NOS-NOC User's Manual
Preliminary version

Armin Nurkanović and Moritz Diehl [1]

March, 2022

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

Contents

1 Introduction

NOS-NOC 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 high accuracy simulation and numerical optimal control algorthims for a general class of piecewise smooth systems (PSS) of the form:

$$\dot{x} = f_i(x,u), \text{ if } x \in R_i \subset \mathbb{R}^{n_x},\ i \in \mathcal{I} \coloneqq \{1,\dots,n_f\}, \tag{1}$$

where R_i are disjoint, nonempty, connected and open sets. The functions $f_i(\cdot)$ are assumed to be smooth on an open neighborhood of $\overline{R}_i$ and n_f is a positive integer. The event of $x(\cdot)$ reaching some boundary ∂R_i is called a *switch*. The right hand side (r.h.s.) of (1) is in general discontinuous in x and u is an externally chosen control function.

Several important classes of systems with state jumps can be reformulated into the form of (1) 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 NOS-NOC [7], the FESD paper [6] and the time-freezing papers [4, 5, 6].

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

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

$$\dot{x} \in F_{\mathrm{F}}(x,u) = \Big\{ F(x)\theta \mid \sum_{i\in\mathcal{I}} \theta_i = 1,\ \theta_i \geq 0,\ \theta_i = 0 \text{ if } x \notin \overline{R_i}, \forall i \in \mathcal{I} \Big\}, \tag{2}$$

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

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:

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

Thus the sets R_i are compactly denoted by:

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

We use Stewart's reformulation of DCS into PSS [8]. In this case, it is assumed that the sets R_i are represent via the *discriminant functions* $g_i(\cdot)$:

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

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

$$g(x) = -Sc(x). \tag{6}$$

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) [8], and (2) is equivalent to

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

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

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

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

$$0 = g(x) - \lambda - \mu(t)e, \tag{8b}$$

$$1 = e^\top \theta, \tag{8c}$$

$$0 \leq \theta \perp \lambda \geq 0, \tag{8d}$$

where $\lambda \in \mathbb{R}^{n_f}_{\geq 0}$ 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

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

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

$$\dot{x} = F(x, u)\theta, \tag{10a}$$
$$0 = G_{\mathrm{LP}}(x, \theta, \lambda, \mu). \tag{10b}$$

2.2 The continuous-time optimal control problem

We regard the continuous-time optimal control problem

$$\begin{aligned}
\min_{x(\cdot), u(\cdot), z(\cdot)} \quad & \int_0^T f_{\mathrm{q}}(x(t), u(t))\mathrm{d}t + f_{\mathrm{qT}}(x(T)) && && \text{(11a)}\\
\text{s.t.} \quad & x_0 = s_0, && && \text{(11b)}\\
& \dot{x}(t) = F(x(t), u(t))\theta(t), && t \in [0, T], && \text{(11c)}\\
& 0 = g(x(t)) - \lambda(t) - \mu(t)e, && t \in [0, T], && \text{(11d)}\\
& 0 \leq \theta(t) \perp \lambda(t) \geq 0, && t \in [0, T], && \text{(11e)}\\
& 1 = e^\top \theta(t), && t \in [0, T], && \text{(11f)}\\
& x_{\mathrm{lb}} \leq x(t) \leq x_{\mathrm{ub}}, && t \in [0, T], && \text{(11g)}\\
& u_{\mathrm{lb}} \leq u(t) \leq u_{\mathrm{ub}}, && t \in [0, T], && \text{(11h)}\\
& G_{\mathrm{ineq,lb}} \leq G_{\mathrm{ineq}}(x(t), u(t)) \leq G_{\mathrm{ineq,ub}}, && t \in [0, T], && \text{(11i)}\\
& G_{\mathrm{T,lb}} \leq G_{\mathrm{T}}(x(t)) \leq G_{\mathrm{T,ub}},, && && \text{(11j)}
\end{aligned}$$

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

2.3 Standard IRK Discretization of a Single Control Interval

We briefly state how a single control interval is discretized in `NOS-NOC`. 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_{fe} finite elements (i.e., integration intervals) $[t_n, t_{n+1}]$ via the grid points $0 = t_0 < t_1 < \ldots < t_{N_{\mathrm{fe}}} = T$.

