Preamble. This is a reprint of the article:

M. Schulze Darup and M. Mönnigmann. Explicit feasible initialization for nonlinear MPC with guaranteed stability. In *Proc. of the 50th IEEE Conference on Decision and Control*, pp. 2674–2679, 2011.

The digital object identifier (DOI) of the original article is:

`10.1109/CDC.2011.6160715`

Explicit feasible initialization for nonlinear MPC with guaranteed stability

Moritz Schulze Darup[†] *and* M. Mönnigmann[†]

Abstract

We present a method for the computation of control invariant (c.i.) sets and a simple suboptimal explicit controller for a large class of constrained nonlinear discrete time systems. The explicit controller provides, for any point in the c.i. set, a finite sequence of input values that drive the system to the origin. These input sequences may serve as feasible initializations for the nonlinear program (NLP) associated to nonlinear model predictive control (NMPC). The proposed method is a straightforward extension of an existing method for the computation of c.i. sets, which we augment by a mechanism to record feasible control actions. In contrast to existing explicit NMPC approaches, the explicit control law is calculated without solving the underlying NLP. The method is computationally expensive, but most of the computational effort can be moved offline, and the evaluation of the resulting explicit controller is quite fast.

1 Introduction

Model predictive control (MPC) is a powerful instrument provided that closed-loop stability of the dynamical system and feasibility of the underlying nonlinear program (NLP) can be ensured. Stability is often guaranteed by computing a positively invariant (p.i.) *terminal set* $\mathcal{T}$ to which each trajectory is driven within the prediction horizon (enforced by a terminal constraint, see [3] or [8]). The *feasible set* $\mathcal{F}$, i.e. the largest state space subset for which there exists a control sequence that satisfies all constraints, plays a crucial role in this context. Feasible sets can be computed based on orthogonal projections [4]; newer methods make use of set relations [9]. However, implementations of these approaches only exist for special system classes, such as linear systems, polytopic systems, or piece-wise affine systems (see [2] and [6]). In contrast, Bravo et al. [2] presented a method for estimating the feasible set for a general class of nonlinear systems using *step sets* $\mathcal{S}$. The present paper deals with the extension of the approach suggested by Bravo

[†] M. Schulze Darup and M. Mönnigmann are with Automatic Control and Systems Theory, Department of Mechanical Engineering, Ruhr-Universität Bochum, 44801 Bochum, Germany. E-mail: `moritz.schulzedarup@rub.de`.

et al. [2]. It is our aim to compute feasible initializations for MPC optimization problems for discrete-time, time-invariant, nonlinear systems of the form

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

that are subject to state constraints $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and input constraints $u \in \mathcal{U} \subseteq \mathbb{R}^m$, where $k \in \mathbb{N}$ and where f is defined on at least $\mathcal{X} \times \mathcal{U}$ and maps into $\mathbb{R}^n$.

Existing algorithms for the computation of $\mathcal{F}$ provide the subset of the state space for which the MPC optimization is feasible or an estimation of this subset. We present a simple but useful method that provides both such an estimation $\hat{\mathcal{F}} \subseteq \mathcal{F}$ and, for any initial point $x_0 \in \hat{\mathcal{F}}$, an input sequence $u(k)$ that is guaranteed to drive the system to the terminal set. This input sequence is in general not the optimal solution, but it serves as a feasible solution to the MPC optimization problem. In contrast to existing explicit NMPC approaches (e.g. [11]), our method provides an explicit suboptimal control law without ever solving the underlying NLP.

The paper is organized as follows. In Sect. 2 we state the problem and collect some basic results on feasibility and step sets. Section 3 deals with the approximation of step sets and the computation of an explicit control law. The main results of the paper, i.e. the identification of feasible inputs (without solving the NLP) and the compact representation of feasible sets are treated in Sect. 3.3 and 3.4. Section 4 illustrates the method with an example. Conclusions are given in Sect. 5.

2 Preliminaries and problem statement

Let the tuple $(\breve{x}, \breve{u})$ denote an equilibrium of system (1), i.e. $\breve{x} = f(\breve{x}, \breve{u})$ with $\breve{x} \in \mathcal{X}$ and $\breve{u} \in \mathcal{U}$. Let $\mathcal{T}_g$ be a terminal set for $\breve{x}$, where a terminal set is defined as follows.

Definition 1: *The set $\mathcal{T}_g \subseteq \mathcal{X}$ is called a terminal set for the equilibrium $\breve{x}$ of (1) subject to the control law $u = g(x)$, if $\breve{x} \in \mathcal{T}_g$ and*

$$f(x, g(x)) \in \mathcal{T}_g, \quad g(x) \in \mathcal{U}$$

for all $x \in \mathcal{T}_g$.

Obviously, a terminal set always represents a p.i. set for the system $f(x, g(x))$, i.e. $f(x, g(x)) \in \mathcal{T}_g$ for all $x \in \mathcal{T}_g$, and a control invariant (c.i) set for $f(x, u)$, i.e. for every $x \in \mathcal{T}_g$ there exists an $u \in \mathcal{U}$ such that $f(x, u) \in \mathcal{T}_g$.

Terminal sets are used to guarantee stability in NMPC problems. Specifically, we treat the discrete time nonlinear receding horizon optimization problem of the following form (with $k \geq 0$). Let $\mathbb{N}_i^j = \{k \in \mathbb{N}_0 \,|\, i \leq k \leq j\}$ for brevity.

$$\min_{\hat{U}(k)} J(\hat{X}(k), \hat{U}(k)) \tag{2}$$

subject to

$$\begin{aligned}
\hat{x}(k|k) &= x(k), \\
\hat{x}(i+1|k) &= f(\hat{x}(i|k), \hat{u}(i|k)), \; \forall\, i \in \mathbb{N}_k^{k+h-1}, \\
\hat{x}(i+1|k) &\in \mathcal{X}, \; \forall\, i \in \mathbb{N}_k^{k+h-1}, \\
\hat{u}(i|k) &\in \mathcal{U}, \; \forall\, i \in \mathbb{N}_k^{k+h-1}, \\
\hat{x}(k+h|k) &\in \mathcal{T}_g,
\end{aligned} \tag{3}$$

where

$$\hat{X}(k) := \begin{pmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+h|k) \end{pmatrix} \quad \text{and} \quad \hat{U}(k) := \begin{pmatrix} \hat{u}(k|k)^T \\ \vdots \\ \hat{u}(k+h-1|k)^T \end{pmatrix}.$$

In this context, $\hat{x}(k+i+1|k)$ refers to the predicted state vector at time $k+i+1$ based on the application of the input sequence $\hat{u}(k|k), \dots, \hat{u}(k+i|k)$ starting from $x(k)$. In the current formulation, the cost function is defined as

$$J(\hat{X}(k), \hat{U}(k)) = \varphi(\hat{x}(k+h|k)) + \sum_{i=k}^{k+h-1} l(\hat{x}(i|k), \hat{u}(i|k))$$

with the stage cost $l(\hat{x}, \hat{u}) = \|\hat{x} - \breve{x}\|_Q^2 + \|\hat{u} - \breve{u}\|_R^2$, the terminal cost $\varphi(\hat{x}) = \|\hat{x} - \breve{x}\|_P^2$, the positive definite (p.d.) matrices P, Q, R and the weighted norm $\|x\|_P^2 = x^T P\, x$.

