Second-Order Partial Outer Convexification for Switched Dynamical Systems

Christoph Plate, Sebastian Sager, Martin Stoll, and Manuel Tetschke

***Abstract*—Mixed-integer optimal control problems arise in many practical applications combining nonlinear, dynamic, and combinatorial features. To cope with the resulting complexity, several approaches have been suggested in the past. Some of them rely on solving a reformulated and relaxed control problem, referred to as partial outer convexification (POC). Inspired by an efficient algorithm for switching time optimization by Stellato and coworkers, SwitchTimeOpt.jl, we developed an algorithmic approach for POC implemented in a Julia package. Both approaches are based on linearization and exponential integration to obtain second derivatives. We show the efficiency and applicability of the novel approach by comparing it to SwitchTimeOpt.jl, by extending the concept and calculations to the treatment of constraints, and by investigating warm starting of switching time optimization. An additional comparison to a CasADi-based standard single shooting approach shows a significant reduction in computational time despite an increase in iterations. The new solver facilitates the reliable and fast solution of mixed-integer optimal control problems.**

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

I. Introduction

In THIS article, we are interested in optimal control of switched dynamical systems, i.e., systems consisting of multiple subsystems typically described by ordinary differential equations (ODEs) or differential-algebraic equations (DAEs). The degree of freedom to minimize a given cost function is the choice of the active subsystem at each point in time. Switched dynamical systems are prevalent in control applications. Typical examples are the choice of gear in vehicles in automotive control, traffic networks, operating strategies of hybrid vehicles, ON/OFF positions of valves, performing measurements yes or no, or the activation of whole units in process engineering applications; see [17] for an online benchmark library of such applications with further references.

Control problems involving switched dynamical systems belong to the problem class of mixed-integer optimal control problems (MIOCPs). Several algorithmic approaches have been developed to tackle this class of problems. A survey is beyond the scope of this article and we refer to [7], [9], and [22] for further references. For direct methods that discretize control functions with finitely many degrees of freedom, different approaches to address the integrality of control functions have been proposed. Two of these are in the focus of this article: *switching time optimization (STO)* and *partial outer convexification (POC)*. In STO [3], [7], a predefined switching sequence of the modes is assumed. Using a time transformation argument, the problem of determining the right mode for each point in time can be translated into finding the optimal switching times. Recently, an efficient structure-exploiting algorithm for solving STO problems was proposed in [20], based on a linearization of the dynamics and integration via matrix exponentials.

In POC [8], [18], a binary control function is introduced for each mode of the switched system. The integrality constraint is then relaxed by allowing the control functions to take values from the interval [0,1], resulting in a standard optimal control problem (OCP). Solving this relaxed OCP is often the first step in a decomposition approach for solving MIOCPs, called combinatorial integral approximation (CIA) [19]. CIA consists of the following three steps: 1) solving the relaxed OCP; 2) approximating the relaxed controls with binary controls; and 3) re-evaluating the MIOCP with fixed binary controls. Several methods for formulating and solving the approximation problem in the second step have been proposed and implemented in the software package `pycombina` [4]. Note that the third step is only necessary if additional continuous controls are present and that step 2 can also be replaced or followed by an STO formulation, as done in Section IV-E.

Here, we focus on the first step, i.e., a new method to solve the relaxed OCP efficiently. We use ideas from [20] as a starting point, adopting the approach of linearizing the dynamics and using matrix exponentials as an integration scheme. We transfer the ideas for evaluating the objective and its derivatives to systems of ODEs of the particular form

$$\dot{x}(t) = f_0\left(x(t)\right) + \sum_{k=1}^{n_\omega} w_k(t) f_k(x(t)) \tag{1}$$

Manuscript received 23 August 2023; revised 26 August 2023; accepted 23 December 2023. Date of publication 9 January 2024; date of current version 28 June 2024. This work was supported in part by Deutsche Forschungsgemeinschaft under Grant 314838170 (RTG 2297 MathCoRe) and Grant 3033031009 (SPP 2331 Power2Methanol). Recommended by Associate Editor E. Trelat. *(Corresponding author: Christoph Plate.)*

Christoph Plate and Sebastian Sager are with the Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, 39106 Magdeburg, Germany, and also with the Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany (e-mail: christoph.plate@ovgu.de; sager@ovgu.de).

Martin Stoll is with the Chemnitz University of Technology, 09111 Chemnitz, Germany (e-mail: martin.stoll@math.tu-chemnitz.de).

Manuel Tetschke is with the Otto von Guericke University Magdeburg, 39106 Magdeburg, Germany (e-mail: tetschke@ovgu.de).

Digital Object Identifier 10.1109/TAC.2024.3351567

that arises from applying the POC reformulation, with n_ω being the number of modes of the switched system and $w_k(t) \in [0,1]$ being the relaxed binary control functions. The drift term f_0 represents the dynamics independent of the choice of the active mode. Also continuous functions entering linearly into the right-hand side are covered by (1). Note that while nonlinearities in integer controls can be reformulated equivalently into form (1), nonlinearities in continuous controls require an additional linearization approach.

In [20], the specific structure of the system dynamics for STO is exploited. There, after linearization, one has structurally $\dot{x}(t) = wAx(t)$ on small intervals for a scalar decision variable $w \in \mathbb{R}_+$ indicating the duration of a mode. Therefore, the solution is given involving a matrix exponential as e^{wAt}. Derivatives with respect to w can be calculated in a straightforward way, using $\frac{\mathrm{d}e^{wA}}{\mathrm{d}w} = Ae^{wA}$. As (1) features the sum of controls and a drift term, the calculation of analytical expressions for derivatives of $e^{(A_0+\sum_{i=1}^{n_\omega} w_i A_i)t}$ becomes more involved for noncommutative matrices A_j. We derive how this can be done efficiently with matrix calculus. As a result, with some numerical overhead compared to STO, second-order optimization becomes also possible for POC control problems.

A. Contributions

In this article, we present a novel algorithm for solving (relaxed) MIOCPs using a direct, *first-discretize-then-optimize* approach. We use existing ideas from an STO algorithm [20] and transfer and extend them to the POC setting. We also generalize the setting to constrained MIOCPs. Our work is implemented in the open source software package SecondOrderPOC.jl, which is available on GitHub.[1] Using the new software, we investigate the performance of the new algorithm by numerical studies of benchmark problems, focusing on a comparison between POC and STO and to a standard way to calculate derivatives, and the possibility to warmstart the STO algorithm with POC solutions.

B. Organization of this Article

The rest of this article is organized as follows. In Section II, we state the problem formulation and explain and derive basic algorithmic ingredients, in particular closed formula for function and derivative evaluation. In Section III, we formulate the main algorithm and explain its implementation in Julia. Numerical results for three benchmark problems and different algorithmic settings are shown in Section IV. Finally, Section V concludes this article. Proofs can be found in the Appendix.

II. Preliminaries

A. Problem Definition

We are interested in optimal control problems of the form

$$\min_{x,w} \int_{t_0}^{t_\mathrm{f}} x(t)^\top \bar{Q} x(t)\, \mathrm{d}t + x(t_\mathrm{f})^\top \bar{E} x(t_\mathrm{f})$$

$$\begin{aligned} \text{s.t.}\quad \dot{x}(t) &= f_0(x(t)) + \sum_{k=1}^{n_\omega} w_k(t) f_k(x(t)) \\ x(t_0) &= x_0, \\ 0 &\geq c(x(t)) \\ 1 &= \sum_{k=1}^{n_\omega} w_k(t) \\ w_k(t) &\in [0,1], \quad k \in [n_\omega] \end{aligned} \tag{POC}$$

on a fixed time horizon $\mathcal{T} = [t_0, t_\mathrm{f}]$ with differential states $x(t) \in \mathbb{R}^{\bar{n}_x}$, initial values $x_0 \in \mathbb{R}^{\bar{n}_x}$, state constraints $c : \mathbb{R}^{\bar{n}_x} \mapsto \mathbb{R}^{\bar{n}_c}$, and a quadratic objective of Bolza type given by symmetric matrices $\bar{Q}, \bar{E} \in \mathbb{R}^{\bar{n}_x \times \bar{n}_x}$. We assume all functions to be sufficiently smooth and Lipschitz continuous. The setting is identical to [20] with two differences: a POC instead of STO reformulation of the dynamical system and the additional possibility to consider state constraints. Here and throughout this article, we use the standard notation $[N] := \{1, 2, \ldots, N\}$ and $[N]_0 := \{0, 1, 2, \ldots, N\}$.

B. Linearization

We follow a direct approach including a discretization of the control functions on N control intervals as well as a linearization of the dynamics using the Taylor expansion. The approach is very closely related to the ideas presented in [20]. In a first step, we introduce the necessary time grids.

Definition 1 (Equidistant time grids): Let $N \in \mathbb{N}$ be the number of control intervals and n_lin be the number of linearization points on each control interval. We define

$$\mathcal{G}_N := [t_0, t_1, t_2, \ldots, t_N = t_\mathrm{f}] \tag{2}$$

with equidistant outer grid size $\Delta t := t_i - t_{i-1}$ and intervals

$$\mathcal{T}_i := [t_{i-1}, t_i], \qquad i \in [N] \tag{3}$$

for a coarse grid partition $\mathcal{T} = \cup_{i\in[N]} \mathcal{T}_i$. For the inner grid, we introduce the fine grid size $\xi := \frac{\Delta t}{n_\mathrm{lin}}$ and write

$$\mathcal{T}_{ij} := [t_{i,j}, t_{i,j+1}], \; i \in [N], j \in [n_\mathrm{lin}] \tag{4}$$

for a fine grid partition $\mathcal{T}_i = \cup_{j\in[n_\mathrm{lin}]} \mathcal{T}_{ij}$ with equidistant inner grid points $t_{i,j} := t_{i-1} + (j-1)\cdot\xi$.

