\tau_i)$.

V. The Trajectory $x(t)$

The trajectory is calculated by numerically integrating (1). To differentiate the trajectory, however, it is helpful to apply the Fundamental Theorem of Calculus to work with the integral form:

$$x_0(t) = x_{init} \tag{3a}$$

$$x_k(t) = x_{k\text{-}1}(\tau_k) + \int_{\tau_k}^{t} f_k(x_k(s), s) \mathrm{d}s \tag{3b}$$

Note that any switching time derivatives of (3a) will identically zero. The rest of the paper assumes $k > 0$ when discussing derivatives.

VI. First Derivative: $D_{\tau_i} x_k(t) \circ \partial\tau_i$

The time derivative of a trajectory segment is known from (1):

$$D_t x_k(t) \circ \partial t = f_k(x_k(t), t) \partial t$$

The first derivative is found by differentiating (3b) while paying careful attention to the Leibniz and chain rules of

differentiation[1]:

$$
\begin{aligned}
&D_{\tau_i}x_k(t)\circ\partial\tau_i = \tfrac{\partial}{\partial\tau_i}\left[x_{k\text{-}1}(\tau_k) + \int_{\tau_k}^{t} f_k(x_k(s),s)\mathrm{d}s\right] \\
&= D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ\partial\tau_i + D_t x_{k\text{-}1}(\tau_k)\circ\tfrac{\partial\tau_k}{\partial\tau_i} - f_k(x_k(\tau_k),\tau_k)\tfrac{\partial\tau_k}{\partial\tau_i} \\
&\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s)\circ D_{\tau_i}x_k(s)\circ\partial\tau_i\mathrm{d}s \\
&= D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ\partial\tau_i + \left[f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k) - f_k(x_k(\tau_k),\tau_k)\right]\tfrac{\partial\tau_k}{\partial\tau_i} \\
&\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s)\circ D_{\tau_i}x_k(s)\circ\partial\tau_i\mathrm{d}s \qquad (4)
\end{aligned}
$$

This is a general form of the derivative that applies for all $k > 0$ and $i > 1$. We gain more insight and simplify the results by considering relationships between k and i.

Note that:

$$
\tfrac{\partial\tau_k}{\partial\tau_i} = \begin{cases} \partial\tau_i & k = i \\ 0 & k \neq i \end{cases} \qquad (5)
$$

which is simply formalizing the idea that the derivative of a independent variable with respect to itself is the identity while the derivative with respect to a different independent variable is zero.

A. First Derivative: $k < i$

In this case, the trajectory segment occurs before the switching time. Causality implies that a change in the future will not affect the present, so we expect this derivative to always be zero. Recognizing that $k < i$ implies $k \neq i$ and applying (5), Eq. (4) simplifies to:

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

When $k = 1$, the above is equivalent to a linear differential equation with zero initial conditions. The solution is therefore identically zero. The same then occurs for $k = 2, 3, \ldots$ up until $k = i$ at which point the above equation no longer applies.

B. First Derivative: $k = i$

In this case, the previous result shows that the first term of (4) disappears and we find:

$$
\begin{aligned}
D_{\tau_i}x_k(t)\circ\partial\tau_i &= f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\partial\tau_i - f_k(x_k(\tau_k),\tau_k)\partial\tau_j \\
&\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s)\circ D_{\tau_i}x_k(s)\circ\partial\tau_i\mathrm{d}s \qquad (6)
\end{aligned}
$$

The above can be transformed into a differential form by the Fundamental Theorem of Calculus:

$$
\begin{aligned}
&D_{\tau_i}x_k(\tau_k)\circ\partial\tau_i = \\
&\qquad f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\partial\tau_i - f_k(x_k(\tau_k),\tau_k)\partial\tau_j \qquad (7a) \\
&\tfrac{\partial}{\partial t}[D_{\tau_i}x_k(\tau_k)\circ\partial\tau_i] = \\
&\qquad \int_{\tau_k}^{t} D_1 f_k(x_k(s),s)\circ D_{\tau_i}x_k(s)\circ\partial\tau_i\mathrm{d}s \qquad (7b)
\end{aligned}
$$

[1]In particular: $\tfrac{\partial}{\partial\tau_i}[x_k(\tau_k)] = D_{\tau_i}x_k(\tau_k)\circ\partial\tau_i + D_t x_k(\tau_k)\circ\tfrac{\partial\tau_k}{\partial\tau_i}$

This form provides a way to concretely calculate $D_{\tau_i}x_k(t)\circ\partial\tau_i$: After evaluating the initial condition with (7a), we numerically integrate (7b) up until τ_{k+1}.

