Second-Order Switching Time Optimization for Switched Dynamical Systems

Bartolomeo Stellato, Sina Ober-Blöbaum, and Paul J. Goulart

***Abstract*—Switching time optimization arises in finite-horizon optimal control for switched systems where, given a sequence of continuous dynamics, one minimizes a cost function with respect to the switching times. We propose an efficient method for computing the optimal switching times for switched linear and nonlinear systems. A novel second-order optimization algorithm is introduced where, at each iteration, the dynamics are linearized over an underlying time grid to compute the cost function, the gradient, and the Hessian efficiently. With the proposed method, the most expensive operations at each iteration are shared between the cost function and its derivatives, thereby greatly reducing the computational burden. We have implemented the algorithm in the Julia package SwitchTimeOpt, allowing users to easily solve switching time optimization problems. In the case of linear dynamics, many operations can be further simplified and benchmarks show that our approach is able to provide optimal solutions in just a few millisecond. In the case of nonlinear dynamics, our method provides optimal solutions with up to two orders of magnitude time reductions over state-of-the-art approaches.**

***Index Terms*—Hybrid systems, optimal control, optimal switching times, optimization algorithms, switched systems.**

I. Introduction

Hybrid dynamical models arise where continuous dynamics interact with discrete events, and have been used in a wide variety of fields such as process control, automotive industry, power systems, traffic control, and many others. Despite recent advances [1], [2], optimal control of hybrid systems presents several challenges because it involves the solution of NP-hard optimization problems [3].

Switched systems are a particular class of hybrid systems consisting of a collection of continuous subsystems with a switching law defining the active one at each time instant, see [4] for a recent survey. In this paper, we focus on optimal control of autonomous switched systems where the sequence of continuous dynamics is fixed. In particular, we study the problem of computing the optimal switching instants at which the ordered dynamics must change in order to minimize a given cost function. This problem is usually referred to as *switching time optimization* and has been studied extensively in the last decade.

In [5], Seatzu *et al.* construct offline a mapping from the initial state to the optimal switching times for linear switched systems, leading to the high storage requirements typical for explicit control approaches [6].

Manuscript received January 31, 2017; accepted April 10, 2017. Date of publication April 24, 2017; date of current version September 25, 2017. This work was supported by the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme (FP7/2007-2013) under REA Grant Agreement 607957 (TEMPO). Recommended by Associate Editor Z. Sun. *(Corresponding author: Bartolomeo Stellato.)*

The authors are with the University of Oxford, Oxford OX1 3PJ, U.K. (e-mail: bartolomeo.stellato@eng.ox.ac.uk; sina.ober-blobaum@eng.ox.ac.uk; paul.goulart@eng.ox.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2017.2697681

Recent methods focus on finding optimal switching times using iterative optimization. In [7], Das and Mukherjee use Pontryagin's Minimum Principle to find the optimal switching times for linear dynamics by solving a two-point boundary value problem. The performance of this approach, however, is strongly dependent on the time discretization and it is not straightforward to extend it to nonlinear dynamics. In [8], an expression for the gradient of the cost function with respect to the switching times is derived for the case of nonlinear systems. A first-order method based on Armijo step sizes is then adopted to find the optimal switching times. An extension for discrete-time switched nonlinear systems is given in [9]. However, first-order methods are very sensitive to the problem data and can exhibit slow convergence [10]. In [11], an expression for the Hessian of the cost function is derived for nonlinear dynamics and a second-order method is adopted to find the optimal switching times showing significant improvements on the number of iterations compared to the first-order method in [8]. However, both of these first- and second-order approaches suffer from the computational complexity of multiple numerical integrations required to solve the differential equations used to define the cost function, the gradient, and the Hessian (in the second-order case). Note that the Hessian definition in [11] requires an additional set of integrations to be performed.

In this paper, we present a novel method to solve switching time optimization problems efficiently for linear and nonlinear dynamics. We develop efficiently computable expressions for the cost function, the gradient, and the Hessian, exploiting shared terms in the most expensive computations. In this way, at each iteration of the optimization algorithm there is no significant increase in complexity in computing the gradient or the Hessian once the cost function is evaluated. These easily computable expressions are obtained thanks to linearizations of the system dynamics around equally spaced grid points, and then integrated via independent matrix exponentials. In the case of linear dynamics, our method can be greatly simplified and the matrix exponentials decomposed into independent scalar exponentials that can be parallelized to further reduce the computation times.

Our method has been implemented in the open-source Julia package `SwitchTimeOpt` [12] with a simple interface that allows the user to easily define and solve switching time optimization problems. Through Julia's `MathProgBase` interface, `SwitchTimeOpt` supports a wide variety of nonlinear solvers that can be quickly interchanged.

We provide two examples to benchmark the performance of our method. The first, from [13], is a system with two unstable switched dynamics whose optimal switching times are obtained in few millisecond with our approach. The second is the so-called Lotka–Volterra fishing problem [14] with nonlinear dynamics, integer control inputs, and constant steady-state values to be tracked. In this example, our method, which is implemented on a high-level language, exhibits up to two orders of magnitude improvements over tailored state-of-the-art software tools for nonlinear optimal control.

The rest of the paper is organized as follows. Section II reformulates the problem in terms of switching time intervals. Section III introduces some mathematical preliminaries. In Section IV, we present the main result. In Section V, we propose simplifications in the case of linear dynamics. In Section VI, we present our open-source Julia package and two example problems. Conclusions are provided in Section VII.

II. Problem Statement

Consider a switched autonomous system switching between N modes, the dynamics for which can be expressed as

$$\dot{x}(t) = f_i(x(t)), \quad t \in [\tau_i, \tau_{i+1}), \quad i = 0, \dots, N \tag{1}$$

with each $f_i : \mathbb{R}^n \to \mathbb{R}^n$ assumed differentiable and the state $x(t) \in \mathbb{R}^{n_x}$ assumed to have initial state $x(0) = x_0$. We will refer to the times τ_i as the *switching times*, and define also the *switching intervals* $\delta_i := \tau_{i+1} - \tau_i, \ i = 0, \dots, N$ so that each $\tau_i = \sum_{j=0}^{i-1} \delta_j$. In the sequel, we will take the set of switching intervals $\delta := \{\delta_i\}_{i=0}^N$ as decision variables to be optimized, but will occasionally use the switching times $\tau := \{\tau_i\}_{i=0}^{N+1}$ for convenience of notation. We define the final time as $T_\delta := \sum_i \delta_i = \tau_{N+1}$, with initial time $\tau_0 = 0$.