Since $\hat{x}(k+h|k) \in \mathcal{T}_g$, the control law

$$\hat{u}(i|k) = g(\hat{x}(i|k))$$

can be applied for $i = k+h, \dots, \infty$ to ensure stability. This results in a quasi-infinite horizon, stable nonlinear model predictive controller [3].

The nonlinear program (NLP) (2)-(3) is feasible for the current state $x(k)$ at time k, if there exists a *feasible control sequence*, i.e. a sequence $\hat{U}(k)$ such that the constraints (3) are satisfied. This leads to the following definition of the feasible set.

Definition 2: *The set of states $x_0 \in \mathcal{X}$ for which the NLP (2)-(3) is feasible for all $k \geq 0$ is called feasible set and denoted by $\mathcal{F}_h(\mathcal{T}_g)$, where h and $\mathcal{T}_g$ are as in Eqs. (2)-(3).*

Definition 2 requires the NLP (2)-(3) to be feasible for all $k \in \mathbb{N}_0^h$. It is well-known, however, that feasibility at $k = 0$ implies feasibility for all $k > 0$.

Lemma 1: *[10] The NLP (2)-(3) is feasible for all $k > 0$, if it is feasible for $k = 0$.*

Lemma 1 implies that we can drop the feasibility requirement for $k = 1, \dots, h$ in Def. 2 of the feasible set. For ease of reference this is summarized in Lemma 2, which we state without proof.

Lemma 2: *The feasible set $\mathcal{F}_h(\mathcal{T}_g)$ is equal to the set of all states $x_0 \in \mathcal{X}$ for which there exists a control sequence $\hat{U}(0)$ that satisfies the constraints (3).*

In order to calculate an approximation of $\mathcal{F}_h(\mathcal{T}_g)$ based on Prop. 2, the so called *one-step set* $\mathcal{Q}(\mathcal{T})$ [1] is needed. The one-step set is defined as the set of states $x_0 \in \mathcal{X}$ which can be steered to the target set $\mathcal{T} \subseteq \mathcal{X}$ in one time-step, i.e.

$$\mathcal{Q}(\mathcal{T}) = \{x \in \mathbb{R}^n \,|\, \exists\, u \in \mathcal{U} : f(x, u) \in \mathcal{T}\}.$$

The generalization of the one-step set leads to the definition of σ-step sets.

Definition 3: *The σ-step set $\mathcal{S}_\sigma(\mathcal{T}) \subseteq \mathcal{X}$ is defined as the set of states $x_0 \in \mathcal{X}$ which can be steered to $\mathcal{T} \subseteq \mathcal{X}$ in σ or fewer steps by a sequence of feasible inputs.*

The σ-step sets can be calculated by repeatedly calculating one-step sets [7]. More precisely, let $\mathcal{S}_0(\mathcal{T}_g) = \mathcal{T}_g$. Then

$$\mathcal{S}_i(\mathcal{T}_g) = \mathcal{Q}(\mathcal{S}_{i-1}(\mathcal{T}_g)) \cap \mathcal{X} \tag{4}$$

for all $i \in \mathbb{N}_1^\sigma$.

As a final preparation we state without proof the following relation between the σ-step sets $\mathcal{S}_\sigma(\mathcal{T}_g)$ and the feasible set $\mathcal{F}_h(\mathcal{T}_g)$.

Proposition 1: *[6] Let $\sigma \leq h$. Then $\mathcal{S}_\sigma(\mathcal{T}_g) \subseteq \mathcal{F}_h(\mathcal{T}_g)$. Furthermore, $\sigma = h$ implies $\mathcal{S}_\sigma(\mathcal{T}_g) = \mathcal{F}_h(\mathcal{T}_g)$ (the converse does in general not hold).*

3 Simultaneous computation of approximate step sets and feasible inputs

We approximate the feasible set $\mathcal{F}_h(\mathcal{T}_g)$ by calculating subsets $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ of the step sets $\mathcal{S}_i(\mathcal{T}_g)$ for $i = 1, \dots, \sigma$ with $\sigma \leq h$. Since $\hat{\mathcal{S}}_i(\mathcal{T}_g) \subseteq \mathcal{S}_i(\mathcal{T}_g)$, this results in $\hat{\mathcal{S}}_\sigma(\mathcal{T}_g) \subseteq \mathcal{S}_\sigma(\mathcal{T}_g) \subseteq \mathcal{F}_h(\mathcal{T}_g)$ according to Prop. 1. The computation of $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ is carried out in the style of [2]. Specifically, we also decompose the state space $\mathcal{X}$ into a set of hyperrectangles and use interval arithmetic to verify the membership of a particular hyperrectangle to $\hat{\mathcal{S}}_i(\mathcal{T}_g)$. In contrast to [2], however, we record the feasible inputs for the identified hyperrectangles within the step sets in an appropriate data structure. To this end it turns out to be necessary to represent $\mathcal{U}$ by a grid of points as opposed to a set of hyperrectangles as in [2].

3.1 State space bisection and input space grid

The proposed approach relies on bisecting $\mathcal{X}$ into a set of hyperrectangles. The resulting hyperrectangles can conveniently be represented by a binary tree. Furthermore, the bisection induces a grid for $\mathcal{U}$. While we attempt to keep these technical aspects to a minimum, the following notation needs to be introduced.

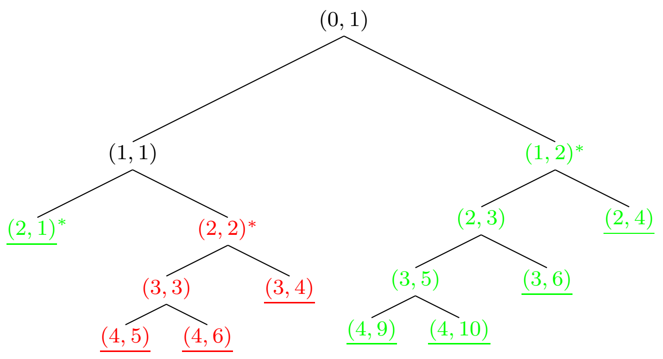

Figure 1: Sample binary tree of maximal depth $d = 4$ with nodes (δ, β). The underlined green (red) leaf nodes refer to hyperrectangles, which are (are not) members of a sample step set $\hat{\mathcal{S}}_i(\mathcal{T}_g)$. The nodes labeled with an asterisk give the most compact representation of $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ and the complementary set $\hat{\mathcal{S}}_i^c(\mathcal{T}_g)$. This is explained in more detail in Sect. 3.4.

