2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Finite Elements with Switch Detection for Direct Optimal Control of Nonsmooth Systems with Set-Valued Step Functions

Armin Nurkanović[1], Jonathan Frey[1,2], Anton Pozharskiy[1], Moritz Diehl[1,2]

***Abstract*— This paper extends the Finite Elements with Switch Detection (FESD) method [19] to optimal control problems with nonsmooth systems involving set-valued step functions. Logical relations and common nonsmooth functions within a dynamical system can be expressed using linear and nonlinear expressions involving step functions. A prominent subclass of these systems are Filippov systems. The set-valued step function can be expressed by the solution map of a linear program, and using its KKT conditions allows one to transform the initial system into an equivalent dynamic complementarity system (DCS). Standard Runge-Kutta (RK) methods applied to DCS have only first-order accuracy. The FESD discretization makes the step sizes degrees of freedom and adds further constraints that ensure exact switch detection to recover the high-accuracy properties that RK methods have for smooth ODEs. We use the novel FESD method for the direct transcription of optimal control problems. All methods and examples in this paper are implemented in the open-source software package `NOSNOC`.**

I. Introduction

In this paper, we introduce a high-accuracy method for discretizing and solving nonsmooth Optimal Control Problems (OCPs) of the following form:

$$\min_{x(\cdot),u(\cdot)} \quad \int_0^T L(x(t),u(t))\mathrm{d}t + R(x(T)) \tag{1a}$$

$$\text{s.t.} \quad x_0 = s_0, \tag{1b}$$

$$\dot{x}(t) \in F(x(t),u(t),\Gamma(c(x(t)))), \text{ a.a. } t\in[0,T], \tag{1c}$$

$$0 \geq G_{\mathrm{p}}(x(t),u(t)), \ t\in[0,T], \tag{1d}$$

$$0 \geq G_{\mathrm{t}}(x(T)), \tag{1e}$$

where $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the running cost and $R : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal cost, $s_0 \in \mathbb{R}^{n_x}$ is a given initial value. The right-hand side of the Differential Inclusion (DI) in Eq. (1c) is the set-valued mapping $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_c} \to \mathcal{P}(\mathbb{R}^{n_x})$. The path and terminal constraints are defined by the functions $G_{\mathrm{p}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_{\mathrm{p}}}$ and $G_{\mathrm{t}} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{\mathrm{t}}}$, respectively.

The OCP is nonsmooth due to the DI in Eq. (1c). The function $c(x) \in \mathbb{R}^{n_c}$ contains n_c *switching functions*. The set-valued function $\Gamma : \mathbb{R}^{n_c} \to \mathcal{P}(\mathbb{R}^{n_c})$ is defined as the concatenation of scalar step functions, i.e., for $y \in \mathbb{R}^{n_c}$ we have $\Gamma(y) = [\gamma(y_1),\ldots,\gamma(y_{n_c})]^\top \in \mathbb{R}^{n_c}$, where $\gamma : \mathbb{R} \to \mathcal{P}(\mathbb{R})$ is defined as:

$$\gamma(y_i) = \begin{cases} \{1\}, & y_i > 0, \\ [0,1], & y_i = 0, \\ \{0\}, & y_i < 0. \end{cases} \tag{2}$$

Note that there are no particular restrictions on how the components $\Gamma(c(x))$ enter the r.h.s. of the DI (1c). This DI is an instance of so-called Aizerman–Pyatnitskii DIs., cf. [8, page 55, Definition c]. A prominent and well-studied subclass of these DIs, on which we focus in the sequel, are Filippov DIs.

Set-valued step functions provide an intuitive way to model logical *if-else* and *and-or* relations in dynamical systems. In addition, several other common nonsmooth functions, such as $\mathrm{sign}, \min$, and $\max$, can be easily expressed via step functions. Smoothed versions of the step function are often used in the numerical simulation of Piecewise Smooth Systems (PSS) [9], [12]. Step functions are often used in the modeling of gene regulatory networks [2], [12]. Another application is to express Filippov sets in sliding modes on surfaces with co-dimension higher than one [7]. Moreover, several classes of systems with state jumps can be reformulated into PSS via the time-freezing reformulation [13], [18], [15], [11]. Thus, the formulation (1) covers a wide class of practical problems.

The OCP (1) is difficult to solve numerically for several reasons. Direct methods first discretize the infinite-dimensional OCP (1) and solve then a finite-dimensional Nonlinear Program (NLP). However, the discretization of a DI (1c) presents several pitfalls. First, standard time-stepping methods for DIs have only first-order accuracy [1]. Stewart and Anitescu [23] have shown that the numerical sensitivities obtained from standard time-stepping are incorrect regardless of the integrator step size. Moreover, the sensitivities of smoothed approximations of (1c) are correct only under the very restrictive assumption that the step size is sufficiently smaller than the smoothing parameter. Smoothing and wrong sensitivities can lead to artificial local minima and jeopardize the progress of NLP solvers [14]. Furthermore, the discretized OCPs are nonsmooth NLPs. In summary, even for moderate accuracy, many optimization variables and a huge computational load are required.

Some of these drawbacks are overcome by the recently introduced Finite Elements with Switch Detection (FESD) method [19]. FESD was originally developed for Filippov DIs, which are transformed into equivalent Dynamic Complementarity System (DCS) via Stewart's reformulation [22].

This research was supported by the DFG via Research Unit FOR 2401 and project 424107692 and by the EU via ELO-X 953348.

[1]Department of Microsystems Engineering (IMTEK), [2]Department of Mathematics, University of Freiburg, Germany, {armin.nurkanovic, jonathan.frey, moritz.diehl}@imtek.uni-freiburg.de, anton.pozharskiy@merkur.uni-freiburg.de

This method starts with a standard Runge-Kutta (RK) discretization of the DCS and, inspired by [5], allows the integrator step sizes to be degrees of freedom. Additional constraints ensure exact switch detection (and thus higher-order accuracy) and correct computation of numerical sensitivities (and avoid convergence to spurious solutions). This overcomes the aforementioned fundamental limitations of standard time-stepping discretization methods.

Contributions: In this paper, we extend the FESD method to DIs (and the associated OCP (1)) governed by set-valued step functions. Our approach is to express the step function as the solution map of a linear program, and using its KKT conditions, we write the DI into an equivalent DCS. The DCS is discretized with a standard RK method. However, we let the step sizes be degrees of freedom and introduce additional equations that ensure exact switch detection. This recovers the high-accuracy properties that RK methods have for smooth ODEs. FESD results in Mathematical Programs with Complementarity Constraints (MPCC), which can be efficiently solved in a homotopy loop with off-the-shelf NLP solvers [21], [4]. We illustrate the efficacy of the new formulation on numerical simulation and OCP examples. All methods and examples of this paper are implemented in the open-source package `NOSNOC` [16][1].

Notation: The complementary conditions for two vectors $x, y \in \mathbb{R}^n$ read as $0 \leq x \perp y \geq 0$, where $x \perp y$ means $x^\top y = 0$. For two scalar variables a, b the so-called C-functions have the property $\phi(a,b) = 0 \iff a \geq 0, b \geq 0, ab = 0$. A famous example is the Fischer-Burmeister function $\phi_{\mathrm{FB}}(a,b) = a + b - \sqrt{a^2+b^2}$. If $x, y \in \mathbb{R}^n$, we use $\phi(\cdot)$ component-wise and define $\Phi(x,y) = (\phi(x_1,y_1),\ldots,\phi(x_n,y_n))$. The concatenation of two column vectors $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ is denoted by $(x,y) \coloneqq [x^\top, y^\top]^\top$. Given a matrix $S \in \mathbb{R}^{n\times m}$, its i-th row is denoted by $S_{i,\bullet}$ and its j-th column is denoted by $S_{\bullet,j}$.