Our goal is to find the optimal switching intervals δ^* minimizing an objective function in Bolza form

$$\psi(\delta) + \mathcal{L}(\delta) := x(T_\delta)^\top \bar{E} x(T_\delta) + \int_0^{T_\delta} x(t)^\top \bar{Q} x(t) dt. \tag{2}$$

The Mayer term ψ penalizes the final state at time T_δ with weights defined by the symmetric positive semidefinite matrix $\bar{E} = \bar{E}^\top \in \mathbb{S}_+^{n_x}$. The Lagrange term $\mathcal{L}$ penalizes the integral between 0 and T_δ of the quadratic state penalty weighted by the matrix $\bar{Q} = \bar{Q}^\top \in \mathbb{S}_+^{n_x}$.

We include a set of constraints on the switching intervals

$$\Delta := \left\{\delta \in \mathbb{R}_+^{N+1} \mid 0 \le \underline{b}_i \le \delta_i \le \bar{b}_i \ i = 0, \dots, N \ \wedge \ T_\delta = T\right\}$$

which requires all switching times to be nonnegative and the final time T_δ to be equal to some desired final time T. In addition, in case the ith dynamics must be active for a minimum or maximum time, we allow lower and upper bounds $\underline{b}_i$ and $\bar{b}_i$, respectively. If neither minimum nor maximum constraints are imposed for interval δ_i, we set $\underline{b}_i = 0$ and $\bar{b}_i = \infty$.

The complete switching time optimization problem takes the following form:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & \int_0^{T_\delta} x(t)^\top \bar{Q} x(t) dt + x(T_\delta)^\top \bar{E} x(T_\delta) \\ \text{subject to} \quad & \dot{x}(t) = f_i(x(t)), \quad t \in [\tau_i, \tau_{i+1}), \quad i = 0, \dots, N \\ & x(0) = x_0 \\ & \delta \in \Delta. \end{aligned} \tag{$\mathcal{P}$}$$

Although we restrict ourselves to the case where the switching order of the $N+1$ modes is prespecified, we allow the system dynamics to be identical for different i. If we set some $\delta_i = 0$, then the ith interval collapses and the dynamics switch directly from the $(i-1)$th to the $(i+1)$th mode. This allows some dynamics to be bypassed and an arbitrary switching order realized without recourse to integer optimization; see [15]. For example, given N_{dyn} different dynamics, one can cycle through all of them in the same predefined ordering N_{cyc} times for a total of $N_{\text{dyn}} N_{\text{cyc}}$ intervals and $N_{\text{dyn}} N_{\text{cyc}} - 1$ switching times, thereby allowing the dynamics to be visited in arbitrary order. We illustrate the use of this approach in the examples in Section VI.

The cost function in problem $(\mathcal{P})$ is nonconvex in general, but it is smooth [16] and its first and second derivatives can be used efficiently within a nonlinear optimization algorithm to obtain locally optimal switching times. In order to obtain a solution method implementable in real-time, we derive tractable formulations of the cost function, the gradient, and the Hessian based on linearizations of the system dynamics.

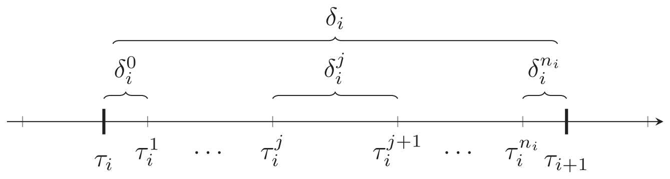

Fig. 1. Switching times within the time grid.

III. Preliminaries

A. Time Grid and Dynamics Linearization

In order to integrate the switched nonlinear dynamics (1), we define an equally spaced "background" grid of n_{grid} time-points from 0 to the final time T, and hold these background grid points fixed regardless of the choice of switching times τ_i. Note that, depending on the intervals δ, the switching times τ can be in different positions relative to the background grid while maintaining the ordering $\tau_i \le \tau_{i+1}$.

We subdivide each interval δ_i according to the background grid points falling between τ_i and τ_{i+1}, with τ_i^j denoting the jth grid point after the switching time τ_i. The number of such background grid points between switching times τ_i and τ_{i+1} is denoted by n_i, which is itself a function of the switching times τ. Note that we set $n_{N+1} = 0$.

For notational convenience, we will define $\tau_i^0 := \tau_i$ and $\tau_i^{n_i+1} := \tau_{i+1}$ for $i = 0, \dots, N$. We further define a partitioning of the switching intervals such that δ_i^j is the jth subdivision of interval i, so that $\delta_i = \sum_{j=0}^{n_i} \delta_i^j$ and $\tau_i^k = \tau_i + \sum_{j=0}^{k-1} \delta_i^j$ for all $k \le n_i$.

In subsequent sections, we will define a number of vector and matrix quantities to be associated with the time instants τ_i^j, and will adopt complementary notation, e.g., vector x_i^j and matrix M_i^j are associated with the jth grid point after switching time τ_i. Likewise $x_i^{n_i+1} := x_{i+1}$, $x_i^0 := x_i$, $M_i^{n_i+1} := M_{i+1}$, and $M_i^0 := M_i$.

A portion of the grid is presented in Fig. 1 where the smaller ticks represent the background grid points.

In order to make the computations of the cost function and its derivatives numerically efficient, we linearize the dynamics around each time instant of the background grid and each switching time.

For a given time instant τ_i^j, we consider the linearized dynamics around the state $x_i^j := x(\tau_i^j)$ with $j = 0, \dots, n_i + 1$ (for ease of notation we define $x_i := x_i^0 := x(\tau_i)$) by writing

$$\begin{aligned} \dot{x}(t) &\approx f_i(x_i^j) + \frac{\partial f_i}{\partial x}(x_i^j)(x(t) - x_i^j) \\ &= \frac{\partial f_i}{\partial x}(x_i^j) x(t) + \left(f_i(x_i^j) - \frac{\partial f_i}{\partial x}(x_i^j) x_i^j\right) \end{aligned}$$

where $\frac{\partial f_i}{\partial x}(x_i^j)$ is the Jacobian of the ith nonlinear dynamics evaluated at x_i^j. We can obtain an approximate linear model by augmenting the dynamics with an additional constant state so that

$$\dot{x}(t) = A_i^j x(t), \quad t \in [\tau_i^j, \tau_i^{j+1}) \tag{3}$$

where

$$A_i^j = \begin{bmatrix} \frac{\partial f_i}{\partial x}(x_i^j) & f_i(x_i^j) - \frac{\partial f_i}{\partial x}(x_i^j) x_i^j \\ 0 & 0 \end{bmatrix} \tag{4}$$

Algorithm 1: Solve Switching Time Optimization problem $(\mathcal{P})$.

```
1: function SWITCHINGTIMEOPTIMIZATION
2:     while Termination conditions not met do
3:         Linearize problem $(\mathcal{P})$
4:         Compute $J(\delta)$, $\nabla J(\delta)$ and $H_J(\delta)$ for $(\mathcal{P}_{\text{lin}})$
5:         Perform one NLP solver iteration obtaining new $\delta$
6:     end while
7: end function
```

and $x(t)$ is an augmented version of the previous state definition, i.e., $x(t) \leftarrow (\, x(t),\ 1\,)$.