Consider an arbitrary closed hyperrectangle $\mathcal{B} = [\underline{b}_1, \overline{b}_1] \times \cdots \times [\underline{b}_n, \overline{b}_n] \subset \mathbb{R}^n$. By bisection we refer to the operation that divides $\mathcal{B}$ into two closed hyperrectangles by splitting the interval of largest width $\omega_i(\mathcal{B}) = \overline{b}_i - \underline{b}_i$ of $\mathcal{B}$ into two intervals of equal length. Recursive bisection results in a hierarchy of hyperrectangles that have pairwise disjoint interiors and cover $\mathcal{B}$. Such a collection of hyperrectangles can conveniently be represented by a binary tree (cf. Fig. 1). We assume the hyperrectangle with the smaller values of the bisected interval is assigned to the left child node to ensure uniqueness of the tree. Figure 1 illustrates that every hyperrectangle can uniquely be identified with the depth δ of its node in the tree and the position β of the node in its level. Hyperrectangles that correspond to the node at depth δ and position β are denoted by $\boxplus_{\delta\beta}\mathcal{B}$. Hyperrectangles that correspond to leaf nodes are referred to as leaf hyperrectangles for short.

A desired resolution of the mesh can be enforced by requiring a certain number of bisections ν_i along each of the coordinate axes $i = 1, \ldots, n$. Without giving details we claim that for all i and all hyperrectangles $\boxplus_{d\beta}\mathcal{B}$ at leaf nodes with maximal depth (i.e. $\delta = d$)

$$\epsilon/2 < \omega_i(\boxplus_{d\beta}\mathcal{B}) \leq \epsilon,$$

if the maximal tree depth d is set to $d = \sum_{i=1}^{n} \nu_i(\epsilon)$, where $\nu_i(\epsilon) = \text{ceil}\left(\log_2\left(\epsilon^{-1}\,\omega_i(\mathcal{B})\right)\right)$. Finally, the set of all nodes (δ, β) of a tree of depth d is given by

$$\mathcal{N}_d = \{(\delta, \beta) \,|\, \delta \in \mathbb{N}_0^d, \beta \in \mathbb{N}_1^{2^\delta}\}.$$

The vertices of the 2^d leaf hyperrectangles of a perfect tree span an equidistant grid of $\gamma = \prod_{i=1}^{n}(2^{\nu_i} + 1)$ points. This set is denoted as

$$\{\boxdot_1\mathcal{B}, \cdots, \boxdot_\gamma\mathcal{B}\},$$

where the grid points $\boxdot_j\mathcal{B}$ may be enumerated in any arbitrary but fixed order.

As mentioned above, we represent the state space $\mathcal{X}$ by a set of hyperrectangles $\boxplus_{\delta\beta}\mathcal{X}$ and the input space $\mathcal{U}$ by a set of grid points $\boxdot_j\mathcal{U}$. This requires $\mathcal{X}$ and $\mathcal{U}$ to be hyperrectangles, i.e. $\mathcal{X} = [x_1] \times \cdots \times [x_n]$ and $\mathcal{U} = [u_1] \times \cdots \times [u_m]$ with $[x_i] = [\underline{x}_i, \overline{x}_i] \subset \mathbb{R}$, $[u_j] \subset \mathbb{R}$. Note that the resolution of the bisection of $\mathcal{X}$ and the resolution of the bisection of $\mathcal{U}$ may be chosen independently of one another.

3.2 Step set approximation

In order to approximate the one-step set, we need to investigate the images of the hyperrectangles $\boxplus_{\delta\beta}\mathcal{X}$ under f for the inputs $\boxdot_j\mathcal{U}$. In other words we need to investigate sets of the form

$$\{f(x, \boxdot_j\mathcal{U}) | x \in \boxplus_{\delta\beta}\mathcal{X}\}\,, \tag{5}$$

which we denote by $f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})$. The calculation of sets of the type (5) is in general difficult, if not impossible. It suffices, however, to calculate supersets

$$\mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})) \supseteq f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})$$

instead. Such a superset can be determined with, for example, interval arithmetic ([2,5]).

Assuming that $\mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U}))$ is available, an approximation of the one-step set can be determined by collecting all leaf hyperrectangles $\boxplus_{\delta\beta}\mathcal{X}$ for which there exists a control action $\boxdot_j\mathcal{U}$ such that $\mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})) \subseteq \mathcal{T}$. This is summarized in the following lemma.

Lemma 3: *Let* $\hat{\mathcal{Q}}(\mathcal{T}) = \bigcup_{(\delta,\beta)\in\tilde{\mathcal{M}}} \boxplus_{\delta\beta}\mathcal{X}$*, where*

$$\tilde{\mathcal{M}} = \{(\delta, \beta) \in \mathcal{N}_d \,|\, \exists\, j \in \mathbb{N}_1^\gamma : \mathcal{P}(f(\mathcal{B}, \boxdot_j\mathcal{U})) \subseteq \mathcal{T}\}.$$

Then $\hat{\mathcal{Q}}(\mathcal{T}) \subseteq \mathcal{Q}(\mathcal{T})$.

Note that the inclusion $\mathcal{T} \subseteq \mathcal{Q}(\mathcal{T})$ holds for the exact one-step set for any c.i. set $\mathcal{T}$. However, the corresponding inclusion does not apply to the one-step set estimates $\hat{\mathcal{Q}}(\mathcal{T}) \subseteq \mathcal{Q}(\mathcal{T})$ from Lemma 3. In fact there may be hyperrectangles $\boxplus_{d\beta}\mathcal{X}$ which are contained in $\mathcal{T}$, but are not contained in the estimate $\hat{\mathcal{Q}}(\mathcal{T})$. As a consequence, we have to replace Eq. (4) by

$$\hat{\mathcal{S}}_i(\mathcal{T}_g) = \hat{\mathcal{Q}}(\hat{\mathcal{S}}_{i-1}(\mathcal{T}_g)) \cup \hat{\mathcal{S}}_{i-1}(\mathcal{T}_g),$$

during the approximate calculation of step sets according to the following proposition, which we state without proof.