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 $A \in$

$\mathbb{R}^{n_\mathrm{s}\times n_\mathrm{s}},\ b \in \mathbb{R}^{n_\mathrm{s}},\ c \in \mathbb{R}^{n_\mathrm{s}}$ [9]. The fixed step-size reads as $h_n = t_{n+1} - t_n,\ n = 0,\dots,N_\mathrm{fe}-1$. The approximation of the differential state at the grid points t_n is denoted by $x_n \approx x(t_n)$. The derivative of state at the stage points $t_n + c_i h_n,\ i = 1,\dots,n_\mathrm{s}$, for a single finite element are summarized in the vector $V_n \coloneqq (v_{n,1},\dots,v_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s}\cdot n_x}$. The stage values for the algebraic variables are collected in the vectors: $\Theta_n \coloneqq (\theta_{n,1},\dots,\theta_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s}\cdot n_f}$, $\Lambda_n \coloneqq (\lambda_{n,1},\dots,\lambda_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s}\cdot n_f}$ and $M_n \coloneqq (\mu_{n,1},\dots,\mu_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s}}$. 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^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 $\lambda_{n,n_\mathrm{s}+1},\ \mu_{n,n_\mathrm{s}+1}$, respectively. Due to continuity, we impose that $\lambda_{n,n_\mathrm{s}+1} = \lambda_{n+1,0}$ and $\mu_{n,n_\mathrm{s}+1} = \mu_{n+1,0}$ and compute explicitly only the right boundary points of the finite elements. These values can be computed from an LP solve for x_n^next:

$$0 = G_\mathrm{LP}(x_n^\mathrm{next}, \theta_{n,n_\mathrm{s}+1}, \lambda_{n,n_\mathrm{s}+1}, \mu_{n,n_\mathrm{s}+1}). \tag{12}$$

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

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

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 Lobbatto schemes [9]), i.e., $c_{n_\mathrm{s}} = 1$, thus we have $\lambda_{n,n_\mathrm{s}} = \lambda_{n,n_\mathrm{s}+1}\ \mu_{n,n_\mathrm{s}} = \mu_{n,n_\mathrm{s}+1}$ and we do not need the additional constraint (12).

The next equations summarize all N_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 $\mathbf{x} = (x_0, x_0^\mathrm{next}, \dots, x_{N_\mathrm{fe}}) \in \mathbb{R}^{(2N_\mathrm{fe}+1)n_x}$, $\mathbf{V} = (V_0,\dots,V_{N_\mathrm{fe}-1}) \in \mathbb{R}^{N_\mathrm{fe} n_\mathrm{s} n_x}$ and $\mathbf{h} \coloneqq (h_0,\dots,h_{N_\mathrm{fe}-1}) \in \mathbb{R}^{N_\mathrm{fe}}$. For the algebraic variables, we collect the stage values and the newly introduced boundary values into the vectors $\mathbf{\Theta} = (\Theta_0, \theta_{0,n_\mathrm{s}+1}, \dots, \Theta_{N_\mathrm{fe}-1}) \in \mathbb{R}^{N_\mathrm{fe}(n_\mathrm{s}+1)n_f}$. The vectors $\mathbf{\Lambda} \in \mathbb{R}^{N_\mathrm{fe}(n_\mathrm{s}+1)n_f}$, $\mathbf{M} \in \mathbb{R}^{N_\mathrm{fe}(n_\mathrm{s}+1)}$ are defined accordingly. The vector $\mathbf{Z} = (\mathbf{x}, \mathbf{\Theta}, \mathbf{\Lambda}, \mathbf{M}, \mathbf{V})$ 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:

$$s_1 = F_{\mathrm{std}}(\mathbf{x}), \tag{14a}$$
$$0 = G_{\mathrm{std}}(s_0, \mathbf{Z}, q), \tag{14b}$$

with $F_{\mathrm{std}}(\mathbf{x}) = x_{N_{\mathrm{fe}}}$ and

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

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

2.4 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 [10]) 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

$$0 = G_{\mathrm{cross}}(\mathbf{\Theta}, \mathbf{\Lambda}) \coloneqq \begin{bmatrix} \sum_{i=1}^{n_{\mathrm{s}}} \sum_{j=1, j\neq i}^{n_{\mathrm{s}}+1} \theta_{1,i}^{\top} \lambda_{1,j} \\ \vdots \\ \sum_{i=1}^{n_{\mathrm{s}}} \sum_{\substack{j=0, \\ j\neq i}}^{n_{\mathrm{s}}} \theta_{N_{\mathrm{fe}}-1,i}^{\top} \lambda_{N_{\mathrm{fe}}-1,j} \end{bmatrix}. \tag{15}$$

Due to the nonnegativity of the variables $\mathbf{\Theta}$ and $\mathbf{\Lambda}$ 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 `NOS-NOC`.

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

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

The switch indicating quantities reads as

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

The joint effect of all components is collected in the product

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

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

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

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.*

$$s_1 = F_{\mathrm{fesd}}(\mathbf{x}), \tag{17a}$$
$$0 = G_{\mathrm{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q), \tag{17b}$$

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

$$G_{\mathrm{fesd}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q) \coloneqq \begin{bmatrix} G_{\mathrm{std}}(\mathbf{x}, \mathbf{h}, \mathbf{Z}, q) \\ G_{\mathrm{cross}}(\mathbf{\Theta}, \mathbf{\Lambda}) \\ G_{\mathrm{eq}}(\mathbf{h}, \mathbf{\Theta}, \mathbf{\Lambda}) \end{bmatrix}.$$

For a known control function q, the formulation (17) can be used as a integrator with exact switch detection for PSS (1). This feature is implemented in `NOS-NOC` via the function `integrator_fesd()`. The FESD scheme can automatically handle all kind of switching cases such as [11]:

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

2.5 Discretization of the OCP

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

$$\begin{aligned}
\min_{\mathbf{s},\mathbf{q},\mathcal{H},\mathcal{Z}} \quad & \sum_{k=1}^{N_{\mathrm{stg}}-1} \hat{f}_{\mathrm{q}}(s_k, \mathbf{x}_k, q_k) + \hat{f}_{\mathrm{qT}}(s_{N_{\mathrm{stg}}}) && && \text{(18a)}\\
\text{s.t.} \quad & s_0 = \bar{x}_0, && && \text{(18b)}\\
& s_{k+1} = F_{\mathrm{fesd}}(\mathbf{x}_k), && k = 0, \dots, N_{\mathrm{stg}}-1, && \text{(18c)}\\
& 0 = G_{\mathrm{fesd}}(\mathbf{x}_k, \mathbf{h}_k, \mathbf{Z}_k, q_k), && k = 0, \dots, N_{\mathrm{stg}}-1, && \text{(18d)}\\
& G_{\mathrm{ineq,lb}} \leq G_{\mathrm{ineq}}(s_k, q_k) \leq G_{\mathrm{ineq,ub}}, && k = 0, \dots, N_{\mathrm{stg}}-1, && \text{(18e)}\\
& x_{\mathrm{lb}} \leq s_k \leq x_{\mathrm{ub}}, && k = 0, \dots, N_{\mathrm{stg}}, && \text{(18f)}\\
& u_{\mathrm{lb}} \leq q_k \leq u_{\mathrm{ub}}, && k = 0, \dots, N_{\mathrm{stg}}-1, && \text{(18g)}\\
& G_{\mathrm{T,lb}} 0 \leq G_{\mathrm{T}}(s_{N_{\mathrm{stg}}}) \leq G_{\mathrm{T,ub}}, && && \text{(18h)}
\end{aligned}$$