B. Solution Approach

Once the dynamics are linearized as in (3), and after defining the augmented cost function weights $Q := \text{blkdiag}(\bar{Q}, 0)$ and $E := \text{blkdiag}(\bar{E}, 0)$, we can approximate the problem $(\mathcal{P})$ as

$$\begin{aligned}
\underset{\delta}{\text{minimize}} \quad & J(\delta) = \int_0^{T_\delta} x(t)^\top Q x(t) dt + x(T_\delta)^\top E x(T_\delta) \\
\text{subject to} \quad & \dot{x}(t) = A_i^j x(t), \;\; t \in [\tau_i^j, \tau_i^{j+1}), \;\; i = 0, \ldots, N, \\
& \qquad j = 0, \ldots, n_i \\
& x(0) = x_0 \\
& \delta \in \Delta. \qquad\qquad (\mathcal{P}_{\text{lin}})
\end{aligned}$$

We will make use of problem $(\mathcal{P}_{\text{lin}})$ to approximate the original problem $(\mathcal{P})$ at each iteration of a standard second-order nonlinear programming routine such as IPOPT [17]. By linearization of the system dynamics around the state trajectory, we can directly construct problem $(\mathcal{P}_{\text{lin}})$.

In the remainder of this paper, we focus on the numerical evaluation of the cost function $J(\delta)$, the gradient $\nabla J(\delta)$, and the Hessian $H_J(\delta)$ for problem $(\mathcal{P}_{\text{lin}})$, all of which can be computed efficiently.

A prototype algorithm is sketched in Algorithm 1. Note that in line 3 the act of linearizing problem $(\mathcal{P})$ produces the majority of the computational work with the benefit that the cost function and its derivatives can be then computed efficiently in line 4.

C. Definitions

We next present some preliminary definitions required to develop our main result.

Definition 1 (State evolution): The matrix $\Phi(t, \tau_i^j)$ is the state-transition matrix of the linearized system from τ_i^j to t, and is defined as

$$\Phi(t, \tau_i^j) := e^{A_l^m (t - \tau_\ell^m)} \cdot \left[\prod_{p=0}^{m-1} e^{A_\ell^p \delta_\ell^p}\right] \left[\prod_{q=i+1}^{\ell-1} \prod_{p=0}^{n_q} e^{A_q^p \delta_q^p}\right] \left[\prod_{p=j}^{n_i} e^{A_i^p \delta_i^p}\right] \tag{5}$$

where τ_ℓ and τ_ℓ^m are the last switching time and the last grid point before t, respectively.

Given a time instant τ_i^j and a time $t \in \mathbb{R}_+$ such that $t \geq \tau_i^j$ we can define the state $x(t)$ as $x(t) = \Phi(t, \tau_i^j) x_i^j$. Observe that if we consider a transition between two switching times τ_i and τ_ℓ with $\tau_i \leq \tau_\ell$, the state-transition matrix in (5) simplifies to

$$\Phi(\tau_\ell, \tau_i) = \prod_{q=i}^{\ell-1} \prod_{p=0}^{n_q} e^{A_q^p \delta_q^p} \tag{6}$$

which will be used extensively in most of the computations in the remainder of this paper.

Definition 2 (Cost-to-go matrices): Given the time τ_i^j, define matrix $P_i^j \in \mathbb{S}_+^{n_x}$ as

$$P_i^j := \int_{\tau_i^j}^{T_\delta} \Phi(t, \tau_i^j)^\top Q \Phi(t, \tau_i^j) dt \tag{7}$$

where $\Phi(t, \tau_i^j)$ is the state-transition matrix in Definition 1. Define the matrix $F_i \in \mathbb{S}_+^{n_x}$ as

$$F_i^j := \Phi(T_\delta, \tau_i^j)^\top E \Phi(T_\delta, \tau_i^j). \tag{8}$$

Define the sum of these two matrices as

$$S_i^j := P_i^j + F_i^j, \quad i = 0, \ldots, N+1. \tag{9}$$

Following the convention described in Section III-A, we will denote $P_i^0 := P_i, F_i^0 := F_i, S_i^0 := S_i, P_i^{n_i+1} := P_{i+1}, F_i^{n_i+1} := F_{i+1}$, and $S_i^{n_i+1} := S_{i+1}$.

Definition 3 (Matrix C): Given matrices S_i with $i = 0, \ldots, N+1$ and $A_i^{n_i}$ with $i = 0, \ldots, N$, define matrices $C_i \in \mathbb{S}_+^{n_x}$ as

$$C_i = Q + (A_i^{n_i})^\top S_{i+1} + S_{i+1} A_i^{n_i}, \quad i = 0, \ldots, N. \tag{10}$$

IV. Numerical Solution Method

We are now in the position to derive the cost function and its first and second derivatives for problem $(\mathcal{P}_{\text{lin}})$.

Theorem 1: (Cost function $J(\delta)$, gradient $\nabla J(\delta)$, and Hessian $H_J(\delta)$): The following hold.

1) The cost function $J(\delta)$ is the following quadratic function of the initial state
$$J(\delta) = x_0^\top S_0 x_0. \tag{11}$$
2) The gradient $\nabla J(\delta)$ of the cost function can be computed as
$$\nabla J(\delta)_i = \frac{\partial J(\delta)}{\partial \delta_i} = x_{i+1}^\top C_i x_{i+1}, \;\; i = 0, \ldots, N. \tag{12}$$
3) The Hessian $H_J(\delta)$ of the cost function can be computed as
$$\begin{aligned}
H_J(\delta)_{i,\ell} &= \frac{\partial^2 J(\delta)}{\partial \delta_i \partial \delta_\ell} \\
&= \begin{cases} 2x_{\ell+1}^\top C_\ell \Phi(\tau_{\ell+1}, \tau_{i+1}) A_i^{n_i} x_{i+1} & \ell \geq i \\ H_J(\delta)_{\ell,i} & \ell < i \end{cases}
\end{aligned} \tag{13}$$
where $i, \ell = 0, \ldots, N$.

The proof can be found in Appendix A.

Regardless of the second-order optimization method employed, most of the numerical operations needed to compute $J(\delta), \nabla J(\delta)$, and $H_J(\delta)$ at each iteration are shared. Thus, it is necessary to perform them only once per solver iteration.