C. First Derivative: $k > i$

In this case (4) becomes

$$
\begin{aligned}
D_{\tau_i}x_k(t)\circ\partial\tau_i &= D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ\partial\tau_i \qquad (8) \\
&\quad + \int_{\tau_k}^{t} D_1 f_k(x_k(s),s)\circ D_{\tau_i}x_k(s)\circ\partial\tau_i\mathrm{d}s
\end{aligned}
$$

As in the previous section, this yields a initial condition/differential equation pair that can be numerically integrated from τ_k to τ_{k+1}. Unlike the previous section, this equation must recursively evaluate itself to determine the initial condition. The recursion is guaranteed to terminate as k is always decreasing towards i so that when $k = i$, the non-recursive (7) will be evaluated instead.

The results from the previous three sections provide a complete set of equations for calculating the first derivative of any trajectory segment with respect to any switching time.

VII. Consolidating the First Derivative

The above derivation is clear requires a new integration for each switching time. This section introduces a new operator that improves the algorithm. The operator is obtained by a single integration over the time horizon and can calculate the derivative with respect to any switching time as a simple linear composition.

A. State Transition Matrix

Recall that solutions to a linear differential equation can be written in terms of a state transition matrix $\Phi(t,\tau)$ [2]:

$$
\begin{aligned}
&z(\tau) = z_0 \\
&\dot{z}(t) = A(t)z(t)
\end{aligned}
\quad\Longleftrightarrow\quad z(t) = \Phi(t,\tau)\circ z_0 \qquad (10)
$$

The state transition matrix is found by numeric integration:

$$
\begin{aligned}
\Phi(\tau,\tau) &= \mathbf{I}_{n\times n} \\
\tfrac{\partial}{\partial t}[\Phi(t,\tau)] &= A(t)\circ\Phi(t,\tau)
\end{aligned} \qquad (11)
$$

B. $D_{\tau_i}x_k(t)\partial\tau_i$ using $\Phi_k(t,\tau_k)$

In (6) and (8), the solutions for $D_{\tau_i}x_k(t)\circ\tau_i$ are governed by the same linear differential equation. Let $\Phi_k(t,\tau_k)$ be the state transition matrix associated with that differential equation:

$$
\tfrac{\partial}{\partial t}[\Phi_k(t,\tau_k)] = D_1 f_k(x_k(s),s)\circ\Phi_k(t,\tau_k)
$$

The first derivative is rewritten in terms of Φ_k:

$$
X_k^i = \begin{cases} 0 & k < i \\ f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\partial\tau_i - f_k(x_k(\tau_k),\tau_k)\partial\tau_i & k = i \\ D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ\partial\tau_i & k > i \end{cases}
$$

$$
D_{\tau_i}x_k(t)\circ\partial\tau_i = \Phi_k(t,\tau_k)\circ X_k^i
$$

were X_k^i is the initial condition $D_{\tau_i}x_k(\tau_k)\circ\partial\tau_i$. This solution has the extremely desirable property that Φ can be numerically integrated *once* and then used to calculate any first-order derivative with simple linear compositions.

