

# NOSNOC User's Manual

## Preliminary version

Armin Nurkanović, Jonathan Frey, Moritz Diehl <sup>1</sup>

March, 2022

<sup>1</sup>Department of Microsystems Engineering (IMTEK) and Department of Mathematics, University of Freiburg, Germany  
`{armin.nurkanovic,jonathan.frey,moritz.diehl}@imtek.uni-freiburg.de`

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Problem Formulation</b>	<b>2</b>
2.1	Notation summary . . . . .	3
2.2	Stewart’s reformulation of the PSS into a Dynamic Complementarity System . . . . .	3
2.3	The continuous-time optimal control problem . . . . .	5
2.4	Standard IRK Discretization of a Single Control Interval . . . . .	6
2.5	FESD Discretization of a Single Control Interval . . . . .	7
2.6	Discretization of the OCP . . . . .	9
<b>3</b>	<b>Settings and Model</b>	<b>10</b>
3.1	The <code>model</code> struct . . . . .	10
3.2	All <code>settings</code> at a glance . . . . .	13
3.3	The <code>solver_initialization</code> . . . . .	18
3.4	The <code>results</code> . . . . .	18
3.5	List of examples . . . . .	18
<b>4</b>	<b>Homotopy and MPCC settings</b>	<b>19</b>
4.0.1	Direct solve ( <code>mpcc_mode =1</code> ) . . . . .	19
4.0.2	Smoothing ( <code>mpcc_mode =2</code> ) and Relaxation ( <code>mpcc_mode =3</code> ) . . . . .	19
4.0.3	$\ell_1$ -Penalty ( <code>mpcc_mode =4</code> ) . . . . .	20
4.0.4	Elastic Mode ( <code>mpcc_mode = 5</code> ) . . . . .	20
4.0.5	Barrier Controlled Penalty Homotopy ( <code>mpcc_mode =8</code> ) . . . . .	20
4.0.6	Objective Scaling . . . . .	20
4.1	Cross complementarities . . . . .	21
4.1.1	Complement all stage values with each other (stage-stage level) . . . . .	21
4.1.2	Sum all $\theta$ and complement with every stage of $\lambda$ and vice versa (FE-stage level) . . . . .	22
4.1.3	Complement sum of all $\theta$ and sum of all $\lambda$ at a FE (FE-FE level) . . . . .	23
4.1.4	Complement sum of all $\theta$ and sum of all $\lambda$ at a control interval (FE-FE level summarized) . . . . .	23
4.1.5	Complement sum of all $\theta$ and sum of all $\lambda$ at a control interval (FE-FE level summarized) . . . . .	23
4.2	Step equilibration . . . . .	24
4.3	FESD Integrator . . . . .	25
<b>5</b>	<b>Structure of the Software</b>	<b>25</b>

# 1 Introduction

NOSNOC is an open source software package for Nonsmooth Numerical Optimal Control. It is a modular tool based on CasADi [1], IPOPT [2] and MATLAB, for numerically solving Optimal Control Problems (OCP) with piecewise smooth systems (PSS). It relies on the recently introduced Finite Elements with Switch Detection [3] which enables high accuracy optimal control and simulation of PSS. The time-freezing reformulation, which transforms several classes of systems with state jumps into PSS is supported as well. This enables the treatment of a broad class of nonsmooth systems in a unified way. The algorithms and reformulations yield mathematical programs with complementarity constraints. They can be solved with techniques of continuous optimization in a homotopy procedure, without the use of integer variables. The goal of the package is to automate all reformulations and to make nonsmooth optimal control problems practically solvable, without deep expert knowledge.

# 2 Problem Formulation

The goal is to solve optimal control problems of the form:

$$\begin{aligned}
\min_{x(\cdot), u(\cdot)} \quad & \int_0^T f_q(x(t), u(t)) dt + f_T(x(T)) & (1a) \\
\text{s.t.} \quad & x_0 = s_0, & (1b) \\
& \dot{x} = f_i(x, u), \text{ if } x \in R_i \subset \mathbb{R}^{n_x}, \ i \in \mathcal{I} := \{1, \dots, n_f\}, \quad t \in [0, T], & (1c) \\
& G_{\text{ineq}, \text{lb}} \leq G_{\text{ineq}}(x(t), u(t)) \leq G_{\text{ineq}, \text{ub}}, \quad t \in [0, T], & (1d) \\
& x_{\text{lb}} \leq x(t) \leq x_{\text{ub}}, \quad t \in [0, T], & (1e) \\
& u_{\text{lb}} \leq u(t) \leq u_{\text{ub}}, \quad t \in [0, T], & (1f) \\
& G_{T, \text{lb}} \leq G_T(x(t)) \leq G_{T, \text{ub}}, & (1g)
\end{aligned}$$

where  $u(t) \in \mathbb{R}^{n_u}$  is the control function,  $f_q : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$  is stage cost and  $f_T : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$  is the terminal cost. The path and terminal constraints are collected in the functions  $G_{\text{ineq}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_{g1}}$  and  $G_{\text{term}} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_{g2}}$ , respectively. The system (1c) is a *piecewise smooth system*. It is assumed that  $R_i$  are disjoint, nonempty, connected and open sets. The functions  $f_i(\cdot)$  are assumed to be smooth on an open neighborhood of  $\bar{R}_i$  and  $n_f$  is a positive integer. The event of  $x(\cdot)$  reaching some boundary  $\partial R_i$  is called a *switch*. The right hand side (r.h.s.) of (1c) is in general discontinuous in  $x$ .

Several important classes of systems with state jumps can be reformulated into the form of (1c) via the *time-freezing* reformulation [4, 5, 6]. Therefore, the focus on PPS enables a unified treatment of several different classes of nonsmooth systems.

Much more mathematical details can be found in paper that introduces NOSNOC [7], the FESD paper [6] and the time-freezing papers [4, 5, 6].

## 2.1 Notation summary

The complementary conditions for two vectors  $a, b \in \mathbb{R}^n$  read as  $0 \leq a \perp b \leq 0$ , where  $a \perp b$  means  $a^\top b = 0$ . For two scalar variables  $a, b$  the so-called C-functions have the property  $\phi(a, b) = 0 \iff a \geq 0, b \geq 0, ab = 0$ . Examples are the natural residual functions  $\phi_{\text{NR}}(a, b) = \min(a, b)$  or the Fischer-Burmeister function  $\phi_{\text{FB}}(a, b) = a + b - \sqrt{a^2 + b^2}$ . If  $a, b \in \mathbb{R}^n$ , we use  $\phi(\cdot)$  component-wise and define  $\Phi(a, b) = (\phi(a_1, b_1), \dots, \phi(a_n, b_n))$ . All vector inequalities are to be understood element-wise,  $\text{diag}(x) \in \mathbb{R}^{n \times n}$  returns a diagonal matrix with  $x \in \mathbb{R}^n$  containing the diagonal entries. The concatenation of two column vectors  $a \in \mathbb{R}^{n_a}, b \in \mathbb{R}^{n_b}$  is denoted by  $(a, b) := [a^\top, b^\top]^\top$ , the concatenation of several column vectors is defined in an analogous way. The identity matrix is denoted by  $I \in \mathbb{R}^{n \times n}$  and a column vector with all ones is denoted by  $e = (1, 1, \dots, 1) \in \mathbb{R}^n$ , their dimension is clear from the context. The closure of a set  $C$  is denoted by  $\overline{C}$ , its boundary as  $\partial C$  and  $\text{conv}(C)$  is its convex hull. Given a matrix  $M \in \mathbb{R}^{n \times m}$ , its  $i$ -th row is denoted by  $M_{i, \bullet}$  and its  $j$ -th column is denoted by  $M_{\bullet, j}$ . For a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  we denote by  $Df(x) = \frac{\partial f(x)}{\partial x} \in \mathbb{R}^{m \times n}$  the Jacobian matrix and by  $\nabla f(x) := \frac{\partial f(x)}{\partial x}^\top$  its transpose. When clear from context, we often drop the dependency on time  $t$ .