II. Filippov systems and the equivalent dynamic complementarity system

In this section, we show how the Filippov convexification of a PSS can be expressed via step functions, and how to transform this system into an equivalent DCS.

A. Filippov systems via step functions

We focus on the PSS systems and their Filippov convexification as the most prominent representative of DIs with set-valued step functions. Most developments are straightforwardly generalized. A controlled PSS is defined as

$$\dot{x}(t) = f_i(x(t),u(t)), \text{ if } x(t) \in R_i \subset \mathbb{R}^{n_x}, i \in \mathcal{J}, \quad (3)$$

where R_i are disjoint, connected, and open sets, and $\mathcal{J} \coloneqq \{1,\ldots,n_f\}$. The sets R_i are assumed to be nonempty and to have piecewise-smooth boundaries ∂R_i. It is assumed that $\overline{\bigcup_{i\in\mathcal{J}} R_i} = \mathbb{R}^{n_x}$, and that $\mathbb{R}^{n_x} \setminus \bigcup_{i\in\mathcal{J}} R_i$ is a set of measure zero. The functions $f_i(\cdot)$ are assumed to be Lipschitz and at least twice continuously differentiable functions on an open neighborhood of $\overline{R}_i$. Here, $u(t)$ is a sufficiently regular externally chosen control function, e.g., obtained as a solution to an Optimal Control Problem (OCP).

The event of $x(t)$ reaching or leaving some boundary ∂R_i is called a switch. In this paper, we consider systems with a finite number of switches on finite time intervals, i.e., Zeno trajectories are excluded. The ODE (3) is not defined on the region boundaries ∂R_i, and classical notions of solutions [6] are not sufficient to treat the rich behavior that emerges in a PSS. For example, during *sliding modes* $x(t)$ must evolve on ∂R_i [6], [8]. A sufficiently regular and practical notion is given by the Filippov extension for (3). The special structure of the PSS allows the definition of a finite number of convex multipliers θ_i and the Filippov DI reads as [8], [22]:

$$\begin{aligned}\dot{x} \in F_{\mathrm{F}}(x,u) = \Big\{ \sum_{i\in\mathcal{J}} f_i(x,u)\,\theta_i \mid \sum_{i\in\mathcal{J}} \theta_i = 1, \theta_i \geq 0, \\ \theta_i = 0 \text{ if } x \notin \overline{R}_i, \forall i \in \mathcal{J} \Big\}.\end{aligned} \quad (4)$$

Let the regions R_i be defined by smooth switching functions $c_j(x), j \in \mathcal{C} \coloneqq \{1,\ldots,n_c\}$. The definition of some set R_i does not have to depend on all functions $c_j(x)$. Therefore, with n_c scalar functions we define up to $n_f \leq 2^{n_c}$ regions. For example:

$$\begin{aligned} R_1 &= \{x \in \mathbb{R}^{n_x} \mid c_1(x) > 0\},\\ R_2 &= \{x \in \mathbb{R}^{n_x} \mid c_1(x) < 0, c_2(x) > 0\},\\ &\vdots\\ R_{n_f} &= \{x \in \mathbb{R}^{n_x} \mid c_1(x) < 0, c_2(x) < 0, \ldots, c_{n_c}(x) < 0\}. \end{aligned}$$

Note that the boundaries of the regions ∂R_i are subsets of the zero-level sets of appropriate functions $c_j(x)$. We can compactly express the definitions of the sets R_i via a matrix $S \in \mathbb{R}^{n_f \times n_c}$, which in our example reads as:

$$S = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ -1 & 1 & \ldots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ -1 & -1 & \ldots & -1 \end{bmatrix}. \quad (5)$$

It is not allowed for S to have a row with all zeros. The sparsity in the matrix S arises from the geometry of the regions R_i. Furthermore, for every region R_i, we define an index set containing the indices of all switching functions relevant to its definition, i.e.,

$$\mathcal{C}_i = \{j \in \mathcal{C} \mid S_{i,j} \neq 0\}, \text{ for all } i \in \mathcal{J}.$$

Now, the matrix S enables us to compactly express the definitions of the regions R_i as:

$$R_i = \{x \in \mathbb{R}^{n_x} \mid S_{i,j} c_j(x) > 0,\ j \in \mathcal{C}_i\}. \quad (6)$$

The next question to be answered is: Given the definitions of R_i via the switching functions $c(x)$, how can we compute the Filippov multipliers θ in (4)? To derive such expressions, we make use of the set-valued step functions and the definition

[1]MATLAB: `https://github.com/nurkanovic/nosnoc`, Python: `https://github.com/FreyJo/nosnoc_py`

of the regions R_i via the switching functions $c_j(x)$. Let us first illustrate the development so far with an example.

Example 1. *We regard three regions defined via the switching functions $c_1(x)$ and $c_2(x)$: $R_1 = \{x \in \mathbb{R}^{n_x} \mid c_1(x) > 0\}$, $R_2 = \{x \in \mathbb{R}^{n_x} \mid c_1(x) < 0, c_2(x) > 0\}$ and $R_3 = \{x \in \mathbb{R}^{n_x} \mid c_1(x) < 0, c_2(x) < 0\}$, and the associated vector fields $f_i(x), i = 1, 2, 3$. By using* (6) *these sets can be compactly defined via the matrix*

$$S = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$$

Next, let $\alpha \in \Gamma(c(x)) \in \mathbb{R}^2$. A selection of the Filippov set (4) *and the associated ODE reads as:*

$$\dot{x} = \alpha_1 f_1(x) + (1-\alpha_1)\alpha_2 f_2(x) + (1-\alpha_1)(1-\alpha_2) f_3(x).$$

By inspection, we conclude that $\theta_1 = \alpha_1$, $\theta_2 = (1-\alpha_1)\alpha_2$ and $\theta_3 = (1-\alpha_1)(1-\alpha_2)$. Since $\alpha_1, \alpha_2 \in [0,1]$ it is clear that $\theta_i \in [0,1], i = 1,2,3$. Similarly, by direct calculation, we verify that $\theta_1 + \theta_2 + \theta_3 = 1$. Observe that the entries of $S_{i,j}$ determine how α_j enters the expression for θ_i. For $S_{i,j} = 1$ we have α_j, for $S_{i,j} = -1$ we have $(1-\alpha_j)$ and for $S_{i,j} = 0$, α_j does not appear in the expression for θ_i.

We generalize the patterns observed in our example and define the set

$$F_{\mathrm{S}}(x) := \Big\{ \sum_{i=1}^{n_f} \prod_{j \in \mathcal{C}_i} \Big(\frac{1 - S_{i,j}}{2} + S_{i,j}\alpha_i \Big) f_i(x) \mid \alpha \in \Gamma(c(x)) \Big\}. \tag{7}$$

Note that we have

$$\frac{1 - S_{i,j}}{2} + S_{i,j}\alpha_i = \begin{cases} \alpha_i, & \text{if } S_{i,j} = 1, \\ 1 - \alpha_i, & \text{if } S_{i,j} = -1. \end{cases}$$

Similar definitions of the set $F_{\mathrm{S}}(x)$ can be found in [7, Section 4.2] and [9, Section 2.1]. However, they are restricted to fully dense matrices S and do not focus on developing high-accuracy discretization methods for such systems. Next we show that $F_{\mathrm{S}}(x)$ is indeed the same set as $F_{\mathrm{F}}(x)$, i.e., the set in the r.h.s. of (4).

Lemma 1 (Lemma 1.5 in [7])**.** *Let $a_1, a_2, \dots a_m \in \mathbb{R}$. Consider the 2^m non-repeated products of the form $p_i = (1 \pm a_1)(1 \pm a_2) \cdots (1 \pm a_m)$, then it holds that $\sum_i^{2^m} p_i = 2^m$.*

Proposition 2. *Let*

$$\theta_i = \prod_{j \in \mathcal{C}_i} \Big(\frac{1 - S_{i,j}}{2} + S_{i,j}\alpha_j \Big), \tag{8}$$

for all $i \in \mathcal{J} = \{1, \dots, n_f\}$, then it holds that $F_{\mathrm{F}}(x) = F_{\mathrm{S}}(x)$.