$$D_{\tau_j}D_{\tau_i}x_k(\tau_k)\circ(\partial\tau_j,\partial\tau_i)=X_k^{i,j}=$$

$$\begin{cases} D_1f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\circ f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\partial\tau_j\partial\tau_i+D_1f_k(x_k(\tau_k),\tau_k)\circ f_k(x_k(\tau_k),\tau_k)\partial\tau_j\partial\tau_i & \\ \quad D_2f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\circ\partial\tau_j\partial\tau_i-D_2f_k(x_k(\tau_k),\tau_k)\circ\partial\tau_j\partial\tau_i & k=i,\quad k=j \\ \quad -2D_1f_k(x_k(\tau_k),\tau_k)\circ f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\partial\tau_j\partial\tau_i & \\ D_1f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\circ D_{\tau_j}x_{k\text{-}1}(\tau_k)\circ\partial\tau_j\partial\tau_i-D_1f_k(x_k(\tau_k),\tau_k)\circ D_{\tau_j}x_k(\tau_k)\circ\partial\tau_j\partial\tau_i & k=i,\quad k>j \\ D_1f_{k\text{-}1}(x_{k\text{-}1}(\tau_k),\tau_k)\circ D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ\partial\tau_i\partial\tau_j-D_1f_k(x_k(\tau_k),\tau_k)\circ D_{\tau_i}x_k(\tau_k)\circ\partial\tau_i\partial\tau_j & k>i,\quad k=j \\ D_{\tau_j}D_{\tau_i}x_{k\text{-}1}(\tau_k)\circ(\partial\tau_j,\partial\tau_i) & k>i,\quad k>j \\ 0 & \text{Allothercases} \end{cases} \tag{9a}$$

$$\frac{\partial}{\partial t}[D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)]= D^2f_k(x_k(t),t)\circ(D_{\tau_j}x_k(t)\circ\partial\tau_j,D_{\tau_i}x_k(t)\circ\partial\tau_i)+D_1f_k(x_k(t),t)\circ D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i) \tag{9b}$$

Fig. 1. The second derivative of x_k is discussed in Sec. VIII

VIII. Second Derivative: $D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)$

The second derivative, $D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)$, is derived in same way as the first; normal derivative rules are applied to (6) and (8), being particularly careful to correctly apply the Leibniz and chain rules. The results are simplified for the individual cases of $k<j$, $k=j$, and $k>j$. The derivation is long and and uses no special techniques so it is omitted here. The final result is a single differential equation with initial conditions that depend on the relationship among k, i, and j. The result is given by (9) where $X_k^{i,j}$ represents the initial condition $D_{\tau_j}D_{\tau_i}x_k(\tau_k)\circ(\partial\tau_j,\partial\tau_i)$.

We again see a recursive dependence for $k>i$,$k>j$. The recursion is still guaranteed to terminate, though now there are three potential terminating equations instead of one.

Equation (9) provides a complete solution to calculate the second derivative of a trajectory. But like the original first derivative formulation, this method requires a new integration for each combination of i and j which will cause the method to become slow as N increases. In the next section, we refine the second derivative so that the computational complexity is independent of the number of switching times.

IX. Improving the Second Derivative

Equation (9b) describes the dynamics of all four non-zero derivative conditions. It is not linear and so cannot be simply represented with a state transition matrix.

However, it is an *affine* differential equation. If we consider the offset term to be an input, we can model (9b) as a *forced* linear system, as seen in Fig. 2.

The linear component of the forced system is the same as that of the first derivative, so the forced linear model will use the same state transition matrix—this does not add any extra computational cost.

The trajectory of a forced linear system is:

$$x(t)=\Phi(t,t_0)\circ\left[X(t_0)+\int_{t_0}^{t}\Phi(t_0,s)\circ B(s)\circ u(s)\mathrm{d}s\right]$$

$$\begin{aligned} D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)= \\ \Phi_k(t,\tau_k)\circ\left[X_k^{i,j}+\int_{\tau_k}^{t}\Phi_k(\tau_k,s)\circ D^2f_k(x_k(s),s)\circ\right. \\ \left.(D_{\tau_j}x_k(s)\circ\partial\tau_j,D_{\tau_i}x_k(s)\circ\partial\tau_i)\mathrm{d}s\right] \\ =\Phi_k(t,\tau_k)\circ\left[X_k^{i,j}+\int_{\tau_k}^{t}\Phi_k(\tau_k,s)\circ D^2f_k(x_k(s),s)\circ\right. \\ \left.(\Phi_k(s,\tau_k)\circ X_k^j,\Phi_k(s,\tau_k)\circ X_k^i)\mathrm{d}s\right] \end{aligned} \tag{12}$$

Lemma: There exists a bilinear operator $M_k\circ(\cdot,\cdot)$, independent of X_k^i, X_k^j, such that

$$D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)=\Phi_k(t,\tau_k)\circ\left[X_k^{i,j}+M_k\circ(X_k^j,X_k^i)\right] \tag{13}$$

is equivalent to (12).