where $\hat{f}_{\mathrm{qT}} : \mathbb{R}^{n_x} \times \mathbb{R}^{(N_{\mathrm{fe}}+1)n_s n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\hat{f}_{\mathrm{qT}} : \mathbb{R}^{n_x} \to \mathbb{R}$ are the discrteized stage and terminal costs. The path constraint $G_{\mathrm{ineq}}(\cdot)$ 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**
User input			
`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 `double` vector of size $[1, n_{\mathrm{s}}]$ can be passed to determined the number of finite elements $N_{\mathrm{fe}k}$ 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 `g_ind` 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.
`c`	CasADi `expr`	no, if `g_ind` not given	The constraint function $c(x)$ that defines the regions R_i in (4)

`g_ind`	CasADi `sym expr`	yes, if no `S` and `c` not given	The discriminant function $g(x)$ (6).
`f_q`	CasADi `sym expr`	no	CasADi symbolic expression for the stage cost $f_q(\cdot,\cdot)$ in the OCP (11).
`f_qT`	CasADi `sym expr`	no	CasADi symbolic expression for the terminal cost $f_{q,\mathrm{T}}(\cdot)$ in the OCP (11).
`g_ineq`	CasADi `sym expr`	no	CasADi symbolic expression for the path constraints $G_{\mathrm{ineq}}(\cdot)$ in the OCP (11).
`g_terminal`	CasADi `sym expr`	no	CasADi symbolic expression for the terminal constraints $G_{\mathrm{T}}(\cdot)$ in the OCP (11).
`lbx`	`double`	no	The upper bound for the differential state x in (11g). If not provided, its entries are by default set to `inf`.
`ubx`	`double`	no	The lower bound for the differential state x in (11g). If not provided, its entries are by default set to `-inf`.
`lbu`	`double`	no	The lower bounds for the control u in (11h). If not provided, its entries are by default set to `-inf`.
`ubu`	`double`	no	The upper bound for the control u in (11h). If not provided, its entries are by default set to `inf`.
`u0`	`double`	no	Initial guess for the control u in (11). If not provided, its entries are by default set to `0`.
`lb_g_ineq`	`double`	no	The lower bound for the general inequality constraints $G_{\mathrm{ineq}}(x,u)$ in (11i). If not provided, its entries are by default set to `-inf`.
`ub_g_ineq`	`double`	no	The upper bound for the general inequality constraints $G_{\mathrm{ineq}}(x,u)$ 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 $G_{\mathrm{T}}(x)$ 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 $G_{\mathrm{T}}(x)$ in (11j). If not provided, its entries are by default set to `0`.
`n_simplex`	`int`	·	to be explained.
Auto generated			
Dimensions			
`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 θ.
`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.
CasADi `Functions` and misc.			
`h`	`int`	·	Nominal step size `h = T/N_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 $G_{\mathrm{LP}}(x,z)$.
`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 σ_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 `w`.
`ind_u`	`int`	·	Indices of all $\mathbf{q}$ in `w`.
`ind_z`	`int`	·	Indices of all $(\mathbf{\Theta}, \mathbf{\Lambda}, \mathbf{M})$ in `w`.
`ind_g_clock_state`	`int`	·	to be explained.
`ind_boundary`	`int`	·	to be explained.
`ind_tf`	`int`	·	to be explained.
`ind_h`	`int`	·	Indices of all $\mathcal{H}$ in `w`, i.e., all variables for the step-size $h_{k,i}$.
`ind_total`	`int`	·	All indices of `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			
`solver_name`	'solver_fesd'	string	Name of the CasADi function calling the NLP solver.