Using the Taylor expansion, we linearize the dynamics

$$\dot{x}(t) = f_0(x(t)) + \sum_{k=1}^{n_\omega} w_k(t) f_k(x(t)) \tag{5}$$

around the state $x_{i,j} = x(t_{i,j})$. Also, we discretize the controls w as piecewise constant functions

$$w_k(t) = w_{ik}, \; t \in \mathcal{T}_i, \; i \in [N] \tag{6}$$

with $w_{ik} \in [0,1], \; i \in [N], k \in [n_\omega]$. This yields

$$\begin{aligned} \dot{x}(t) \approx f_0(x_{i,j}) + \sum_{k=1}^{n_\omega} w_{ik} f_k(x_{i,j}) + \Big[J_{f_0}(x_{i,j}) \\ + \sum_{k=1}^{n_\omega} w_{ik} J_{f_k}(x_{i,j}) \Big] (x(t) - x_{i,j}), \; t \in \mathcal{T}_{ij}. \end{aligned} \tag{7}$$

[1] [Online]. Available: https://github.com/chplate/SecondOrderPOC.jl.git

Here, J_{f_k} represents the Jacobian of the mode f_k, i.e.,

$$J_{f_k}(x) = \frac{\partial f_k(x)}{\partial x}, \quad k \in [n_\omega]_0. \tag{8}$$

If we augment the state variables by a constant state

$$x(t) \leftarrow \left[x(t)^\top, 1\right]^\top \tag{9}$$

and from now on $x(t) \in \mathbb{R}^{n_x}$ with $n_x = \bar{n}_x + 1$, we can write the approximation (7) of the (nonlinear) differential equation as a linear ODE on each interval $\mathcal{T}_{ij}$, i.e.,

$$\dot{x}(t) = A_{i,j}x(t), \; t \in \mathcal{T}_{ij} \tag{10}$$

where the system matrix $A_{i,j} \in \mathbb{R}^{n_x \times n_x}$ (the dependence on ω_i is omitted) follows from reordering the terms in (7) is as follows:

$$\begin{aligned} A_{i,j} :=& \begin{pmatrix} J_{f_0}(x_{i,j}) & f_0(x_{i,j}) - J_{f_0}(x_{i,j})x_{i,j} \\ 0 & 0 \end{pmatrix} \\ &+ \sum_{k=1}^{n_\omega} w_{ik} \begin{pmatrix} J_{f_k}(x_{i,j}) & f_k(x_{i,j}) - J_{f_k}(x_{i,j})x_{i,j} \\ 0 & 0 \end{pmatrix} \\ =:& A^0_{i,j} + \sum_{k=1}^{n_\omega} w_{ik} A^k_{i,j}. \end{aligned} \tag{11}$$

Here, $A^k_{i,j}$ represents the unweighted contribution of the kth mode to the overall linearized dynamic. Considering the increase in dimension due to the linearization, we also augment the matrices $\bar{Q}, \bar{E} \in \mathbb{R}^{\bar{n}_x \times \bar{n}_x}$ and use

$$Q = \begin{pmatrix} \bar{Q} & \\ & 0 \end{pmatrix} \in \mathbb{R}^{n_x \times n_x}, \quad E = \begin{pmatrix} \bar{E} & \\ & 0 \end{pmatrix} \in \mathbb{R}^{n_x \times n_x}. \tag{12}$$

This leads to a slight modification of problem (POC) with a linear ODE, which we will from now on consider

$$\begin{aligned} \min_{x,w} \quad & \int_{t_0}^{t_\mathrm{f}} x(t)^\top Q x(t) \, \mathrm{d}t + x(t_\mathrm{f})^\top E x(t_\mathrm{f}) \\ \text{s.t.} \quad \dot{x}(t) &= A_{i,j}x(t) \\ x(t_0) &= x_0 \\ w_k(t) &\in [0,1], \; k \in [n_\omega] \\ 1 &= \sum_{k=1}^{n_\omega} w_k(t) \\ 0 &\geq c(x(t_i)), \; i \in [N]. \qquad (\mathrm{POC-lin}) \end{aligned}$$

The ODE holds piecewise for $i \in [N], j \in [n_\mathrm{lin}], t \in \mathcal{T}_{ij}$.

C. Exponential Integrator

The linearized ODE (10) can be solved analytically using the matrix exponential of the system matrix (11), i.e.,

$$x(t_{i,j+1}) = e^{A_{i,j}\xi} x(t_{i,j}) \tag{13}$$

noting that the integration step length ξ is constant by virtue of Definition 1. To simplify notation, we define as follows (generally omitting dependencies, e.g., on ω_i).

Definition 2 (Auxiliary matrices): The matrix exponential $\mathcal{E}_{i,j} \in \mathbb{R}^{n_x \times n_x}$ of the matrix $A_{i,j}, i \in [N], j \in [n_\mathrm{lin}]$ is

$$\mathcal{E}_{i,j} := e^{A_{i,j}\xi}. \tag{14}$$

Moreover, we define the matrix $M_{i,j} \in \mathbb{R}^{n_x \times n_x}$

$$M_{i,j} := \int_0^\xi e^{A_{i,j}^\top \eta} Q e^{A_{i,j}\eta} \mathrm{d}\eta \tag{15}$$

as the Lagrange term of the objective on the interval $\mathcal{T}_{ij}$.

Both quantities can be computed through a single matrix exponential due to [13, Th. 1]. For this, a temporary matrix is created and its matrix exponential is computed

$$Z_{i,j} := \exp\left(\xi \cdot \begin{bmatrix} -A_{i,j}^\top & Q \\ 0 & A_{i,j} \end{bmatrix}\right) =: \begin{bmatrix} Z^1_{i,j} & Z^2_{i,j} \\ 0 & Z^3_{i,j} \end{bmatrix}. \tag{16}$$

With (16), the expressions (14) and (15) can be obtained via

$$\begin{aligned} \mathcal{E}_{i,j} &= Z^3_{i,j} \\ M_{i,j} &= Z^{3\top}_{i,j} Z^2_{i,j}. \end{aligned} \tag{17}$$

The procedure of linearization and computation of the next state can be applied iteratively and be generalized as follows.

Definition 3 (State transition matrices Φ): The state transition matrix $\Phi(t_j, t_i)$ for the transition of the state $x(t_i)$ to the state $x(t_j)$ with $t_0 \leq t_i < t_j \leq t_N$ is defined as

$$\Phi(t_j, t_i) := \prod_{m=i+1}^{j} \prod_{n=1}^{n_\mathrm{lin}} \mathcal{E}_{m,n}. \tag{18}$$

Note that the multiplications in Definition 3 need to be multiplications from the left, such that the matrix exponential belonging to the first considered time step stays rightmost. Using these transition matrices Φ, one can now propagate the state over multiple timesteps at once, i.e.,

$$x(t_j) = \Phi(t_j, t_i) x(t_i). \tag{19}$$

Definition 4 (Cost-to-go matrices): For a given grid point $t_a \in \mathcal{G}_N$ with $t_a \leq t_N$, the cost-to-go-matrices P_a, F_a, and S_a are defined as

$$\begin{aligned} F_a &:= \Phi(t_N, t_a)^\top E \Phi(t_N, t_a) \\ P_a &:= \int_{t_a}^{t_N} \Phi(t, t_a)^\top Q \Phi(t, t_a) \, \mathrm{d}t \\ S_a &:= F_a + P_a. \end{aligned} \tag{20}$$

For the computation of $S_a \in \mathbb{R}^{n_x \times n_x}$, we make use of the following recursion, which was also used in [20]. The main difference is our choice of equidistant grids.

Lemma 5 (Recursive evaluation of cost-to-go-matrices): Given matrices $M_{i,j}$ and $\mathcal{E}_{i,j}$ with $i \in [N], j \in [n_{lin}]$ (and for notational convenience using $S_{i+1} := S_{i,n_{lin}+1}$), the following recursion holds:

$$S_{N+1} = E$$

$$S_{i,j} = M_{i,j} + \mathcal{E}_{i,j}^{\top} S_{i,j+1} \mathcal{E}_{i,j} \tag{21}$$

Proof: Follows the proof in [20]. ■

D. Derivatives

The key to solving the problem (POC-lin) via an iterative optimization algorithm is computing the derivatives of the objective function and constraints with respect to the controls w. In this section, the necessary quantities are derived.

1) Derivatives of Matrix Exponentials: To compute derivatives of our integration method, we are combining two methods: the block-triangular method [14] and the complex step method [1]. In a first step, we state [14, Th. 2.1], which is used to compute the derivative of a single matrix exponential with respect to a control w_{ik}.

Theorem 6: Let D denote an open subset of $\mathbb{R}$ or $\mathbb{C}$ and $M_n = \mathbb{C}^{n\times n}$. Let $M_n(D,m)$ denote the subset of matrices with spectrum contained in D and largest Jordan block of size at most m. Let f be $m-1$ times differentiable on D. Let $A(t)$ be differentiable at $t = t_0$ and assume that $A(t) \in M_n(D,m)$ for all t in some neighborhood of t_0. Then

$$\frac{d}{dt} f(A(t))|_{t=t_0} = \left[f \begin{pmatrix} A(t_0) & A'(t_0) \\ 0 & A(t_0) \end{pmatrix} \right]_{1,2} \tag{22}$$

where the subscript 1,2 indicates the (1,2)-block of the result of applying the matrix function.

Proof: See [14, Th. 2.1]. ■

This theorem can be transferred to our setting by identifying f as the exponential function for matrices and $A(t)$ as (11) with the controls w_{ik} taking the place of the variable t, thus

$$\mathcal{E}_{i,j}^{k} := \frac{\partial \mathcal{E}_{i,j}}{\partial w_{ik}} = \left[\exp\left(\xi \cdot \begin{bmatrix} A_{i,j} & A_{i,j}^{k} \\ 0 & A_{i,j} \end{bmatrix} \right) \right]_{1,2}. \tag{23}$$

The same quantity can be computed via the complex step method [1] with some small $h > 0$, avoiding the doubling in size of the matrix exponential as in (23), as

$$\mathcal{E}_{i,j}^{k} = \operatorname{Im} \frac{\exp\left(\xi \cdot (A_{i,j} + ihA_{i,j}^{k})\right)}{h}. \tag{24}$$

For computing second derivatives of a matrix exponential $\mathcal{E}_{i,j}$ with respect to two controls on $\mathcal{T}_i$ different approaches are possible. One could apply Theorem 6 to (23). This does, however, result in a further doubling of the dimensions of the matrix exponential to be calculated.

$$\begin{aligned} \mathcal{E}_{i,j}^{k,l} &:= \frac{\partial \mathcal{E}_{i,j}^{k}}{\partial w_{il}} = \frac{\partial^2 e^{A_{i,j}\xi}}{\partial w_{ik} \partial w_{il}} \\ &= \left[\exp\left(\xi \cdot \begin{bmatrix} A_{i,j} & A_{i,j}^{k} & A_{i,j}^{l} & \frac{\partial A_{i,j}^{k}}{\partial w_{il}} \\ 0 & A_{i,j} & 0 & A_{i,j}^{l} \\ 0 & 0 & A_{i,j} & A_{i,j}^{k} \\ 0 & 0 & 0 & A_{i,j} \end{bmatrix} \right) \right]_{1,4} \\ &= \left[\exp\left(\xi \cdot \begin{bmatrix} A_{i,j} & A_{i,j}^{k} & A_{i,j}^{l} & 0 \\ 0 & A_{i,j} & 0 & A_{i,j}^{l} \\ 0 & 0 & A_{i,j} & A_{i,j}^{k} \\ 0 & 0 & 0 & A_{i,j} \end{bmatrix} \right) \right]_{1,4}. \end{aligned} \tag{25}$$

