Second-order Partial Outer Convexification for Switched Dynamical Systems

Christoph Plate, Sebastian Sager, Martin Stoll, 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. Inspired by an efficient algorithm for switching time optimization by Stellato and coworkers, SwitchTimeOpt.jl, we developed an algorithmic approach for partial outer convexification 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. 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

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 [14] 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 paper and we refer to [6], [8], [19] 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 paper: **switching time optimization (STO)** and **partial outer convexification (POC)**. In STO [2], [6] 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 [17], based on a linearization of the dynamics and integration via matrix exponentials.

In POC [7], [15] 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) [16]. CIA consists of three steps: 1) solving the relaxed OCP, 2) approximating the relaxed controls with binary controls, 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` [3].

Here, we focus on the first step, i.e., a new method to solve the relaxed OCP efficiently. We use ideas from [17] 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

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

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.

In [17], 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

Submitted December 2022. "This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, 314838170) under Grants GRK 2297 MathCoRe and SPP 2331."

C. Plate, S. Sager, and M. Tetschke are with the Otto von Guericke University Magdeburg, Germany (e-mails: christoph.plate@ovgu.de, sager@ovgu.de, manuel.tetschke@ovgu.de).

M. Stoll is with the Technical University Chemnitz (e-mail: martin.stoll@math.tu-chemnitz.de).

expressions for derivatives of $e^{(A_0+\sum_{i=1}^{n_\omega} w_i A_i)t}$ becomes more involved for non-commutative 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 paper, we present a novel algorithm for solving (relaxed) MIOCPs using a direct, *first-discretize-than-optimize* approach. We use existing ideas from an STO algorithm [17] 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 the possibility to warmstart the STO algorithm with POC solutions.

B. Organization of the Paper

The paper 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. We conclude with a discussion in Section V. Proofs can be found in the Appendix.

II. Preliminaries

A. Problem definition

We are interested in optimal control problems of the form

$$\begin{aligned}\min_{x,w}\ & \int_{t_0}^{t_f} x(t)^\top \bar{Q} x(t)\,\mathrm{d}t + x(t_f)^\top \bar{E} x(t_f)\\ \text{s.t.}\ & \dot{x}(t) = f_0(x(t)) + \textstyle\sum_{k=1}^{n_\omega} w_k(t) f_k(x(t)))\\ & x(t_0) = x_0,\\ & 0 \geq c(x(t))\\ & 1 = \textstyle\sum_{k=1}^{n_\omega} w_k(t),\\ & w_k(t) \in [0,1], \quad k \in [n_\omega],\end{aligned} \tag{POC}$$

on a fixed time horizon $\mathcal{T} = [t_0, t_f]$ with differential states $x \in \mathbb{R}^{\bar{n}_x}$, initial values $x_0 \in \mathbb{R}^{\bar{n}_x}$, state constraints $c : \mathbb{R} \times \mathbb{R}^{\bar{n}_x} \mapsto \mathbb{R}^{\bar{n}_x}$ 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 [17] with two differences: a POC instead of STO reformulation of the dynamical system and the additional possibility to consider state constraints. Here and throughout the paper we use the standard notation $[N] := \{1, 2, \ldots, N\}$ and $[N]_0 := \{0, 1, 2, \ldots, N\}$.

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

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 Taylor expansion. The approach is very closely related to the ideas presented in [17]. 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

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

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 $\xi := \frac{\Delta t}{n_{lin}}$ and write

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

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

Using 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

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

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 $A_{i,j} \in \mathbb{R}^{n_x \times n_x}$ follows from reordering the terms in (7):

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

Here, $A^k_{i,j}$ represents the unweighted contribution of the k-th 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

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

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

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

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

C. Exponential integrator

For the linearized ODE (10) we can directly state the solution 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.

Definition 2 (Auxiliary matrices): The matrix exponential $\mathcal{E}_{i,j} \in \mathbb{R}^{n_x \times n_x}$ of the matrix $A_{i,j}, i \in [N], j \in [n_{lin}]$ 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 [11, Theorem 1]. For this, a temporary matrix is created and its matrix exponential is computed:

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

Having computed (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 by the following definition.