Proposition 2: *Let $i \in \mathbb{N}_1^\sigma$ and $\hat{\mathcal{S}}_0(\mathcal{T}_g) = \mathcal{S}_0(\mathcal{T}_g) = \mathcal{T}_g$. Define the initial set $\tilde{\mathcal{M}}_0 = \{(\delta, \beta) \in \mathcal{N}_d \,|\, \boxplus_{\delta\beta}\, \mathcal{X} \subseteq \hat{\mathcal{S}}_0(\mathcal{T}_g)\}$, the candidate set*

$$\tilde{\mathcal{C}}_i = \{(\delta, \beta) \in \mathcal{N}_d \setminus \tilde{\mathcal{M}}_{i-1} \,|\, \nexists\, (\tilde{\delta}, \tilde{\beta}) \in \tilde{\mathcal{M}}_{i-1} : \boxplus_{\tilde{\delta}\tilde{\beta}}\mathcal{X} \subset \boxplus_{\delta\beta}\mathcal{X}\}$$

and the improvement

$$\Delta\tilde{\mathcal{M}}_i = \{(\delta, \beta) \in \tilde{\mathcal{C}}_i \,|\, \exists\, j \in \mathbb{N}_1^\gamma : \mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})) \subseteq \hat{\mathcal{S}}_{i-1}(\mathcal{T}_g)\}.$$

Then the set $\hat{\mathcal{S}}_i(\mathcal{T}_g) = \bigcup_{(\delta,\beta) \in \tilde{\mathcal{M}}_i} \boxplus_{\delta\beta}\mathcal{X}$ with $\tilde{\mathcal{M}}_i = \tilde{\mathcal{M}}_{i-1} \cup \Delta\tilde{\mathcal{M}}_i$ represents an underestimation of the i-step set $\mathcal{S}_i(\mathcal{T}_g)$, i.e. $\hat{\mathcal{S}}_i(\mathcal{T}_g) \subseteq \mathcal{S}_i(\mathcal{T}_g)$.

The sets $\tilde{\mathcal{M}}$ in Lemma 3 and Prop. 2 contain redundant information, since some hyperrectangles may be contained in others in that there may exist $(\delta, \beta) \in \mathcal{M}$, $(\tilde{\delta}, \tilde{\beta}) \in \mathcal{M}$ such that $\boxplus_{\tilde{\delta}\tilde{\beta}}\mathcal{X} \subset \boxplus_{\delta\beta}\mathcal{X}$. Cases of this type may arise, since $\tilde{\mathcal{B}} \subseteq \mathcal{B}$ implies $f(\tilde{\mathcal{B}}, u) \subseteq f(\mathcal{B}, u)$, where we assume that this inclusion property is preserved in overestimations, i.e. $\mathcal{P}(f(\tilde{\mathcal{B}}, u)) \subseteq \mathcal{P}(f(\mathcal{B}, u))$. The redundancy can be removed by selecting those nodes whose parents are not part of the considered step sets, since these nodes correspond to the largest regions for which a feasible input can be found. The set

$$\mathcal{M} = \{(\delta, \beta) \in \tilde{\mathcal{M}} \,|\, (\delta - 1, \lceil \beta/2 \rceil) \notin \tilde{\mathcal{M}}\}$$

describes the desired nodes. The underlined green leaf nodes in Fig. 1 illustrate the set $\mathcal{M}$.

Note that algorithm 1 given in the appendix implements are more efficient way to compute $\mathcal{S}_i(\mathcal{T}_g)$ than Prop. 2. These improvements are technical, however, and therefore a discussion is omitted here. Figure 2 illustrates some step sets for the example discussed further below (in Sect. 4). The growth of the sets as well as the hyperrectangle hierarchy (cf. Sect. 3.4) is apparent from this figure.

3.3 Calculation of suboptimal feasible inputs

As pointed out before, we intend to calculate feasible input sequences that may serve, for example, as feasible initial guesses for the NMPC optimization problem (2)-(3). Having established the necessary tools and notation, this aim can be stated more precisely. We would like to determine a sequence of inputs such that, for any initial condition $x(0) \in \boxplus_{\delta\beta}\mathcal{X} \subseteq \hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$, the dynamical system (1) is driven to the terminal set $\mathcal{T}_g$ in σ or fewer steps. The mentioned hyperrectangle $\boxplus_{\delta\beta}\mathcal{X}$ with $(\delta, \beta) \in \mathcal{M}_\sigma$ is a subset of $\hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$ by definition, but it may be part of a step set $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ with $i < \sigma$. The smallest number of steps i such that the leaf hyperrectangle is contained in $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ is given by

$$i_{\delta\beta} = \min(\{i \in \mathbb{N}_0^\sigma \mid \boxplus_{\delta\beta}\, \mathcal{X} \subseteq \hat{\mathcal{S}}_i(\mathcal{T}_g)\}).$$

We intend to identify inputs $\boxdot_j\mathcal{U}$ that steer, for every $\boxplus_{\delta\beta}\mathcal{X} \subseteq \hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$ with $(\delta, \beta) \in \mathcal{M}_\sigma$ and $\boxplus_{\delta\beta}\mathcal{X} \nsubseteq \mathcal{T}_g$ (i.e. $i_{\delta\beta} > 0$), $\boxplus_{\delta\beta}\mathcal{X}$ into the previous step set $\hat{\mathcal{S}}_{i_{\delta\beta}-1}(\mathcal{T}_g)$. Note that one such $\boxdot_j\mathcal{U}$ arises in Prop. 2 during step $i_{\delta\beta}$. However, we opt for determining all possible $\boxdot_j\mathcal{U}$, since the quality of the NMPC initializations can be improved this way. Fortunately, the identification of the remaining $\boxdot_j\mathcal{U}$ can be carried out during step $i_{\delta\beta}$ as well. In fact those $\boxdot_j\mathcal{U}$ with $j \in \mathcal{I}_{\delta\beta}$, where

$$\mathcal{I}_{\delta\beta} = \{j \in \mathbb{N}_1^\gamma \,|\, \mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})) \subseteq \hat{\mathcal{S}}_{i_{\delta\beta}-1}(\mathcal{T}_g)\}$$

steer $\boxplus_{\delta\beta}\mathcal{X}$ to $\hat{\mathcal{S}}_{i_{d\beta}-1}(\mathcal{T}_g)$. We refer to the set $\mathcal{I}_{\delta\beta}$ as input set for short.