## 2.2 Stewart's reformulation of the PSS into a Dynamic Complementarity System

To have a computationally more useful we transform the PSS (1c) into a Dynamic Complementarity System (DCS). First, to have a meaningful notion of solution we regard the Filippov convexification of (1c), which reads as:

$$\dot{x} \in F_F(x, u) = \left\{ F(x)\theta \mid \sum_{i \in \mathcal{I}} \theta_i = 1, \theta_i \geq 0, \theta_i = 0 \text{ if } x \notin \overline{R_i}, \forall i \in \mathcal{I} \right\}, \quad (2)$$

with  $\theta = (\theta_1, \dots, \theta_{n_f}) \in \mathbb{R}^{n_f}$  and  $F(x) := [f_1(x), \dots, f_{n_f}(x)] \in \mathbb{R}^{n_x \times n_f}$ .

Second, we assume the sets  $R_i$  are define by the zero level sets of functions  $c_i(x) = 0, i = 1, \dots, n_c$ , which are collected into the vector  $c(x) = (c_1(x), \dots, c_{n_c}(x)) \in \mathbb{R}^{n_c}$ . Without lost of generality, the sets are defined as  $R_1 = \{x \in \mathbb{R}^{n_x} \mid c_1(x) > 0, \dots, c_{n_c}(x) > 0\}$ ,  $R_2 = \{x \in \mathbb{R}^{n_x} \mid c_1(x) > 0, \dots, c_{n_c}(x) < 0\}$ , and so on. This can compactly encoded with a sign matrix  $S \in \mathbb{R}^{n_f \times n_c}$  which has no repeating rows:

$$S = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & -1 \\ \vdots & \vdots & \dots & \vdots & \\ -1 & -1 & \dots & -1 & -1 \end{bmatrix}. \quad (3)$$

Thus the sets  $R_i$  are compactly denoted by:

$$R_i = \{x \in \mathbb{R}^{n_x} \mid \text{diag}(S_{i, \bullet})c(x) > 0\}. \quad (4)$$

We use Stewart's reformulation of DCS into PSS [9]. In this case, it is assumed that the sets  $R_i$  are represent via the *discriminant functions*  $g_i(\cdot)$ :

$$R_i = \{x \in \mathbb{R}^{n_x} \mid g_i(x) < \min_{j \in \mathcal{I}, j \neq i} g_j(x)\}. \quad (5)$$

Using the representation via the sign matrix  $S$  in Eq. (4), it can be shown that the function  $g : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_f}$  whose components are  $g_i(x)$  can be found as [3]:

$$g(x) = -Sc(x). \quad (6)$$

**Example 1.** Note that the zero level sets of  $c(x) = 0$  represent the region boundaries. Regard  $x \in \mathbb{R}^2$  and the following two functions  $c_1(x) = x_1$  and  $c_2(x) = x_2$  which result in four regions:

$$\begin{aligned} R_1 &= \{x \mid c_1(x) > 0, c_2(x) > 0\}, \\ R_2 &= \{x \mid c_1(x) > 0, c_2(x) < 0\}, \\ R_3 &= \{x \mid c_1(x) < 0, c_2(x) > 0\}, \\ R_4 &= \{x \mid c_1(x) < 0, c_2(x) < 0\}. \end{aligned}$$

We define also  $c(x) = (c_1(x), c_2(x)) \in \mathbb{R}^2$ . Then the corresponding sign matrix  $S$  (Eq. (4)) reads as

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

Now using Eq. (4), we can see that, for  $R_2$  we have  $S_{2,\bullet} = [1 \quad -1]$  and that  $\text{diag}(S_{2,\bullet}) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ . Therefore from (4) we obtain

$$\begin{aligned} R_2 &= \{x \in \mathbb{R}^2 \mid \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} c(x) > 0\} \\ &= \{x \in \mathbb{R}^2 \mid c_1(x) > 0, -c_2(x) > 0\} \\ &= \{x \in \mathbb{R}^2 \mid c_1(x) > 0, c_2(x) < 0\}. \end{aligned}$$

□

Using Stewart's definition of the sets via Eq. (5) the multipliers  $\theta(\cdot)$  in the r.h.s. of (2) can be found as a solution of a Linear Program (LP) [9], and (2) is equivalent to

$$\dot{x} = F(x, u)\theta(x), \quad (7a)$$

$$\theta(x) \in \arg \min_{\tilde{\theta} \in \mathbb{R}^{n_f}} g(x)^\top \tilde{\theta} \quad \text{s.t.} \quad e^\top \tilde{\theta} = 1, \tilde{\theta} \geq 0. \quad (7b)$$

Using the KKT conditions of the LP we can rewrite the last system into the following DCS:

$$\dot{x} = F(x, u)\theta \quad (8a)$$

$$0 = g(x) - \lambda - \mu e, \quad (8b)$$

$$1 = e^\top \theta, \quad (8c)$$

$$0 \leq \theta \perp \lambda \geq 0, \quad (8d)$$

where  $\lambda \in \mathbb{R}_{\geq 0}^{n_f}$  and  $\mu \in \mathbb{R}$  are the Lagrange multipliers associated to the constraints of the LP (7b) and  $z := (\theta, \lambda, \mu) \in \mathbb{R}^{n_z}$ ,  $n_z = 2n_f + 1$  are algebraic variables.

For compact notation, we use a C-function  $\Phi(\cdot, \cdot)$  for the complementarity conditions and rewrite the KKT conditions of the LP as a nonsmooth equation

$$G_{\text{LP}}(x, \theta, \lambda, \mu) := \begin{bmatrix} g(x) - \lambda - \mu e \\ 1 - e^\top \theta \\ \Phi(\theta, \lambda) \end{bmatrix} = 0. \quad (9)$$

Finally, the Filippov system is equivalent to the following DCS (which can be interpreted as a nonsmooth differential algebraic equation):

$$\dot{x} = F(x, u)\theta, \quad (10a)$$

$$0 = G_{\text{LP}}(x, \theta, \lambda, \mu). \quad (10b)$$

## 2.3 The continuous-time optimal control problem

We regard the continuous-time optimal control problem

$$\min_{x(\cdot), u(\cdot), z(\cdot)} \int_0^T f_q(x(t), u(t))dt + f_{qT}(x(T)) \quad (11a)$$

$$\text{s.t. } x_0 = s_0, \quad (11b)$$

$$\dot{x}(t) = F(x(t), u(t))\theta(t), \quad t \in [0, T], \quad (11c)$$

$$0 = g(x(t)) - \lambda(t) - \mu(t)e, \quad t \in [0, T], \quad (11d)$$

$$0 \leq \theta(t) \perp \lambda(t) \geq 0, \quad t \in [0, T], \quad (11e)$$

$$1 = e^\top \theta(t), \quad t \in [0, T], \quad (11f)$$

$$x_{\text{lb}} \leq x(t) \leq x_{\text{ub}}, \quad t \in [0, T], \quad (11g)$$

$$u_{\text{lb}} \leq u(t) \leq u_{\text{ub}}, \quad t \in [0, T], \quad (11h)$$

$$G_{\text{ineq,lb}} \leq G_{\text{ineq}}(x(t), u(t)) \leq G_{\text{ineq,ub}}, \quad t \in [0, T], \quad (11i)$$

$$G_{\text{T,lb}} \leq G_{\text{T}}(x(t)) \leq G_{\text{T,ub}}, \quad (11j)$$

where  $u(t) \in \mathbb{R}^{n_u}$  is the control function,  $f_{qT} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$  is the Mayer term and  $f_q : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$  is the Lagrange objective term. The path and terminal constraints are collected in the functions  $G_{\text{ineq}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_{g1}}$  and  $G_{\text{term}} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_{g2}}$ , respectively.

## 2.4 Standard IRK Discretization of a Single Control Interval

We briefly state how a single control interval is discretized in **NOSNOC**. We start with the standard IRK discretization and later we detail how this is extended to obtain FESD. Consider the single control interval  $[0, T]$  with a constant externally choose control input  $q$ . Suppose the initial value  $x_0 = s_0$  to be given. The control interval is divided into  $N_{\text{FE}}$  finite elements (i.e., integration intervals)  $[t_n, t_{n+1}]$  via the grid points  $0 = t_0 < t_1 < \dots < t_{N_{\text{FE}}} = T$ .