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

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

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, 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 [17]. 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

$$\begin{aligned} S_{N+1} &= E \\ S_{i,j} &= M_{i,j} + \mathcal{E}_{i,j}^\top S_{i,j+1} \mathcal{E}_{i,j} \end{aligned} \tag{21}$$

Proof: Follows the proof in [17]. ∎

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 [12] and the complex step method [1]. In a first step, we state [12, Theorem 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 $\mathbb{C}$, M_n be the set of $n \times n$ complex matrices, and let $M_n(D, m)$ denote the set of $n \times n$ 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 [12, Theorem 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

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

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,

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

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) (see top of page 5). 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 $n \in \mathbb{N}$ evalutation 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

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

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

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

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

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

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

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

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

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 I. 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

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

and

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

The proof is given in Appendix II.

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 $t_0 < t_j \leq t_i$

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

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 definitions above, 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 that:

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
$$
\begin{aligned}
\frac{\partial^2 J(w)}{\partial w_{ik} \partial w_{il}} &= x_{i-1}^\top \Big(G_i^{k,l} + 2C_i^{l\top} S_i C_i^k \\
&\quad + 2\Phi(t_i, t_{i-1})^\top S_i D_i^{k,l} \Big) x_{i-1}
\end{aligned} \tag{42}
$$
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
$$
\begin{aligned}
\frac{\partial^2 J(w)}{\partial w_{ik} \partial w_{jl}} &= 2x_{j-1}^\top \left(L_j^l + \Phi(t_j, t_{j-1})^\top S_j C_j^l \right. \\
&\quad \left. + C_j^{l\top} S_j \Phi(t_j, t_{j-1}^\top) \right) \cdot \\
&\quad \Phi(t_{j-1}, t_i) C_i^k x_{i-1}
\end{aligned} \tag{43}
$$

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

III. Implementation

Our implementation is based on SwitchTimeOpt.jl [17], 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 [10]. With this approach we can interface our code to a variety of NLP solvers, for example IPOPT [18] or KNITRO [4]. The procedure of linearizing and integrating the dynamics is shown in Alg. 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.

Algorithm 1 Linearize and propagate states

Initialize $x_{1,1} = x_0$, ▷ Fixed initial state
function LINMATEXPPROP
  for $i \in [N]$ **do**
    for $j \in [n_{lin}]$ **do**
      $A_{i,j} \leftarrow$ Eq. (11) ▷ Linearize dynamic
      $Z_{i,j} \leftarrow$ Eq. (16) ▷ Matrix exponential
      $\mathcal{E}_{i,j}, M_{i,j} \leftarrow$ Eq. (17)
      $x_{i,j+1} \leftarrow \mathcal{E}_{i,j} x_{i,j}$ ▷ Compute next state
    end for
  end for
  return $x_{i,j}, A_{i,j}, \mathcal{E}_{i,j}, M_{i,j}, \quad i \in [N], j \in [n_{lin}]$
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 which performs an update on the current iterate. This process is iterated until a convergence criterion is fulfilled.

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
  $\nabla\mathcal{E}, \nabla^2\mathcal{E} \leftarrow$ Eq. (26)
  $\Phi \leftarrow$ Eq. (18) ▷ Transition matrices
  $C, D \leftarrow$ Eq. (27), Eq. (28)
  $L, G \leftarrow$ Eq. (31), Eq. (32)
Evaluation: ▷ Theorem 9
  $J(w) \leftarrow$ Eq. (40)
  $\nabla J(w) \leftarrow$ Eq. (41)
  $H_J(w) \leftarrow$ Eq. (42), (43)
end function

We further improved the efficiency of 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 [9], reducing the number of necessary matrix multiplications. We illustrate this procedure in Alg. 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 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 $\mathcal{E}^{k}_{i,j}$.

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

IV. NUMERICAL RESULTS

In this section we present some numerical results. First we compare the efficacy of the package SwitchTimeOpt.jl [17] 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. Concluding this section, 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 the publication [17], but was adapted to the Julia version 1.7. It can be found on GitHub.[2]

A. Test Problems

Our three test problems were also used in [17] and are contained in the MIOCP benchmark library [14].

1) Egerstedt: This problem has switched linear dynamics. It was first proposed in [5] with two modes and extended in [15] with a third mode. It reads

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

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] https://github.com/chplate/SwitchTimeOpt.jl.git.