A. State Propagation and Matrix Exponentials

We now define the auxiliary matrices needed for our computations.

Definition 4 (Auxiliary matrices): We define the matrix exponential of the linearized system between time instants τ_i^j and τ_i^{j+1} with $i = 0, \ldots, N$ and $j = 0, \ldots, n_i$ as

$$\mathcal{E}_i^j := e^{A_i^j \delta_i^j}. \tag{14}$$

Algorithm 2; Linearize, Compute mat. exp. and Propagate.

```
 1: function, LinMatExpProp
 2:     for $i = 0, \dots, N$ do
 3:         for $j = 0, \dots, n_i$ do
 4:             $A_i^j \leftarrow$ Eq. (4)                  ▷ Linearize dynamics
 5:             $Z_i^j \leftarrow$ Eq. (16)                 ▷ Matrix exponential
 6:             $\mathcal{E}_i^j, M_i^j \leftarrow$ Eq. (17)
 7:             $x_i^{j+1} \leftarrow \mathcal{E}_i^j x_i^j$
 8:         end for
 9:     end for
10:     return $x_i, \quad i = 0, \dots, N+1$
11:     return $M_i^j, \mathcal{E}_i^j, \quad j = 0, \dots, n_i \quad i = 0, \dots, N$
12: end function
```

Moreover, we define the matrices $M_i^j \in \mathbb{S}_+^{n_x}$ as

$$M_i^j := \int_0^{\delta_i^j} e^{A_i^{j\top}\eta} Q e^{A_i^j \eta} d\eta. \tag{15}$$

Both of these matrices can be computed via the single matrix exponential

$$Z_i^j = e^{G_i^j \delta_i^j} := \begin{bmatrix} Z_{i,1}^j & Z_{i,2}^j \\ 0 & Z_{i,3}^j \end{bmatrix} \tag{16}$$

with $Z_{i,1}^j, Z_{i,2}^j, Z_{i,3}^j \in \mathbb{R}^{n_x \times n_x}$ and matrices G_i^j being defined as $G_i^j := \begin{bmatrix} -A_i^{j\top} & Q \\ 0 & A_i^j \end{bmatrix}$ with $G_i^j \in \mathbb{R}^{2n_x \times 2n_x}$.

After computing Z_i^j, matrices $\mathcal{E}_i^j$ and M_i^j can be obtained as

$$\mathcal{E}_i^j = Z_{i,3}^j \quad \text{and} \quad M_i^j = Z_{i,3}^{j\top} Z_{i,2}^j. \tag{17}$$

For more details, see [18, Th. 1].

In Algorithm 2, we describe the subroutine to propagate the state, linearize the dynamics, and obtain matrices $\mathcal{E}_i^j$ and M_i^j. At every instant τ_i^j, the dynamics are linearized, matrices $\mathcal{E}_i^j, M_i^j$ are computed, and the state x_i^j is propagated. Note that, as we described in Section III-A, we consider $x_i^0 = x_i$ and $x_i^{n_i+1} = x_{i+1}$.

There are several methods to compute the matrix exponential as discussed in [19] and [20]. In our work we use "Method 3" in [20, Section 3], i.e., the scaling and squaring method explained in detail in [21], which is the most common method used for computing the matrix exponential because of its efficiency and precision. However, in the case of linear dynamics discussed in Section V, the matrices A_i^j are always constant and many operations can be precomputed, thereby increasing the speed of the algorithm.

Observe that the matrix exponentials employed in this section are an implementation of the first-order forward Euler exponential integrator [22] that performs well in many cases of stiff systems.

B. State Transition Matrices Φ

From Theorem 1, we need the state transition matrices between the switching instants. They can be computed recursively using Definition 1 which, combined with the definition of the matrix exponentials in (14), can be written as

$$\Phi(\tau_\ell, \tau_i) = \prod_{q=i}^{\ell-1} \prod_{p=0}^{n_q} \mathcal{E}_q^p. \tag{18}$$

Note that we only need to compute the state transition matrices in the cases $\ell \geq i$, so that the transition goes forward in time.

Algorithm 3: Compute $J(\delta), \nabla J(\delta)$ and $H_J(\delta)$.

```
1: function ComputeCostFunctionAndDerivatives
   Shared Precomputations:
2:     $x_i, \mathcal{E}_i^j, M_i^j \leftarrow$ LinMatExpProp          ▷ Algorithm 2
3:     $S_i \leftarrow$ ComputeS                            ▷ Proposition 1
4:     $C_i \leftarrow$ Eq. (10)                            ▷ Definition 3
5:     $\Phi(\tau_l, \tau_i) \leftarrow$ ComputeΦ                     ▷ (18)
   Compute $J(\delta), \nabla J(\delta)$ and $H_J(\delta)$                ▷ Theorem 1
6:     $J(\delta) \leftarrow$ Eq. (11)
7:     $\nabla J(\delta) \leftarrow$ Eq. (12)
8:     $H_J(\delta) \leftarrow$ Eq. (13)
9: end function
```

C. Matrices S_i^j

To obtain the cost function and its first and second derivatives, we need to compute matrices S_i. Given matrices $\mathcal{E}_i^j$ and M_i^j, S_i can be obtained with the following proposition.

Proposition 1: Matrix S_i^j with $i = 0, \dots, N+1$ and $j = 0, \dots, n_i$ satisfy the following recursion:

$$S_N = E \tag{19}$$

$$S_i^j = M_i^j + \mathcal{E}_i^{j\top} S_i^{j+1} \mathcal{E}_i^j. \tag{20}$$

The proof is given in Appendix B.

Note that we are considering $S_i^0 := S_i$ and $S_i^{n_i+1} := S_{i+1}$, as discussed in Section III-A.

D. Complete Algorithm to Compute $J(\delta), \nabla J(\delta)$, and $H_J(\delta)$

The complete algorithm to linearize problem $(\mathcal{P})$ and compute the cost function, the gradient, and the Hessian of $(\mathcal{P}_{\text{lin}})$ with respect to the switching intervals is shown in Algorithm 3.

After performing the shared precomputations, the cost function and its derivatives can be computed using Theorem 1 with no significant increase in computation to obtain also the Hessian needed in a second-order method.

V. Linear Switched Systems

When the system has linear switched dynamics of the form

$$\dot{x}(t) = A_i x(t), \quad t \in [\tau_i, \tau_{i+1}), \quad i = 0, \dots, N \tag{21}$$

the computations can be greatly simplified. In Algorithm 1, there is no need to resort to an auxiliary problem with linearized dynamics. In this case, the main result in Theorem 1 applies directly to the cost function and derivatives of the original problem $(\mathcal{P})$. There is no need for a linearization grid when dealing with linear systems. Thus, we simplify all the results for nonlinear dynamics by removing the indices j and setting $n_i = 0$ with $i = 0, \dots, N+1$.