`use_fesd`	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.
IRK and FESD Settings			
`d`	2	$\{1,\dots,9\}$	Number of stages in IRK scheme.
`collocation_scheme`	'radau'	'radau','legendre'	Choose which IRK Scheme Familiy (and corresponding Butcher Tabelu) is used in the FESD scheme
`cross_comp_mode`	3	$\{1,\dots,9\}$	Determines which form of the cross complementarity, cf. Section **??**.
`gamma_h`	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.
`lp_initalization`	0	$\{0,1\}$	Solve a parametric LP (9) to get a feasible guess for the algebraic variables $\theta_{n,m}$,$\lambda_{n,m}$ and $\mu_{n,m}$.
`initial_theta`	1	$\mathbb{R}_{\geq 0}$	Initial guess for the convex multiplers θ.
`initial_lambda`	1	$\mathbb{R}_{\geq 0}$	Initial guess for the dual variable in the DCS λ.
`initial_mu`	1	$\mathbb{R}_{\geq 0}$	Initial guess for the dual variable in the DCS μ.
MPCC and Homotopy Settings			
`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 therms 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`	$\left\lceil \left\lvert \frac{\log(\frac{\sigma_N}{\sigma_0})}{\log \kappa} \right\rvert \right\rceil$	$\mathbb{N}$	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.
Settings for Barrier Based MPCC Heuristic			
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 `time_freezing` 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_variable`	1	$\{0,1\}$	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.
`s_sot_min`	s_sot_max^{-1}	≥ 0	Lower bound for the speed of time varaibles.
`impose_terminal_phyisical_time`	1	$\{0,1\}$	This option is only avilable if time-freezing is used. It imposes that the final phyisical time is equl to the provided T (except in time optimal control problems where it is a degree of freedom, then it determines is the phyisical or numericla timize minimized). If it is off, it turns off `time_rescaling` during `time_freezing`.
IPOPT Settings			
Integrator Settings			

3.3 The `solver_initalization`

3.4 The `results`

3.5 List of examples

This table provides a list of example currently available in the `NOS-NOC` repository. It gives a classification regarding number of control and variables, number of regions R_i (i.e., number of mode of the PSS), type of problem: OCP or SIM (for simulation), time-freezing type: elastic impacts (TF-E), inelastic impacts (TF-I), hyseresis (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

$$\begin{aligned} \min_{w} \quad & f(w) && \text{(19a)}\\ \text{s.t.} \quad & 0 \leq h(w), && \text{(19b)}\\ & 0 \leq w_1 \perp w_2 \geq 0, && \text{(19c)} \end{aligned}$$

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 [12]. Fortunately, they can often be solved efficiently via reformulations and homotopy approaches [12, 13, 14]. We briefly discuss the standard ways of solving MPCC that are implemented in `NOS-NOC`. 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 $\sigma_i \to 0$ (or often even for a finite i and σ_i) the solution of the relaxed NLP matches a solution of (19). `NOS-NOC` supports the following approaches:

4.0.1 Direct solve (`mpcc_mode =1`)

The bilinear term is treated directly as 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 [12]. 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 σ_i is solved and under certain assumptions for $\sigma_i \to 0$ a solution of the initial MPCC (19) is obtained [13].

4.0.3 ℓ_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 ℓ_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) [14].

4.0.4 Elastic Mode (`mpcc_mode = 5`)

In *elastic mode* (sometimes called ℓ_∞-approach) [12] 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) [12].

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

As controlling the homotopy parameters σ_i might sometimes be difficult we propose a heuristic approach, were the homotopy parameter is indirectly controlled by the barrier parameter in `IPOPT` [2]. The formulation works as follows. We introduce two scalar slack variables τ and δ. We add the constraint

$$0 \leq \tau \leq \bar{\tau}, \tag{20}$$

$$0 \leq \delta \leq \bar{\delta}, \tag{21}$$

$$\tau \leq e^{-\delta}. \tag{22}$$

and to the objective the term $-\rho_\delta \delta^2$. This objective term will favor larger δ which imply very small values for τ. Intuitively, as the interior point iterations proceed, τ 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 τ instead of σ_i in all of the reformulations above and solve a single NLP.