2) Lotka: This problem is given by

$$\begin{aligned}\min_{x,w} \quad & \int_0^{12} (x_1(t)-1)^2 + (x_2(t)-1)^2 \,\mathrm{d}t \\ \text{s.t.} \quad \dot{x}_1(t) &= x_1(t) - x_1(t)x_2(t) - w(t)c_1x_1(t) \\ \dot{x}_2(t) &= -x_2(t) + x_1(t)x_2(t) - w(t)c_2x_2(t) \\ x(0) &= [0.5, 0.7], \\ w_1(t) &\in [0,1],\end{aligned} \tag{Lotka}$$

with parameters $c_1 = 0.4, c_2 = 0.2$ which describe the rates with which the populations x_1 and x_2 are reduced when the control w is active. The control goal for the Lotka-Volterra system is to reach the steady state $x_1 = x_2 = 1$.

3) Tank: The double-tank control problem

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

has parameters $c_1 = 1, c_2 = 2$ representing two possible flowrates 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 [2], however for a constant reference level x_3. The specific choice of right-hand side for x_3 goes back to SwitchTimeOpt.jl [17].

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.,

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

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

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

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

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

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 [18] version 3.14.4 with tolerance 10^{-6} and limits of at most 500 iterations and at most 1000 seconds computation time.

C. Comparison POC vs STO

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

- rate of success, i.e., percentage of optimally solved instances,
- number of iterations for optimally solved instances and
- 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 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. 1 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

TABLE I: Comparison of performance indicators for POC and STO and all three test instances. Success rates are favorable for POC. The other indicators were evaluated only among the successfully solved instances. While POC needs fewer iterations, the STO approach is considerably faster. However, the quality of solutions may be worse than for POC.

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

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.

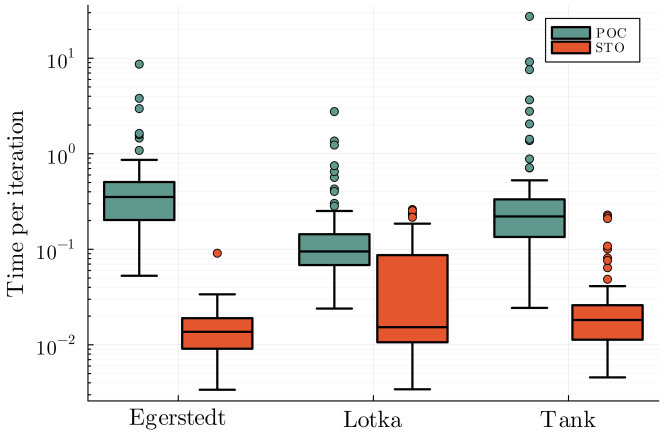

Fig. 1: Comparison of time per iteration of solved instances of the three test problems for POC and STO. Iterations are cheaper by roughly one order of magnitude for the STO approach, mainly due to simpler formulas to calculate the derivatives.

In Fig. 2, 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.