Since the dynamics matrices do not change during the optimization, we precompute the matrices $G_i = G_i^0$ in (16) offline. In addition, if some of the G_i are diagonalizable, they can be factorized offline as $G_i = Y_i^\top \Lambda_i Y_i$, where Λ_i are the diagonal matrices of eigenvalues and Y_i are the nonsingular matrices of right eigenvectors. Thus, matrix exponentials (16) can be computed online as simple scalar exponentials of the diagonal elements of Λ_i via

$$Z_i = Y_i^\top e^{\Lambda_i \delta_i} Y_i \tag{22}$$

which corresponds to "Method 14" in [19, Sec. 6] and [20]. Note that the scalar exponentials are independent and can be computed in parallel to minimize the computation time. If a particular G_i is not diagonalizable, we compute its matrix exponential as in the nonlinear system case with the scaling and squaring method [20, Sec. 3].

Further improvements in computational efficiency can be obtained in the case of linear dynamics by executing the main loop in Algorithm 2 in parallel, since there is no need to propagate the state and iteratively linearize the system.

VI. Software and Examples

All algorithms and examples described in this paper have been implemented in the Julia language in the open-source package `SwitchTimeOpt`, available at the URL given in [12]. This package allows the user to easily define and efficiently solve switching time optimization problems for linear and nonlinear systems. `SwitchTimeOpt` supports a wide variety of nonlinear solvers through `MathProgBase` interface such as IPOPT [17] or KNITRO [23].

For each of the examples described in this section, we interfaced `SwitchTimeOpt` with the IPOPT solver [17] with default options on a late 2013 Macbook Pro with Intel Core i7 and 16GB of RAM. The examples are initialized with τ_i equally spaced between 0 and T.

A. Unstable Switched Dynamics

Consider the switched system from [13] described by the two unstable dynamics $A_1 = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}$ and $A_2 = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}$. Note that A_1 and A_2 have no common eigenvectors. The system transitions happen $N = 5$ times between 0 and $T = 1$ according to the modes sequence $\{1, 2, 1, 2, 1, 2\}$ and the cost function matrix is $Q = I$. The approach converges to the IPOPT precision 10^{-8} in roughly 3.5 ms producing the optimal switching times $\tau^* = (\,0.100,\ 0.297,\ 0.433,\ 0.642,\ 0.767\,)$, which correspond to the same solution obtained in [13]. However, no timing is reported in that work.

B. Lotka–Volterra Type Fishing Problem

The Lotka–Volterra fishing problem has been studied for almost a century [14] and analyzed from an integer optimal control point of view in [24] and [25]. Given an integer input sequence $\{u_i\}_{i=0}^N,\ u_i \in \{0, 1\}$ defining whether to fish $u_i = 1$ or not to fish $u_i = 0$ and N switching times τ_i, the Lotka–Volterra system can be described with the following switched dynamics:

$$\dot{x}(t) = f_i(x(t)) = \begin{bmatrix} x_1(t) - x_1(t)x_2(t) - c_1 x_1(t)u_i \\ -x_2(t) + x_1(t)x_2(t) - c_2 x_2(t)u_i \end{bmatrix} \tag{23}$$

with $t \in [\tau_i, \tau_{i+1}),\ i = 0, \ldots, N$. The prey biomass $x_1(t)$ assumed to grow exponentially and the predator $x_2(t)$ one assumed to decrease exponentially. In addition, there is a coupling term describing the interaction of the biomasses when the predators eat the prey. We choose $c_1 = 0.4$ and $c_2 = 0.2$ defining the amount of predators and prey caught when fishing occurs.

The goal is to fish responsibly in order to bring both the biomasses from an initial value of $x_0 = (\,0.5,\ 0.7\,)$ to the steady-state value $(\,1,\ 1\,)$ within the time $T = 12$. In other words, the optimal control problem constitutes a tracking problem where we penalize the deviations from the reference values $x_r(t) = (\,1,\ 1\,)$ by deciding when to start and stop fishing.

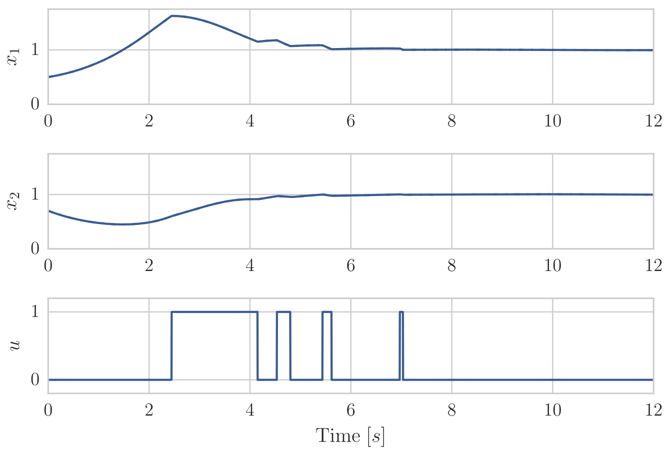

Fig. 2. Fishing problem. States and input behaviors at the optimal switching times τ^*. The states of the simulated nonlinear system (blue line) and the linearized system (dot-dashed green line) show a very close match.

TABLE I
Results for Lotka–Volterra Fishing Problem After 20 Iterations

n_{grid}	$J_{\text{ode45}}(\delta^*)$	$J(\delta^*)$	ΔJ [%]	$n_{J,\text{eval}}$	Time [s]
100	1.3500	1.3508	0.065	177	0.65
150	1.3454	1.3459	0.033	56	0.27
200	1.3456	1.3459	0.016	51	0.29
250	1.3454	1.3455	0.010	54	0.38

The complete optimal control problem can be written as

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & \int_0^{T_\delta} \|x(t) - x_r(t)\|_2^2 dt \\ \text{subject to} \quad & \dot{x}(t) = f_i(x(t)), \quad t \in [\tau_i, \tau_{i+1}),\ i = 0, \ldots, N \\ & x(0) = x_0 \\ & \delta \in \Delta. \end{aligned} \tag{24}$$

The problem is easily brought into the state-regulation form $(\mathcal{P})$ by augmenting the state, with $x_r(t)$ having $\dot{x}_r(t) = 0$, and minimizing the deviations between $x(t)$ and $x_r(t)$.