4.0.6 Objective Scaling

Furthermore, in the ℓ_1 and ℓ_∞ approaches, *direct* ($f(w) + \frac{1}{\sigma_i}\psi(w)$) and *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

4.1.1 `cross_complementarity_mode =1`

$$0 = \mathrm{diag}(\theta_{0,m})\lambda_{0,m'}, \ m, \ m' \in \mathcal{I}_{\mathrm{stg}}, \tag{23a}$$

$$0 = \mathrm{diag}(\theta_{n,m})\lambda_{n,m'}, \ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}}, m' \in \mathcal{I}_{\mathrm{stg}} \cup \{0\}. \tag{23b}$$

4.1.2 `cross_complementarity_mode =2`

$$0 = \theta_{0,m}^{\top} \lambda_{0,m'},\ m,\ m' \in \mathcal{I}_{\mathrm{stg}}, \tag{24a}$$
$$0 = \theta_{n,m}^{\top} \lambda_{n,m'},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}}, m' \in \mathcal{I}_{\mathrm{stg}} \cup \{0\}. \tag{24b}$$

4.1.3 `cross_complementarity_mode =3`

Case 3: For every stage point one vector-valued constraint via sum of all $\lambda_{n,m}$.

$$0 = \mathrm{diag}(\theta_{0,m}) \sum_{k=1}^{n_{\mathrm{s}}} \lambda_{0,k},\ m \in \mathcal{I}_{\mathrm{stg}}, \tag{25a}$$
$$0 = \mathrm{diag}(\theta_{n,m}) \sum_{k=0}^{n_{\mathrm{s}}} \lambda_{n,k},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}}. \tag{25b}$$

4.1.4 `cross_complementarity_mode =4`

Now we do the same via sum of θ for the remaining caseses. Note that here we have one more constraint as we make an extra constraint for $\lambda_{0,n}$. For every stage point one vector-valued constraint via sum of all θ.

$$0 = \mathrm{diag}(\lambda_{0,m}) \sum_{k=1}^{n_{\mathrm{s}}} \theta_{0,k},\ m \in \mathcal{I}_{\mathrm{stg}}, \tag{26a}$$
$$0 = \mathrm{diag}(\lambda_{n,m}) \sum_{k=1}^{n_{\mathrm{s}}} \theta_{n,m},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}} \cup \{0\}. \tag{26b}$$

4.1.5 `cross_complementarity_mode =5`

Case 5: Per stage point one scalar-valued constraint via sum of λ.

$$0 = \theta_{0,m}^{\top} \sum_{k=1}^{n_{\mathrm{s}}} \lambda_{0,k},\ m \in \mathcal{I}_{\mathrm{stg}}, \tag{27a}$$
$$0 = \theta_{n,m}^{\top} \sum_{k=0}^{n_{\mathrm{s}}} \lambda_{n,k},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}}. \tag{27b}$$

4.1.6 `cross_complementarity_mode =6`

Case 6: For every collocation point one scalar-valued constraint via sum of all θ.

$$0 = \lambda_{0,m}^{\top} \sum_{k=1}^{n_{\mathrm{s}}} \theta_{0,k},\ m \in \mathcal{I}_{\mathrm{stg}}, \tag{28a}$$
$$0 = \lambda_{n,m}^{\top} \sum_{k=1}^{n_{\mathrm{s}}} \theta_{n,m},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}} \cup \{0\}. \tag{28b}$$

4.1.7 `cross_complementarity_mode =7`

One constraint per finite element. Case 7: Per stage (finite element) one vector valued constraint via sum of λ.