On each of these intervals an  $n_s$ -stage Implicit Runge-Kutta (IRK) scheme is applied. An IRK scheme is defined by its Butcher tableau entries  $A \in \mathbb{R}^{n_s \times n_s}$ ,  $b \in \mathbb{R}^{n_s}$ ,  $c \in \mathbb{R}^{n_s}$  [10]. The fixed step-size reads as  $h_n = t_{n+1} - t_n$ ,  $n = 0, \dots, N_{\text{FE}} - 1$ . The approximation of the differential state at the grid points  $t_n$  is denoted by  $x_n \approx x(t_n)$ . The derivative of state at the stage points  $t_n + c_i h_n$ ,  $i = 1, \dots, n_s$ , for a single finite element are summarized in the vector  $V_n := (v_{n,1}, \dots, v_{n,n_s}) \in \mathbb{R}^{n_s \cdot n_x}$ . The stage values for the algebraic variables are collected in the vectors:  $\Theta_n := (\theta_{n,1}, \dots, \theta_{n,n_s}) \in \mathbb{R}^{n_s \cdot n_f}$ ,  $\Lambda_n := (\lambda_{n,1}, \dots, \lambda_{n,n_s}) \in \mathbb{R}^{n_s \cdot n_f}$  and  $M_n := (\mu_{n,1}, \dots, \mu_{n,n_s}) \in \mathbb{R}^{n_s}$ . We also define the vector  $Z_n = (x_n, \Theta_n, \Lambda_n, M_n, V_n)$ , which collects all internal variables.

With  $x_n^{\text{next}}$  we denote the value at the next time step  $t_{n+1}$ , which is obtained after a single IRK step. For the FESD scheme, additional to the stage values of  $\lambda_{n,i}$ ,  $\mu_{n,i}$  we need their values on  $t_n$  and  $t_{n+1}$  which are denoted by  $\lambda_{n,0}$ ,  $\mu_{n,0}$  and  $\lambda_{n,n_s+1}$ ,  $\mu_{n,n_s+1}$ , respectively. Due to continuity, we impose that  $\lambda_{n,n_s+1} = \lambda_{n+1,0}$  and  $\mu_{n,n_s+1} = \mu_{n+1,0}$  and compute explicitly only the right boundary points of the finite elements. These values can be computed from an LP solve for  $x_n^{\text{next}}$ :

$$0 = G_{\text{LP}}(x_n^{\text{next}}, \theta_{n,n_s+1}, \lambda_{n,n_s+1}, \mu_{n,n_s+1}). \quad (12)$$

Now we write the IRK equations for the DCS (10), including the additional Eq. (12) for the boundary value, in a compact *differential* form. The IRK equations of a single integration step are summarized in

$$G_{\text{irk}}(x_n^{\text{next}}, h_n, Z_n, q) := \begin{bmatrix} v_{n,1} - F(x_n + h_n \sum_{j=1}^{n_s} a_{1,j} v_{n,j}, q) \theta_{n,1} \\ \vdots \\ v_{n,n_s} - F(x_n + h_n \sum_{j=1}^{n_s} a_{n_s,j} v_{n,j}, q) \theta_{n,n_s} \\ G_{\text{LP}}(x_n + h_n \sum_{j=1}^{n_s} a_{1,j} v_{n,j}, \theta_{n,1}, \lambda_{n,1}, \mu_{n,1}) \\ \vdots \\ G_{\text{LP}}(x_n + h_n \sum_{j=1}^{n_s} a_{n_s,j} v_{n,j}, \theta_{n,n_s}, \lambda_{n,n_s}, \mu_{n,n_s}) \\ G_{\text{LP}}(x_n^{\text{next}}, \lambda_{n,n_s+1}, \theta_{n,n_s+1}, \mu_{n,n_s+1}) \\ x_n^{\text{next}} - x_n - h_n \sum_{i=1}^{n_s} b_i v_{n,i} \end{bmatrix}. \quad (13)$$

We remind the reader that  $h_n$  in (13) are implicitly fixed by the chosen discretization grid. We also want to highlight, that some IRK schemes contain the right boundary points as a stage point (e.g., Radau and Lobatto schemes [10]),

i.e.,  $c_{n_s} = 1$ , thus we have  $\lambda_{n,n_s} = \lambda_{n,n_s+1}$   $\mu_{n,n_s} = \mu_{n,n_s+1}$  and we do not need the additional constraint (12).

The next equations summarize all  $N_{\text{FE}}$  IRK steps over the control interval in the same discrete-time system style. We introduce some new notation. All variables for a single control interval of all finite elements are collected in the following vectors  $\mathbf{x} = (x_0, x_0^{\text{next}}, \dots, x_{N_{\text{FE}}}) \in \mathbb{R}^{(2N_{\text{FE}}+1)n_x}$ ,  $\mathbf{V} = (V_0, \dots, V_{N_{\text{FE}}-1}) \in \mathbb{R}^{N_{\text{FE}}n_s n_x}$  and  $\mathbf{h} := (h_0, \dots, h_{N_{\text{FE}}-1}) \in \mathbb{R}^{N_{\text{FE}}}$ . For the algebraic variables, we collect the stage values and the newly introduced boundary values into the vectors  $\Theta = (\Theta_0, \theta_{0,n_s+1}, \dots, \Theta_{N_{\text{FE}}-1}) \in \mathbb{R}^{N_{\text{FE}}(n_s+1)n_f}$ . The vectors  $\Lambda \in \mathbb{R}^{N_{\text{FE}}(n_s+1)n_f}$ ,  $\mathbf{M} \in \mathbb{R}^{N_{\text{FE}}(n_s+1)}$  are defined accordingly. The vector  $\mathbf{Z} = (\mathbf{x}, \Theta, \Lambda, \mathbf{M}, \mathbf{V})$  collects all *internal* variables.

Finally, we can summarize all computations over a single control interval which we call here the **standard discretization**. We interpret it as a discrete-time nonsmooth system:

$$s_1 = F_{\text{std}}(\mathbf{x}), \quad (14a)$$

$$0 = G_{\text{std}}(s_0, \mathbf{Z}, q), \quad (14b)$$

with  $F_{\text{std}}(\mathbf{x}) = x_{N_{\text{FE}}}$  and

$$G_{\text{std}}(s_0, \mathbf{h}, \mathbf{Z}, q) := \begin{bmatrix} x_1 - x_0^{\text{next}} \\ G_{\text{irk}}(x_0^{\text{next}}, Z_n, q) \\ \vdots \\ x_{N_{\text{FE}}} - x_{N_{\text{FE}}-1}^{\text{next}} \\ G_{\text{irk}}(x_{N_{\text{FE}}-1}^{\text{next}}, Z_{N_{\text{FE}}-1}, q) \end{bmatrix}.$$

Note that  $s_0$  and  $\mathbf{h}$  are given parameters.

## 2.5 FESD Discretization of a Single Control Interval

In Finite Elements with Switch Detection (FESD) scheme the step-sizes  $h_n$  are left as degrees of freedom (as first proposed by [11]) such that the grid points  $t_n$  can coincide with the switching times. To exploit the additional degrees of freedom and to achieve exact switch detection we introduce additional conditions to the standard IRK equations (14) called *cross complementaries* and *step-equilibration*.

**The cross complementaries** The cross complementarity conditions avoid switching inside a finite element and make exact switch detection possible. They read as

$$0 = G_{\text{cross}}(\Theta, \Lambda) := \begin{bmatrix} \sum_{i=1}^{n_s} \sum_{j=1, j \neq i}^{n_s+1} \theta_{1,i}^\top \lambda_{1,j} \\ \vdots \\ \sum_{i=1}^{n_s} \sum_{j=0, j \neq i}^{n_s} \theta_{N_{\text{FE}}-1,i}^\top \lambda_{N_{\text{FE}}-1,j} \end{bmatrix}. \quad (15)$$



Due to the nonnegativity of the variables  $\Theta$  and  $\Lambda$  there are ten equivalent formulations of these constraints which have different degrees of sparsity, see Section 4.1 for an overview. All variants are supported in **NOSNOC**.

