# Set-Membership Computation of Admissible Controls for Trajectory Tracking

Olivier Mullier, Estelle Courtial

# Set-Membership Computation of Admissible Controls for Trajectory Tracking*

Olivier Mullier
University of Orléans, PRISME Laboratory
Olivier.Mullier@univ-orleans.fr

Estelle Courtial
University of Orléans, PRISME Laboratory
Estelle.Courtial@univ-orleans.fr

**Abstract**

In a context of predictive control strategy, this paper addresses the computation of admissible control set for a trajectory tracking over a prediction horizon. The proposed method combines numerical methods based on set-membership computation and control methods based on a flatness concept. It makes possible i) to provide a guaranteed computation of admissible controls, ii) to deal with uncertain reference trajectory, iii) to reduce the time complexity of the algorithm compared to the existing approach. Simulations illustrate the efficiency of the developed methods in two different cases. For Single Input - Single Output (SISO) systems, generalized affine forms are computed otherwise a Branch & Prune algorithm with an inner inclusion test is used for Multi Inputs - Multi Outputs (MIMO) systems. The computational time is reduced significantly compared to the one required by the existing approach.

**Keywords:** set-membership computation, flatness, nonlinear predictive control, discrete-time systems.
**AMS subject classifications:** 65P99, 93C55

## 1 Introduction

Among the control methods capable of tracking a reference trajectory, Nonlinear Model Predictive Control (NMPC) is well-adapted, especially in the presence of constraints on state and/or input variables [1]. The aim of NMPC is to determine a sequence of controls by solving a constrained optimization problem at time $k$ over a prediction horizon $n_p$. Only the first component of the sequence is really applied to the process.

*Submitted: February 19, 2016; Revised: December 16, 2016; Accepted: April 5, 2017.

At time $k+1$, information is updated thanks to measurements, the prediction horizon moves one sampling period ahead, and the whole optimization procedure is repeated.

In some cases, for example in economic application [2], the reference trajectory may not be known precisely due to uncertainties or confident intervals of the targeted variable. Hence, we consider not a reference trajectory but a set of possible reference trajectories to which the reference trajectory must belong. The sequence of controls consequently becomes a set of controls. Computing such a set is intractable in general, but set-membership computation can be a very useful tool to provide outer or inner approximations of this set.

Several methods are based on set-membership computations: linear matrix inequalities [5], affine forms [20], interval analysis [15] or generalized affine forms [4]. Classical intervals [17] are used in many situations to compute rigorously sets of values with intervals instead of reals, leading to outer approximations of the desired sets.

Interval analysis has already been used in robotics to solve control problems such as navigation and localisation [12], collision avoidance or reliability [14]. Modelling errors can be taken into account in the interval formulation. Robust control approaches based on interval analysis have largely been developed to deal with uncertain parameters for linear systems [18]. More recently set-membership computations were considered for nonlinear estimation [9] or control [10].

In our problem, since the objective is to determine the set of admissible controls, the computation of an inner approximation of the control set is mandatory. In this context, most methods use the Branch & Prune algorithm (B&P)[6] combined with interval analysis. In [7], the SIVIA algorithm based on this kind of combination has been used for parameter estimation, then was enhanced in [8]. In [9], among all the guaranteed sequences of control, the search of only one was considered. In [10], the computation of all the guaranteed sequences of control was considered, but only for a short prediction horizon because of the time complexity of the SIVIA algorithm. Indeed, the major drawback of the B&P algorithm is its time complexity, exponential in the dimension of the control sequence. Considering a control input of dimension $m$, the dimension of the control sequence is $m \times n_p$, leading to a time complexity $O(\exp(m \times n_p))$.

To avoid the aforementioned drawback, we propose a direct method to compute an inner approximation of the admissible control set for tracking an uncertain reference trajectory. The flatness [3], an intrinsic property of dynamical systems, is used and makes possible to guarantee the inner approximation result with the algorithm complexity reduced to $O(n_p \times \exp(m \times n))$ with $n \leqslant n_p$ the dimension of the state. Two different methods are developed according to the dimension of the control $m$. For the generic case ($m \geqslant 1$), the method is based on the B&P algorithm with an inner test. For the special case ($m = 1$), it is based on generalized affine forms.

This paper is organized as follows: brief reviews of set-membership computation and interval analysis are presented in Section 2. The existing approach and its limitation are detailed. Section 3 is devoted to the flatness concept and to the main results of this paper. Finally simulations on SISO and MIMO systems illustrate the computation time and precision efficiency of the proposed methods.