Proof. We only need to show that $\theta_i \geq 0$ for all $i \in \mathcal{J}$ and $\sum_{i \in \mathcal{J}} \theta_i = 1$. It is easy to see that $\theta_i \in [0,1]$ as it consists of a product of terms that takes value in $[0,1]$. Next we show that $\sum_{i \in \mathcal{J}} \theta_i = 1$. We introduce the change of variables: $\frac{1+b_j}{2} = \alpha_j$, $\frac{1-b_j}{2} = 1 - \alpha_j$. Then all θ_i are of the form

$$\theta_i = 2^{-|\mathcal{C}_i|} \prod_{j \in \mathcal{C}_i} (1 \pm b_j).$$

If the matrix S is dense we have that $\mathcal{C}_i = \mathcal{C}$ for all $i \in \mathcal{J}$ and $n_f = 2^{n_c}$. By applying Lemma 1 we conclude that $\sum_{i \in \mathcal{J}} \theta_i = 1$ and the proof is complete. On the other hand, if the matrix S has zero entries, we have that $n_f < 2^{n_c}$. We extend sequence $\{\theta_i\}_{i=1}^{n_f}$ to $\{\tilde{\theta}_l\}_{l=1}^{2^{n_c}}$, where the terms $\tilde{\theta}_l$ are defined as follows. If $\mathcal{C}_i = \mathcal{C}$, then $\tilde{\theta}_i = \theta_i$. Now let $\mathcal{C}_i \subset \mathcal{C}$ and $\mathcal{C} \setminus \mathcal{C}_i = \{k\}$, i.e., only one $S_{i,k}$ in $S_{i,\bullet}$ is zero. We can use the simple identity $\theta_i = \theta_i \frac{(1+b_k)}{2} + \theta_i \frac{(1-b_k)}{2}$, and let two additional $\tilde{\theta}_l$ be the two terms in the extended sum above. Applying this procedure inductively we obtain for all i where $\mathcal{C}_i \subset \mathcal{C}$ terms $\tilde{\theta}_l$ of the form $\frac{(1 \pm b_1)}{2} \cdots \frac{(1 \pm b_{n_c})}{2}$. Now we can apply Lemma 1 and conclude that $\sum_{i=1}^{n_f} \theta_i = \sum_{l=1}^{2^{n_c}} \tilde{\theta}_l = 1$. □

B. The equivalent dynamic complementarity system

Next, we pass from the abstract definition of the Filippov systems via set-valued step functions to the computationally more practical formulation of a DCS. The complementarity conditions encode all the combinatorial structure and nonsmoothness in the system but can still be efficiently treated via derivative-based optimization methods [21], [4].

To perform this transition, we express the set-valued step function $\Gamma(c(x))$ as the solution map of a linear program parametric in x [1], [2]:

$$\Gamma(c(x)) = \arg \min_{\alpha \in \mathbb{R}^{n_c}} \; -c(x)^\top \alpha \tag{9a}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq 1, \; i = 1, \dots, n_c. \tag{9b}$$

Let $\lambda^{\mathrm{n}}, \lambda^{\mathrm{p}} \in \mathbb{R}^{n_c}$ be the Lagrange multipliers for the lower and upper bound on α in (9b), respectively. The KKT conditions of (9) read as

$$c(x) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}, \tag{10a}$$

$$0 \leq \lambda^{\mathrm{n}} \perp \alpha \geq 0, \tag{10b}$$

$$0 \leq \lambda^{\mathrm{p}} \perp e - \alpha \geq 0. \tag{10c}$$

We look closer at a single component α_j and the associated function $c_j(x)$. From the LP (9) and its KKT conditions, one can see that for $c_j(x) > 0$, we have $\alpha_j = 1$. From (10a) and the complementarity condition (10b) it follows that $\lambda^{\mathrm{n}}_{j} = c_j(x) > 0$. The lower bound is inactive, thus, $\lambda^{\mathrm{n}}_j = 0$. Similarly, for $c_j(x) < 0$, if follows that $\alpha_j = 0$, $\lambda^{\mathrm{p}}_j = 0$ and $\lambda^{\mathrm{n}}_j = -c_j(x) > 0$. Lastly, $c_j(x) = 0$ implies that $\alpha_j \in [0,1]$ and $\lambda^{\mathrm{p}}_j = \lambda^{\mathrm{n}}_j = 0$. From these discussions, it is clear that $c(x)$, λ^{n} and λ^{p} are related by following expressions:

$$\lambda^{\mathrm{p}} = \max(c(x), 0), \;\; \lambda^{\mathrm{n}} = -\min(c(x), 0). \tag{11}$$

That is, λ^{p} collects the positive parts of $c(x)$ and λ^{n} the absolute value of the negative parts of $c(x)$. From the continuity of $c(x(t))$, it follows that the functions $\lambda^{\mathrm{p}}(t)$ and $\lambda^{\mathrm{n}}(t)$ are continuous in t as well.

TABLE I: Expressions of θ_i for different definitions of R_i.

Definition R_i	Expression θ_i
$R_i = A$	$\theta_i = \alpha_1$
$R_i = A \cup B$	$\theta_i = \alpha_1 + \alpha_2$
$R_i = A \cap B$	$\theta_i = \alpha_1 \alpha_2$
$R_i = \text{int}(\mathbb{R}^{n_x} \setminus A) = \{x \mid c_1(x) < 0\}$	$\theta_i = 1 - \alpha_1$
$R_i = A \setminus B$	$\theta_i = \alpha_1 - \alpha_2$

Using KKT systems (10) and combining this with the definition of the Filippov set in (7) and the expression for θ_i in (8), we obtain the following DCS:

$$\dot{x} = F(x,u)\,\theta, \tag{12a}$$
$$\theta_i = \prod_{j \in \mathcal{C}_i} \frac{1 - S_{i,j}}{2} + S_{i,j}\alpha_j, \text{ for all } i \in \mathcal{J}, \tag{12b}$$
$$c(x) = \lambda^{\mathrm{p}} - \lambda^{\mathrm{n}}, \tag{12c}$$
$$0 \le \lambda^{\mathrm{n}} \perp \alpha \ge 0, \tag{12d}$$
$$0 \le \lambda^{\mathrm{p}} \perp e - \alpha \ge 0, \tag{12e}$$

where $F(x) = [f_1(x,u), \ldots, f_{n_f}(x,u)] \in \mathbb{R}^{n_x \times n_f}$, $\theta = (\theta_1, \ldots, \theta_{n_f}) \in \mathbb{R}^{n_f}$ and $\lambda^{\mathrm{p}}, \lambda^{\mathrm{n}}, \alpha \in \mathbb{R}^{n_c}$. We group all algebraic equations into a single function and use a C-function $\Psi(\cdot,\cdot)$ for the complementarity condition to obtain a more compact expression:

$$G(x,\theta,\alpha,\lambda^{\mathrm{p}},\lambda^{\mathrm{n}}) := \begin{bmatrix} \theta_1 - \prod_{j \in C_1} \frac{1-S_{1,j}}{2} + S_{1,j}\alpha_j \\ \vdots \\ \theta_{n_f} - \prod_{j \in C_{n_f}} \frac{1-S_{n_f,j}}{2} + S_{n_f,j}\alpha_j \\ c(x) - \lambda^{\mathrm{p}} + \lambda^{\mathrm{n}} \\ \Psi(\lambda^{\mathrm{n}}, \alpha) \\ \Psi(\lambda^{\mathrm{p}}, e - \alpha) \end{bmatrix}.$$

Finally, we obtain a compact representation of (12) in the form of a nonsmooth DAE:

$$\dot{x} = F(x,u)\theta, \tag{13a}$$
$$0 = G(x,\theta,\alpha,\lambda^{\mathrm{p}},\lambda^{\mathrm{n}}). \tag{13b}$$

In Table I we summarize the elementary algebraic expressions for the multipliers θ_i depending on the geometric definition of the regions R_i. Thereby, we regard the two sets $A = \{x \in \mathbb{R}^{n_x} \mid c_1(x) > 0\}$ and $B = \{x \in \mathbb{R}^{n_x} \mid c_2(x) > 0\}$. All other more complicated expressions can be obtained by combining these elementary operations.