By combining the two approaches, the renewed doubling in size of the matrix exponential can be avoided. We thus use

$$\mathcal{E}_{i,j}^{k,l} = \operatorname{Im} \left[\frac{\exp\left(\xi \cdot \begin{bmatrix} A_{i,j} + ihA_{i,j}^{k} & A_{i,j}^{l} \\ 0 & A_{i,j} + ihA_{i,j}^{k} \end{bmatrix} \right)}{h} \right]_{1,2}. \tag{26}$$

2) Derivatives of Transition Matrices: With the previous results one can easily calculate the derivative for the transition matrices Φ by applying the product rule

$$\begin{aligned} C_i^k &:= \frac{\partial \Phi(t_i, t_{i-1})}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}} \left(\prod_{l=1}^{n_{\text{lin}}} \mathcal{E}_{i,l} \right) \\ &= \sum_{l=1}^{n_{\text{lin}}} \left[\left(\prod_{m=1}^{l-1} \mathcal{E}_{i,m} \right) \mathcal{E}_{i,l}^{k} \left(\prod_{n=l+1}^{n_{\text{lin}}} \mathcal{E}_{i,n} \right) \right]. \end{aligned} \tag{27}$$

Again, note that the matrix belonging to the first time step needs to stay rightmost. For the second derivative $D_i^{k,p}$ of a given transition matrix $\Phi(t_i, t_{i-1})$ with respect to two controls w_{ik} and w_{ip} on the same interval $\mathcal{T}_i$, one needs to determine the derivative of (27). With (24), (26), and the product rule, this quantity can be computed as in (28) shown at the bottom of the next page, Due to the application of the product rule and the resulting structure of the expression, this quantity is expensive to calculate. However, as discussed later in Section III, using Horner's scheme to compute (27) and (28) can help reducing the number of necessary operations.

Using the cost-to-go-matrices introduced in Definition 4 will later allow us to evaluate the objective function and its derivatives. Therefore, these quantities are investigated in the following. First, we make use of an approximation for computing the derivative of the cost-to-go-matrix P_a. Recall that for any continuous function $f : [a,b] \mapsto \mathbb{R}$ and $n \in \mathbb{N}$ evaluation points on $[a,b]$, it is possible to approximate the integral of f on $[a,b]$ as a Riemann sum, i.e.,

$$\int_a^b f(x)\, \mathrm{d}x \approx \sum_{i=1}^{n} \left(\frac{b-a}{n} \right) \cdot f\left(a + \frac{i \cdot (b-a)}{n} \right). \tag{29}$$

Using the n_{lin} equidistant linearization points, we can approximate the Lagrange term on a given interval $\mathcal{T}_i$ as

$$\begin{aligned} &\int_{t_{i-1}}^{t_i} \Phi(t, t_{i-1})^{\top} Q \Phi(t, t_{i-1})\, \mathrm{d}t \\ &\quad \approx \sum_{j=1}^{n_{\text{lin}}} \xi \cdot \Phi(t_{i,j}, t_{i-1})^{\top} Q \Phi(t_{i,j}, t_{i-1}) \\ &\quad = \xi \cdot \sum_{j=1}^{n_{\text{lin}}} \left(\prod_{k=1}^{j} \mathcal{E}_{i,k} \right)^{\top} Q \left(\prod_{k=1}^{j} \mathcal{E}_{i,k} \right) \end{aligned} \tag{30}$$

noting that the fine grid size ξ is constant. Now, each summand can be differentiated independently using the product rule

$$\begin{aligned} L_i^k &:= \frac{\partial}{\partial w_{ik}} \int_{t_{i-1}}^{t_i} \Phi(t,t_{i-1})^\top Q \Phi(t,t_{i-1})\,\mathrm{d}t \\ &\approx \xi \cdot \frac{\partial}{\partial w_{ik}} \sum_{j=1}^{n_{\text{lin}}} \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right) \\ &= \xi \cdot \sum_{j=1}^{n_{\text{lin}}} \left[\left(\frac{\partial}{\partial w_{ik}} \prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right) \right. \\ &\quad \left. + \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \left(\frac{\partial}{\partial w_{ik}} \prod_{k=1}^{j} \mathcal{E}_{i,k}\right) \right]. \end{aligned} \tag{31}$$

As we will need second derivatives for our optimization, we differentiate (31) again to get

$$\begin{aligned} G_i^{k,l} &:= \frac{\partial}{\partial w_{il}} L_i^k \\ &= \frac{\partial^2}{\partial w_{ik}\partial w_{il}} \int_{t_{i-1}}^{t_i} \Phi(t,t_{i-1})^\top Q \Phi(t,t_{i-1})\,\mathrm{d}t \\ &\approx \xi \cdot \frac{\partial^2}{\partial w_{ik}\partial w_{ip}} \sum_{j=1}^{n_{\text{lin}}} \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right) \\ &= \xi \cdot \sum_{j=1}^{n_{\text{lin}}} \left[\left(\frac{\partial^2}{\partial w_{ik}\partial w_{il}} \prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \prod_{k=1}^{j} \mathcal{E}_{i,k} \right. \\ &\quad \left. + \left(\prod_{k=1}^{j} \mathcal{E}_{i,k}\right)^\top Q \left(\frac{\partial^2}{\partial w_{ik}\partial w_{il}} \prod_{k=1}^{j} \mathcal{E}_{i,k}\right) \right]. \end{aligned} \tag{32}$$

Note that the derivatives of products of matrix exponentials appearing in (31) and (32) can be computed via the same routines as for (27) and (28), respectively. With these introductory results, the first derivatives of the cost-to-go-matrices can be stated, summarized in the following lemma.

Lemma 7: Let F_a and P_a be given as in Definition 4. Then, it holds that

$$\begin{aligned} \frac{\partial F_a}{\partial w_{ik}} &= \Phi(t_{i-1},t_a)^\top \left[C_i^{k\top} F_i \Phi(t_i,t_{i-1}) \right. \\ &\quad \left. + \Phi(t_i,t_{i-1})^\top F_i C_i^k\right] \Phi(t_{i-1},t_a) \end{aligned} \tag{33}$$

and

$$\begin{aligned} \frac{\partial P_a}{\partial w_{ik}} &= \Phi(t_{i-1},t_a)^\top \left(L_i^k + C_i^{k\top} P_i \Phi(t_i,t_{i-1}) \right. \\ &\quad \left. + \Phi(t_i,t_{i-1})^\top P_i C_i^k\right) \Phi(t_{i-1},t_a). \end{aligned} \tag{34}$$

Combining the two results, one obtains

$$\begin{aligned} \frac{\partial S_a}{\partial w_{ik}} &= \frac{\partial F_a}{\partial w_{ik}} + \frac{\partial P_a}{\partial w_{ik}} \\ &= \Phi(t_{i-1},t_a)^\top \left(L_i^k + \Phi(t_i,t_{i-1})^\top S_i C_i^k \right. \\ &\quad \left. + C_i^{k\top} S_i \Phi(t_i,t_{i-1})\right) \Phi(t_{i-1},t_a). \end{aligned} \tag{35}$$

The proof can be found in Appendix A. For second derivatives of S_a, we consider the derivative of (35) with respect to a second control either on the same control interval $\mathcal{T}_i$ or on an interval $\mathcal{T}_j$ with $t_i < t_j$ as follows.

Lemma 8: The second derivatives of S_a as defined in Definition 4 for two grid points $t_i, t_j \in \mathcal{G}_N$ with $t_i < t_j$ are

$$\begin{aligned} \frac{\partial^2 S_a}{\partial w_{ik}\partial w_{il}} &= \Phi(t_{i-1},t_a)^\top \left[G_i^{k,l} + C_i^{l\top} S_i C_i^k \right. \\ &\quad + \Phi(t_i,t_{i-1})^\top S_i D_i^{k,l} \\ &\quad + D_i^{kl\top} S_i \Phi(t_i,t_{i-1}) \\ &\quad \left. + C_i^{k\top} S_i C_i^l\right] \Phi(t_{i-1},t_a) \end{aligned} \tag{36}$$

and

$$\begin{aligned} \frac{\partial^2 S_a}{\partial w_{ik}\partial w_{jl}} &= 2\Phi(t_{j-1},t_a)^\top \left[L_j^l + \Phi(t_j,t_{j-1})^\top S_j C_j^l \right. \\ &\quad \left. + C_j^{l\top} S_j \Phi(t_j,t_{j-1})\right] \Phi(t_{j-1},t_i) C_i^k \cdot \\ &\quad \Phi(t_{i-1},t_a). \end{aligned} \tag{37}$$

The proof is given in Appendix B.