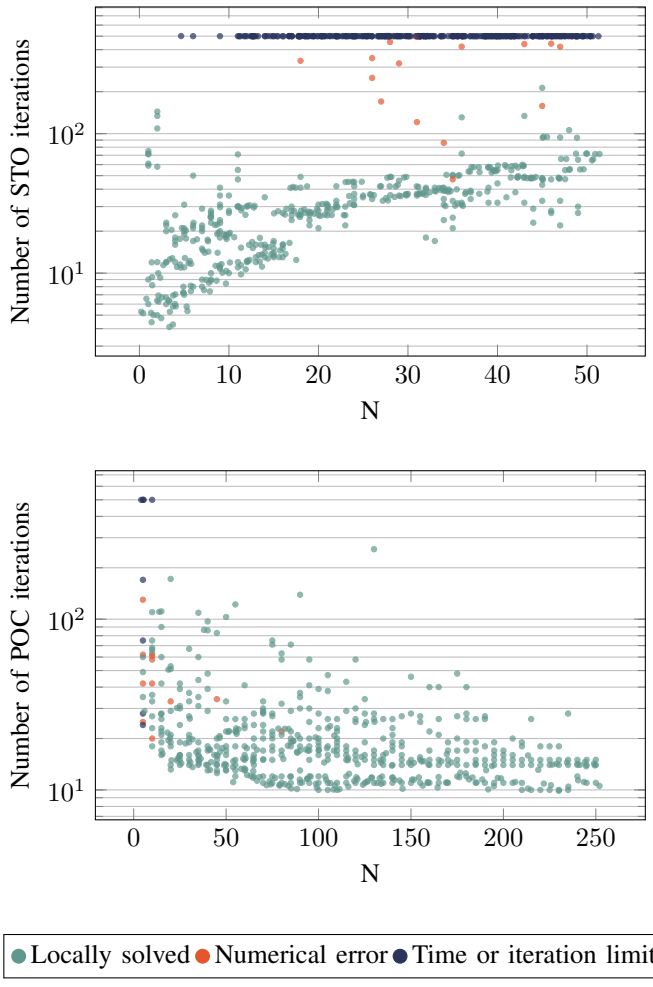

Fig. 2: Iterations and termination status over number of control intervals N for all instances of the three test problems in STO (top) and POC (bottom) formulation. The POC formulation results in a larger number of locally solved instances.

D. 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.

Figure 3 gives an overview over computation times and termination status for all three problems and all presented 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 seconds 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.

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 Figure 3 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.

E. Constrained POC

As a novel feature compared to SwitchTimeOpt.jl, in SecondOrderPOC.jl also state constraints

$$c(t, x(t)) \leq 0$$

for differentiable $c : \mathcal{T} \times \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_c}$ can be considered. As a proof of concept, we solved (Egerstedt) with the constraint

$$x_1(t) \geq 0.4 \quad t \in \mathcal{T},$$

as proposed in [15]. Figure 4 depicts a solution including this constraint in comparison to the unconstrained optimum.

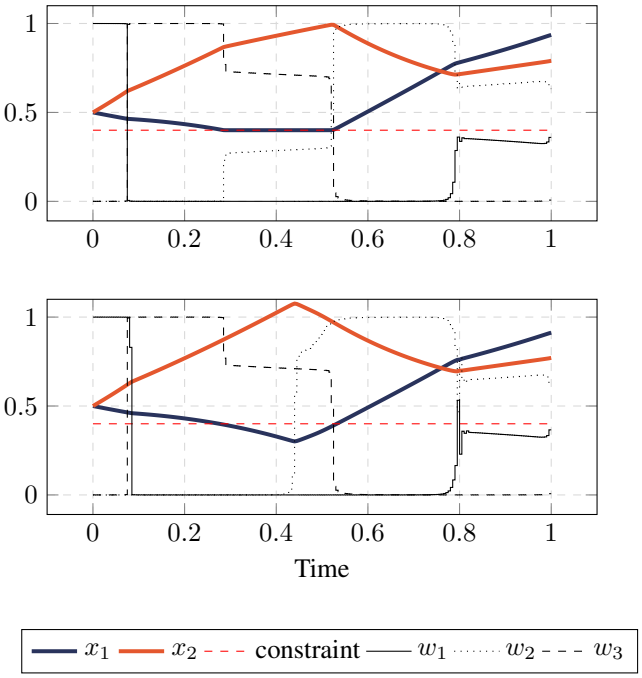

Fig. 4: Optimal controls and corresponding differential states with (top) and without (bottom) path constraint $x_1(t) \geq 0.4$ for (Egerstedt) with discretization $N = 200, n_{lin} = 3$.

V. Conclusion

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 paper). 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 [13].

In our numerical study, 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.

There are several directions for future work. Regarding the implementation, using an integration scheme of higher order would certainly help to be able to tackle more difficult problems with highly nonlinear dynamics. Also, the implementation could be parallelized efficiently. In view of the higher per-iteration costs of our method, it would also make sense to investigate whether initializing SwitchTimeOpt.jl with not fully converged solutions of our method can also improve its convergence properties.