III. Finite Elements with Switch Detection

A. Standard Runge-Kutta discretization

As a starting point for our analysis, we regard a standard RK discretization for the nonsmooth DAE formulation of the DCS (13). We remind the reader that (13b) collects all algebraic equations including the complementarity conditions (12d)-(12e). For ease of exposition, we regard a single control interval $[0,T]$ with a fixed control input $q \in \mathbb{R}^{n_u}$, i.e., we set $u(t) = q$ for $t \in [0,T]$. In Section III-E, we will treat the discretization of OCPs with multiple control intervals. Let $x(0) = s_0$ be the initial value. The control interval $[0,T]$ is divided into into N_{FE} finite elements (integration intervals) $[t_n, t_{n+1}]$ via the grid points $0 = t_0 < t_1 < \ldots < t_{N_{\mathrm{FE}}} = T$. In each finite elements we regard an n_{s}-stage RK method which is characterized by the Butcher tableau entries $a_{i,j}, b_i$ and c_i with $i,j \in \{1,\ldots,n_{\mathrm{s}}\}$ [10]. The step sizes are denoted by $h_n = t_{n+1} - t_n,\ n = 0,\ldots,N_{\mathrm{FE}} - 1$. The approximation of the differential state at the grid points t_n is denoted by $x_n \approx x(t_n)$.

We regard the differential representation of the RK method. Hence, the derivatives of states at the stage points $t_{n,i} := t_n + c_i h_n,\ i = 1,\ldots,n_{\mathrm{s}}$, are degrees of freedom. For a single finite element, we group them in the vector $V_n := (v_{n,1},\ldots,v_{n,n_{\mathrm{s}}}) \in \mathbb{R}^{n_{\mathrm{s}} n_x}$. Similarly, the stage values for the algebraic variables are collected in the vectors: $\Theta_n := (\theta_{n,1},\ldots,\theta_{n,n_{\mathrm{s}}}) \in \mathbb{R}^{n_{\mathrm{s}} \cdot n_f}$, $A_n := (\alpha_{n,1},\ldots,\alpha_{n,n_{\mathrm{s}}}) \in \mathbb{R}^{n_{\mathrm{s}} \cdot n_c}$, $\Lambda_n^{\mathrm{p}} := (\lambda_{n,1}^{\mathrm{p}},\ldots,\lambda_{n,n_{\mathrm{s}}}^{\mathrm{p}}) \in \mathbb{R}^{n_{\mathrm{s}} \cdot n_c}$ and $\Lambda_n^{\mathrm{n}} := (\lambda_{n,1}^{\mathrm{n}},\ldots,\lambda_{n,n_{\mathrm{s}}}^{\mathrm{n}}) \in \mathbb{R}^{n_{\mathrm{s}} \cdot n_c}$. We collect all *internal* variables in the vector $Z_n = (x_n, \Theta_n, A_n, \Lambda_n^{\mathrm{p}}, \Lambda_n^{\mathrm{n}}, V_n)$.

The vector x_n^{next} denotes the value at t_{n+1}, which is obtained after a single integration step. Now, we can state the RK equations for the DCS (13) for a single finite element as

$$0 = G_{\mathrm{rk}}(x_n^{\mathrm{next}}, Z_n, h_n, q) := \tag{14}$$
$$\begin{bmatrix} v_{n,1} - F(x_n + h_n \sum_{j=1}^{n_{\mathrm{s}}} a_{1,j} v_{n,j}, q)\theta_{n,1} \\ \vdots \\ v_{n,n_{\mathrm{s}}} - F(x_n + h_n \sum_{j=1}^{n_{\mathrm{s}}} a_{n_{\mathrm{s}},j} v_{n,j}, q)\theta_{n,n_{\mathrm{s}}} \\ G(x_n + h_n \sum_{j=1}^{n_{\mathrm{s}}} a_{1,j} v_{n,j}, \theta_{n,1}, \alpha_{n,1}, \lambda_{n,1}^{\mathrm{p}}, \lambda_{n,1}^{\mathrm{n}}) \\ \vdots \\ G(x_n + h_n \sum_{j=1}^{n_{\mathrm{s}}} a_{n_{\mathrm{s}},j} v_{n,j}, \theta_{n,n_{\mathrm{s}}}, \alpha_{n,n_{\mathrm{s}}}, \lambda_{n,n_{\mathrm{s}}}^{\mathrm{p}}, \lambda_{n,n_{\mathrm{s}}}^{\mathrm{n}}) \\ x_n^{\mathrm{next}} - x_n - h_n \sum_{i=1}^{n_{\mathrm{s}}} b_i v_{n,i} \end{bmatrix}.$$

Next, we summarize the equations for all N_{FE} finite elements over the entire interval $[0,T]$ in a discrete-time system format. To simplify the statement, we need additional shorthand notation to collect all variables, on all finite elements, within the regarded control interval: $\mathbf{x} = (x_0,\ldots,x_{N_{\mathrm{FE}}}) \in \mathbb{R}^{(N_{\mathrm{FE}}+1)n_x}$, $\mathbf{V} = (V_0,\ldots,V_{N_{\mathrm{FE}}-1}) \in \mathbb{R}^{N_{\mathrm{FE}} n_{\mathrm{s}} n_x}$ and $\mathbf{h} := (h_0,\ldots,h_{N_{\mathrm{FE}}-1}) \in \mathbb{R}^{N_{\mathrm{FE}}}$. Recall that the simple continuity condition $x_{n+1} = x_n^{\mathrm{next}}$ holds. We collect the stage values of the Filippov multipliers in the vector $\mathbf{\Theta} = (\Theta_0,\ldots,\Theta_{N_{\mathrm{FE}}-1}) \in \mathbb{R}^{n_\theta}$ and $n_\theta = N_{\mathrm{FE}} n_{\mathrm{s}} n_f$. Similarly, we collect the stage values of the algebraic variables specific to the step representation in vectors $\mathbf{A}, \mathbf{\Lambda}^{\mathrm{p}}, \mathbf{\Lambda}^{\mathrm{n}} \in \mathbb{R}^{n_\alpha}$, where $n_\alpha = N_{\mathrm{FE}} n_{\mathrm{s}} n_c$. Finally, we collect all internal variables in the vector $\mathbf{Z} = (\mathbf{x}, \mathbf{V}, \mathbf{\Theta}, \mathbf{A}, \mathbf{\Lambda}^{\mathrm{p}}, \mathbf{\Lambda}^{\mathrm{n}}) \in \mathbb{R}^{n_{\mathbf{Z}}}$, where $n_{\mathbf{Z}} = (N_{\mathrm{FE}}+1)n_x + N_{\mathrm{FE}} n_{\mathrm{s}} n_x + n_\theta + 3n_\alpha$.