3) Derivatives of Constraints: The one-hot-constraints are linear in the controls, with constant first derivatives

$$\frac{\partial}{\partial w_{jl}} \sum_{k=1}^{n_\omega} w_{ik} = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases}, \quad i,j \in [N], l \in [n_\omega]. \tag{38}$$

The path constraints $c(x(t)) \leq 0$ are only evaluated at the grid points $t_i \in \mathcal{G}_N$. By using (27) and the chain rule, it follows for some $t_i, t_j \in \mathcal{G}_N$ with $t_0 < t_j \leq t_i$

$$\frac{\partial c(x(t_i))}{\partial w_{jk}} = J_c(x(t_i)) \Phi(t_i,t_j) C_j^k x(t_{j-1}),\ k \in [n_\omega] \tag{39}$$

$$\begin{aligned} D_i^{k,p} &:= \frac{\partial C_i^k}{\partial w_{ip}} = \frac{\partial}{\partial w_{ip}} \sum_{l=1}^{n_{\text{lin}}} \left[\left(\prod_{m=1}^{l-1} \mathcal{E}_{i,m}\right) \mathcal{E}_{i,l}^k \left(\prod_{m=l+1}^{n_{\text{lin}}} \mathcal{E}_{i,m}\right)\right] = \sum_{l=1}^{n_{\text{lin}}} \frac{\partial}{\partial w_{ip}} \left(\left(\prod_{m=1}^{l-1} \mathcal{E}_{i,m}\right) \mathcal{E}_{i,l}^k \left(\prod_{m=l+1}^{n_{\text{lin}}} \mathcal{E}_{i,m}\right)\right) \\ &= \sum_{l=1}^{n_{\text{lin}}} \left[\sum_{m=1}^{l-1} \left[\left(\prod_{n=1}^{m-1} \mathcal{E}_{i,n}\right) \mathcal{E}_{i,m}^p \left(\prod_{n=m+1}^{l-1} \mathcal{E}_{i,n}\right)\right] \mathcal{E}_{i,l}^k \left(\prod_{m=l+1}^{n_{\text{lin}}} \mathcal{E}_{i,m}\right) + \left(\prod_{m=1}^{l-1} \mathcal{E}_{i,m}\right) \mathcal{E}_{i,l}^{k,p} \left(\prod_{m=l+1}^{n_{\text{lin}}} \mathcal{E}_{i,m}\right) \right. \\ &\quad \left. + \left(\prod_{m=1}^{l-1} \mathcal{E}_{i,m}\right) \mathcal{E}_{i,l}^k \sum_{m=l+1}^{n_{\text{lin}}} \left[\left(\prod_{n=l+1}^{m-1} \mathcal{E}_{i,n}\right) \mathcal{E}_{i,m}^p \left(\prod_{n=m+1}^{n_{\text{lin}}} \mathcal{E}_{i,n}\right)\right]\right]. \end{aligned} \tag{28}$$

where J_c denotes the Jacobian, i.e., $J_c(x(t)) = \frac{\partial c(x(t))}{\partial x}$.

E. Evaluation of Objective Function and Derivatives

Summarizing the previous results, the following theorem states the expressions for evaluating the objective function of (POC-lin) and its derivatives for a given control w.

Theorem 9: With the aforementioned definitions, given a control $w \in [0,1]^{N \times n_\omega}$ and two grid points $t_i, t_j \in \mathcal{G}_N$ with $t_0 < t_i < t_j \leq t_N$, it holds the following.

1) The objective function of (POC-lin) can be evaluated as
$$J(w) = x_0^\top S_0 x_0. \tag{40}$$
2) The gradient can be computed as
$$\frac{\partial J(w)}{\partial w_{ik}} = x_{i-1}^\top \left(L_i^k + 2\Phi(t_i, t_{i-1})^\top S_i C_i^k\right) x_{i-1}. \tag{41}$$
3) The elements of the Hessian $H_J(w)$ involving two controls on $\mathcal{T}_i$ can be computed as
$$\frac{\partial^2 J(w)}{\partial w_{ik} \partial w_{il}} = x_{i-1}^\top \Big(G_i^{k,l} + 2C_i^{l\top} S_i C_i^k + 2\Phi(t_i, t_{i-1})^\top S_i D_i^{k,l}\Big) x_{i-1}. \tag{42}$$
4) The elements of the Hessian $H_J(w)$ involving one control on $\mathcal{T}_i$ and one on $\mathcal{T}_j$ can be computed as
$$\frac{\partial^2 J(w)}{\partial w_{ik} \partial w_{jl}} = 2x_{j-1}^\top \left(L_j^l + \Phi(t_j, t_{j-1})^\top S_j C_j^l + C_j^{l\top} S_j \Phi(t_j, t_{j-1}^\top)\right) \cdot \Phi(t_{j-1}, t_i) C_i^k x_{i-1}. \tag{43}$$

Proof: Summary of the previous results, especially Lemmas 7 and 8. ∎

As a novel feature compared to SwitchTimeOpt.jl, in SecondOrderPOC.jl also state constraints
$$c(x(t)) \leq 0$$
for differentiable $c : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_c}$ can be considered. Their derivatives are calculated in a similar way as the derivatives of the objective function.

F. Error Estimation

Piecewise linearization and integration via matrix exponentials introduces an error. A detailed error analysis for exponential integration can be found in [16]. Essentially, the maximum error depends on the right-hand sides through their Lipschitz constants and bounds to the norm of their second derivatives as well as the maximum discretization stepsize. In numerical studies, the authors found the piecewise linearization method to be more accurate than the second-order explicit improved Euler's method (also known as Heun's method) and the trapezoidal rule and less accurate compared to the explicit fourth-order Runge–Kutta method, all using fixed stepsizes. The integration error also carries over to the computation of sensitivities. As Lipschitz constants are difficult to assess in practice, we compared the first- and second-order sensitivities of (POC-lin) numerically. We did this exemplarily for the well-known Lotka–Volterra fishing problem, which is given by

$$\begin{aligned} \min_{x,w} \quad & \int_0^{12} (x_1(t)-1)^2 + (x_2(t)-1)^2 \, \mathrm{d}t \\ \text{s.t.} \quad & \dot{x}_1(t) = x_1(t) - x_1(t)x_2(t) - w(t)c_1 x_1(t) \\ & \dot{x}_2(t) = -x_2(t) + x_1(t)x_2(t) - w(t)c_2 x_2(t) \\ & x(0) = [0.5, 0.7] \\ & w_1(t) \in [0,1] \qquad \text{(Lotka)} \end{aligned}$$

with parameters $c_1 = 0.4$ and $c_2 = 0.2$. The control goal for the Lotka–Volterra system is to reach the steady state $x_1 = x_2 = 1$. A comparison of an explicit fourth-order Runge–Kutta integrator for the nonlinear dynamics and exponential integration resulted in consistent values, as shown in Figs. 1 and 2. The impact of the inaccuracies on the performance of optimization is investigated in Section IV-D.

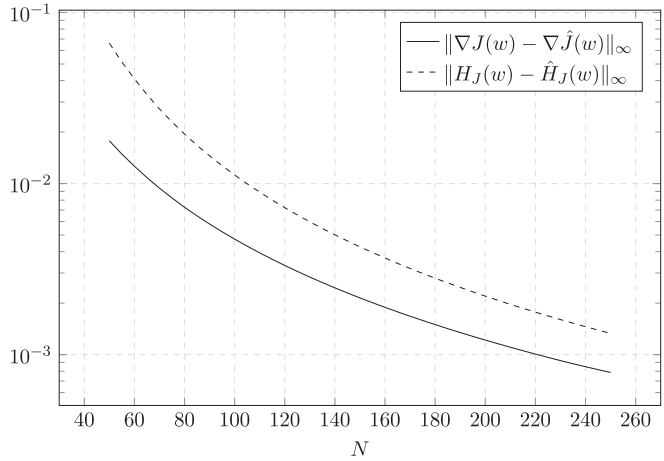


Fig. 1. Maximum deviation of first- and second-order derivatives of (Lotka) with respect to controls over the number of control intervals N for $n_{\text{lin}} = 1$. Our approach of integrating the piecewise linearized dynamics via matrix exponentials and evaluating derivatives as proposed in Lemmas 7 and 8 is compared with integrating the nonlinear dynamics using an adaptive fourth-order Runge–Kutta method and computing sensitivities using forward-mode of automatic differentiation.

III. Implementation

Our implementation is based on SwitchTimeOpt.jl [20], a software package implemented in the programming language Julia. It consists mainly of the necessary computations to evaluate the objective function and its derivatives. These evaluations must then be passed to a suitable NLP solver, which is made possible in a simple way via the Julia package MathOptInterface.jl [12]. With this approach, we can interface our code to a variety of NLP solvers, for example, IPOPT [21] or KNITRO [5]. The procedure of linearizing and integrating the dynamics is shown in Algorithm 1. For a given iterate, the differential states as well as auxiliary matrices like the system matrices $A_{i,j}$ and their matrix exponentials are calculated and temporarily saved as they are necessary for the subsequent computation of derivatives.

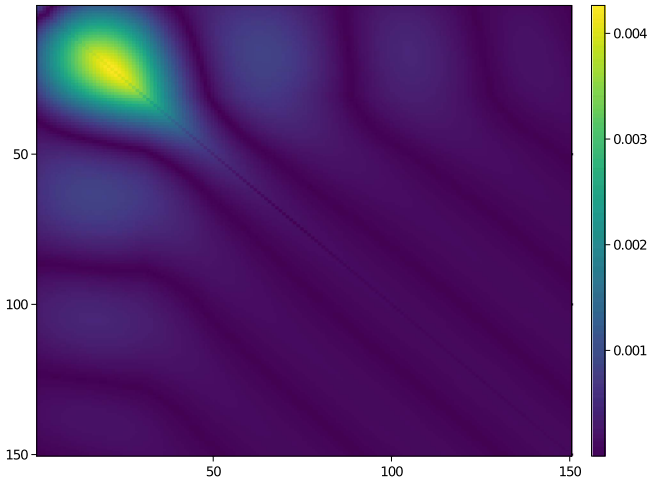


Fig. 2. Absolute error in the second-order sensitivities of (Lotka) of our approach as proposed in Lemma 8. Shown here is the case $N = 150$, compare also Fig. 1, with a maximum error of 0.0043.

Algorithm 1: Linearize and Propagate States.

Initialize $x_{1,1} = x_0$, ▷ Fixed initial state
function LINMATEXPPROP
  for $i \in [N]$ **do**
    for $j \in [n_{\text{lin}}]$ **do**
      $A_{i,j} \leftarrow (11)$ ▷ Linearize dynamic
      $Z_{i,j} \leftarrow (16)$ ▷ Matrix exponential
      $\mathcal{E}_{i,j}, M_{i,j} \leftarrow (17)$
      $x_{i,j+1} \leftarrow \mathcal{E}_{i,j} x_{i,j}$ ▷ Compute next state
    end for
  end for
  return $x_{i,j}, A_{i,j}, \mathcal{E}_{i,j}, M_{i,j}, \quad i \in [N], j \in [n_{\text{lin}}]$
end function

Algorithm 2 shows the necessary computations to perform one iteration of an NLP solver. The main steps consist of linearizing and integrating the dynamics, computing the cost-to-go-matrices using Lemma 5 and the derivatives of the individual matrix exponentials using Theorem 6. Having computed these, the composite expressions (e.g., the transition matrices Φ) can be differentiated using the product rule. Finally, the objective and its derivatives can be evaluated via Theorem 9. These evaluated quantities are passed to the solver that performs an update on the current iterate. This process is iterated until a convergence criterion is fulfilled. We further improved the efficiency of the derivative calculation as follows.