Proof: Begin by setting the right sides of the equations for $D_{\tau_j}D_{\tau_i}x_k(t)\circ(\partial\tau_j,\partial\tau_i)$ equal:

$$\begin{aligned} \Phi_k(t,\tau_k)\circ\left[X_k^{i,j}+M_k(t,\tau_k)\circ(X_k^i,X_k^j)\right]= \\ \Phi_k(t,\tau_k)\circ\left[X_k^{i,j}+\int_{\tau_k}^{t}\Phi_k(\tau_k,s)\circ D^2f_k(x_k(s),s)\circ\right. \\ \left.(\Phi_k(s,\tau_k)\circ X_k^j,\Phi_k(s,\tau_k)\circ X_k^i)\mathrm{d}s\right] \end{aligned}$$

State transition matrices are always invertible:

$$\begin{aligned} X_k^{i,j}+M_k(t,\tau_k)\circ(X_k^i,X_k^j)=X_k^{i,j}+ \\ \int_{\tau_k}^{t}\Phi_k(\tau_k,s)\circ D^2f_k(x_k(s),s)\circ \\ (\Phi_k(s,\tau_k)\circ X_k^j,\Phi_k(s,\tau_k)\circ X_j^i)\mathrm{d}s \end{aligned}$$

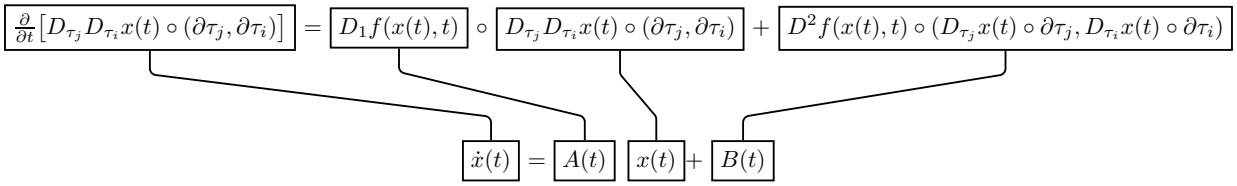

Fig. 2. The second derivative of $x_k(t)$ has the form of a forced linear system.

$$M_k(t, \tau_k) \circ (X_k^i, X_k^j) = \tag{14}$$

$$\int_{\tau_k}^{t} \Phi_k(\tau_k, s) \circ D^2 f_k(x_k(s), s) \circ \tag{15}$$

$$(\Phi_k(s, \tau_k) \circ X_k^j, \Phi_k(s, \tau_k) \circ X_k^i) \mathrm{d}s \tag{16}$$

Equation (16) is a composition of linear and bilinear operations. The most important property however, is that X_k^i and X_k^j can be pulled out of the integral. The entire composed operator can be integrated over $t = [\tau_k, \tau_{k+1}]$ and then used by (13) to calculate any second derivative using simple linear compositions.

A. Review Final Form of $D_{\tau_j} D_{\tau_i} x_k(t) \circ (\partial\tau_j, \partial\tau_i)$

With $\Phi_k(t, \tau_k)$ and $M_k(t, \tau_k)$ in hand, the second derivative is straightforward linear composition requiring additionally integration:

$$D_{\tau_j} D_{\tau_i} x_k(t) \circ (\partial\tau_j, \partial\tau_i) = \Phi_k(t, \tau_k) \circ X_k^{i,j} + M_k \circ (X_k^j, X_k^i)$$

with X_k^i and $X_k^{i,j}$ as defined earlier.