# 2 Review of Set Membership Computation and Existing Approach

The set computation usually provides two kinds of approximations: an inner approximation and/or an outer approximation. For a given set $\mathcal{P}$, the set $\mathcal{Q}$ is an inner approximation of $\mathcal{P}$ if these sets verify the quantified proposition

$$(Q \subseteq P) \Leftrightarrow (\forall q \in \mathcal{Q})(\exists p \in \mathcal{P})(p = q); \tag{1}$$

the set $\mathcal{Q}$ is an outer approximation of $\mathcal{P}$ if they verify

$$(P \subseteq Q) \Leftrightarrow (\forall p \in \mathcal{P})(\exists q \in \mathcal{Q})(p = q). \tag{2}$$

If $\mathcal{P} = \mathcal{Q}$, both quantified propositions are verified.

In the following, classical results on interval analysis used in Section 2.2 and Section 3 are reviewed briefly.

## 2.1 Interval Analysis

We denote an interval by $[x] = [\underline{x}, \overline{x}]$ with $\underline{x} \leqslant \overline{x}$ and the set of intervals by $\mathbb{IR} = \{[x] = [\underline{x}, \overline{x}] \mid \underline{x}, \overline{x} \in \mathbb{R},\ \underline{x} \leqslant \overline{x}\}$. The Cartesian product of intervals $[x] \in \mathbb{IR}^n$ is called a box. The main result of interval analysis is its fundamental theorem [17] stating that the evaluation of an expression using intervals leads to an *outer approximation* of the resulting set of values for this expression whatever the values considered in the intervals. To deal with interval functions, an interval inclusion function also known as interval extension of a function can be defined.

**Definition 2.1 (Inclusion function [15])** *Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$. The interval function $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ is an interval inclusion function of $f$ if the evaluation of $[f]([x])$ gives an outer approximation of the image of $[x]$ by the function $f$, denoted by $f([x]) = \{f(x), x \in [x]\}$,*

$$\forall [x] \in \mathbb{IR}^n, f([x]) = \{f(x), x \in [x]\} \subseteq [f]([x]). \tag{3}$$

*This definition can also be expressed as a quantified proposition,*

$$(\forall x \in [x])(\exists z \in [f]([x]))(z = f(x)). \tag{4}$$

**Remark 2.1** *The quantified proposition* (4) *corresponds to the one described in Eq.* (2) *with $\mathcal{Q} = f([x])$ and $\mathcal{P} = [f]([x])$.*

Many interval extensions of function can be defined that satisfy the quantified proposition (4). We can cite the natural extension [15], which replaces the operations on reals by their interval counterparts using interval arithmetic, and the mean value extension [15], which linearizes the function around its mean value.

Inclusion functions allow us to compute outer approximations. For inner approximations, much harder to compute, the SIVIA algorithm [7] is the most frequently used. It is based on the B&P algorithm with an inclusion test (verification of a property). To compute the set of values $x \in \mathcal{X}$ which verify a property $P$, the SIVIA algorithm defines two lists of boxes, $\mathcal{L}_{\texttt{Inside}}$ and $\mathcal{L}_{\texttt{Boundary}}$, such that

$$(\cup\mathcal{L}_{\texttt{Inside}}) \subseteq \{x \in \mathcal{X} \mid P(x)\} \subseteq (\cup\mathcal{L}_{\texttt{Inside}}) \cup (\cup\mathcal{L}_{\texttt{Boundary}}), \tag{5}$$

with $(\cup\mathcal{L}_{\texttt{Inside}})$ and $(\cup\mathcal{L}_{\texttt{Boundary}})$, the union of the boxes contained in these lists. We require an interval evaluation $[P]([x])$ of the property $P$ (inclusion test)

$$[P]([x]) = \begin{cases} 0 & \text{if } \forall x \in [x], x \text{ verifies } P \\ 1 & \text{if } \forall x \in [x], x \text{ does not verify } P \\ [0,1] & \text{otherwise.} \end{cases}$$

The B&P algorithm is described in Algorithm 1. The resulting list $\mathcal{L}_{\texttt{Inside}}$ is a subpaving of $\{x \in \mathcal{X} \mid P(x)\}$.