1) Horner's Scheme for Evaluating Derivatives: The derivatives of products of matrix exponentials appear in several places throughout Section II, e.g., in (27), (28), (31), and (32). They can be calculated in multiple ways. We use Horner's scheme [11], reducing the number of necessary matrix multiplications. We illustrate this procedure in Algorithm 3 with the example of (27).

2) Joint Evaluation of First and Second Derivatives: The first and second derivatives of our matrix exponentials (14) can be computed jointly, reducing the number of operations. This

Algorithm 2: Compute $J(w), \nabla J(w), H_J(w)$.

Input w ▷ Current iterate
function COMPUTECOSTFUNCTIONANDDERIVATIVES
  Precomputations:
  $x, A, \mathcal{E}, M \leftarrow$ LINMATEXPPROP ▷ Algorithm 1
  $S \leftarrow$ COMPUTES ▷ Lemma 5
  $\nabla\mathcal{E}, \nabla^2\mathcal{E} \leftarrow (26)$
  $\Phi \leftarrow (18)$ ▷ Transition matrices
  $C, D \leftarrow (27), (28)$
  $L, G \leftarrow (31), (32)$ Evaluation: ▷ Theorem 9
  $J(w) \leftarrow (40)$
  $\nabla J(w) \leftarrow (41)$
  $H_J(w) \leftarrow (42), (43)$
end function

Algorithm 3: Horner's Scheme for Computing (27).

function COMPUTECHORNER
  for $i \in [N]$ **do**
    for $j \in [n_\omega]$ **do**
      $P \leftarrow \mathcal{E}_{i,1}$
      $S \leftarrow \mathcal{E}^j_{i,1}$
      for $k = 2 \dots n_{lin}$ **do**
        $S \leftarrow \mathcal{E}_{i,k} S + \mathcal{E}^j_{i,k} P$
        $P \leftarrow \mathcal{E}_{i,k} P$
      end for
      $C^j_i \leftarrow S$
    end for
  end for
end function

is due to the structure of the appearing matrices in (26) and the power series definition of the matrix exponential. In addition to extracting the (1,2)-block in (26), and thereby, obtaining the second derivative $\mathcal{E}^{k,l}_{i,j}$, extracting the (1,1)-block obviously gives the first derivative $\mathcal{E}^k_{i,j}$.

IV. NUMERICAL RESULTS

In this section, we present some numerical results. First, we compare the efficacy of the package SwitchTimeOpt.jl [20] with our approach for three test problems and show empirically that our method is more likely to converge with optimality and needs fewer iterations on average. Second, we compare the performance of our approach to a similar setting, but with derivatives calculated with an adaptive integrator instead of exponential integration. Third, we study the effect of using the rounded solution of (POC-lin) as an initial guess for the problem in STO formulation in order to mitigate the convergence problems.

The SwitchTimeOpt.jl code used for the numerical experiments is basically identical with [20], but was adapted to the Julia version 1.7. It can be found on GitHub.[2]

[2][Online]. Available: https://github.com/chplate/SwitchTimeOpt.jl.git.

A. Test Problems

Our three test problems were also used in [20] and are contained in the MIOCP benchmark library [17]. In addition to (Lotka), we study the following two problems.

1) Egerstedt: This problem has switched linear dynamics. It was first proposed in [6] with two modes and extended in [18] with a third mode. It reads

$$\begin{aligned}\min_{x,w} \quad & \int_0^1 \|x(t)\|_2^2 \,\mathrm{d}t \\ \text{s.t.} \quad & \dot{x}(t) = \sum_{i=1}^{3} w_i(t) A_i x(t) \\ & x(0) = [0.5, 0.5], \\ & w_i(t) \in [0,1], \quad i \in [3] \\ & \sum_{i=1}^{3} w_i(t) = 1 \qquad \text{(Egerstedt)}\end{aligned}$$

where the right-hand side of the differential equation is described by the three-system matrices

$$A_1 = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}, \text{and } A_3 = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

2) Tank: The double-tank control problem

$$\begin{aligned}\min_{x,w} \quad & \int_0^{10} (x_2(t) - x_3(t))^2 \,\mathrm{d}t \\ \text{s.t.} \quad & \dot{x}_1(t) = -\sqrt{x_1(t)} + w_1(t)c_1 + w_2(t)c_2 \\ & \dot{x}_2(t) = \sqrt{x_1(t)} - \sqrt{x_2(t)} \\ & \dot{x}_3(t) = -0.05 \\ & x(0) = [2,2,3] \\ & w_i(t) \in [0,1], \quad i \in [2] \\ & 1 = w_1(t) + w_2(t) \quad \text{(Tank)}\end{aligned}$$

has parameters $c_1 = 1, c_2 = 2$ representing two possible flow rates into the upper tank x_1. The deviation of the level of the lower tank x_2 from the reference level x_3 shall be minimized. The problem was investigated in [3], however for a constant reference level x_3. The specific choice of right-hand side for x_3 goes back to SwitchTimeOpt.jl [20].

B. Discretization and Settings

We solved each problem for different discretizations. Five different numbers of linearization points on $\mathcal{T}$

$$n_{grid} \in \{100, 200, 300, 400, 500\}$$

were investigated. For SwitchTimeOpt.jl, we also varied the number of switches, ranging from $N = 1$ up to $N = 50$, i.e.,

$$N_{STO} \in \{1, 2, \ldots, 50\}.$$

In this scenario, allowing for $n \in \mathbb{N}$ switches, gives $n+1$ intervals of variable length each with a specific right-hand side. The choice of the right-hand side for each interval is given by the switching sequence $\sigma : [N+1] \mapsto [n_\omega]$. It is chosen such that it repeats a predefined order, i.e.,

$$\sigma(i) = 1 + (i-1) \bmod n_\omega, \quad \text{for } i \in [N+1].$$

If $n_\omega = 1$, we alternate between $w = 1$ and $w = 0$, beginning with $w = 1$. This setup gives a total of 250 instances for each problem in the STO formulation.

For POC, we varied N in the range between $N = 5$ and $N = 250$ with steps of five, more specifically

$$N_{\text{POC}} \in \{5, 10, 15, \ldots, 250\}.$$

Note that we use N in two different contexts. In the context of problems in the STO formulation, N describes the number of allowed switches between the modes. In the context of POC, N represents the number of control intervals as introduced in Definition 1.

The parameter n_{lin} was chosen such that the discretization is comparable to the five stages given by n_{grid}. Therefore, each n_{lin} was calculated via

$$n_{\text{lin}}(N) = \left\{ \max\left(1, \left\lfloor \frac{n_t}{N} \right\rceil\right), \quad n_t \in n_{\text{grid}} \right\}.$$

Here, we used

$$\lfloor x \rceil = \begin{cases} \lceil x \rceil, & x - \lfloor x \rfloor \geq 0.5 \\ \lfloor x \rfloor, & x - \lfloor x \rfloor < 0.5 \end{cases}$$

as a notation for rounding to the nearest integer. In total, this approach yields 189 distinct instances for each problem in POC formulation with a comparable effort for numerical integration as in the STO formulation. All instances were solved with IPOPT [21] version 3.14.4 with tolerance 10^{-6} and limits of at most 500 iterations and at most 1000 s computation time.

C. Comparison POC Versus STO

In Table I, we compare results obtained using SwitchTimeOpt.jl and SecondOrderPOC.jl, focusing on

1) rate of success, i.e., percentage of optimally solved instances;
2) number of iterations for optimally solved instances;
3) time per iteration for optimally solved instances.

First, it shows the distribution of termination status of all problem instances for POC and STO. Evidently, the rate of success for STO strongly depends on the chosen problem. While (Egerstedt) could be solved in nearly all cases, roughly every other instance of (Tank) could not be solved. For (Lotka), only one in five instances terminated successfully. In contrast, all three problems have success rates greater than 90% in POC formulation. Second, it lists the best objectives among the optimally solved instances as well as the maximum and median deviation from it. The best objectives among the optimally solved instances of (Tank) and (Egerstedt) are the same for POC and STO up to three digits. For (Lotka), the best objective is slightly lower for POC. More interestingly, the maximal deviation from the best objective among the optimally solved instances for POC is at most 0.29%. In contrast, for STO this value is 599.21%. This suggests that if one does not know the approximate structure of the optimal solution (e.g., number of switches) and does not

TABLE I
COMPARISON OF PERFORMANCE INDICATORS FOR POC AND STO AND ALL THREE TEST INSTANCES

Problem	Formulation	Termination status			Success rate	Best objective	Deviation from best objective (in %)		Median time	Median number of iterations
		Solved	Error	Limit			Maximum	Median		
Egerstedt	POC	179	9	1	94.7%	0.9891	**0.17**	0.005	6.05	**17**
	STO	249	1	-	**99.6%**	0.9891	32.8	**0.003**	**0.49**	36
Lotka	POC	183	-	6	**96.8%**	1.3442	**0.29**	**0.039**	2.22	**22**
	STO	50	6	194	20%	1.3449	599.21	0.361	**0.36**	26.5
Tank	POC	186	2	1	**98.4%**	1.858	**0.05**	0.0065	2.65	**11**
	STO	140	9	101	56%	1.858	111.82	**0.0035**	**0.22**	13

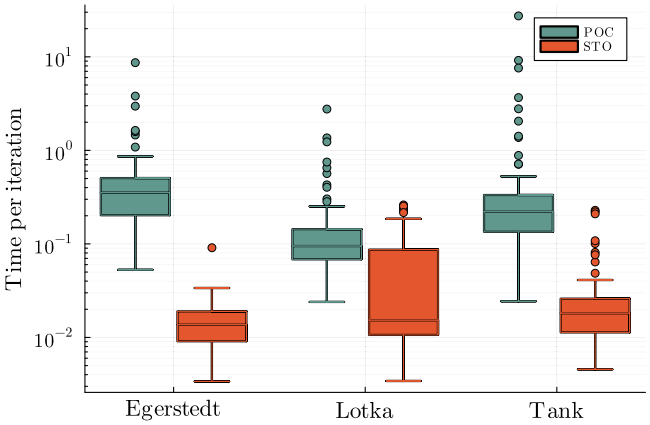


Fig. 3. Comparison of time per iteration of solved instances of the three test problems for POC and STO. Iterations are cheaper by roughly one order of magnitude for the STO approach, mainly due to simpler formulas to calculate the derivatives.

provide this prior knowledge when initializing the problem in STO formulation, the found solution can be arbitrarily far from the optimum. Third, the computation times among the optimally solved instances of the three problems are roughly one order of magnitude larger for POC compared to STO, whereas the numbers of necessary iterations is consistently lower.