All computations over a single control interval of the standard RK discretization are summarized in:

$$s_1 = F_{\mathrm{std}}(\mathbf{Z}), \tag{15a}$$
$$0 = G_{\mathrm{std}}(\mathbf{Z}, \mathbf{h}, s_0, q), \tag{15b}$$

where $s_1 \in \mathbb{R}^{n_x}$ is the approximation of $x(T)$ and

$$F_{\mathrm{std}}(\mathbf{Z}) = x_{N_{\mathrm{FE}}},$$
$$G_{\mathrm{std}}(\mathbf{Z}, \mathbf{h}, s_0, q) \coloneqq \begin{bmatrix} x_0 - s_0 \\ G_{\mathrm{rk}}(x_1, Z_0, h_0, q) \\ \vdots \\ G_{\mathrm{rk}}(x_{N_{\mathrm{FE}}}, Z_{N_{\mathrm{FE}}-1}, h_{N_{\mathrm{FE}}-1}, q) \end{bmatrix}.$$

In (15), $\mathbf{h}$ is a given parameter and implicitly fixes the discretization grid. We proceed by letting $\mathbf{h}$ be degrees of freedom and introduce the cross complementarity conditions.

B. Cross complementarity

For brevity, we regard in this paper only RK methods with $c_{n_\mathrm{s}} = 1$, which already covers many schemes, e.g., Radau IIA and several Lobatto methods [10]. This means that the right boundary point of a finite element is a stage point since $t_{n+1} = t_n + c_{n_\mathrm{s}} h_n$. We will provide extensions for $c_{n_\mathrm{s}} \neq 1$ in future work.

As in any event based method, we assume that there is a finite number of switches. To be able to detect all switches, we assume that N_{FE} is greater then total the number of switches. Our goal is to derive additional constraints that will allow active-set changes only at the boundary of a finite element. Moreover, in this case, the step size h_n should adapt such that all switches are detected exactly. Note that in the standard discretization, at every RK-stage point, we have for $n = 1, \dots, N_{\mathrm{FE}}$, the complementarity conditions:

$$0 \leq \lambda^{\mathrm{n}}_{n,m} \perp \alpha_{n,m} \geq 0, \ m = 1, \dots, n_\mathrm{s}, \tag{16a}$$
$$0 \leq \lambda^{\mathrm{p}}_{n,m} \perp e - \alpha_{n,m} \geq 0, \quad m = 1, \dots, n_\mathrm{s}. \tag{16b}$$

As a first step, we exploit the continuity of the Lagrange multipliers λ^{p} and λ^{n}. For this purpose, we regard the boundary values of the approximation of λ^{p} and λ^{n} on an interval $[t_n, t_{n+1}]$, which are denoted by $\lambda^{\mathrm{p}}_{n,0}$, $\lambda^{\mathrm{n}}_{n,0}$ at t_n and $\lambda^{\mathrm{p}}_{n,n_\mathrm{s}}$, $\lambda^{\mathrm{n}}_{n,n_\mathrm{s}}$ at t_{n+1}. We impose a continuity condition for the discrete-time versions of λ^{p} and λ^{n} for $n = 0, \dots, N_{\mathrm{FE}} - 1$:

$$\lambda^{\mathrm{p}}_{n,n_\mathrm{s}} = \lambda^{\mathrm{p}}_{n+1,0}, \ \lambda^{\mathrm{n}}_{n,n_\mathrm{s}} = \lambda^{\mathrm{n}}_{n+1,0}, \tag{17}$$

In the sequel, we use only the right boundary points $\lambda^{\mathrm{p}}_{n,n_\mathrm{s}}$ and $\lambda^{\mathrm{n}}_{n,n_\mathrm{s}}$, which are for $c_{n_\mathrm{s}} = 1$, already variables in the RK equations (15).

Remark 3. *It is important to note that $\lambda^{\mathrm{p}}_{0,0}$ and $\lambda^{\mathrm{n}}_{0,0}$ are not defined via Eq.* (17)*, as we do not have a preceding finite element for $n = 0$. However, they are crucial for determining the active set in the first finite element. They are not degrees of freedom but can be pre-computed for a given x_0. Using equation* (11) *we have $\lambda^{\mathrm{p}}_{0,0} = \max(c(x_0), 0)$ and $\lambda^{\mathrm{n}}_{0,0} = -\min(c(x_0), 0)$.*

At a switch of the PSS, i.e., at an active-set change in the DCS (12), we have $c_i(x) = 0$. From Eq. (17) and, due to continuity, it follows that $\lambda^{\mathrm{p}}_i(t)$ and $\lambda^{\mathrm{n}}_i(t)$ must be zero at an active-set change, as well. Moreover, on an interval $t \in (t_n, t_{n+1})$ with a fixed active set, the components of these multipliers are either zero or positive on the whole interval. We must now impose that their discrete-time counterparts, i.e., the stage values $\lambda^{\mathrm{p}}_{n,m}$ and $\lambda^{\mathrm{n}}_{n,m}$, have similar properties. We achieve this with the cross complementarity conditions, which read for $n = 0, \dots, N_{\mathrm{FE}} - 1$, $m = 1, \dots, n_\mathrm{s}$, $m' = 0, \dots, n_\mathrm{s}$, and $m \neq m'$ as:

$$0 = \mathrm{diag}(\lambda^{\mathrm{n}}_{n,m'}) \alpha_{n,m}, \tag{18a}$$
$$0 = \mathrm{diag}(\lambda^{\mathrm{p}}_{n,m'})(e - \alpha_{n,m}), \tag{18b}$$

In contrast to Eq. (16), we have conditions relating variables corresponding to different RK stages within a finite element.

We formalize the claims about the constraints (18) in the next lemma. Recall that in our notation, $\alpha_{n,m,j}$ is the j-th component of the vector $\alpha_{n,m}$.

Lemma 4. *Regard a fixed $n \in \{0, \dots, N_{\mathrm{FE}} - 1\}$ and a fixed $j \in \mathcal{C}$. If any $\alpha_{n,m,j}$ with $m \in \{1, \dots, n_\mathrm{s}\}$ is positive, then all $\lambda^{\mathrm{n}}_{n,m',j}$ with $m' \in \{0, \dots, n_\mathrm{s}\}$ must be zero. Conversely, if any $\lambda^{\mathrm{n}}_{n,m',j}$ is positive, then all $\alpha_{n,m,j}$ are zero.*

Proof. Let $\alpha_{n,m,i}$ be positive, and suppose $\lambda^{\mathrm{n}}_{n,j,i} = 0$ and $\lambda^{\mathrm{n}}_{n,k,i} > 0$ for some $k, j \in \{0, \dots, n_\mathrm{s}\}, k \neq j$, then $\alpha_{n,m,i} \lambda^{\mathrm{n}}_{n,k,i} > 0$ which violates (18), thus all $\lambda^{\mathrm{n}}_{n,m',i} = 0$, $m' \in \{0, \dots, n_\mathrm{s}\}$. The converse is proven similarly. □

An analogous statement holds for $\lambda^{\mathrm{p}}_{n,m}$ and $(e - \alpha_{n,m})$. For the switch detection, it is crucial to include the boundary points of the previous finite element in the cross complementarity conditions (18), namely $\lambda^{\mathrm{p}}_{n+1,0} = \lambda^{\mathrm{p}}_{n,0}$ and $\lambda^{\mathrm{n}}_{n+1,0} = \lambda^{\mathrm{n}}_{n,0}$. A consequence of Lemma 4 is that, if the active-set changes in the j-th component between the n-th and $n+1$-st finite element, then it must hold that $\lambda^{\mathrm{p}}_{n,n_\mathrm{s},j} = \lambda^{\mathrm{p}}_{n+1,0,j} = 0$ and $\lambda^{\mathrm{n}}_{n,n_\mathrm{s},j} = \lambda^{\mathrm{n}}_{n+1,0,j} = 0$. Since $x^{\mathrm{next}}_n = x_{n+1}$, we have from (10a) and (14) the condition

$$c_j(x_{n+1}) = 0,$$

which defines the switching surface between two regions. Therefore, we have implicitly a constraint that forces h_n to adapt such that the switch is detected exactly.