**The step-equilibration** If now switches happen the cross complementarity conditions are trivial satisfied and spurious degrees of freedom in  $h_n$  appear. For detailed explanations see [3]. We introduce the following backward and forward sums:

$$\begin{aligned}\sigma_n^{\lambda,B} &= e^\top \Lambda_{n-1}, \\ \sigma_n^{\lambda,F} &= e^\top \Lambda_n, \\ \sigma_n^{\theta,B} &= e^\top \Theta_{n-1}, \\ \sigma_n^{\theta,F} &= e^\top \Theta_n,\end{aligned}$$

The switch indicating quantities reads as

$$\begin{aligned}\pi_n^\lambda &= \text{diag}(\sigma_n^{\lambda,B}) \sigma_n^{\lambda,F}, \\ \pi_n^\theta &= \text{diag}(\sigma_n^{\theta,B}) \sigma_n^{\theta,F}, \\ v_n &= \pi_n^\lambda + \pi_n^\theta.\end{aligned}$$

The joint effect of all components is collected in the product

$$\eta_n(\Theta, \Lambda) := \eta(\Theta_{n-1}, \Lambda_{n-1}, \Theta_n, \Lambda_n) = \prod_{i=1}^{n_f} (v_n)_i.$$

The constraints that remove possible spurious degrees of freedom in  $h_n$  read as:

$$0 = G_{\text{eq}}(\mathbf{h}, \Theta, \Lambda) := \begin{bmatrix} (h_1 - h_0) \eta_1(\Theta, \Lambda) \\ \vdots \\ (h_{N_{\text{FE}}-1} - h_{N_{\text{FE}}-2}) \eta_{N_{\text{FE}}-1}(\Theta, \Lambda) \\ \sum_{n=0}^{N_{\text{FE}}-1} h_n - T \end{bmatrix}. \quad (16)$$

These constraint are sometimes very nonlinear and might harm the convergence. Several different formulations and heuristic are offered to help the convergence. For an overview see Section 4.2.

**A single FESD step** We summarize the developemnts of this subsection which result in the FESD discretization. We use again the same discrete-timer representation and refer to this as the *FESD discretization*.

$$s_1 = F_{\text{fesd}}(\mathbf{x}), \quad (17a)$$

$$0 = G_{\text{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q), \quad (17b)$$

and  $F_{\text{fesd}}(\mathbf{x}) = x_{N_{\text{FE}}}$  renders the state transition map and the equation  $0 = G_{\text{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q)$  collects all other internal computations including all IRK steps within the regarded control interval:

$$G_{\text{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q) := \begin{bmatrix} G_{\text{std}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q) \\ G_{\text{cross}}(\boldsymbol{\Theta}, \boldsymbol{\Lambda}) \\ G_{\text{eq}}(\mathbf{h}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}) \end{bmatrix}.$$

For a known control function  $q$ , the formulation (17) can be used as a integrator with exact switch detection for PSS (1c). This feature is implemented in `NOSNOC` via the function `integrator_fesd()`. The FESD scheme can automatically handle all kind of switching cases such as [12]:

1. crossing a discontinuity,
2. sliding mode,
3. leaving a sliding mode and
4. spontaneous switches.

## 2.6 Discretization of the OCP

Using the FESD (recommended) or standard discretization of a single control interval we can discretized and OCP with  $N_{\text{stg}} \geq 1$  control intervals indexed by  $k$ . We assume piecewise constant controls collected in  $\mathbf{q} = (q_0, \dots, q_{N_{\text{stg}}-1}) \in \mathbb{R}^{N_{\text{stg}} n_u}$ . All internal variables of every control interval are additionally equipped with an index  $k$ . On every control interval  $k$  we apply a discretization (17) with  $N_{\text{fe},k}$  internal finite elements. The state values at the control interval boundaries are collected in  $\mathbf{s} = (s_0, \dots, s_{N_{\text{stg}}}) \in \mathbb{R}^{(N_{\text{stg}}+1)n_x}$ . The following vectors collect all internal variables of all discretization steps:  $\mathcal{H} = (\mathbf{h}_0, \dots, \mathbf{h}_{N_{\text{stg}}-1})$  and  $\mathcal{Z} = (\mathbf{Z}_0, \dots, \mathbf{Z}_{N_{\text{stg}}-1})$ . Finally the discretized version of the OCP (11) reads as:

$$\min_{\mathbf{s}, \mathbf{q}, \mathcal{H}, \mathcal{Z}} \sum_{k=1}^{N_{\text{stg}}-1} \hat{f}_{\text{q}}(s_k, \mathbf{x}_k, q_k) + \hat{f}_{\text{qT}}(s_{N_{\text{stg}}}) \quad (18a)$$

$$\text{s.t. } s_0 = \bar{x}_0, \quad (18b)$$

$$s_{k+1} = F_{\text{fesd}}(\mathbf{x}_k), \quad k = 0, \dots, N_{\text{stg}} - 1, \quad (18c)$$

$$0 = G_{\text{fesd}}(\mathbf{x}_k, \mathbf{h}_k, \mathbf{Z}_k, q_k), \quad k = 0, \dots, N_{\text{stg}} - 1, \quad (18d)$$

$$G_{\text{ineq,lb}} \leq G_{\text{ineq}}(s_k, q_k) \leq G_{\text{ineq,ub}}, \quad k = 0, \dots, N_{\text{stg}} - 1, \quad (18e)$$

$$x_{\text{lb}} \leq s_k \leq x_{\text{ub}}, \quad k = 0, \dots, N_{\text{stg}}, \quad (18f)$$

$$u_{\text{lb}} \leq q_k \leq u_{\text{ub}}, \quad k = 0, \dots, N_{\text{stg}} - 1, \quad (18g)$$

$$G_{\text{T,lb}} 0 \leq G_{\text{T}}(s_{N_{\text{stg}}}) \leq G_{\text{T,ub}}, \quad (18h)$$

where  $\hat{f}_{\text{qT}} : \mathbb{R}^{n_x} \times \mathbb{R}^{(N_{\text{FE}}+1)n_x n_u} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$  and  $\hat{f}_{\text{q}} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$  are the discrteized stage and terminal costs. The path constraint  $G_{\text{ineq}}(\cdot)$  in this formulation is evaluated only on the control discretization grid but the possibility to evaluate the constrain on a finer grid is supported as well.

### 3 Settings and Model

This sections provides details on the user settings and model input.

#### 3.1 The model struct

This table gives an overview of the model data and what is mandatory user input.

Name	Type	Is it mandatory	Description
<b>User input</b>			
N_stages	double	yes	This is the number of control intervals $n_s$ .
N_finte_elements	double	yes	This is the number of finite elements $N_{FE}$ per control interval/stage. If a single value is passed the all stages have the same number of finite elements. Alternatively, a <b>double</b> vector of size $[1, n_s]$ can be passed to determined the number of finite elements $N_{FEk}$ for every stage.
T	double	yes	Length of the control horizon $T$ in the OCP (11). Note that if time-freezing is used this is the total numerical time horizon and a time scaling will be introduced to achieve the same terminal physical time.
x0	double	no	This is the initial value of the PSS $s_0$ .
x	CasADi SX or MX	yes	A CasADi symbolic vector for the differential states.
u	CasADi SX or MX	yes	A CasADi symbolic vector for the control variables.
F	CasADi expr	yes	The Matrix $F(x)$ that stores all modes $f_i(x)$ of the PSS.
S	double	yes, if <b>g_ind</b> not given	The sign matrix $S$ for encoding the regions $R_i$ (4). Note that if this matrix is not passed, it is assumed that the function $g(x)$ (6) is provided.