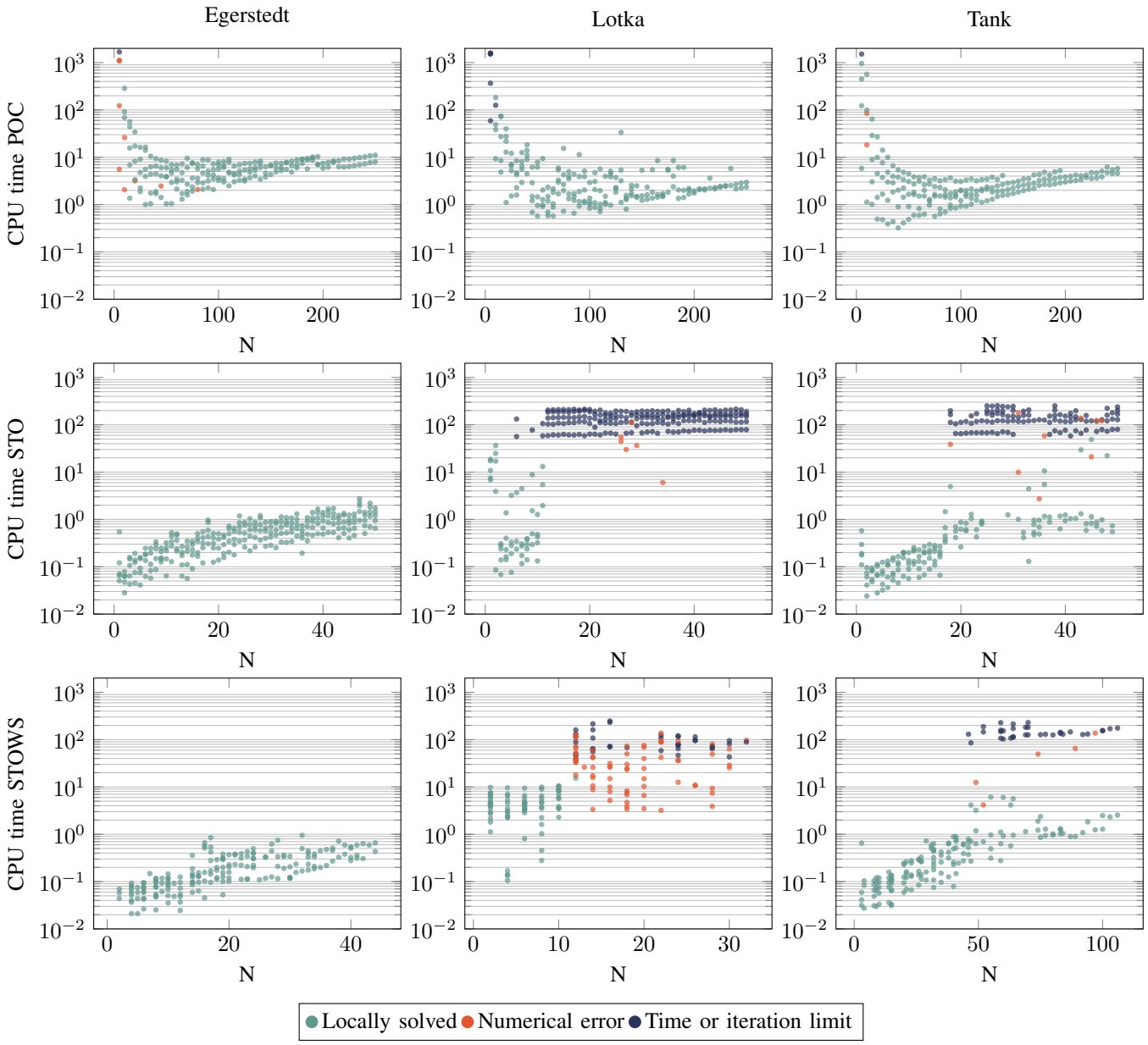

Fig. 3: Solution times and status for all problems and approaches. The columns correspond to the three test problems, the rows to the approaches POC, STO and STOWS. Times for STOWS do not take into account the initialization step. In addition to the advantages of the warm started STOWS approach with respect to the quality of local solutions as discussed in Section IV-D, an improvement with respect to solution time and status is visible for the Tank instances.

APPENDIX I
PROOF OF LEMMA 7

Proof: We begin by analyzing the component P_a.

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

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.