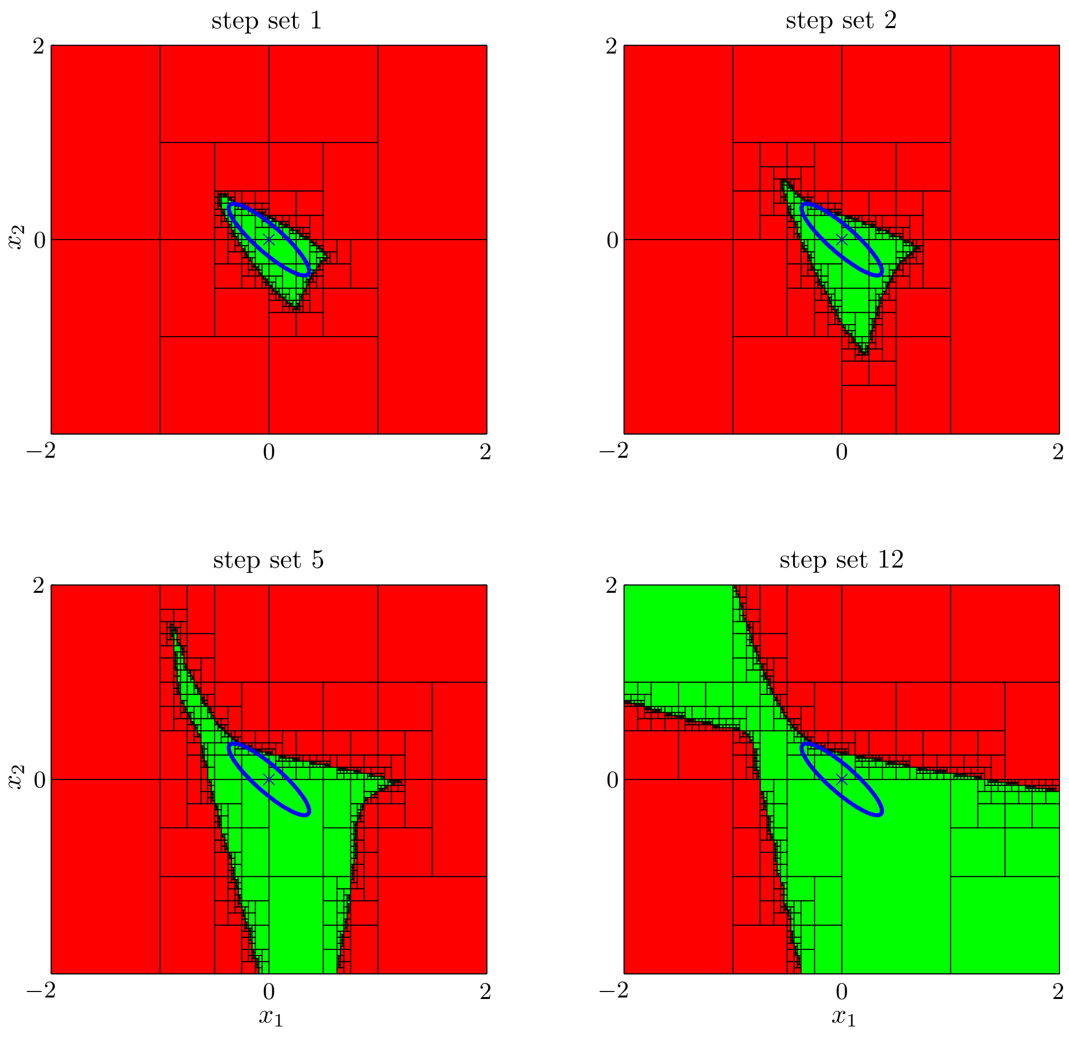

Figure 2: Sets $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ for the steps $i = \{1, 2, 5, 12\}$. The hyperrectangles which are members (no members) of the particular step sets are colored green (red). The terminal set and the setpoint $\breve{x}$ are represented by a blue ellipse and blue cross, respectively.

It remains to state an explicit formulation of the desired initialization sequences, or equivalently, an explicit control law. To this end we assign a single input value $u_{\delta\beta}$ to each node (δ, β), which is part of the final member set, i.e. $(\delta, \beta) \in \mathcal{M}_\sigma$. This results in a piecewise constant suboptimal control law

$$\tilde{u}(x) = \begin{cases} u_{\delta\beta} & \text{if } x \in \boxplus_{\delta\beta}\mathcal{X},\ (\delta, \beta) \in \mathcal{M}_\sigma,\ \boxplus_{\delta\beta}\mathcal{X} \nsubseteq \mathcal{T}_g \\ g(x) & \text{if } x \in \mathcal{T}_g \end{cases} \tag{6}$$

with guaranteed closed-loop stability and convergence to $\mathcal{T}_g$ (for $x_0 \in \hat{\mathcal{S}}_\sigma$) by construction. In order to select $u_{\delta\beta}$ we consider the midpoint of the current box (i.e. set $x_0 = \mathrm{mid}(\boxplus_{\delta\beta}\mathcal{X})$) and evaluate the input $\boxdot_{j^*}\mathcal{U}$ with $j^* \in \mathcal{I}_{\delta\beta}$ with the smallest cost function value for the resulting trajectory. More formally, this amounts to the following optimization problem, which can be solved by enumeration, and setting $u_{\delta\beta} = \boxdot_{j^*}\mathcal{U}$.

$$j^* = \arg \min_{j \in \mathcal{I}_{\delta\beta}} J(\hat{X}(0), \hat{U}(0)) \tag{7}$$

subject to

$$\begin{aligned} \hat{x}(0|0) &= \mathrm{mid}(\boxplus_{\delta\beta}\mathcal{X}) \\ \hat{u}(0|0) &= \boxdot_j\mathcal{U} \\ \hat{x}(k+1|0) &= f(\hat{x}(k|0), \hat{u}(k|0)),\ \forall\, k \in \mathbb{N}_0^{h-1}, \\ \hat{u}(k|0) &= \tilde{u}(\hat{x}(k|0)),\ \forall\, k \in \mathbb{N}_1^{h-1} \end{aligned} \tag{8}$$

In Alg. 1, the computation of $u_{\delta\beta}$ is carried out in line 21. Figure 3 visualizes the resulting control law $\tilde{u}(x)$ for the example discussed in Sect. 4. Based on Eq. (6), a feasible initialization for the NMPC optimization problem (2)-(3) can be generated, for any $x_0 \in \hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$, by setting $\hat{u}(k|0) = \tilde{u}(\hat{x}(k|0))$ with $\hat{x}(0|0) = x_0$ and $k \in \mathbb{N}_0^{h-1}$.