<code>c</code>	CasADi expr	no, if <code>g_ind</code> not given	The constraint function $c(x)$ that defines the regions $R_i$ in (4)
<code>g_ind</code>	CasADi sym expr	yes, if no <code>S</code> and <code>c</code> not given	The discriminant function $g(x)$ (6).
<code>f_q</code>	CasADi sym expr	no	CasADi symbolic expression for the stage cost $f_q(\cdot, \cdot)$ in the OCP (11).
<code>f_qT</code>	CasADi sym expr	no	CasADi symbolic expression for the terminal cost $f_{q,T}(\cdot)$ in the OCP (11).
<code>g_ineq</code>	CasADi sym expr	no	CasADi symbolic expression for the path constraints $G_{\text{ineq}}(\cdot)$ in the OCP (11).
<code>g_terminal</code>	CasADi sym expr	no	CasADi symbolic expression for the terminal constraints $G_T(\cdot)$ in the OCP (11).
<code>lbx</code>	<code>double</code>	no	The upper bound for the differential state $x$ in (11g). If not provided, its entries are by default set to <code>inf</code> .
<code>ubx</code>	<code>double</code>	no	The lower bound for the differential state $x$ in (11g). If not provided, its entries are by default set to <code>-inf</code> .
<code>lbu</code>	<code>double</code>	no	The lower bounds for the control $u$ in (11h). If not provided, its entries are by default set to <code>-inf</code> .
<code>ubu</code>	<code>double</code>	no	The upper bound for the control $u$ in (11h). If not provided, its entries are by default set to <code>inf</code> .
<code>u0</code>	<code>double</code>	no	Initial guess for the control $u$ in (11). If not provided, its entries are by default set to 0.
<code>lb_g_ineq</code>	<code>double</code>	no	The lower bound for the general inequality constraints $G_{\text{ineq}}(x, u)$ in (11i). If not provided, its entries are by default set to <code>-inf</code> .

ub_g_ineq	double	no	The upper bound for the general inequality constraints $G_{\text{ineq}}(x, u)$ in (11i). If not provided, its entries are by default set to 0.
lb_g_terminal	double	no	The lower bound for the terminal constraints $G_T(x)$ in (11j). If not provided, its entries are by default set to 0.
ub_g_terminal	double	no	The upper bound for the terminal constraints $G_T(x)$ in (11j). If not provided, its entries are by default set to 0.
n_simplex	int	·	to be explained.
<b>Auto generated</b>			
<b>Dimensions</b>			
n_x	int	·	Dimension of state vector $x$ .
n_u	int	·	Dimension of control $u$ .
n_z	int	·	Dimension of all algebraic variables $z$ .
n_theta	int	·	Dimension of $\theta$ .
n_p	int	·	Dimension of parameter vector $p$ .
m_ind_vec	int	·	to be explained
n_cross_comp	int	·	Number of complementarity constraint per stage or per finite element.
<b>CasADi Functions and misc.</b>			
h	int	·	Nominal step size $h = T/N_{\text{stages}}$
c_fun	CasADi Function	·	Function for the evaluation of $c(x)$ .
f_x_fun	CasADi Function	·	Function for the evaluation of the r.h.s. $F(x)\theta$ .
f_q_fun	CasADi Function	·	Function for the evaluation stage cost.
f_qT_fun	CasADi Function	·	Function for the evaluation of the terminal cost .
f_z_fun	CasADi Function	·	Function for the evaluation of $G_{\text{LP}}(x, z)$ .
g_ind_all_fun	CasADi Function	·	Function should be g ind function, for $g(\cdot)$ .
g_ineq_fun	CasADi Function	·	Function for the evaluation of general path constraints.
g_terminal_fun	CasADi Function	·	Function for the evaluation of the terminal constraints.

NLP solver			
w	CasADi sym	.	Vector that contains all decision variables in the finite-dimensional NLP (18).
g	CasADi exp	.	All nonlinear constraints functions in the finite-dimensional NLP (18).
J	CasADi	.	Symoblic expression for the objective of the finite dimenisional NLP (18).
sigma	CasADi MX	.	The homotopy parameters $\sigma_i$ .
J_fun	CasADi sym	.	Function for the NLP objective.
f_J_cc	CasADi sym	.	Function for evaluating the constraint residual.
nlp	struct	.	MATLAB struct containing all NLP elements to create a CasADi NLP solver Function.
prob	struct	.	difference to nlp???
solver	CasADi Function	.	Function for solving an NLP via IPOPT.
comp_res	CasADi Function	??	
Index sets			
ind_x	int	.	Indices of all $\mathbf{x}$ in $\mathbf{w}$ .
ind_u	int	.	Indices of all $\mathbf{q}$ in $\mathbf{w}$ .
ind_z	int	.	Indices of all $(\Theta, \Lambda, \mathbf{M})$ in $\mathbf{w}$ .
ind_g_clock_state	int	.	to be explained.
ind_boundary	int	.	to be explained.
ind_sot	int	.	Indices of all "speed of time" variables, which are used in time-freezing and time optimal control problems.
ind_h	int	.	Indices of all $\mathcal{H}$ in $\mathbf{w}$ , i.e., all variables for the step-size $h_{k,i}$ .
ind_total	int	.	All indices of $\mathbf{w}$ .

TODO: delete f\_z\_cc .n\_algebraic\_constraints n\_bilinear\_cc

### 3.2 All settings at a glance

Name	Default value	Possible values	Description
General			

<code>solver_name</code>	<code>'nosnoc_solver'</code>	string	Name of the CasADi function calling the NLP solver.
<code>use_fesd</code>	1	$\{0, 1\}$	Determine if the FESD scheme from ?? is used. If turned off, a standard IRK schemes for a DCS from Section ?? with a fixed step size is used.
<b>IRK and FESD Settings</b>			
<code>n_s</code>	2	$\{1, \dots, 9\}$	Number of stages in IRK scheme.
<code>irk_scheme</code>	<code>'radau'</code>	<code>'radau','legendre'</code>	Choose which IRK Scheme Family (and corresponding Butcher Tabelu) is used in the FESD scheme
<code>irk_representation</code>	<code>'integral'</code>	<code>'integral','differential'</code>	Choose are the IRK equations written in differential or integral form. Integral form is usually used in direct collocation.
<code>cross_comp_mode</code>	3	$\{1, \dots, 9\}$	Determines which form of the cross complementarity, cf. Section ??.
<code>gamma_h</code>	1	$[0, 1]$	Determines the lower and upper bound for $h_n$ as $(1 - \gamma_h) \frac{T}{N}$ and $(1 + \gamma_h) \frac{T}{N}$ , respectively.
<code>lp_initalization</code>	0	$\{0, 1\}$	Solve a parametric LP (9) to get a feasible guess for the algebraic variables $\theta_{n,m}, \lambda_{n,m}$ and $\mu_{n,m}$ .
<code>initial_theta</code>	0	$\mathbb{R}_{\geq 0}$	Initial guess for the convex multipliers $\theta$ .
<code>initial_lambda</code>	0	$\mathbb{R}_{\geq 0}$	Initial guess for the dual variable in the DCS $\lambda$ .

initial_mu	0	$\mathbb{R}_{\geq 0}$	Initial guess for the dual variable in the DCS $\mu$ .
<b>OCP Settings</b>			
g_ineq_constraint	0	$\{0,1\}$	Indicates if there is a general nonlinear path constraint (11i). If provided by the user the flag is set to 1. It can be turned off even if the constraint is given.
g_ineq_at_fe	0	$\{0,1\}$	Evaluate the general nonlinear path constraint (11i) at every finite element boundary point. Note that, if the path constraint depends only on $u$ , it usually does not make much sense to evaluate in on a finer grid.
g_ineq_at_stg	0	$\{0,1\}$	Evaluate the general nonlinear path constraint (11i) at every finite element boundary point. Note that, if the path constraint depends only on $u$ , it usually does not make much sense to evaluate in on a finer grid.
x_box_at_fe	1	$\{0,1\}$	Evaluate box constraint for $x$ (11g) at every finite element boundary point.