In Fig. 3, the distribution of computation times per iteration for all successfully solved instances of the three test problems are shown. Confirming the impression from Table I, the POC approach is taking roughly one order of magnitude more time per iteration for the test problems. This behavior can be expected as it requires the computation of many additional matrix exponentials in order to calculate derivatives. In SwitchTimeOpt.jl, the evaluation of objective and its derivatives is less expensive. There, the main computational burden lies in integrating the dynamical system, after which first derivatives can be computed cheaply by evaluating only a few matrix products and additions. Also, due to shared terms in the expressions, computing second derivatives comes at no significant additional cost.

In Fig. 4, the number of iterations for all three test problems are shown. Two observations can be made. First, the vast majority of instances are successfully solved in POC, whereas several instances are terminating due to resource limits in STO formulation. This is especially apparent when the number of allowed switches is large. Moreover, the number of allowed switches clearly influences whether SwitchTimeOpt.jl is able to converge at all. Second, the POC approach needs less iterations on average for converging successfully, again confirming the data in Table I.

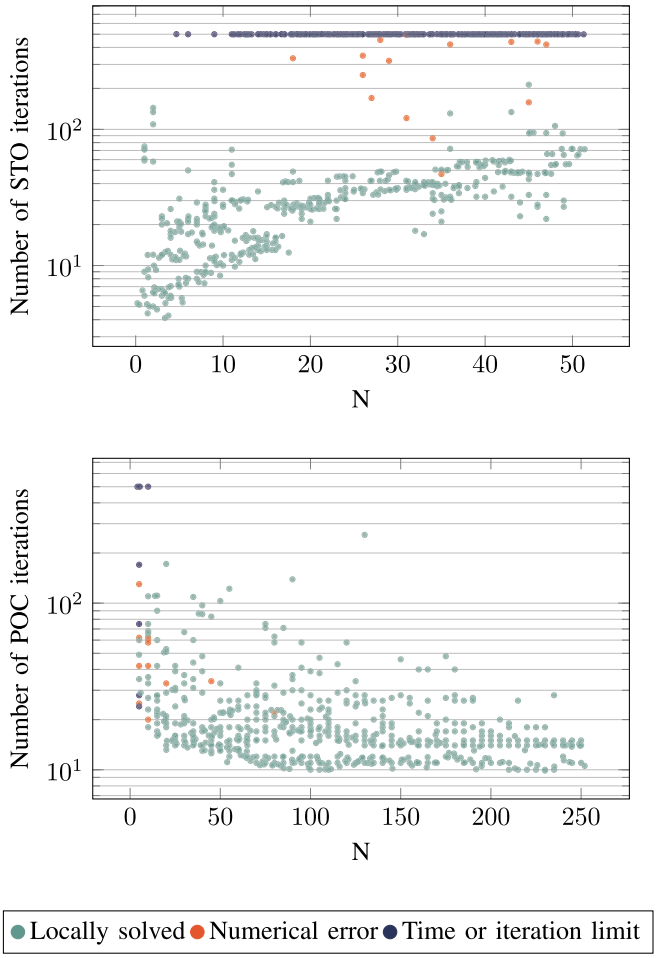


Fig. 4. Iterations and termination status over number of control intervals N for all instances of the three test problems in (top) STO and (bottom) POC formulation. The POC formulation results in a larger number of locally solved instances.

D. Comparison With CasADi

CasADi [1] is an open-source software for solving nonlinear optimization and optimal control problems, with efficient automatic differentiation capabilities and a direct interface to NLP solvers, such as Ipopt, and integrators from the Sundials suite [10]. We compared the performance of CasADi using direct single shooting with the adaptive ODE solver CVODES with that of our approach on the three test problems for the same control discretizations and solver options as described in Section IV-B. Results for our approach were averaged among the successful instances with the same number of control intervals but differences in the number of overall linearization points. Fig. 5 shows

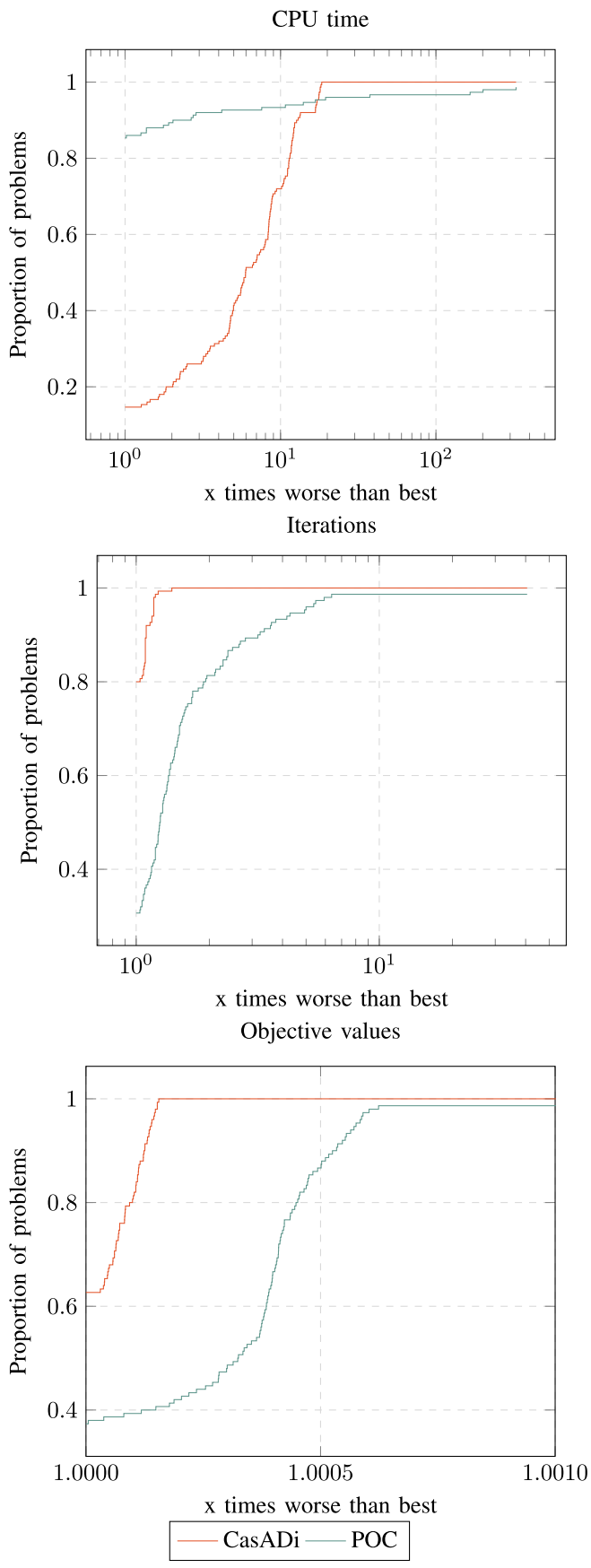


Fig. 5. Performance profiles comparing the proposed exponential integration approach (POC) and solution via CasADi [2] and the adaptive integrator CVODES (Sundials).

the results of this comparison. The computation times are lower for most instances with our approach, despite an increase in the number of iterations. The latter may be due to the inaccuracy introduced by the piecewise linearization of the ODE. The best found objectives are practically indistinguishable. We speculate that the exponential integration approach may perform better for larger problems with more derivatives that need to be calculated per iteration. In addition, it seems plausible (and is supported by a comparison between the three benchmarks, data not shown) that more iterations are needed for the exponential integration approach for more nonlinear systems.

E. Warmstart

As a second numerical study, we investigate the benefits of using the rounded solution of (POC-lin) to warmstart the problem in STO formulation and solve it with SwitchTimeOpt.jl. For this, we used the same problem instances for POC as described in Section IV-C and proceeded as follows:

1) solve problem in POC formulation; proceed if and only if the problem terminates with optimality;
2) use sum-up-rounding to get switching times τ^* and integer controls w_{SUR};
3) solve problem in STO formulation; initialize the problem with integer controls w_{SUR} and switching times τ^*.

We will call this approach STOWS in the following. The instances of STOWS are not directly corresponding to the instances of STO from Section IV-C in the sense that they do not have the same number of switches or the same predefined switching sequence σ. This is due to the fact that for STOWS these parameters are determined by the solution obtained from applying sum-up-rounding to the solution of (POC-lin). Nevertheless, it makes sense comparing the two approaches STO and STOWS in terms of success rates and the quality of the found solutions. Note that it would also be possible to calculate the solutions of steps 1) and/or 3) in a different way, e.g., using a standard way to calculate derivatives as in the previous section. However, we expect the new algorithms based on exponential integration to be competetive algorithms in both cases. Fig. 6 gives an overview over computation times and termination status for all problems and approaches.

In the case of (Egerstedt), all instances solved in the initialization step with POC could also be solved with the STOWS approach. The best found objective was 0.9891, the same as with STO. However, the maximal deviation from that value could be reduced to only 0.52%, with a median deviation of 0.002%. Also, the computation time and number of iterations could be reduced to 0.16 s and 18 iterations, respectively.

Concerning (Lotka), we found a success rate of 41.5% among the instances of STOWS, which is a doubling compared to STO. The best found objective was the same as with STO and again, the maximal and median deviation from the best objective could be drastically reduced to 2.92% and 0.16%, respectively. It is also interesting to see that the structure of the relaxed solution (starting and ending with control $w = 0$) leads to only even numbers of switches.