We consider a sequence of $N = 8$ switchings between the two possible input values $\{u_i\}_{i=0}^N = \{0, 1, 0, 1, 0, 1, 0, 1, 0\}$ giving a total of 9 dynamics. We run the algorithm for 20 iterations for increasing number of fixed-grid points $100, 150, 200,$ and 250. The optimal switching times for $n_{\text{grid}} = 200$ are $\tau^* = (\,2.446,\ 4.150,\ 4.533,\ 4.799,\ 5.436,\ 5.616,\ 6.969,\ 7.033\,)$ and the state behavior is displayed in Fig. 2. The linearized system is also plotted as a dot-dashed green line showing an almost indistinguishable curve.

The results are shown in Table I. The system is simulated at the optimal intervals δ^* with an ode45 integrator obtaining the cost function value $J_{\text{ode45}}(\delta^*)$ and with the grid linearizations obtaining $J(\delta^*)$—their difference described by $\Delta J = \|J_{\text{ode45}}(\delta^*) - J(\delta^*)\|/\|J_{\text{ode45}}(\delta^*)\|$ converges to 0 as the number of grid points increases. The number of cost function evaluations $n_{J,\text{eval}}$ and the computation time are also given in Table I. For the chosen solver IPOPT, increasing the number of grid points does not necessarily mean a higher computation time, because the latter is strictly related to the number of cost function evaluations that depends on the line search steps.

Our results are very close to the solutions in [24], which are obtained by discretizing the problem in 60 time instants leading to a mixed-integer optimization problem with 2^{60} possible input combinations. In [24], the best cost function value 1.3451 is obtained after solving the relaxed integer optimal control problem, applying sum-up-rounding and using the obtained result as a warm-starting point for the switching time optimization problem solution with multiple shooting. Timings are provided in the report [26, Sec. 5.5] where the execution times are approximately ten times slower than the ones obtained in this paper. Note that the implementations in [24] and [26] use the software package MUSCOD-II [27], which is an optimized C++ implementation of the multiple shooting methods, while our approach has been implemented in the high-level language Julia.

VII. Conclusion

We presented a novel method for computing the optimal switching times for linear and nonlinear switched systems. By reformulating the problem with the switching intervals as optimization variables, we derive efficiently computable expressions for the cost function, the gradient, and the Hessian, which share the most expensive computations. Once the cost function value is obtained, at each iteration of the optimization algorithm, there is no significant increase in complexity in computing the gradient and the Hessian. In addition, we showed that in the case of linear dynamics, many operations can be performed offline and many online operations parallelized greatly reducing the computation times.

We implemented our method in the open-source Julia package `SwitchTimeOpt`, which allows the user to quickly define and solve switching time optimization problems. An example with switched linear dynamics shows that our method can solve switching time optimization problems in milliseconds time scale. In a switched nonlinear dynamics example, we show that our high-level Julia implementation can solve these problems up to two orders of magnitude faster than state-of-the-art approaches.

Appendix A
Proof of Theorem 1

In order to prove the main result, we first require the following two lemmas.

Lemma 1 (State-transition matrix derivative): Given two switching times τ_a and τ_{i+1} with $\tau_a \le \tau_{i+1}$ such that τ_{i+1} does not coincide with any point of the background grid and the switching interval δ_i, the first derivative of the state-transition matrix between τ_a and τ_{i+1} with respect to δ_i can be written as

$$\frac{\partial \Phi(\tau_{i+1},\tau_a)}{\partial \delta_i} = A_i^{n_i}\Phi(\tau_{i+1},\tau_a). \tag{25}$$

Note that in the case when τ_{i+1} coincides with a fixed-grid point, the derivative is not defined since at $\tau_{i+1}+\epsilon$ a new linearization is introduced breaking the smoothness of the state-transition matrix. Our derivations still hold in that case by considering, instead of the gradient, the subgradient equal to the one-sided limit of the derivative from below.

Proof: We can rewrite $\Phi(\tau_{i+1},\tau_a)$ using Definition 1 as

$$\Phi(\tau_{i+1},\tau_a) = e^{A_i^{n_i}\left(\delta_i - \sum_{p=0}^{n_i-1}\delta_i^p\right)}\left(\prod_{p=0}^{n_i-1} e^{A_i^p\delta_i^p}\right)\Phi(\tau_i,\tau_a) \tag{26}$$

by using the relation $\delta_i = \sum_{j=0}^{n_i}\delta_i^j$. Taking the derivative of (26), we obtain

$$\begin{aligned}\frac{\partial \Phi(\tau_{i+1},\tau_a)}{\partial \delta_i} &= A_i^{n_i} e^{A_i^{n_i}\left(\delta_i - \sum_{p=0}^{n_i-1}\delta_i^p\right)}\left(\prod_{p=0}^{n_i-1} e^{A_i^p\delta_i^p}\right)\Phi(\tau_i,\tau_a)\\ &= A_i^{n_i}\Phi(\tau_{i+1},\tau_a)\end{aligned}$$

where we made use of the properties of the matrix exponential $e^{X(a+b)} = e^{Xa} + e^{Xb}$ and $\frac{\partial}{\partial c}e^{Xc} = Xe^{Xc}$ with $X \in \mathbb{R}^{n_x}$ and a, b, $t \in \mathbb{R}$. ■

Matrices S_i and their first derivatives play an important role in the proof. We here derive the first derivative of S_i.

Lemma 2 (Derivative of matrices S_i): Given the switching times τ_a and τ_{i+1} so that $\tau_a \le \tau_{i+1}$ and that τ_{i+1} does not coincide with any point of the background grid and the interval δ_i, the derivative of S_a with respect to δ_i is given as

$$\frac{\partial S_a}{\partial \delta_i} = \Phi(\tau_{i+1},\tau_a)^\top C_i \Phi(\tau_{i+1},\tau_a). \tag{27}$$

Proof: From (9), we can write the derivative as

$$\frac{\partial S_a}{\partial \delta_i} = \frac{\partial P_a}{\partial \delta_i} + \frac{\partial F_a}{\partial \delta_i}. \tag{28}$$

Let us analyze the two components separately. We decompose the integral defined by P_a as

$$\begin{aligned}\frac{\partial P_a}{\partial \delta_i} &= \frac{\partial}{\partial \delta_i}\left(\int_{\tau_i}^{\tau_{i+1}} \Phi(t,\tau_a)^\top Q\Phi(t,\tau_a)dt\right)\\ &\quad + \frac{\partial}{\partial \delta_i}\left(\int_{\tau_{i+1}}^{T_\delta} \Phi(t,\tau_a)^\top Q\Phi(t,\tau_a)dt\right).\end{aligned} \tag{29}$$