x_box_at_stg	1	$\{0,1\}$	Evaluate box constraint for $x$ (11g) at every stage point. IT is set to 0 per default if <code>irk_representation = 'differential'</code> , as it becomes a linear instead of box constraint.
terminal_constraint	0	$\{0,1\}$	Indicates if there is a terminal constraint (11j). If provided by the user the flag is set to 1. It can be turned off even if the constraint is given.
time_optimal_problem	1	$\{0,1\}$	Indicate is the OCP a time optimal control problem.
<b>MPCC and Homotopy Settings</b>			
mpcc_mode	5	$\{1, \dots, 10\}$	Choose which MPCC approach is used, cf. Section ??
comp_tol	$10^{-16}$	$\mathbb{R}_{>0}$	Stopping criterion for the homotopy loops in terms of the complementarity residual.
objective_scaling	1	$\{0,1\}$	objective scaling.
sigma_0	1	$\mathbb{R}_{>0}$ and $> \sigma_N$	Initial value for homotopy parameter.
sigma_N	$10^{-14}$	$\mathbb{R}_{>0}$ and $< \sigma_0$	Final value of the homotopy parameter.
kappa	0.1	0, 1	Constant in the homotopy parameter update rule $\sigma_{i+1} = \kappa \sigma_i$ .
N_homotopy	$\left\lceil \left  \frac{\log(\frac{\sigma_N}{\sigma_0})}{\log \kappa} \right  \right\rceil$	N	Maximum number of homotopy parameters.
s_elastic_0	1	$\mathbb{R}_{>0}$	elastic mode.
s_elastic_max	100	$\mathbb{R}_{>0}$	elastic mode.
s_elastic_min	0	$\mathbb{R}_{>0}$	elastic mode.
store_homotopy_iterates	1	$\{0,1\}$	Store the solution of every NLP in the homotopy loop.
<b>Settings for Barrier Based MPCC Heuristic</b>			

Step-Equilibration			
Time Settings			
time.optimal_problem	0	0,1	is the given OCP a time optimal OCP? If yes, removes the constraints that the sum of all $h_n$ equals T and adds the parameter T to the objective and to the vector of optimization variables w.
time.freezing	0	{0,1}	Is time freezing used, this is useful to isolate the clock state and other time transformation variables.
time.rescaling	0	{0,1}	This enables to extend the bounds of h, so that a desired terminal time can be achieved. If only <b>time_freezing</b> is true, speed of time and step size all lumped together, e.g., $\hat{h}_{n,m} = s_n h_{n,m}$ , hence the bounds need to be extended.”
use_speed_of_time_variables	1	{0,1}	gggggggg
local_speed_of_time_variable1		{0,1}	Add a speed of time variable s(.) for every control interval.
stagewise_clock_constraint	1	{0,1}	Determine is there after every stage a ”terminal” constraint for the time that has passed so far, if on. Or if off, just add a final terminal constraint for the clock state.
s_sot0	1	> 0	Initial guess for the speed of time variables.
s_sot_max	25	≥ 0	Upper bound for the speed of time variables.

<code>s_sot_min</code>	<code>s_sot_max<sup>-1</sup></code>	$\geq 0$	Lower bound for the speed of time variables.
<code>impose_terminal_physical_time</code>		$\{0, 1\}$	This option is only available if time-freezing is used. It imposes that the final physical time is equal to the provided $T$ (except in time optimal control problems where it is a degree of freedom, then it determines if the physical or numerical time is minimized). If it is off, it turns off <code>time_rescaling</code> during <code>time_freezing</code> .
<b>IPOPT Settings</b>			
<b>Integrator Settings</b>			

List of supported IRK schemes			
IRK scheme	Number of stages	Order of diff. state	Mode supported
'Radau-I'	$\{2, 3\}$	$2n_s - 1$	differential
'Radau-IA'	$\{1, 2, 3\}$	$2n_s - 1$	differential
'Radau-IIA'	$\{1, \dots, 9\}$	$2n_s - 1$	differential, integral
'Gauss-Legendre'	$\{1, \dots, 9\}$	$2n_s$	differential, integral
'Lobatto-III'	$\{2, 3, 4, 5\}$	$2n_s - 2$	differential
'Lobatto-IIIA'	$\{2, 3, 4, 5\}$	$2n_s - 2$	differential
'Lobatto-IIIB'	$\{2, 3, 4, 5\}$	$2n_s - 2$	differential
'Lobatto-IIIC'	$\{2, 3, 4, 5\}$	$2n_s - 2$	differential
'Explicit-RK'	$\{1, 2, 3, 4\}$	$n_s$ (if $n_s \leq 4$ )	differential

### 3.3 The solver initialization

### 3.4 The results

### 3.5 List of examples

This table provides a list of examples currently available in the NOSNOC repository. It gives a classification regarding number of control and variables, number of regions  $R_i$  (i.e., number of modes of the PSS), type of problem: OCP or SIM (for simulation), time-freezing type: elastic impacts (TF-E), inelastic impacts (TF-I), hysteresis (TF-H).

Example Name	$n_f$	$n_x, n_u$	Class
time_freezing_thworing_a_ball	2	5,2	OCP, TF-E
time_optimal_control_car	4	5,1	OCP, TF-H
thermostat	4	3,0	SIM, TF-H

## 4 Homotopy and MPCC settings

The discrete-time OCP from the last section obtained via direct transcription using FESD is an MPCC. It can be written more compactly as

$$\min_w f(w) \quad (19a)$$

$$\text{s.t. } 0 \leq h(w), \quad (19b)$$

$$0 \leq w_1 \perp w_2 \geq 0, \quad (19c)$$

where  $w = (w_0, w_1, w_2) \in \mathbb{R}^{n_w}$  is a given decomposition of the problem variables. MPCC are difficult nonsmooth NLP which violate e.g., the MFCQ at all feasible points [13]. Fortunately, they can often be solved efficiently via reformulations and homotopy approaches [13, 14, 15]. We briefly discuss the standard ways of solving MPCC that are implemented in NOSNOC. They differ in how Eq. (19c) is handled. In all cases  $w_1, w_2 \geq 0$  is kept unaltered and the bilinear constraint  $w_1^\top w_2 = 0$  is treated differently.

In a homotopy procedure we solve a sequence of more regular, relaxed NLP related to (19) and parameterized by a homotopy parameter  $\sigma_i \in \mathbb{R}_{\geq 0}$ . Every new NLP is initialized with the solution of the previous one. In all approaches the homotopy parameter is updated via the rule:  $\sigma_{i+1} = \kappa \sigma_i$ ,  $\kappa \in (0, 1)$ ,  $\sigma_0 > 0$ , where  $i$  is the index of the NLP in the homotopy. In the limit as  $\sigma_i \rightarrow 0$  (or often even for a finite  $i$  and  $\sigma_i$ ) the solution of the relaxed NLP matches a solution of (19). NOSNOC supports the following approaches:

### 4.0.1 Direct solve (mpcc\_mode =1)

The bilinear term is treated directly as  $w_1^\top w_2 \leq 0$  and a single NLP is solved. This results in a degenerate NLP where the MFCQ is violated at all feasible points [13]. This approach usually yields poor performance.

### 4.0.2 Smoothing (mpcc\_mode =2) and Relaxation (mpcc\_mode =3)

In smoothing the bilinear term is replaced by the simpler constraint  $w_1^\top w_2 = \sigma_i$  and in *relaxation* by  $w_1^\top w_2 \leq \sigma_i$ . A sequence of NLP for a decreasing  $\sigma_i$  is solved and under certain assumptions for  $\sigma_i \rightarrow 0$  a solution of the initial MPCC (19) is obtained [14].

#### 4.0.3 $\ell_1$ -Penalty (`mpcc_mode = 4`)

In this approach the bilinear constraint is discarded and the term  $\frac{1}{\sigma_i} w_1^\top w_2$  is added to the objective, which is a penalized  $\ell_1$  norm of the complementarity residual. When the penalty  $\frac{1}{\sigma_i}$  exceeds a certain (often finite) threshold we have  $w_1^\top w_2 = 0$  and the solution of such an NLP is a solution to (19) [15].

#### 4.0.4 Elastic Mode (`mpcc_mode = 5`)

In *elastic mode* (sometimes called  $\ell_\infty$ -approach) [13] a bounded scalar slack variable  $\gamma \in [0, \bar{\gamma}]$  is introduced. The relaxed bilinear constraint reads as  $w_1^\top w_2 \leq \gamma$  and we add to the objective  $\frac{1}{\sigma_i} \gamma$ . Variants with  $w_1^\top w_2 = \gamma$  (`mpcc_mode = 6`) and  $-\gamma \leq w_1^\top w_2 \leq \gamma$  (`mpcc_mode = 7`) are supported as well. Once the penalty  $\frac{1}{\sigma_i}$  exceeds a certain (often finite) threshold we have  $\gamma = 0$  and we recover a solution of (19) [13].