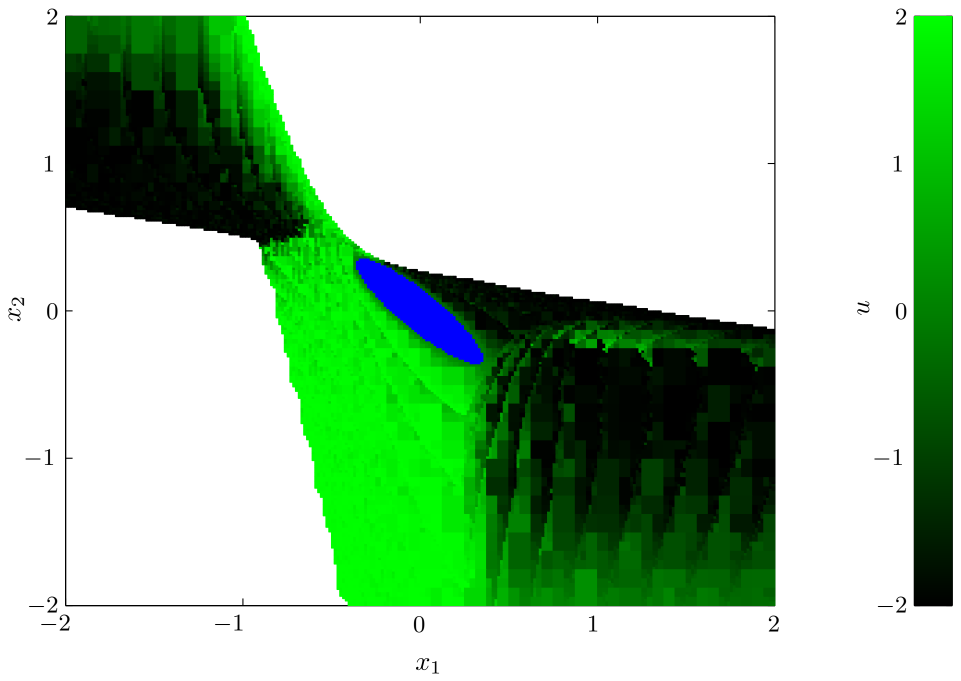

Figure 3: Visualization of the piecewise constant, suboptimal control law $\tilde{u}(x)$ for the example discussed further below (in Sect. 4). The green colorbar refers to the input value of the particular regions. The blue ellipse represents the terminal set, where the control law $g(x)$ is applied.

3.4 Node aggregation

The procedure described in Sect. 3.2 and 3.3 provides an approximation of the feasible set $\mathcal{S}_\sigma(\mathcal{T}_g)$ and a suboptimal explicit control law $\tilde{u}(x)$ with guaranteed stability. The size of the trees that are needed to represent the partition of the feasible set and the control law can become a limiting factor of the method. The tree that represents the feasible set can be reduced in size considerably by merging hyperrectangles based on the following observation: If the hyperrectangles associated with two sibling nodes are part of $\hat{\mathcal{S}}_i(\mathcal{T}_g)$, then the parent hyperrectangle is also a member of $\hat{\mathcal{S}}_i(\mathcal{T}_g)$. Formally, this aggregation of nodes corresponds to solving the optimization problem

$$\mathcal{M}_i^* = \arg\min_{\mathcal{M}_i} |\mathcal{M}_i| \quad \text{s.t.} \bigcup_{(\delta,\beta)\in\mathcal{M}_i} \boxplus_{\delta\beta}\mathcal{X} = \mathcal{S}_i(\mathcal{T}_g). \tag{9}$$

A corresponding size reduction for the tree that represents the control law $\tilde{u}(x)$ is not straightforward.

Figure 4 visualizes the aggregation that results from Eq. (9) for the system discussed in Sect. 4. Obviously, the reduction can be significant. The advantage of this aggregation becomes evident during the calculation of step sets. The decision whether a superset $\mathcal{P}(f(\boxplus_{\delta\beta}\mathcal{X},\boxdot_j\mathcal{U}))$ is a subset of $\hat{\mathcal{S}}_{i-1}(\mathcal{T}_g)$ or not (cf. Prop. 2), can be carried out much more efficiently using $\mathcal{M}_i^*$ instead of $\mathcal{M}_i$.

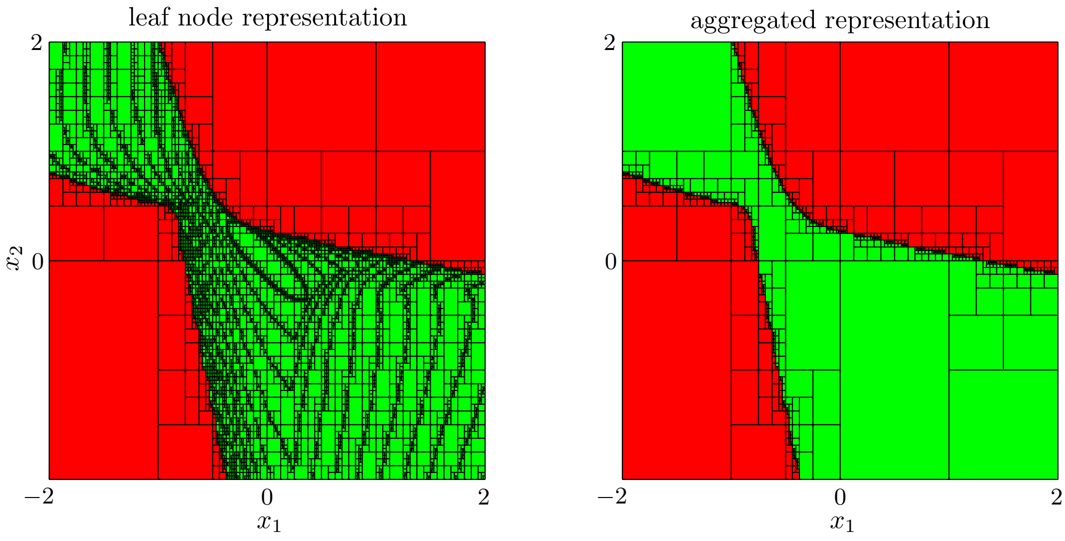

Figure 4: Two representations of the step set $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$ for the system discussed in Sect. 4. The associated perfect box-tree exhibits a total of 65536 leaf nodes, of which 30034 would be part of $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$ (i.e. 35502 would not). The 5861 green rectangles in the left figure represent boxes, for which at least one feasible input was detected, which steers the box to a previous step set. Merging the corresponding nodes (according to (9)) results in the aggregated representation on the right, which describes $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$ based on only 452 boxes. The visualization of the complementary set $\hat{\mathcal{S}}_{12}^c(\mathcal{T}_g)$ (red) requires 446 boxes in both cases.

4 Application and Example

We apply the proposed approach to the nonlinear system $x(k+1) = f(x(k), u(k))$ with

$$\begin{aligned} f_1(x,u) =& \tfrac{201}{200}x_1 + \tfrac{601}{6000}x_2 + \big(\tfrac{631}{12000} + \tfrac{299}{6000}x_1 - \tfrac{3}{400}x_2\big)u \\ &+ \big(\tfrac{9}{8000} + \tfrac{1}{800}x_1 + \tfrac{13}{24000}x_2\big)u^2 + \tfrac{1}{48000}\,(1+x_1)u^3 \\ f_2(x,u) =& \tfrac{601}{6000}x_1 + \tfrac{201}{200}x_2 + \big(\tfrac{631}{12000} - \tfrac{3}{400}x_1 - \tfrac{2407}{12000}x_2\big)u \\ &+ \big(-\tfrac{41}{8000} + \tfrac{13}{24000}x_1 + \tfrac{1}{50}x_2\big)u^2 + \tfrac{1}{3000}\,(1-4\,x_2)u^3, \end{aligned}$$

where $\mathcal{X} = [-2,2] \times [-2,2]$ and $\mathcal{U} = [-2,2]$. The system is a discrete-time variant[1] of the continuous time system treated in [2] and [3].