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

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 (45) and (44) we conclude the overall proof. ∎

APPENDIX II
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.

$$\begin{aligned}
\frac{\partial^2 S_a}{\partial w_{ik}\partial w_{il}} &= \frac{\partial}{\partial w_{il}}\left[\Phi_{t_a}^{t_{i-1}\top}\left(L_i^k + \Phi_{t_{i-1}}^{t_i\top} S_i C_i^k\right.\right.\\
&\quad \left.\left. + C_i^{k\top} S_i \Phi_{t_{i-1}}^{t_i}\right)\Phi_{t_a}^{t_{i-1}}\right]\\
&= \Phi_{t_a}^{t_{i-1}\top}\left[\frac{\partial L_i^k}{\partial w_{il}} + C_i^{l\top} S_i C_i^k\right.\\
&\quad + \Phi_{t_{i-1}}^{t_i\top} S_i D_i^{k,l} + D_i^{kl\top} S_i \Phi_{t_{i-1}}^{t_i}\\
&\quad \left. + C_i^{k\top} S_i C_i^l\right] \Phi_{t_a}^{t_{i-1}}\\
&= \Phi_{t_a}^{t_{i-1}\top}\left[G_i^{k,l} + C_i^{l\top} S_i C_i^k + \Phi_{t_{i-1}}^{t_i\top} S_i D_i^{k,l}\right.\\
&\quad \left. + D_i^{kl\top} S_i \Phi_{t_{i-1}}^{t_i} + C_i^{k\top} S_i C_i^l\right] \Phi_{t_a}^{t_{i-1}}
\end{aligned} \tag{46}$$

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

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

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

REFERENCES

[1] Awad Al-Mohy and Nicholas Higham. The complex step approximation to the fréchet derivative of a matrix function. *Numerical Algorithms*, 53, 01 2010.

[2] H. Axelsson, M. Egerstedt, Y. Wardi, and G. Vachtsevanos. Algorithm for switching-time optimization in hybrid dynamical systems. In *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, pages 256–261, Limassol, Cyprus, 2005. IEEE.

[3] 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. volume 53, pages 6502–6508, 2020.

[4] R.H. Byrd, J. Nocedal, and R.A. Waltz. Knitro: An Integrated Package for Nonlinear Optimization. In G. Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 35–59. Springer US, 2006.

[5] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-time optimization for switched-mode dynamical systems. *IEEE Transactions on Automatic Control*, 51:110–115, 2006.

[6] M. Gerdts. A variable time transformation method for mixed-integer optimal control problems. *Optimal Control Applications and Methods*, 27(3):169–182, 2006.

[7] Simone Göttlich, Andreas Potschka, and Claus Teuber. A partial outer convexification approach to control transmission lines. *Computational Optimization and Applications*, 72(2):431–456, 2019.

[8] E. Hellström, M. Ivarsson, J. Aslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17:245–254, 2009.

[9] William George Horner. Xxi. a new method of solving numerical equations of all orders, by continuous approximation. *Philosophical Transactions of the Royal Society of London*, (109):308–335, 1819.

[10] Benoit Legat, Oscar Dowson, Joaquim Garcia, and Miles Lubin. MathOptInterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689, 2021.

[11] C. Van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23:395–404, 1978.

[12] Roy Mathias. A chain rule for matrix functions and applications. *SIAM Journal on Matrix Analysis and Applications*, 17(3):610–620, 1996.

[13] Florian Messerer, Katrin Baumgärtner, and Moritz Diehl. Survey of sequential convex programming and generalized gauss-newton methods. *ESAIM. Proceedings and Surveys*, 71:64, 2021.

[14] S. Sager. A benchmark library of mixed-integer optimal control problems. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 631–670. Springer, 2012.

[15] S. Sager, H.G. Bock, and M. Diehl. The Integer Approximation Error in Mixed-Integer Optimal Control. *Mathematical Programming A*, 133(1–2):1–23, 2012.

[16] S. Sager, M. Jung, and C. Kirches. Combinatorial Integral Approximation. *Mathematical Methods of Operations Research*, 73(3):363–380, 2011.

[17] B. Stellato, S. Ober-Blöbaum, and P. J. Goulart. Second-order switching time optimization for switched dynamical systems. *IEEE Transaction on Automatic Control*, 62(10):5407–5414, 2017.

[18] A. Wächter and L.T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006.

[19] C. Zeile, N. Robuschi, and S. Sager. Mixed-integer optimal control under minimum dwell time constraints. *Mathematical Programming*, 188(2):653–694, 2021.