#### 4.0.5 Barrier Controlled Penalty Homotopy (`mpcc_mode = 8`)

As controlling the homotopy parameters  $\sigma_i$  might sometimes be difficult we propose a heuristic approach, where the homotopy parameter is indirectly controlled by the barrier parameter in IPOPT [2]. The formulation works as follows. We introduce two scalar slack variables  $\tau$  and  $\delta$ . We add the constraint

$$0 \leq \tau \leq \bar{\tau}, \quad (20)$$

$$0 \leq \delta \leq \bar{\delta}, \quad (21)$$

$$\tau \leq e^{-\delta}. \quad (22)$$

and to the objective the term  $-\rho_\delta \delta^2$ . This objective term will favor larger  $\delta$  which imply very small values for  $\tau$ . Intuitively, as the interior point iterations proceed,  $\tau$  will be implicitly controlled by the barrier parameter as the constraints get more "active", cf. Fig. ?? . The positive scalars  $\bar{\tau}, \bar{\delta}, \rho_\delta$  are tuning parameters. We can now use  $\tau$  instead of  $\sigma_i$  in all of the reformulations above and solve a single NLP.

#### 4.0.6 Objective Scaling

Furthermore, in the  $\ell_1$  and  $\ell_\infty$  approaches:

- *direct*:  $f(w) + \frac{1}{\sigma_i} \psi(w)$
- *indirect*:  $\sigma_i f(w) + \psi(w)$

objective scaling are available, which are mathematically equivalent but often result in different performance.

## 4.1 Cross complementarities

In the implementation we include the "standard" complementarity conditions as well. To have a more compact notation, let us define the following variables for the sum of all stage values of  $\Lambda_n$  and  $\Theta_n$  over a single finite element  $i$ :

$$\begin{aligned}\sigma_{k,i}^\theta &= \sum_{j=1}^{n_s} \theta_{k,i,j}, & k = 0, \dots, N_{\text{stg}} - 1, i = 0, \dots, N_{\text{FE},k}, \\ \sigma_{k,i}^\lambda &= \sum_{j=0}^{n_s+1} \lambda_{k,i,j}, & k = 0, \dots, N_{\text{stg}} - 1, i = 0, \dots, N_{\text{FE},k} - 1, \\ \sigma_{k,N_{\text{FE},k}}^\lambda &= \sum_{j=0}^{n_s} \lambda_{k,i,j}, & k = 0, \dots, N_{\text{stg}} - 1, i = N_{\text{FE},k}.\end{aligned}$$

In this notation the index  $k$  is for the control interval,  $i$  for the finite element in this control interval and  $j$  for the stage in this finite element, with keeping in mind that

$$\lambda_{k,i} = \lambda_{k,i-1}, \quad k = 0, \dots, n_s - 1, i = 0, \dots, N_{\text{FE},k} - 1,$$

Note that in the sum for  $\lambda$  the values of  $\lambda_{k,i,0}$  and  $\lambda_{k,i,n_s+1}$  are not included.

### 4.1.1 Complement all stage values with each other (stage-stage level)

**cross\_complementarity\_mode =1** For all  $k = 0, \dots, n_s - 1, i = 0, \dots, N_{\text{FE},k} - 1$ , i.e., we impose on every finite element and every stage the following constraints:

$$0 = \text{diag}(\theta_{0,i,j}) \lambda_{0,i,j'}, \quad j = 1, \dots, n_s, j' = 1, \dots, n_s + 1, \quad (23a)$$

$$0 = \text{diag}(\theta_{k,i,j}) \lambda_{k,i,j'}, \quad j = 1, \dots, n_s, j' = 0, \dots, n_s + 1, \quad (23b)$$

$$0 = \text{diag}(\theta_{k,N_{\text{FE},k}-1,j}) \lambda_{k,N_{\text{FE},k}-1,j'}, \quad j = 1, \dots, n_s, j' = 0, \dots, n_s. \quad (23c)$$

For every stage we obtain up to  $n_s + 2$  vector valued constraints.

**cross\_complementarity\_mode =2** Essentially the same as the previous case, but now instead of a vector-valued constraint, at every stage we have a scalar-valued one. For all  $k = 0, \dots, n_s - 1, i = 0, \dots, N_{\text{FE},k} - 1$ , i.e., we impose on every finite element and every stage the following constraints:

$$0 = \theta_{0,i,j}^\top \lambda_{0,i,j'}, \quad j = 1, \dots, n_s, j' = 1, \dots, n_s + 1, \quad (24a)$$

$$0 = \theta_{k,i,j}^\top \lambda_{k,i,j'}, \quad j = 1, \dots, n_s, j' = 0, \dots, n_s + 1, \quad (24b)$$

$$0 = \theta_{k,N_{\text{FE},k}-1,j}^\top \lambda_{k,N_{\text{FE},k}-1,j'}, \quad j = 1, \dots, n_s, j' = 0, \dots, n_s. \quad (24c)$$

For every stage we obtain up to  $n_s + 2$  scalar valued constraints.

#### 4.1.2 Sum all $\theta$ and complement with every stage of $\lambda$ and vice versa (FE-stage level)

The next four cases provide per every stage one scalar or vector value constraint.

**cross\_complementarity\_mode =3** For every control interval  $k$ , every finite-element  $i$  and every stage  $j$ , a vector-valued constraint via sum of all  $\lambda_{k,i,j}$ .

$$0 = \text{diag}(\theta_{k,i,j})\sigma_{k,i}^\lambda, \quad j = 1, \dots, n_s, \quad (25)$$

$$(26)$$

For every of the  $n_s$  stages where  $\theta$  is defined, a vector-valued constraint.

**cross\_complementarity\_mode =4** Similar to case 3, but now with sum of all  $\theta_{k,i,j}$ . At every control interval  $k$ , every finite-element  $i$  and every stage  $j$  (including boundary values we have)

$$0 = \text{diag}(\sigma_{k,0}^\theta)\lambda_{k,0,j} \quad j = 1, \dots, n_s + 1, \quad (27a)$$

$$0 = \text{diag}(\sigma_{k,i}^\theta)\lambda_{k,i,j} \quad i = 1, \dots, N_{\text{FE},k} - 2, \quad 0 = 1, \dots, n_s + 1, \quad (27b)$$

$$0 = \text{diag}(\sigma_{k,N_{\text{FE},k}-1}^\theta)\lambda_{k,N_{\text{FE},k}-1,j} \quad j = 0, \dots, n_s, \quad (27c)$$

$$(27d)$$

For every of the up to  $n_s + 2$  stages where  $\lambda$  is defined, a vector-valued constraint.

**cross\_complementarity\_mode =5** Same as case 3, but now with the inner product hence we obtain scalar valued constraint for every stage. For every control interval  $k$ , every finite-element  $i$  and every stage  $j$ , a vector-valued constraint via sum of all  $\lambda_{k,i,j}$ .

$$0 = \theta_{k,i,j}^\top \sigma_{k,i}^\lambda, \quad j = 1, \dots, n_s, \quad (28)$$

$$(29)$$

For every of the  $n_s$  stages where  $\theta$  is defined, a scalar-valued constraint.

**cross\_complementarity\_mode =6** Same as case 4, for every stage point we obtain a scalar constraint.

$$0 = \sigma_{k,0}^\theta{}^\top \lambda_{k,0,j} \quad j = 1, \dots, n_s + 1, \quad (30a)$$

$$0 = \sigma_{k,i}^\theta{}^\top \lambda_{k,i,j} \quad i = 1, \dots, N_{\text{FE},k} - 2, \quad 0 = 1, \dots, n_s + 1, \quad (30b)$$

$$0 = \sigma_{k,N_{\text{FE},k}-1}^\theta{}^\top \lambda_{k,N_{\text{FE},k}-1,j} \quad j = 0, \dots, n_s, \quad (30c)$$

For every of the up to  $n_s + 2$  stages where  $\lambda$  is defined, a scalar-valued constraint.