Note that the integral from τ_a to τ_i does not depend on δ_i and its derivative is zero. Taking first the leftmost term in (29), the integral from τ_i to τ_{i+1} can be written as

$$\frac{\partial}{\partial \delta_i}\left(\int_{\tau_i}^{\tau_{i+1}} \Phi(t,\tau_a)^\top Q\Phi(t,\tau_a)dt\right) \tag{30}$$

$$\begin{aligned}&= \Phi(\tau_i,\tau_a)^\top \frac{\partial}{\partial \delta_i}\left(\int_0^{\delta_i} \Phi(\eta+\tau_i,\tau_i)^\top\right.\\ &\qquad \left.\cdot\, Q\Phi(\eta+\tau_i,\tau_i)d\eta\right)\Phi(\tau_i,\tau_a)\\ &= \Phi(\tau_{i+1},\tau_a)^\top Q\Phi(\tau_{i+1},\tau_a)\end{aligned} \tag{31}$$

where in the first equality we applied the change of variables $\eta = t - \tau_i$ and in the second equality the fundamental theorem of calculus. Next, taking the rightmost term in (29), the integral from τ_{i+1} to T_δ can be obtained as

$$\begin{aligned}&\frac{\partial}{\partial \delta_i}\left(\int_{\tau_{i+1}}^{T_\delta} \Phi(t,\tau_a)^\top Q\Phi(t,\tau_a)dt\right)\\ &= \frac{\partial}{\partial \delta_i}\left(\Phi(\tau_{i+1},\tau_a)^\top P_{i+1}\Phi(\tau_{i+1},\tau_a)\right)\\ &= \Phi(\tau_{i+1},\tau_a)^\top\left((A_i^{n_i})^\top P_{i+1} + P_{i+1}A_i^{n_i}\right)\Phi(\tau_{i+1},\tau_a).\end{aligned} \tag{32}$$

In the first equality, we decomposed the state transition matrices and used the definition of P_{i+1} of (7). In the second equality, we applied the chain rule noting that P_{i+1} is independent from δ_i and Lemma 1 to compute the derivatives. We rewrite (29) using (31) and (32) obtaining

$$\frac{\partial P_a}{\partial \delta_i} = \Phi(\tau_{i+1},\tau_a)^\top\left(Q + (A_i^{n_i})^\top P_{i+1} + P_{i+1}A_i^{n_i}\right)\Phi(\tau_{i+1},\tau_a). \tag{33}$$

We now focus on the derivative of F_a in (28) that can be written so that

$$\frac{\partial F_a}{\partial \delta_i} = \frac{\partial}{\partial \delta_i}\left(\Phi(\tau_{i+1},\tau_a)^\top F_{i+1}\Phi(\tau_{i+1},\tau_a)\right)$$
$$= \Phi(\tau_{i+1},\tau_a)^\top\left((A_i^{n_i})^\top F_{i+1} + F_{i+1}A_i^{n_i}\right)\Phi(\tau_{i+1},\tau_a). \quad (34)$$

In the first equality, we decomposed the state transition matrices and used the definition of F_{i+1} from (8). In the second equality, we applied the chain rule noting that F_{i+1} is independent from δ_i and Lemma 1 to compute the derivatives.

By adding (33) and (34) as in (28) and applying Definition 2, we obtain

$$\frac{\partial S_a}{\partial \delta_i} = \Phi(\tau_{i+1},\tau_a)^\top\left(Q + (A_i^{n_i})^\top S_{i+1} + S_{i+1}A_i^{n_i}\right)\Phi(\tau_{i+1},\tau_a) \quad (35)$$

and the result from Definition 3. ■

We are now in a position to prove each of the statements in Theorem 1 in turn.

a) Cost function—proof of 1): The cost function in (11) can be derived directly from its definition in problem $(\mathcal{P}_{\text{lin}})$ and Definition 2.

b) Gradient—proof of 2): The gradient of the cost function can be derived by taking the derivative of (11). By considering the component related to δ_i, we can write

$$\frac{\partial J(\delta)}{\partial \delta_i} = x_0^\top\frac{\partial S_0}{\partial \delta_i}x_0 = x_0^\top\Phi(\tau_{i+1},0)^\top C_i\Phi(\tau_{i+1},0)x_0$$
$$= x_{i+1}^\top C_i x_{i+1}. \quad (36)$$

In the first equality, the initial state has been taken out from the derivative operator since x_0 fixed. In the second equality, we applied Lemma 2, and in the third equality, we used Definition 1 to obtain x_{i+1}. The result holds for $i = 0,\ldots,N$.

c) Hessian—proof of 3): The Hessian of the cost function can be derived by taking the derivative of (36). Let us first take the derivative with respect to the same interval δ_i, writing

$$\frac{\partial^2 J(\delta)}{\partial \delta_i^2} = \frac{\partial}{\partial \delta_i}\left(x_0^\top\Phi(\tau_{i+1},0)^\top C_i\Phi(\tau_{i+1},0)x_0\right)$$
$$= x_0^\top\Phi(\tau_{i+1},0)^\top(A_i^{n_i})^\top C_i\Phi(\tau_{i+1},0)x_0$$
$$+ x_0^\top\Phi(\tau_{i+1},0)^\top C_i A_i^{n_i}\Phi(\tau_{i+1},0)x_0$$
$$= x_{i+1}^\top(A_i^{n_i})^\top C_i x_{i+1} + x_{i+1}^\top C_i A_i^{n_i}x_{i+1}$$
$$= 2x_{i+1}^\top C_i A_i^{n_i}x_{i+1}.$$

In the first equality, we took into account that C_i and x_0 do not depend on δ_i and applied the chain rule. In the second equality, we applied Lemma 1, and in the third equality, we used Definition 1. In the last equality, we took the transpose of the first term that is a scalar. By using identity $C_iA_i^{n_i} = C_i\Phi(\tau_{i+1},\tau_{i+1})A_i^{n_i}$, we obtain the desired result.

In the case when we take the derivative with respect to δ_ℓ with $\ell > i$, we can write

$$\frac{\partial^2 J(\delta)}{\partial \delta_\ell\partial \delta_i} = \frac{\partial}{\partial \delta_\ell}\left(x_{i+1}^\top\left(Q + A_i^{n_i\top}S_{i+1} + S_{i+1}A_i^{n_i}\right)x_{i+1}\right)$$
$$= x_{i+1}^\top A_i^{n_i\top}\frac{\partial}{\partial \delta_\ell}(S_{i+1})x_{i+1} + x_{i+1}^\top\frac{\partial}{\partial \delta_\ell}(S_{i+1})A_i^{n_i}x_{i+1}$$
$$= 2x_{i+1}^\top\frac{\partial}{\partial \delta_\ell}(S_{i+1})A_i^{n_i}x_{i+1}$$
$$= 2x_{\ell+1}^\top C_\ell\Phi(\tau_{\ell+1},\tau_{i+1})A_i^{n_i}x_{i+1}. \quad (37)$$