$$0 = \sum_{k=1}^{n_s} \mathrm{diag}(\theta_{n,k}) \sum_{m=1}^{n_s} \lambda_{0,m}, \tag{29a}$$

$$0 = \sum_{k=1}^{n_s} \mathrm{diag}(\theta_{n,k}) \sum_{m=0}^{n_s} \lambda_{n,m},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}, m \in \mathcal{I}_{\mathrm{stg}}. \tag{29b}$$

4.1.8 `cross_complementarity_mode =8`

Case 8: Per stage (finite element) one scalar constraint via sum of λ.

$$0 = \sum_{k=1}^{n_s} \theta_{0,k}^\top \sum_{m=1}^{n_s} \lambda_{0,m}, \tag{30a}$$

$$0 = \sum_{k=1}^{n_s} \theta_{n,k}^\top \sum_{m=0}^{n_s} \lambda_{n,m},\ n \in \mathcal{I}_{\mathrm{grd}} \setminus \{0\}. \tag{30b}$$

The cases one constraint per stage are the same no matter if they are performed via sum of θ or sum of λ.

4.1.9 `cross_complementarity_mode =9`

All conditions in one vector valued constraint.

$$0 = \sum_{k=1}^{n_s} \mathrm{diag}(\theta_{n,k}) \sum_{m=1}^{n_s} \lambda_{0,m} + \sum_{n=1}^{N_{\mathrm{grid}}-1} \sum_{k=1}^{n_s} \mathrm{diag}(\theta_{n,k}) \sum_{m=0}^{n_s} \lambda_{n,m}. \tag{31}$$

4.1.10 `cross_complementarity_mode =10`

All conditions in one scalar valued constraint.

$$0 = \sum_{k=1}^{n_s} \theta_{n,k}^\top \sum_{m=1}^{n_s} \lambda_{0,m} + \sum_{n=1}^{N_{\mathrm{grid}}-1} \sum_{k=1}^{n_s} \theta_{n,k}^\top \sum_{m=0}^{n_s} \lambda_{n,m}. \tag{32}$$

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

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:

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

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

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

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

4.3 FESD Integrator

5 A Tutorial Example

5.1 A MATLAB Example of an OCP with a System with State Jumps

```
import casadi.*
% Call this function to have an overview of all options.
[settings] = default_settings_fesd();
% Highlight that the problem is treated with time-freezing
    reformulation and change the number of IRK stages
settings.time_freezing = 1;
settings.n_s = 3;
% Discretization data
model.T = 5;
model.N_stages = 15;
model.N_finite_elements = 3;
% Define states, controls and inital values
q = MX.sym('q',2); v = MX.sym('v',2); t = MX.sym('t');
model.x = [q;v;t];
model.x0 = [0;0.5;0;0;0];
u = MX.sym('u',2);
model.u = u;
% Define regions of the PSS
model.S = [1; -1];
model.c = q(2);
% Define modes of PSS
f_1 = [v;u(1);u(2)-9.81;1];
f_2 = [0;v(2);0;-10*(q(2))-0.211989*v(2);0];
model.F = [f_1 f_2];
% Stage and terminal costs
model.f_q = u'*u; model.f_q_T = 10*v'*v;
% Path and terminal constraints
model.g_ineq = u'*u-7^2;
model.g_terminal = q-[4;0.25];
```

```
% Solve OCP with MPCC homotopy and plot results
[solver,solver_initalization, model,settings] = create_nlp_fesd(
      model,settings);
[results,stats] = homotopy_solver(solver,model,settings,
      solver_initalization);
plot_result_ball(model,settings,results,stats)
```

6 Structure of the Software

The aim of this section is to give more detail on the structure of NOS-NOC 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] ——, "Nos-noc: A software package for numerical optimal control of nonsmooth systems," *Submitted to The IEEE Control Systems Letters (L-CSS)*, 2022.

[8] 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.

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

[10] 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.

[11] A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides.* Springer Science & Business Media, Series: Mathematics and its Applications (MASS), 2013, vol. 18.

[12] 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.

[13] 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.

[14] 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.