For clarity, the conditions (18) are given in their sparsest form. However, the nonnegativity of $\alpha_{n,m}, \lambda^{\mathrm{p}}_{n,m}$ and $\lambda^{\mathrm{n}}_{n,m}$ allows many equivalent and more compact forms. For instance, we can use inner products instead of componentwise products, or we can even summarize all constraints for a finite element or all finite elements in a single equation, cf. [19], [16] for a similar discussion. We collect the conditions (18) into the equation $G_{\mathrm{cross}}(\mathbf{A}, \mathbf{\Lambda}^{\mathrm{p}}, \mathbf{\Lambda}^{\mathrm{n}}) = 0$.

C. Step equilibration

To complete the derivation of the FESD method for (12), we need to derive the step equilibration conditions. If no active-set changes happen, the cross complementarity constraints (18) are implied by the standard complementarity conditions (16). Therefore, we end up with a system of equations with more degrees of freedom than conditions. The step equilibration constraints aim to remove the degrees of freedom in the appropriate h_n if no switches happen. We

achieve the goals outlined above via the equation:

$$0 = G_{\text{eq}}(\mathbf{h}, \mathbf{A}, \mathbf{\Lambda}^{\text{p}}, \mathbf{\Lambda}^{\text{n}}) := \begin{bmatrix} (h_1 - h_0)\eta_1(\mathbf{A}, \mathbf{\Lambda}^{\text{p}}, \mathbf{\Lambda}^{\text{n}}) \\ \vdots \\ (h_{N_{\text{FE}}-1} - h_{N_{\text{FE}}-2})\eta_{N_{\text{FE}}-1}(\mathbf{A}, \mathbf{\Lambda}^{\text{p}}, \mathbf{\Lambda}^{\text{n}}) \end{bmatrix}, \tag{19}$$

where η_n is an indicator function that is zero only if a switch occurs, otherwise its value is strictly positive. In other words, if a switch happens, the n-th condition in (19) is trivially satisfied. Otherwise, it provides a condition that removes the spurious degrees of freedom. For brevity, we omit to derive the expressions for η_n. They can be obtained by similar reasoning as in [19, Section 3.2.3].

D. Summary of the FESD discretization

We have now introduced all extensions needed to pass from a standard RK (15) to the FESD discretization. With a slight abuse of notation, we collect all equations in a discrete-time system form:

$$s_1 = F_{\text{fesd}}(\mathbf{Z}), \tag{20a}$$
$$0 = G_{\text{fesd}}(\mathbf{Z}, \mathbf{h}, s_0, q, T), \tag{20b}$$

where $F_{\text{fesd}}(\mathbf{x}) = x_{N_{\text{FE}}}$ is the state transition map and $G_{\text{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q, T)$ collects all other internal computations including all RK steps within the regarded time interval:

$$G_{\text{fesd}}(\mathbf{Z}, \mathbf{h}, s_0, q, T) := \begin{bmatrix} G_{\text{std}}(\mathbf{Z}, \mathbf{h}, s_0, q, T) \\ G_{\text{cross}}(\mathbf{A}, \mathbf{\Lambda}^{\text{p}}, \mathbf{\Lambda}^{\text{n}}) \\ G_{\text{eq}}(\mathbf{h}, \mathbf{A}, \mathbf{\Lambda}^{\text{p}}, \mathbf{\Lambda}^{\text{n}}) \\ \sum_{n=0}^{N_{\text{FE}}-1} h_n - T \end{bmatrix}. \tag{21}$$

Here, the control variable q, horizon length T, and initial value s_0 are given parameters, but $\mathbf{h}$ are degrees of freedom.

E. Direct optimal control with FESD

Next, we discretize this OCP using the FESD method. The discretization process is fully automated within `NOSNOC` [16]. Consider $N \geq 1$ control intervals of equal length, indexed by k. We take piecewise constant control discretization, where the control variables are collected $\mathbf{q} = (q_0, \ldots, q_{N-1}) \in \mathbb{R}^{Nn_u}$. All considerations can be easily extended to different control parametrizations. We add the index k to all internal variables. On each control interval k, we use the FESD discretization (20) with N_{FE} internal finite elements. The state values at the control interval boundaries are grouped in the vector $\mathbf{s} = (s_0, \ldots, s_N) \in \mathbb{R}^{(N+1)n_x}$. In $\mathcal{Z} = (\mathbf{Z}_0, \ldots, \mathbf{Z}_{N-1})$ all internal variables, and in $\mathcal{H} = (\mathbf{h}_0, \ldots, \mathbf{h}_{N-1})$ we collect all step sizes.

The discrete-time variant of (1) read as:

$$\min_{\mathbf{s},\mathbf{q},\mathcal{Z},\mathcal{H}} \quad \sum_{k=0}^{N-1} \hat{L}(s_k, \mathbf{x}_k, q_k) + R(s_N) \tag{22a}$$
$$\text{s.t.} \quad s_0 = \bar{x}_0, \tag{22b}$$
$$s_{k+1} = F_{\text{fesd}}(\mathbf{x}_k), \ k = 0, \ldots, N-1, \tag{22c}$$
$$0 = G_{\text{fesd}}(\mathbf{x}_k, \mathbf{Z}_k, q_k), \ k = 0, \ldots, N-1, \tag{22d}$$
$$0 \geq G_{\text{p}}(s_k, q_k), \ k = 0, \ldots, N-1, \tag{22e}$$
$$0 \geq G_{\text{t}}(s_N), \tag{22f}$$

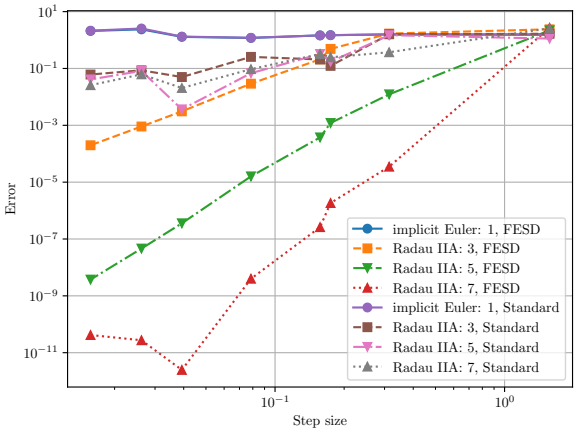

Fig. 1: Accuracy vs. step size: Simulation of example (23) with different RK schemes and step sizes. The number next to the method's name is the order of the underlying RK method.

where $\hat{L} : \mathbb{R}^{n_x} \times \mathbb{R}^{(N_{\text{FE}}+1)n_s n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the discretized running costs. Due to the complementarity constraints in the FESD discretization, (22) is an MPCC. In practice, MPCCs can usually be solved efficiently by solving a sequence of related and relaxed NLPs within a homotopy approach. Such an approach, with some of the standard reformulations [21], [4], is implemented in `NOSNOC`. The underlying NLPs are solved via `IPOPT` [24] called via its `CasADi` interface [3].

IV. Numerical Examples

We show on a numerical simulation example that FESD recovers the order of accuracy that RK methods have for smooth ODEs. We further use a hopping robot OCP example modeled as a system with state jumps in order to show that the step reformulation can outperform Stewart's reformulation used in [19].

A. Integration order experiment

We compare the standard RK time-stepping method for DCS from Eq. (15) to FESD (20) on a simulation example from [19]. Regard the PSS:

$$\dot{x} = f_i(x), \text{ if } x \in R_i, \tag{23}$$

with $f_1(x) = A_1 x, f_2(x) = A_2 x, c(x) = \|x\|_2^2 - 1$, $R_1 = \{x \mid c(x) < 0\}$, $R_2 = \{x \mid c(x) > 0\}$. The system matrices are

$$A_1 = \begin{bmatrix} 1 & 2\pi \\ -2\pi & 1 \end{bmatrix}, \ A_2 = \begin{bmatrix} 1 & -2\pi \\ 2\pi & 1 \end{bmatrix}.$$

The initial value is $x(0) = (e^{-1}, 0)$ and we regard an the interval $t \in [0, T]$ with $T = \frac{\pi}{2}$. It can be shown that the switch happens at $t_{\text{s}} = 1$ and that $x(T) = (\exp(T - t_{\text{s}})\cos(\omega(T - t_{\text{s}})), -\exp(T - t_{\text{s}})\sin(\omega(T - t_{\text{s}})))$, for $T > t_{\text{s}}$. Thus, given a numerical approximation $\hat{x}(t)$, we can determine the global integration error $E(T) = \|x(T) - \hat{x}(T)\|$ and observe the accuracy order of the integrator for a varying step size h.