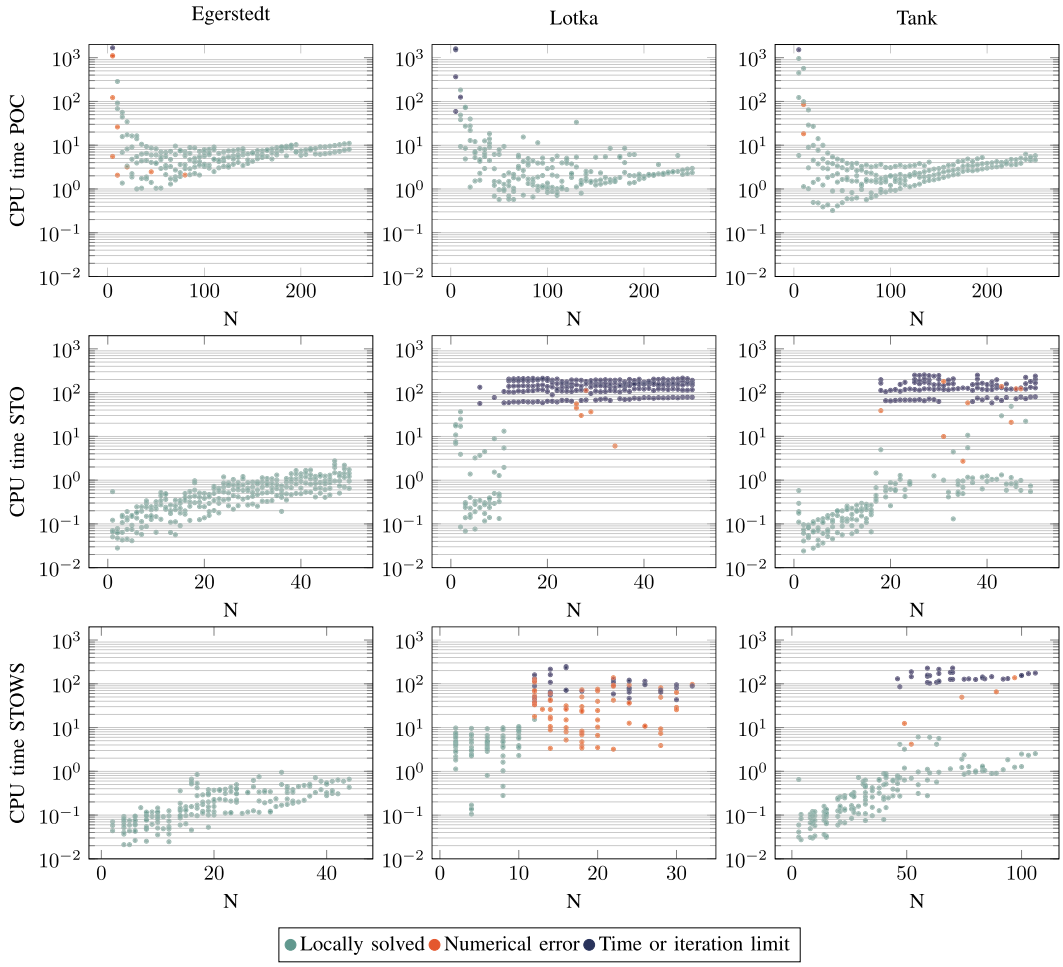


Fig. 6. Solution times and status for all problems and approaches. The columns correspond to the three test problems, the rows to the approaches POC, STO, and STOWS. Times for STOWS do not take into account the initialization step. In addition to the advantages of the warm started STOWS approach with respect to the quality of local solutions as discussed in Section IV-E, an improvement with respect to solution time and status is visible for the Tank instances.

For (Tank), we measured a success rate of 79.6% with STOWS, again a significant increase in comparison to the 56% with the standard initialization of STO. With our initialization, now instances with up to approximately 50 switches could be solved consistently, as Fig. 6 illustrates. The best objective is again the same as with STO, with the worst objective only 0.38% higher, in contrast to a maximal deviation of more than 100% with STO. The median computation time and median number of iterations among the solved instances are slightly larger with 0.27 s and 15 iterations, respectively.

V. Conclusion

In this article, we presented an algorithm for solving the partial outer convexification reformulation of mixed-integer optimal control problems. The algorithm uses a *first-discretize-then-optimize* approach employing exponential integrators, which was adapted from an algorithm for solving switching time optimization problems. In particular, we derived closed formula for the objective and derivative evaluation and implemented them in an open-source Julia package. The efficient calculation of second derivatives motivates the usage of Newton's method to solve STO and POC problems. Newton's method was superior to a BFGS approach (results not included in this article). A second-order method is particularly promising when the objective functions are quadratic and convex, as in our setting. It should also be transferable to other outer convex structures as surveyed in [15].

In our numerical study, we showed that already for small-scale benchmark problems, there is a considerable speed-up compared to standard ways of derivative calculation when the overall computational time is considered. We expect this advantage to increase in the number of degrees of freedom, but to decrease for more nonlinear systems.

In addition, we showed the effectiveness of our method and compared convergence behavior and quality of solutions to the original switching time optimization implementation. The results of our study indicate that the two methods work well together when combined, i.e., using POC solutions as initializations for SwitchTimeOpt.jl helped improve the quality of found solutions. Nevertheless, still for many instances in two out of three of our test problems, no convergence was obtained within the time limit, regardless of the initialization. Thus, future work is necessary to find better STO formulations or tailored STO algorithms. Performing only a fixed number of iterations as done in [20], but initialized with the POC solution, might be

one heuristic approach to obtain good solutions. A comparison between exponential integration and standard derivatives for STO is lacking and might give additional insight. In view of the higher per-iteration costs of POC, one might investigate whether initializing SwitchTimeOpt.jl with not fully converged POC solutions reduces the overall computational time.

There are other directions for future work. An integration scheme of higher order might help to tackle problems with "more nonlinear" dynamics. The implementation could be parallelized efficiently. The single shooting approach could be matched by a structure-exploiting multiple shooting version.

Appendix A
Proof of Lemma 7

Proof: We begin by analyzing the component P_a.

$$\begin{aligned}
\frac{\partial P_a}{\partial w_{ik}} &= \frac{\partial}{\partial w_{ik}} \left(\int_{t_a}^{t_N} \Phi(t,t_a)^\top Q \Phi(t,t_a)\, \mathrm{d}t \right) \\
&= \frac{\partial}{\partial w_{ik}} \left(\int_{t_a}^{t_{i-1}} \Phi(t,t_a)^\top Q \Phi(t,t_a)\, \mathrm{d}t \right. \\
&\quad + \int_{t_{i-1}}^{t_i} \Phi(t,t_a)^\top Q \Phi(t,t_a)\, \mathrm{d}t \\
&\quad \left. + \int_{t_i}^{t_N} \Phi(t,t_a)^\top Q \Phi(t,t_a)\, \mathrm{d}t \right) \\
&= \frac{\partial}{\partial w_{ik}} \left(\Phi(t_{i-1},t_a)^\top \cdot \right. \\
&\quad \int_{t_{i-1}}^{t_i} \Phi(t,t_{i-1})^\top Q \Phi(t,t_{i-1})\, \mathrm{d}t \cdot \Phi(t_{i-1},t_a) \\
&\quad + \Phi(t_i,t_a)^\top \cdot \\
&\quad \left. \int_{t_i}^{t_N} \Phi(t,t_i)^\top Q \Phi(t,t_i)\, \mathrm{d}t \Phi(t_i,t_a) \right) \\
&= \Phi(t_{i-1},t_a)^\top \cdot \\
&\quad \frac{\partial}{\partial w_{ik}} \left(\int_{t_{i-1}}^{t_i} \Phi(t,t_{i-1})^\top Q \Phi(t,t_{i-1})\, \mathrm{d}t \right) \cdot \\
&\quad \Phi(t_{i-1},t_a) \\
&\quad + \Phi(t_{i-1},t_a)^\top C_i^{k\top} \cdot \\
&\quad \int_{t_i}^{t_N} \Phi(t,t_i)^\top Q \Phi(t,t_i)\, \mathrm{d}t \cdot \Phi(t_i,t_a) \\
&\quad + \Phi(t_i,t_a)^\top \cdot \\
&\quad \int_{t_i}^{t_N} \Phi(t,t_i)^\top Q \Phi(t,t_i)\, \mathrm{d}t \cdot C_i^k \Phi(t_{i-1},t_a) \\
&= \Phi(t_{i-1},t_a)^\top \cdot \\
&\quad \frac{\partial}{\partial w_{ik}} \left(\int_{t_{i-1}}^{t_i} \Phi(t,t_{i-1})^\top Q \Phi(t,t_{i-1})\, \mathrm{d}t \right) \cdot \\
&\quad \Phi(t_{i-1},t_a) \\
&\quad + \Phi(t_{i-1},t_a)^\top C_i^{k\top} P_i \Phi(t_i,t_a) \\
&\quad + \Phi(t_i,t_a)^\top P_i C_i^k \Phi(t_{i-1},t_a) \\
&\approx \Phi(t_{i-1},t_a)^\top \left(L_i^k + C_i^{k\top} P_i \Phi(t_i,t_{i-1}) \right. \\
&\quad \left. + \Phi(t_i,t_{i-1})^\top P_i C_i^k \right) \Phi(t_{i-1},t_a). \qquad (44)
\end{aligned}$$

In the first equality, we use Definition 4. In the second equality, we split up the integral into three parts. Then, we bring $\Phi(t_i,t_a)$ and $\Phi(t_{i-1},t_a)$ outside of the integrals since they do not depend on t. Also, we conclude that the last term is zero as w_{ik} only has an impact on the interval $[t_{i-1},t_i]$. After that we bring the constant $\Phi(t_{i-1},t_a)$ outside of the derivative operator and apply the product rule, (27) and (31), which concludes the proof.

$$\begin{aligned}
\frac{\partial F_a}{\partial w_{ik}} &= \frac{\partial}{\partial w_{ik}} \left(\Phi(t_N,t_a)^\top E \Phi(t_N,t_a) \right) \\
&= \frac{\partial}{\partial w_{ik}} \left(\Phi(t_i,t_a)^\top (\Phi(t_N,t_i)^\top E \Phi(t_N,t_i)) \cdot \right. \\
&\quad \left. \Phi(t_i,t_a) \right) \\
&= \frac{\partial}{\partial w_{ik}} \left(\Phi(t_i,t_a)^\top F_i \Phi(t_i,t_a) \right) \\
&= \frac{\partial}{\partial w_{ik}} \left(\Phi(t_{i-1},t_a)^\top \Phi(t_i,t_{i-1})^\top F_i \cdot \right. \\
&\quad \left. \Phi(t_i,t_{i-1}) \Phi(t_{i-1},t_a) \right) \\
&= \Phi(t_{i-1},t_a)^\top \frac{\partial \Phi(t_i,t_{i-1})^\top}{\partial w_{ik}} F_i \Phi(t_i,t_a) \\
&\quad + \Phi(t_i,t_a)^\top F_i \frac{\partial \Phi(t_i,t_{i-1})}{\partial w_{ik}} \Phi(t_{i-1},t_a) \\
&= \Phi(t_{i-1},t_a)^\top C_i^{k\top} F_i \Phi(t_i,t_a) \\
&\quad + \Phi(t_i,t_a)^\top F_i C_i^k \Phi(t_{i-1},t_a) \\
&= \Phi(t_{i-1},t_a)^\top \left(C_i^{k\top} F_i \Phi(t_i,t_{i-1}) \right. \\
&\quad \left. + \Phi(t_i,t_{i-1})^\top F_i C_i^k \right) \Phi(t_{i-1},t_a). \qquad (45)
\end{aligned}$$

We start again with the Definition 4. In the second equation, we split up the transition matrices Φ and identify the inner expression as F_i. Then, we note that F_i does not depend on w_{ik}, and apply the product rule together with (27). By adding (44) and (45), we conclude the overall proof.