X. Example: Linear Time-Invariant Dynamics

This example[2] compares an optimization of a system with three switching times composed by two linear time-invariant systems.

Given

$$A_1 = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}$$

find the switching times $0 \leq \tau_2 \leq \tau_3 \leq \tau_4 \leq 1$ to optimize the incremental cost function:

$$L(x, t) = \tfrac{1}{2} ||x||^2$$

for the linear system governed by the following dynamics:

$$\dot{x} = \begin{cases} f_1(x, t) = A_1 x & 0 < t \leq \tau_2 \\ f_2(x, t) = A_2 x & \tau_2 < t \leq \tau_3 \\ f_3(x, t) = A_1 x & \tau_3 < t \leq \tau_4 \\ f_4(x, t) = A_2 x & \tau_4 < t \leq 1.0 \end{cases} \qquad x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The system was optimized with 1st-order only and a mixed 1st/2nd-order algorithms. Both optimizations used an Armijo line search with $\alpha = \beta = \frac{1}{2}$ and had the same switching times of $(\tau_2, \tau_3, \tau_4) = (0.3, 0.5, 0.7)$. The optimizations terminate once $||\nabla f(x_i)|| < 10^{-4}$.

[2]This example originally appears in [5].

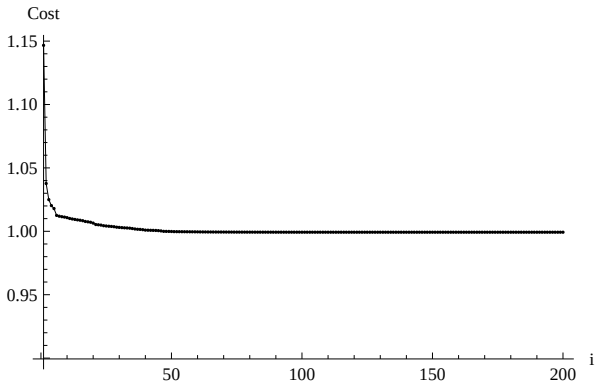

Fig. 3. The cost vs. iteration for the 1st-order only optimization.

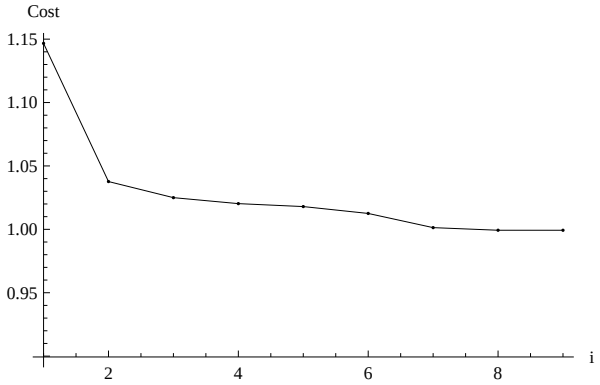

Fig. 4. The cost vs. iteration for the mixed 1st/2nd-order optimization.

The 1st-order optimization converges to $(\tau_2, \tau_3, \tau_4) = (0.5226, 0.7001, 0.8036)$ after 199 iterations. Figure 3 plots the cost vs. iteration. The 1st-order descent performs well at first, but slows down dramatically as it approaches the minimum.

The 1st/2nd-order optimization converges to approximately the same solution at $(\tau_2, \tau_3, \tau_4) = (0.5227, 0.7001, 0.8038)$ in 8 iterations. The first 4 iterations fall back to a 1st order optimization due to a non-positive definite second derivative. The remaining iterations were 2nd-order Newton's method steps. Figure 4 plots the cost vs. iterations.