#### 4.1.3 Complement sum of all $\theta$ and sum of all $\lambda$ at a FE (FE-FE level)

These variants consider for every finite element a single complementarity constraint, vector or scalar valued.

`cross_complementarity_mode =7` For every control interval  $k$ , every finite-element  $i$  a vector-valued constraint via sum of all  $\lambda_{k,i,j}$  and sum of  $\theta_{k,i,j}$ .

$$0 = \text{diag}(\sigma_{k,i}^\theta) \sigma_{k,i}^\lambda, \quad k = 0, \dots, N_{\text{stg}} - 1, i = 0, \dots, N_{\text{FE},k}. \quad (31)$$

For every finite element a vector-valued constraint.

`cross_complementarity_mode =8` The same as case 7 but via an inner product. For every control interval  $k$ , every finite-element  $i$  a vector-valued constraint via sum of all  $\lambda_{k,i,j}$  and sum of  $\theta_{k,i,j}$ .

$$0 = \sigma_{k,i}^\theta{}^\top \sigma_{k,i}^\lambda, \quad k = 0, \dots, N_{\text{stg}} - 1, i = 0, \dots, N_{\text{FE},k}. \quad (32)$$

For every finite element a scalar-valued constraint.

#### 4.1.4 Complement sum of all $\theta$ and sum of all $\lambda$ at a control interval (FE-FE level summarized)

This is formulations summarize all complementarity conditions for a single control interval.

`cross_complementarity_mode =9` For every control interval  $k$ , every finite-element  $i$  a vector-valued constraint via sum of all  $\lambda_{k,i,j}$  and sum of  $\theta_{k,i,j}$ .

$$0 = \sum_{i=0}^{N_{\text{FE},k}-1} \text{diag}(\sigma_{k,i}^\theta) \sigma_{k,i}^\lambda, \quad k = 0, \dots, N_{\text{stg}} - 1. \quad (33)$$

For every control interval we have one vector-valued constraints.

`cross_complementarity_mode =10` Same as previous, but now scalar-valued:

$$0 = \sum_{i=0}^{N_{\text{FE},k}-1} \sigma_{k,i}^\theta{}^\top \sigma_{k,i}^\lambda, \quad k = 0, \dots, N_{\text{stg}} - 1. \quad (34)$$

#### 4.1.5 Complement sum of all $\theta$ and sum of all $\lambda$ at a control interval (FE-FE level summarized)

In the next two cases are conditions are summarized for a single control interval. We have a vector-valued and a scalar-valued case.



`cross_complementarity_mode =11` All conditions in one vector valued constraint.

$$0 = \sum_{k=0}^{N_{\text{stg}}-1} \sum_{i=0}^{N_{\text{FE},k}-1} \text{diag}(\sigma_{k,i}^{\theta}) \sigma_{k,i}^{\lambda}. \quad (35)$$

This is the vector-valued case.

`cross_complementarity_mode =12`

$$0 = \sum_{k=0}^{N_{\text{stg}}-1} \sum_{i=0}^{N_{\text{FE},k}-1} \sigma_{k,i}^{\theta \top} \sigma_{k,i}^{\lambda}. \quad (36)$$

This is the scalar-valued case. This expression additionally gives the total complementarity residual (including cross-complementarity).

Remark: The scalar valued constraint seem to work better with equality type relaxations and the vector valued ones with inequality type ones.

Overview for cross complementarites			
Mode	Output	Resolution	Total number of constraints
1	vector	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_f n_s \cdot (n_s + 1)$
2	scalar	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_s \cdot (n_s + 1)$
3	vector	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_f n_s$
4	vector	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_f (n_s + 1)$
5	scalar	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_s$
6	scalar	at every stage point	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot (n_s + 1)$
7	vector	at every finite element	$N_{\text{stg}} \cdot N_{\text{FE}} \cdot n_f$
8	scalar	at every finite element	$N_{\text{stg}} \cdot N_{\text{FE}}$
9	vector	at every control interval	$N_{\text{stg}} \cdot n_f$
10	scalar	at every control interval	$N_{\text{stg}}$
11	vector	once	$n_f$
12	scalar	once	1

## 4.2 Step equilibration

**Heuristic step-equilibration** The constraint (16) can due to its nonlinearity slow down the convergence and impair the progress of the homotopy loop. Therefore, we propose several heuristic approach to improve the convergence properties. Instead of having the bilinear terms of (16) as constraints we add for every  $k$  the objective the term:

$$\psi_{\text{eq}}(\mathbf{h}, \mathbf{Z}) = \rho_{\text{eq}} \sum_{k=0}^{N-1} \sum_{n=0}^{N_{\text{FE}}-1} \tanh(\eta_{k,n}) (h_{k,n+1} - h_{k,n})^2,$$

where  $\rho_{\text{eq}} > 0$  and the  $\tanh(\cdot)$  bring all  $\eta_n$  to the same scale. An alternative simple heuristic is to add the quadratic cost term

$$\tilde{\psi}_{\text{eq}}(\mathbf{h}, \mathbf{Z}) = \rho_{\text{eq}} \sum_{k=0}^{N-1} \sum_{n=0}^{N_{\text{FE}}-1} (h_{k,n+1} - h_{k,n})^2.$$

However, in this case a too large  $\rho_{\text{eq}}$  biases towards selecting control inputs that lead to switches on an equidistant grid.

### 4.3 FESD Integrator

## 5 Structure of the Software

The aim of this section is to give more detail on the structure of `NOSNOC` and to provide more insight on all functions shipped with this packages. (work in progress...)

## References

- [1] 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*, vol. 11, no. 1, pp. 1–36, 2019.
- [2] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [3] A. Nurkanović, M. Sperl, S. Albrecht, and M. Diehl, “Finite Elements with Switch Detection for Direct Optimal Control of Nonsmooth Systems,” *to be submitted*, 2022.
- [4] A. Nurkanović, T. Sartor, S. Albrecht, and M. Diehl, “A Time-Freezing Approach for Numerical Optimal Control of Nonsmooth Differential Equations with State Jumps,” *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 439–444, 2021.
- [5] A. Nurkanović, S. Albrecht, B. Brogliato, and M. Diehl, “The Time-Freezing Reformulation for Numerical Optimal Control of Complementarity Lagrangian Systems with State Jumps,” *arXiv preprint*, 2021.
- [6] A. Nurkanović and M. Diehl, “Continuous optimization for control of hybrid systems with hysteresis via time-freezing,” *Submitted to The IEEE Control Systems Letters (L-CSS)*, 2022.
- [7] —, “NOSNOC: A software package for numerical optimal control of nonsmooth systems,” *Submitted to The IEEE Control Systems Letters (L-CSS)*, 2022.

- [8] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*, N. York, Ed. Springer-Verlag, 2003, vol. 1-2.
- [9] D. E. Stewart, “A high accuracy method for solving ODEs with discontinuous right-hand side,” *Numerische Mathematik*, vol. 58, no. 1, pp. 299–328, 1990.
- [10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, 2nd ed. Berlin Heidelberg: Springer, 1991.
- [11] B. T. Baumrucker and L. T. Biegler, “MPEC strategies for optimization of a class of hybrid dynamic systems,” *Journal of Process Control*, vol. 19, no. 8, pp. 1248–1256, 2009.
- [12] A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides*. Springer Science & Business Media, Series: Mathematics and its Applications (MASS), 2013, vol. 18.
- [13] M. Anitescu, P. Tseng, and S. J. Wright, “Elastic-mode algorithms for mathematical programs with equilibrium constraints: global convergence and stationarity properties,” *Mathematical Programming*, vol. 110, no. 2, pp. 337–371, 2007.
- [14] S. Scholtes, “Convergence properties of a regularization scheme for mathematical programs with complementarity constraints,” *SIAM Journal on Optimization*, vol. 11, no. 4, pp. 918–936, 2001.
- [15] D. Ralph and S. J. Wright, “Some properties of regularization and penalization schemes for mpecs,” *Optimization Methods and Software*, vol. 19, no. 5, pp. 527–556, 2004.