Obviously, $(\breve{x}, \breve{u}) = (0,0)$ is an equilibrium. We claim without proof that the ellipsoid $\mathcal{E} = \{x \in \mathbb{R}^n \mid \|x - \breve{x}\|_P^2 \leq \alpha^2\}$ with

$$P = \begin{pmatrix} 22.7067 & 19.9485 \\ 19.9485 & 22.7067 \end{pmatrix} \quad \text{and} \quad \alpha = 0.8408$$

[1] Heun's third order discretization method with a sample time $\Delta t = 0.1\,\mathrm{s}$ was applied to the system used in [2].

is a terminal set $\mathcal{T}_g$ for the control law $u = g(x) = -K\,(x - \breve{x}) + \breve{u}$ with $K = \left(2.0057 \quad 2.0057 \right)$. Choosing a prediction horizon $h = 15$ and defining

$$Q = \begin{pmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{pmatrix} \quad \text{and} \quad R = 1.0$$

completes the definition of the NLP (2)-(3).

We choose $\sigma = 12$ and $\epsilon = \epsilon_{\mathcal{X}} = \epsilon_{\mathcal{U}} = 0.02$ for the calculation of step sets and feasible inputs. We note that the associated box-tree exhibits a maximal depth $d = 16$ and a maximum of $2^d = 65536$ leaf nodes. The grid of inputs contains $\gamma = 2^8 + 1 = 257$ elements $\boxdot_j\mathcal{U}$. The resulting step sets $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ and the corresponding control law are illustrated in

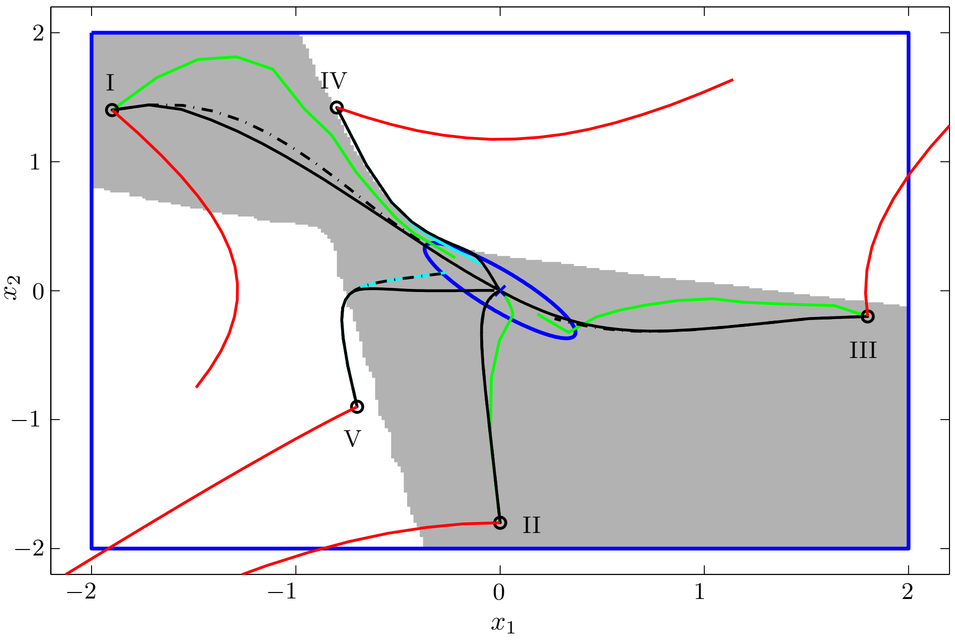

Figure 5: Initial, optimal, and some reference trajectories. Black circles mark initial states x_0. Green trajectories result from applying the suboptimal explicit control law $\tilde{u}(x)$ for input $u(k)$ at step $k = 0, \ldots, h-1$. Black trajectories result from solving the NMPC optimization problem (for 50 steps). Dash-dotted lines represent the solution of the NLP for $k = 0$. Red trajectories result for the choice $u(k) = \breve{u}$. The grey region represents the step set $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$, the blue ellipse marks the terminal set $\mathcal{T}_g = \mathcal{E}$ and the blue square indicates the boundary of $\mathcal{X}$.

Figs. 2 and 3, respectively. Figure 5 illustrates the results of some numerical experiments. For the initial values I to III, which are members of the step set $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$ (grey region), we are able to calculate feasible initializations for the NMPC optimization problem (2)-(3) using the control law $\tilde{u}(x)$ as specified in Sect. 3.3. As expected, the corresponding resulting trajectories of the dynamical system (green curves) never leave $\hat{\mathcal{S}}_{12}(\mathcal{T}_g)$ and always lead to $\mathcal{T}_g$. Note, however, that the optimal trajectory that results from solving the NMPC optimization problem deviates from the one that results from our approach.

For states which are not part of $\hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$ (here $\sigma = 12$), the presented method is not able to provide a feasible initialization. Since $\hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$ in general is an underestimation of the actual feasible set, i.e. $\hat{\mathcal{S}}_\sigma(\mathcal{T}_g) \subset \mathcal{F}_h(\mathcal{T}_g)$, feasible input sequences may exist for points $x_0 \notin \hat{\mathcal{S}}_\sigma(\mathcal{T}_g)$, however. Consider the points in Fig. 5 marked by IV and V, for example. There exist feasible initializations (cyan trajectories) in these cases (found by trial and error here).

Finally, Fig. 5 illustrates that the initialization $\hat{u}(i|0) = \breve{u}$ is in general not a good choice (red trajectories).

5 Conclusions

We extended an existing method for the computation of c.i. sets ([2,3]) to the simultaneous calculation of c.i. sets and an explicit controller that regulates the system to the origin. The control law may be used to compute feasible initializations for NMPC (warm start). By construction the feasible input sequences guarantee stability of the NMPC problem for any initial value in the c.i. set.

The approach was illustrated with a benchmark example [2,3]. Numerical experiments show that the feasible input sequences are excellent initial guesses for NMPC. This is an interesting result by itself, since NMPC optimization problems sometimes cannot be solved if no feasible initial guess for the underlying nonlinear program can be found.