XI. Example: The Kinematic Car

The kinematic car is an interesting non-linear model to use with the optimization. In this example, we create a desired trajectory from a sequence of constant inputs. The

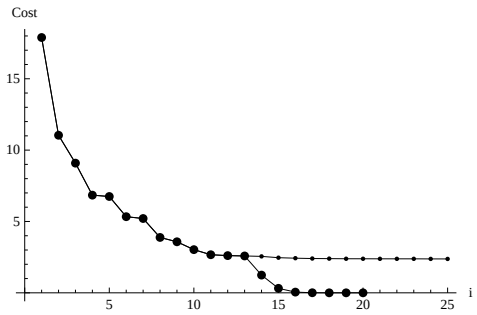

Fig. 5. Convergence of the 1st-order optimization (Small Dots) vs. mixed 1st/2nd order optimization (Large Dots). The mixed optimization converges almost immediately once Newton's method is being used. The 1st-order optimization is seen to converge extremely slowly.

optimization should (assuming the order and magnitude of the inputs are known) determine the switching times from one input to the next.

The dynamic model of the car is:

$$\dot{x} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = f_{car}(x,u) = \begin{bmatrix} u_1 \cos(\theta) \\ u_1 \sin(\theta) \\ u_1 \tan(\phi) \\ u_2 \end{bmatrix} \tag{17}$$

Distinct dynamic models are derived from the kinematic car by using different fixed inputs. For this 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	$u = [0.09, 0]$

These individual movements can only compose into the desired movement if the switching time from one input to the next is chosen properly.

The incremental cost function will be norm of the error between the simulated and desired trajectory:

$$L(x,t) = ||x - x_d(t)||$$

The optimization was performed with a purely 1st-order method and a mixed 1st/2nd-order method (i.e. Steepest descent was used for the first five iterations and Newton's method for the remaining iterations. When the second derivative was not positive definite, the algorithm fell back to steepest descent). The optimizations terminated when $||\nabla J(x_i)|| < 10^{-5}$.

Both optimizations converged to the correct switching times. The mixed method converged after 19 iterations while the steepest descent optimization method took nearly 30,000 iterations. The cost vs. iteration is plotted for both methods for up to 25 iterations in Fig. 5.

Figure 6 illustrates the simulated trajectory at each iteration of the trajectory.

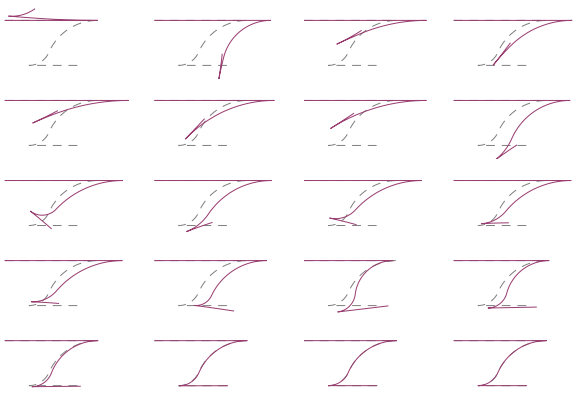

Fig. 6. The simulated car trajectories during each iteration of the mixed 1st/2nd order optimization. The optimization proceeds from left to right, top to bottom. The dashed line indicates the desired trajectory.

XII. Conclusions and Future Work

The tools developed in this paper enable a 2nd-order approach (Newton's method) to optimizing multiple switching times for arbitrary non-linear dynamic systems. The 1st and 2nd order derivatives of the trajectory are each found with a single integration over the entire trajectory going forward in time.

The kinematic car example demonstrates the dramatic reduction in iterations needed to converge that are possible by using a second order optimization.

XIII. Acknowledgements

This material is based upon work partially supported by the National Science Foundation under Grant CCF 0907869. 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.

We would additionally like to acknowledge useful conversations with Prof. Magnus Egerstedt at the Georgia Institute of Technology and Prof. John Hauser at the University of Colorado at Boulder.

References

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

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

[3] Florent C. Delmotte. *Multi-Modal Control: From Motion Description Languages to Optimal Control*. PhD thesis, Georgia Institute of Technology, 2006.

[4] M. Egerstedt, Y. Wardi, and H. Axelsson. Optimal control of switching times in hybrid systems. In *IEEE Methods and Models in Automation and Robotics*, Miedzyzdroje, Poland, Aug. 2003.

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

[6] C.T. Kelley. *Iterative Methods for Optimization*. Society for Industrial Mathematics, 1987.