Appendix B
Proof of Lemma 8

Proof: We introduce $\Phi_{t_a}^{t_i} = \Phi(t_i,t_a)$ as a short notation for the transition matrix from t_a to t_i.

$$\begin{aligned}
\frac{\partial^2 S_a}{\partial w_{ik} \partial w_{il}} &= \frac{\partial}{\partial w_{il}} \left[\Phi_{t_a}^{t_{i-1}\top} \left(L_i^k + \Phi_{t_{i-1}}^{t_i\top} S_i C_i^k \right. \right. \\
&\quad \left. \left. + C_i^{k\top} S_i \Phi_{t_{i-1}}^{t_i} \right) \Phi_{t_a}^{t_{i-1}} \right] \\
&= \Phi_{t_a}^{t_{i-1}\top} \left[\frac{\partial L_i^k}{\partial w_{il}} + C_i^{l\top} S_i C_i^k \right. \\
&\quad + \Phi_{t_{i-1}}^{t_i\top} S_i D_i^{k,l} + D_i^{kl\top} S_i \Phi_{t_{i-1}}^{t_i} \\
&\quad \left. + C_i^{k\top} S_i C_i^l \right] \Phi_{t_a}^{t_{i-1}}
\end{aligned}$$

$$= \Phi_{t_a}^{t_{i-1}\top}\Big[G_i^{k,l} + C_i^{l\top} S_i C_i^k + \Phi_{t_{i-1}}^{t_i\top} S_i D_i^{k,l} + D_i^{kl\top} S_i \Phi_{t_{i-1}}^{t_i} + C_i^{k\top} S_i C_i^l\Big] \Phi_{t_a}^{t_{i-1}}. \quad (46)$$

Here, we started with the result from Lemma 7 and applied the product rule as well as (27), (28), and (32). In the case when we take the derivative with respect to a control w_{jl} from a control interval $\mathcal{T}_j$ with $t_j > t_i$, we can write $\frac{\partial^2 S_a}{\partial w_{ik}\partial w_{jl}}$ as

$$\begin{aligned}
&\frac{\partial\left[\Phi_{t_a}^{t_{i-1}\top}\left(L_i^k + \Phi_{t_{i-1}}^{t_i\top} S_i C_i^k + C_i^{k\top} S_i \Phi_{t_{i-1}}^{t_i}\right)\Phi_{t_a}^{t_{i-1}}\right]}{\partial w_{jl}} \\
&= \Phi_{t_a}^{t_i\top}\frac{\partial S_i}{\partial w_{jl}} C_i^k \Phi_{t_a}^{t_{i-1}} + \Phi_{t_a}^{t_{i-1}\top} C_i^{k\top}\frac{\partial S_i}{\partial w_{jl}}\Phi_{t_a}^{t_i} \\
&= \Phi_{t_a}^{t_i\top}\Big[\Phi_{t_i}^{t_{j-1}\top}\Big(L_j^l + \Phi_{t_{j-1}}^{t_j\top} S_j C_j^l \\
&\quad + C_j^{l\top} S_j \Phi_{t_{j-1}}^{t_j}\Big)\Phi_{t_i}^{t_{j-1}\top}\Big] C_i^k \Phi_{t_a}^{t_{i-1}} \\
&\quad + \Phi_{t_a}^{t_{i-1}\top} C_i^{k\top}\Big[\Phi_{t_i}^{t_{j-1}\top}\Big(L_j^l + \Phi_{t_{j-1}}^{t_j\top} S_j C_j^l \\
&\quad + C_j^{l\top} S_j \Phi_{t_{j-1}}^{t_j}\Big)\Phi_{t_i}^{t_{j-1}}\Big]\Phi_{t_a}^{t_i} \\
&= 2\Phi_{t_a}^{t_{j-1}\top}\Big(L_j^l + \Phi_{t_{j-1}}^{t_j\top} S_j C_j^l \\
&\quad + C_j^{l\top} S_j \Phi_{t_{j-1}}^{t_j}\Big)\Phi_{t_i}^{t_{j-1}} C_i^k \Phi_{t_a}^{t_{i-1}}. \quad (47)
\end{aligned}$$

Again, we applied the product rule, (27), (28), and Lemma 7.

REFERENCES

[1] A. Al-Mohy and N. Higham, "The complex step approximation to the Fréchet derivative of a matrix function," *Numer. Algorithms*, vol. 53, pp. 133–148, 2010.

[2] A. E. Joel, J. Andersson, G. Gillis, J. B. Horn Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[3] H. Axelsson, M. Egerstedt, Y. Wardi, and G. Vachtsevanos, "Algorithm for switching-time optimization in hybrid dynamical systems," in *Proc. IEEE Int. Symp. Mediterrean Conf. Control Automat. Intell. Control*, 2005, pp. 256–261.

[4] A. Bürger, C. Zeile, M. Hahn, A. Altmann-Dieses, S. Sager, and M. Diehl, "Pycombina: An open-source tool for solving combinatorial approximation problems arising in mixed-integer optimal control," *IFAC-PapersOnLine*, vol. 53, pp. 6502–6508, 2020.

[5] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-Scale Nonlinear Optimization of Nonconvex Optimization and Its Applications*, vol. 83, G. Pillo and M. Roma, Eds., Berlin, Germany: Springer, 2006, pp. 35–59.

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

[7] 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.

[8] S. Göttlich, A. Potschka, and C. Teuber, "A partial outer convexification approach to control transmission lines," *Comput. Optim. Appl.*, vol. 72, no. 2, pp. 431–456, 2019.

[9] E. Hellström, M. Ivarsson, J. Aslund, and L. Nielsen, "Look-ahead control for heavy trucks to minimize trip time and fuel consumption," *Control Eng. Pract.*, vol. 17, pp. 245–254, 2009.

[10] A. C. Hindmarsh et al., "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 363–396, 2005.

[11] W. G. Horner, "XXI. A new method of solving numerical equations of all orders, by continuous approximation," *Philos. Trans. Roy. Soc. London*, vol. 109, pp. 308–335, 1819.

[12] B. Legat, O. Dowson, J. Garcia, and M. Lubin, "MathOptInterface: A data structure for mathematical optimization problems," *INFORMS J. Comput.*, vol. 34, no. 2, pp. 672–689, 2021.

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

[14] R. Mathias, "A chain rule for matrix functions and applications," *SIAM J. Matrix Anal. Appl.*, vol. 17, no. 3, pp. 610–620, 1996.

[15] F. Messerer, K. Baumgärtner, and M. Diehl, "Survey of sequential convex programming and generalized Gauss-Newton methods," *ESAIM Proc. Surv.*, vol. 71, 2021, Art. no. 64.

[16] J. I. Ramos and C. M. García-López, "Piecewise-linearized methods for initial-value problems," *Appl. Math. Comput.*, vol. 82, no. 2, pp. 273–302, Mar. 1997.

[17] S. Sager, "A benchmark library of mixed-integer optimal control problems," in *Mixed Integer Nonlinear Programming*, J. Lee and S. Leyffer, Eds. Berlin, Germany: Springer, 2012, pp. 631–670.

[18] S. Sager, H. G. Bock, and M. Diehl, "The integer approximation error in mixed-integer optimal control," *Math. Program. A*, vol. 133, no. 1–2, pp. 1–23, 2012.

[19] S. Sager, M. Jung, and C. Kirches, "Combinatorial integral approximation," *Math. Methods Oper. Res.*, vol. 73, no. 3, pp. 363–380, 2011.

[20] B. Stellato, S. Ober-Blöbaum, and P. J. Goulart, "Second-order switching time optimization for switched dynamical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 5407–5414, Oct. 2017.

[21] 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.

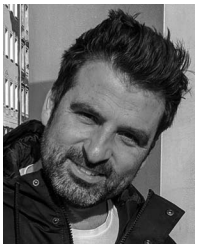
[22] C. Zeile, N. Robuschi, and S. Sager, "Mixed-integer optimal control under minimum dwell time constraints," *Math. Program.*, vol. 188, no. 2, pp. 653–694, 2021.

Christoph Plate received the bachelor's degree in mathematical engineering in 2019, and the master's degree in mathematics in 2021, both from Otto von Guericke University (OVGU) Magdeburg, Germany, where he is currently working toward the Ph.D. degree in mathematics.

Since 2022, he has been a Research Assistant with OVGU. His research interests include optimal control of switched dynamical systems, hybrid modeling, optimization methods for machine learning, and applications in chemical engineering.

Sebastian Sager studied mathematics (diploma, 2001) and received the Ph.D. degree in mathematics with a thesis "Numerical Methods for Mixed-Integer Optimal Control Problems," in 2006, and the habilitation degree in mathematics with a thesis "On the Integration of Optimization Approaches for Mixed-Integer Nonlinear Optimal Control," in 2012, at Ruperto Carola University in Heidelberg, Germany.

Since 2012, he has been a Full Professor in algorithmic optimization with Otto von Guericke University, Magdeburg, Germany. Since 2023, he is Max Planck Fellow at the Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, Germany. His research interests include the development of optimization algorithms for problems that combine the properties integrality, nonlinearity, time dependence, and uncertainty. The algorithms are stimulated by applications in estimation, control, experimental design, and analysis of dynamic systems and machine learning with applications in clinical decision support, mobility, and engineering.



Martin Stoll received the Diploma in mathematics from TU Chemnitz, Chemnitz, Germany, in 2005, and the Ph.D. degree in numerical analysis from the University of Oxford, Oxford, U.K., in 2009.

From 2013 to 2017, he was a Group Leader with the Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany. In 2017, he joined the Department of Mathematics, TU Chemnitz, as a Full Professor in scientific computing. His research interests include the development of efficient methods for problems in large-scale partial differential equation (PDE)-constrained optimization, in dynamical systems, phase-field models, and data science.

Manuel Tetschke received the bachelor's degree in computational mathematics and the master's degree in mathematics from Otto von Guericke University Magdeburg, Magdeburg, Germany, in 2011 and 2014, respectively.

After a short detour working as a Software Developer, he joined OVGU as a Research Assistant. His research interests include the development of efficient methods for the solution of mixed integer optimal control problems and on applications of these methods in the context of medical treatment schedules.