Figure 1 shows the results of our experiment for the Radau IIA methods. In all cases, the standard RK time-stepping (15)

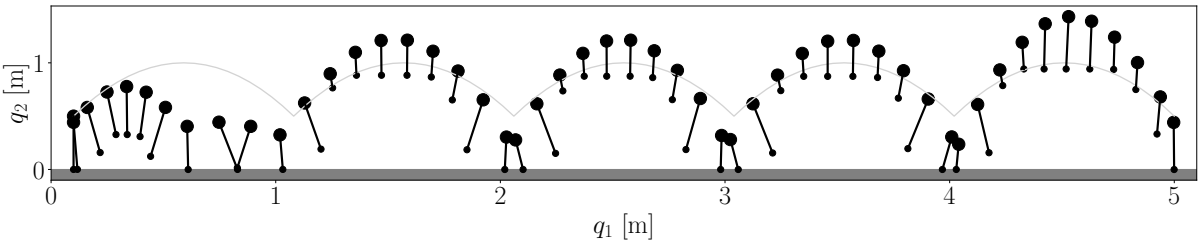

Fig. 2: Several frames of the hopping robot trajectory.

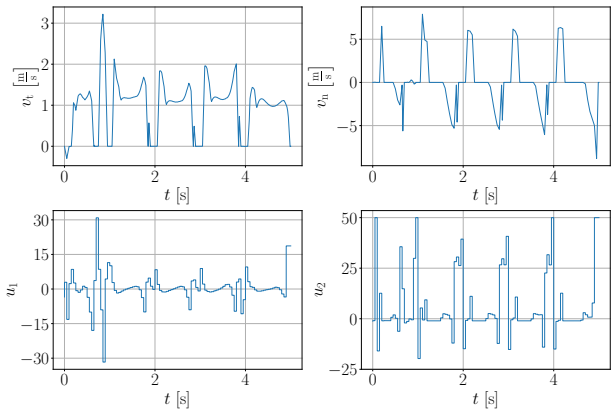

Fig. 3: The top plots show the discontinuous normal and tangential velocity plotted over the *physical time* t, the bottom plots the optimal controls.

has, as expected, only first-order accuracy, i.e., $O(h)$. The FESD method (20) detects the switch and recovers the high integration order properties of the underlying RK method, i.e., $O(h^p)$ with $p = 2n_\mathrm{s} - 1$.

B. Optimal control example with state jumps

We regard the example of a planar hopper with state jumps and friction. Using the time-freezing reformulation, we can transform the system with state jumps into a PSS of the form of (3) [18]. The system is inspired by a reaction wheel and force-controlled single leg as described by [20]. It is assumed that the inertia matrix is a constant diagonal matrix $M = \mathrm{diag}(m_\mathrm{b} + m_\mathrm{l}, m_\mathrm{b} + m_\mathrm{l}, I_\mathrm{b} + I_\mathrm{l}, m_\mathrm{l})$, where $m_\mathrm{b} = 1$ is the mass of the body, $m_\mathrm{l} = 0.25$ is the mass of the link, $I_\mathrm{b} = 0.25$ the inertia of the body, and $I_\mathrm{l} = 0.025$ the inertia of the link.

The configuration $q = (q_1, q_2, \psi, l)$ consists of the 2D position, orientation, and leg length, respectively. The resulting PSS state consists of the position $q(\tau)$, velocity $v(\tau)$ and clock state $t(\tau)$ (needed for time-freezing, cf. [18]), i.e., $x = (q, v, t) \in \mathbb{R}^9$. The robot makes contact with the ground with the tip of its leg, and the gap function is given by $f_c(q) = q_2 - l\cos(\psi)$. We also make use of the contact normal $J_\mathrm{n}(q) = (0, 1, l\sin(\psi), -\cos(\psi))$, and tangent $J_\mathrm{t}(q) = (0, 1, l\cos(\psi), \sin(\psi))$. For the time-freezing reformulation, we regard three switching functions: the gap function, as well as the normal and tangential contact velocities:

$$c(x) = (f_c(q), J_\mathrm{n}(q)^\top v, J_\mathrm{t}(q)^\top v).$$

They define the three regions:

$$\begin{aligned}
R_1 &= \{x \in \mathbb{R}^{n_x} \mid f_c(q) > 0\} \\
&\quad \cup \{x \in \mathbb{R}^{n_x} \mid f_c(q) < 0, J_\mathrm{n}(q)^\top v > 0\}, \\
R_2 &= \{x \in \mathbb{R}^{n_x} \mid f_c(q) < 0, J_\mathrm{n}(q)^\top v < 0, J_\mathrm{t}(q)^\top v > 0\}, \\
R_3 &= \{x \in \mathbb{R}^{n_x} \mid f_c(q) < 0, J_\mathrm{n}(q)^\top v < 0, J_\mathrm{t}(q)^\top v < 0\}.
\end{aligned}$$

In region R_1, we define the unconstrained (free flight) dynamics of the robot, and in regions R_2 and R_3, auxiliary dynamics that mimic state jumps in normal and tangential directions due to frictional impacts, cf. [18]:

$$\begin{aligned}
f_1(x, u) &= (q, M^{-1} f_v(q, u), 1), \\
f_2(x) &= (\mathbf{0}_{4,1}, M^{-1}(J_\mathrm{n}(q) - J_\mathrm{t}(q)\mu)a_\mathrm{n}, 0), \\
f_3(x) &= (\mathbf{0}_{4,1}, M^{-1}(J_\mathrm{n}(q) + J_\mathrm{t}(q)\mu)a_\mathrm{n}, 0),
\end{aligned}$$

with $f_v(q, u) = (-\sin(\psi)u_2, (m_\mathrm{b}+m_\mathrm{l})g + \cos(\psi)u_2, u_1, u_2)$ summarizing all forces acting on the robot, $u = (u_1, u_2) \in \mathbb{R}^2$ are the controls, $\mu = 0.45$ is the coefficient of friction, $g = 9.81$ the gravitational acceleration constant, and $a_\mathrm{n} = 100$ is the auxiliary dynamics constant [18]. Note that the clock state dynamics are $\frac{\mathrm{d}t}{\mathrm{d}\tau} = 1$ in R_1, and $\frac{\mathrm{d}t}{\mathrm{d}\tau} = 0$ in R_2 and R_3. Solution trajectories of the PSS are continuous in time. However, by taking the pieces of the trajectory where $\frac{\mathrm{d}t}{\mathrm{d}\tau} > 0$, we recover the solution of the original system, cf. [18]. To demonstrate the efficiency gained via the step function approach, we run an experiment in which the hopper attempts to cross 5 meters with a given reference trajectory of 5 jumps in $T = 5$ seconds. The initial value is $x(0) = (0.1, 0.5, 0, 0.5, 0, 0, 0, 0)$. Given a reference $x^\mathrm{ref}(t)$, we define the least-squares objective with the running and terminal costs:

$$\begin{aligned}
L(x(\tau), u(\tau)) &= (x(\tau) - x^\mathrm{ref}(\tau))^\top Q (x(\tau) - x^\mathrm{ref}(\tau)) \\
&\quad + \rho_u u(\tau)^\top u(\tau), \\
R(x(T)) &= (x(T) - x^\mathrm{ref}(T)^\top Q_T (x(T) - x^\mathrm{ref}(T)),
\end{aligned}$$