The computational effort of the method is high. Once the step sets $\hat{\mathcal{S}}_i(\mathcal{T}_g)$ and the explicit control law $\tilde{u}(x)$ have been calculated offline, however, feasible initializations can be calculated for any initial condition in the c.i. set very quickly. Compared to existing explicit NMPC approaches [11], our control laws result in a worse performance, but they can be calculated without solving the underlying NLP.

Acknowledgment

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, MO 1086/9).

References

[1] D. P. Bertsekas and I. B. Rhodes. Minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–247, 1971.

[2] J. M. Bravo, D. Limon, T. Alamo, and E. F. Camacho. On the computation of invariant sets for constrained nonlinear systems: An interval arithmetic approach. *Automatica*, 41(9):1583–1589, 2005.

[3] H. Chen and F. Allgower. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[4] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical report, Department of Engineering, University of Cambridge, 2004.

[5] R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer Academic Publishers, 1996.

[6] E. C. Kerrigan. *Robust Constraint Satisfaction: Invariant sets and predictive control.* PhD thesis, University of Cambridge, 2000.

[7] D. Limon, T. Alamo, and E. F. Camacho. Enlarging the domain of attraction of MPC controllers. *Automatica*, 41(4):629–635, 2005.

[8] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.

[9] F. Scibilia, S. Olaru, and M. Hovd. On feasible sets for MPC and their approximations. *Automatica*, 47(1):133–139, 2010.

[10] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[11] S. Summers, D. M. Raimondo, C. N. Jones, J. Lygeros, and M. Morari. Fast explicit nonlinear model predictive control via multiresolution function approximation with guaranteed stability. In *Proc. of 8th IFAC Symposium on Nonlinear Control Systems*, 2010.

A Algorithmic details

Algorithm 1: *Computation of step sets and feasible inputs.*

```
1  $\mathcal{M}_0 \leftarrow \emptyset$; $\mathcal{C} \leftarrow \emptyset$; $\hat{S}_0 \leftarrow \mathcal{T}_g$
2  nodes $\leftarrow$ new stack($(0,1)$)
3  while nodes.length() $> 0$ do
4      $(\delta,\beta) \leftarrow$ nodes.pop()
5      if $\boxplus_{\delta\beta}\mathcal{X} \subseteq \hat{S}_0$ then $\mathcal{M}_0 \leftarrow \mathcal{M}_0 \cup (\delta,\beta)$
6      else
7          if $\delta = d \vee (\boxplus_{\delta\beta}\mathcal{X} \cap \hat{S}_0 = \emptyset)$ then $\mathcal{C} \leftarrow \mathcal{C} \cup (\delta,\beta)$
8          else nodes.push($(\delta+1, 2\beta-1), (\delta+1, 2\beta)$)
9  for $i \leftarrow 1$ to $\sigma$ do
10     $\mathcal{T} \leftarrow \hat{S}_{i-1}$; $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1}$
11     repeat
12         foreach $(\delta,\beta) \in \mathcal{C}$ do
13             if $\boxplus_{\delta\beta}\mathcal{X} \cap \mathcal{T} \neq \emptyset$ then nodes.push($(\delta,\beta)$);
14         $\Delta\mathcal{M} \leftarrow \emptyset$
15         while nodes.length() $> 0$ do
16             $(\delta,\beta) \leftarrow$ nodes.pop(); $\mathcal{I} \leftarrow \emptyset$
17             for $j \leftarrow 1$ to $\gamma$ do
18                 if $\mathcal{P}_{\mathrm{IA}}(f(\boxplus_{\delta\beta}\mathcal{X}, \boxdot_j\mathcal{U})) \subset \hat{S}_{i-1}$ then $\mathcal{I} \leftarrow \mathcal{I} \cup j$
19             if $|\mathcal{I}| > 0$ then
20                 $\Delta\mathcal{M} \leftarrow \Delta\mathcal{M} \cup (\delta,\beta)$; $\mathcal{C} \leftarrow \mathcal{C} \setminus (\delta,\beta)$
21                 solve OP (7)-(8) and set $u_{\delta\beta}$
22             else
23                 if $\delta > d$ then
24                     $\mathcal{C} \leftarrow (\mathcal{C} \setminus (\delta,\beta)) \cup (\delta+1, 2\beta-1) \cup (\delta+1, 2\beta)$
25                     if $\boxplus_{\delta+1,2\beta-1}\mathcal{X} \cap \mathcal{T} \neq \emptyset$ then
26                         nodes.push($(\delta+1, 2\beta-1)$)
27                         if $\boxplus_{\delta+1,2\beta}\mathcal{X} \cap \mathcal{T} \neq \emptyset$ then
28                             nodes.push($(\delta+1, 2\beta)$)
29                     else nodes.push($(\delta+1, 2\beta)$)
30         $\mathcal{T} \leftarrow \bigcup_{(\delta,\beta)\in\Delta\mathcal{M}} \boxplus_{\delta\beta}\mathcal{X}$; $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \Delta\mathcal{M}$
       until $|\Delta\mathcal{M}| = 0$
31     solve OP (9) and set $\mathcal{M}_i^*$
32     $\hat{S}_i \leftarrow \bigcup_{(\delta,\beta)\in\mathcal{M}_i^*} \boxplus_{\delta\beta}\mathcal{X}$
```

Algorithm 1 formalizes the calculation of step sets and feasible inputs. Lines 1-8 carry out preliminary steps. such as the compilation of the sets $\mathcal{M}_0$ and $\mathcal{C}$. Lines 9-31 calculate the step sets $\hat{S}_i(\mathcal{T}_g)$ acc. to Prop. 2 and the piecewise constant control law $\tilde{u}(x)$ corresponding to Sect. 3.3. At first, we identify neighboring boxes of the current target set $\mathcal{T}$ (ll. 12-13). Then, we check these neighbors in terms of membership to the current step set i. In doing so, we calculate the propagation of the considered box $\boxplus_{\delta\beta}\mathcal{X}$ in combination with each input $\boxdot_j\mathcal{U}$ and check whether the (overestimated) image is part of $\hat{S}_{i-1}(\mathcal{T}_g)$ or not (ll. 17-18). If at least one feasible input was detected, we update the member and candidate sets and calculate the suboptimal input $u_{\delta\beta}$ by solving the optimization problem (OP) (7)-(8) (ll. 19-21). Otherwise, the current node-box is bisected (if the maximal depth d is not reached) (ll. 22-28). Now, the update of the target and member set is carried out by taking the obtained improvement ($\Delta\mathcal{M}$) into account (l. 29). The lines 12-29 are repeated until no improvement is achieved. Finally, $\mathcal{M}_i^*$ and the step itself are computed in line 30 and 31. Note that there are more efficient ways to implement Alg. 1. The version shown here is chosen for ease of presentation.