In the first equality, we applied Definition 3. In the second inequality, we brought the terms not depending on δ_ℓ outside of the derivative operator. In the third equality, we took the transpose of the first element that is a scalar. In the fourth equality, we applied Lemma 2 and finally obtained $x_{\ell+1}$ from Definition 1. ■

Appendix B
Proof of Proposition 1

From Definition 2, we can obtain (19) by directly setting $i = N+1$. The recursion (20) can be derived by using Definition 2 as follows:

$$S_i^j = \int_{\tau_i^j}^{\tau_i^{j+1}}\Phi(t,\tau_i^j)^\top Q\Phi(t,\tau_i^j)dt$$
$$+ \Phi(\tau_i^{j+1},\tau_i^j)^\top\left(\int_{\tau_i^{j+1}}^{T_\delta}\Phi(t,\tau_i^{j+1})^\top Q\Phi(t,\tau_i^{j+1})dt\right.$$
$$\left.+\ \Phi(T_\delta,\tau_i^{j+1})^\top E\Phi(T_\delta,\tau_i^{j+1})\right)\Phi(\tau_i^{j+1},\tau_i^j)$$
$$= \int_0^{\delta_i^j}\Phi(\eta+\tau_i^j,\tau_i^j)^\top Q\Phi(\eta+\tau_i^j,\tau_i^j)d\eta$$
$$+ \Phi(\tau_i^{j+1},\tau_i^j)^\top S_i^{j+1}\Phi(\tau_i^{j+1},\tau_i^j)$$
$$= \int_0^{\delta_i^j}e^{A_i^{j\top}\eta}Qe^{A_i^j\eta}d\eta + e^{A_i^{j\top}\delta_i^j}S_i^{j+1}e^{A_i^{j\top}\delta_i^j}.$$

In the first equality, we split the integral in two parts and can bring the matrices $\Phi(\tau_i^{j+1},\tau_i^j)$ and $\Phi(\tau_i^{j+1},\tau_i^j)^\top$ outside the integral since they do not depend on t. In the second equality, we apply the change of variables $\eta = t - \tau_i^j$ and the Definition 2 to obtain S_i^{j+1}. In the third equality, we rewrite the transition matrices using matrix exponentials. Finally, by using Definition 4 we complete the proof. ■

References

[1] S. Sager, "Numerical methods for mixed-integer optimal control problems," Ph.D. dissertation, Interdisciplinary Center Sci. Comput., Universität Heidelberg, Heidelberg, Germany, 2005.

[2] C. Kirches, *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*. Vieweg+Teubner Verlag, Fachmedien Wiesbaden GmbH, Wiesbaden, 2011.

[3] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, Apr. 2013.

[4] F. Zhu and P. J. Antsaklis, "Optimal control of hybrid switched systems: A brief survey," *Discrete Event Dyn. Syst.*, vol. 25, no. 3, pp. 345–364, May 2014.

[5] C. Seatzu, D. Corona, A. Giua, and A. Bemporad, "Optimal control of continuous-time switched affine systems," *IEEE Trans. Automat. Control*, vol. 51, no. 5, pp. 726–741, May 2006.

[6] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[7] T. Das and R. Mukherjee, "Optimally switched linear systems," *Automatica*, vol. 44, no. 5, pp. 1437–1441, May 2008.

[8] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. Automat. Control*, vol. 51, no. 1, pp. 110–115, Jan. 2006.

[9] K. Flaßkamp, T. Murphey, and S. Ober-Blöbaum, "Discretized switching time optimization problems," in *Proc. Eur. Control Conf.*, Jul. 2013, pp. 3179–3184.

[10] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Berlin, Germany: Springer , 2006.

[11] E. R. Johnson and T. D. Murphey, "Second-order switching time optimization for nonlinear time-varying dynamic systems," *IEEE Trans. Automat. Control*, vol. 56, no. 8, pp. 1953–1957, Jul. 2011.

[12] "SwitchTimeOpt.", 2017. [Online]. Available: github.com/oxfordcontrol/SwitchTimeOpt.jl

[13] T. M. Caldwell and T. D. Murphey, "Single integration optimization of linear time-varying switched systems," *IEEE Trans. Automat. Control*, vol. 57, no. 6, pp. 1592–1597, May 2012.

[14] V. Volterra, "Variazioni e fluttuazioni del numero d'individui in specie animali conviv enti," *Mem. R. Accad. Naz. dei Lincei*, vol. VI, no. 2, pp. 31–113, 1926.

[15] M. Gerdts, "A variable time transformation method for mixed-integer optimal control problems," *Optimal Control Appl. Methods*, vol. 27, no. 3, pp. 169–182, 2006.

[16] X. C. Ding, Y. Wardi, and M. Egerstedt, "On-line optimization of switched-mode dynamical systems," *IEEE Trans. Automat. Control*, vol. 54, no. 9, pp. 2266–2271, Aug. 2009.

[17] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[18] C. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Trans. Automat. Control*, vol. AC-23, no. 3, pp. 395–404, Jun. 1978.

[19] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Rev.*, vol. 20, pp. 801–836, 1978.

[20] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Rev.*, vol. 45, no. 1, pp. 3–49, Mar. 2003.

[21] N. J. Higham, "The scaling and squaring method for the matrix exponential revisited," *SIAM Rev.*, vol. 51, no. 4, pp. 747–764, Nov. 2009.

[22] M. Hochbruck and A. Ostermann, "Exponential integrators," *Acta Numerica*, vol. 19, pp. 209–286, May 2010.

[23] R. H. Byrd, J. Nocedal, and R. A. Waltz, "KNITRO: An integrated package for nonlinear optimization," in *Large-Scale Nonlinear Optimization*. Boston, MA, USA: Springer, 2006, pp. 35–59.

[24] S. Sager, H. G. Bock, M. Diehl, G. Reinelt, and J. P. Schlöder, *Numerical Methods for Optimal Control With Binary Control Functions Applied to a Lotka-Volterra Type Fishing Problem* (Lecture Notes in Economics and Mathematical Systems). Berlin, Germany: Springer-Verlag, Dec. 2006, pp. 269–290.

[25] S. Sager, "A benchmark library of mixed-integer optimal control problems," in *Mixed Integer Nonlinear Programming*. New York, NY, USA: Springer, 2012, pp. 631–670.

[26] S. Sager, "On the integration of optimization approaches for mixed-integer nonlinear optimal control ," Habilitationsschrift, Interdisciplinary Center Sci. Comp., Universität Heidelberg, Sep. 2011.

[27] M. Diehl, D. B. Leineweber, A. Schäfer, MUSCOD-II Users Manual, *IWR-Preprint 2001-25*, Universität Heidelberg, 2001.