$Q = \mathrm{diag}(100, 100, 20, 20, 0.1, 0.1, 0.1, 0.1, 0)$ $\rho_u = 0.01$, and $Q_T = \mathrm{diag}(300, 300, 300, 300, 0.1, 0.1, 0.1, 0.1, 0)$. We define the path constraints:

$$x_\mathrm{lb} \leq x \leq x_\mathrm{ub}, \tag{24a}$$

$$u_\mathrm{lb} \leq u(t) \leq u_\mathrm{ub}, \tag{24b}$$

$$J_\mathrm{t}(q)^\top v(1 - \alpha_1)(1 - \alpha_2) = 0, \tag{24c}$$

where $x_\mathrm{ub} = (5.1, 1.5, \pi, 0.5, 10, 10, 5, 5, \infty)$, $x_\mathrm{lb} = (0, 0, -\pi, 0.1, -10, -10, -5, -5, -\infty)$, $u_\mathrm{ub} = (50, 50)$, and $u_\mathrm{lb} = -u_\mathrm{ub}$. The last equality constraint (24c) models the

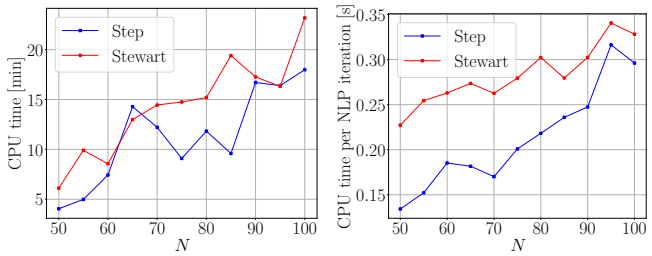

Fig. 4: The total CPU time of the homotopy loop for different N (left plot) and the CPU time per NLP solver iteration (right plot).

following: If the normal contact force, which is proportional to $(1-\alpha_1)(1-\alpha_2)$, is nonnegative (cf. [18]), then $J_{\mathrm{t}}(q)^\top v = 0$. This prevents the optimizer from choosing controls that lead to a lot of slipping when the robot is on the ground. Collecting all the above, we can formulate an OCP of the form of (1), which we discretized with the FESD Radau IIA scheme of order 3 ($n_{\mathrm{s}} = 2$), with $N_{\mathrm{FE}} = 3$ finite elements on every control interval. The OCP is discretized and solved with `nosnoc_py` in a homotopy loop with `IPOPT` [24].

Several frames of an example solution ($N = 100$) can be seen in Figure 2. Figure 3 shows the normal and tangential velocity of the foot tip, and optimal controls. Next, we solve this OCP for 10 different values N (number of control intervals) from 50 to 100 in increments of 5 and compare it to the FESD derived for Stewart's reformulation [19]. We plot the CPU time per NLP iteration and total CPU time for both approaches in Figure 4. The step reformulation leads to faster NLP iterations than the Stewart reformulation, since it needs less variables. The overall computation time is governed by many factors (homotopy loop, initialization, NLP solver performance, etc.) and as such shows a less clear trend. However, we see that in most cases the step approach is still faster by up to 50%.

V. Conclusion and Outlook

This paper introduced an extension of the Finite Elements with Switch Detection (FESD) [19] to optimal control and simulation problems with nonsmooth systems with set-valued step functions. This formulation covers a broad spectrum of practical problems. We show in numerical examples that depending on the switching functions and geometry of the underlying piecewise smooth system it can be computationally more efficient than the formulation in [19]. Several extensions, more numerical benchmarks and a theoretical analysis of the proposed method is available in the journal version of this paper [17].

References

[1] Vincent Acary, Olivier Bonnefon, and Bernard Brogliato. *Nonsmooth modeling and simulation for switched circuits*, volume 69. Springer Science & Business Media, 2010.

[2] Vincent Acary, Hidde De Jong, and Bernard Brogliato. Numerical simulation of piecewise-linear models of gene regulatory networks using complementarity systems. *Physica D: Nonlinear Phenomena*, 269:103–119, 2014.

[3] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[4] Mihai Anitescu, Paul Tseng, and Stephen J Wright. Elastic-mode algorithms for mathematical programs with equilibrium constraints: global convergence and stationarity properties. *Mathematical programming*, 110(2):337–371, 2007.

[5] Brian T Baumrucker and Lorenz T Biegler. Mpec strategies for optimization of a class of hybrid dynamic systems. *Journal of Process Control*, 19(8):1248–1256, 2009.

[6] Jorge Cortes. Discontinuous dynamical systems. *IEEE Control systems magazine*, 28(3):36–73, 2008.

[7] Luca Dieci and Luciano Lopez. Sliding motion on discontinuity surfaces of high co-dimension. a construction for selecting a filippov vector field. *Numerische Mathematik*, 117(4):779–811, 2011.

[8] AF Filippov. *Differential Equations with Discontinuous Righthand Sides: Control Systems*, volume 18. Springer Science & Business Media, 1988.

[9] Nicola Guglielmi and Ernst Hairer. An efficient algorithm for solving piecewise-smooth dynamical systems. *Numerical Algorithms*, 89(3):1311–1334, 2022.

[10] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer, Berlin Heidelberg, 2nd edition, 1991.

[11] Mathew Halm and Michael Posa. Set-valued rigid body dynamics for simultaneous frictional impact. *arXiv preprint arXiv:2103.15714*, 2021.

[12] Anna Machina and Arcady Ponosov. Filippov solutions in the analysis of piecewise linear models describing gene regulatory networks. *Nonlinear Analysis: Theory, Methods & Applications*, 74(3):882–900, 2011.

[13] Armin Nurkanović, Sebastian Albrecht, Bernard Brogliato, and Moritz Diehl. The Time-Freezing Reformulation for Numerical Optimal Control of Complementarity Lagrangian Systems with State Jumps. *Automatica (accepted for publication)*, 2023.

[14] Armin Nurkanović, Sebastian Albrecht, and Moritz Diehl. Limits of MPCC Formulations in Direct Optimal Control with Nonsmooth Differential Equations. In *2020 European Control Conference (ECC)*, pages 2015–2020, 2020.

[15] Armin Nurkanović and Moritz Diehl. Continuous optimization for control of hybrid systems with hysteresis via time-freezing. *IEEE Control Systems Letters*, 2022.

[16] Armin Nurkanović and Moritz Diehl. NOSNOC: A software package for numerical optimal control of nonsmooth systems. *IEEE Control Systems Letters*, 2022.

[17] Armin Nurkanović, Anton Pozharskiy, Jonathan Frey, and Moritz Diehl. Finite elements with switch detection for numerical optimal control of nonsmooth dynamical systems with set-valued step functions. *arXiv preprint arXiv:2307.03482*, 2023.

[18] Armin Nurkanović, Tommaso Sartor, Sebastian Albrecht, and Moritz Diehl. A Time-Freezing Approach for Numerical Optimal Control of Nonsmooth Differential Equations with State Jumps. *IEEE Control Systems Letters*, 5(2):439–444, 2021.

[19] Armin Nurkanović, Mario Sperl, Sebastian Albrecht, and Moritz Diehl. Finite Elements with Switch Detection for Direct Optimal Control of Nonsmooth Systems. 2022.

[20] Marc H Raibert, H Benjamin Brown Jr, Michael Chepponis, Jeff Koechling, and Jessica K Hodgins. Dynamically stable legged locomotion. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1989.

[21] Stefan Scholtes. Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 11(4):918–936, 2001.

[22] David Stewart. A high accuracy method for solving odes with discontinuous right-hand side. *Numerische Mathematik*, 58(1):299–328, 1990.

[23] David E Stewart and Mihai Anitescu. Optimal control of systems with discontinuous differential equations. *Numerische Mathematik*, 114(4):653–695, 2010.

[24] Andreas Wächter and Lorenz 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.