**Algorithm 1:** The B&P algorithm.

**Input:** $[P]$, $[x]$, $\epsilon$
**Output:** $\mathcal{L}_{\texttt{Inside}}$ (list of boxes), $\mathcal{L}_{\texttt{Boundary}}$ (list of boxes)
$\mathcal{L}_{\texttt{Inside}}, \mathcal{L}_{\texttt{Boundary}} \leftarrow \emptyset$; `// empty lists of boxes`
$\mathcal{L}_{\texttt{Domain}} \leftarrow \{[x]\}$; `// list containing only` $[x]$
**while** $\mathcal{L}_{Domain}$ *not empty* **do**
  $[\tilde{x}] \leftarrow \text{Extract}(\mathcal{L}_{\texttt{Domain}})$; `// extraction of an element from` $\mathcal{L}_{\texttt{Domain}}$
  $[\tilde{y}] \leftarrow [P]([\tilde{x}])$; `// Inclusion test`
  **if** $[\tilde{y}] = 0$ **then**
    $\mathcal{L}_{\texttt{Inside}} \leftarrow \mathcal{L}_{\texttt{Inside}} \cup \{[\tilde{y}]\}$;
  **else if** $[\tilde{y}] = [0,1]$ **then**
    **if** $\text{width}([\tilde{x}]) \geq \epsilon$ **then**
      Bisection of $[\tilde{x}]$ to get $[\tilde{x}']$ and $[\tilde{x}'']$;
      $\mathcal{L}_{\texttt{Domain}} \leftarrow \mathcal{L}_{\texttt{Domain}} \cup \{[\tilde{x}'], [\tilde{x}'']\}$;
    **else**
      $\mathcal{L}_{\texttt{Boundary}} \leftarrow \mathcal{L}_{\texttt{Boundary}} \cup \{[\tilde{y}]\}$;
**return** $(\mathcal{L}_{Inside}, \mathcal{L}_{Boundary})$;

## 2.2 Existing Approach for Control Purposes

We consider the class of nonlinear discrete-time systems described by

$$(S_d) \begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = h(x_k), \end{cases} \tag{6}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and $y_k \in \mathbb{R}^p$ are respectively the state, the input and the output of the system at the current time $k$, $x_0$ is the initial condition. The functions $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^p$ are nonlinear vector functions. The reference trajectory to be tracked is not certainly known, but belongs to a set $\mathbb{Y}^{\text{ref}} = \{\mathbb{Y}^{\text{ref}}_{k+1}, \mathbb{Y}^{\text{ref}}_{k+2}, \dots\}$. The control objective is to determine the set of admissible controls $\mathcal{U}^*_i$, at time $i$, such that

$$\mathcal{U}^*_i = \{u_i \in \mathbb{R}^m \mid h(f(x_i, u_i)) \in \mathbb{Y}^{\text{ref}}_{i+1}\}. \tag{7}$$

Considering a prediction horizon $n_p$, we have to compute all the sets $\mathcal{U}_i^*$ for $i = k$ to $k + n_p - 1$ denoted:

$$\mathcal{U}^*_{k..k+n_p-1} = \{(u_k, \ldots, u_{k+n_p-1}) \in \mathbb{R}^{m\times n_p} \mid \quad \forall i = k, \ldots, k+n_p-1, \\ h(f(x_i, u_i)) \in \mathbb{Y}^{\text{ref}}_{i+1}\}. \tag{8}$$

Let us define $g_\ell : \mathbb{R}^{m\times(\ell+1)} \to \mathbb{R}^m$ such that for $u_{k..k+\ell} = (u_k, u_{k+1}, \ldots, u_{k+\ell})$:

$$g_{k+\ell}(u_{k..k+\ell}) = h(\underbrace{f(f(\ldots f}_{\ell+1 \text{ times}}(x_k, u_k), u_{k+1}), \ldots), u_{k+\ell}). \tag{9}$$

Considering Eq. (9), the set expressed in Eq. (8) can be reformulated as:

$$\mathcal{U}^*_{k..k+n_p-1} = \{(u_k, \ldots, u_{k+n_p-1}) \in \mathbb{R}^{m\times n_p} \mid \quad \forall \ell = 0, \ldots, n_p-1, \\ g_{k+\ell}(u_{k..k+\ell}) \in \mathbb{Y}^{\text{ref}}_{k+\ell+1}\}. \tag{10}$$

The computation of the set $\mathcal{U}^*_{k..k+n_p-1}$ is the typical case of a set inversion problem, solved in [10] by using the SIVIA algorithm. This method has a major drawback: the time complexity is exponential on the size of the considered box to split which is here $m \times n_p$. It consequently limits the method for short prediction horizons $n_p$.

In the next section, we present a new way of computing a guaranteed inner approximation of the projection, on each control $u_i$ ($i = k, \ldots, k + n_p - 1$), of the set described in Eq. (10) by using the flatness concept.

# 3 Main Results

The proposed method guarantees the computation of an inner approximation of the set described in Eq. (10) component-wise. It combines numerical methods based on interval analysis or generalized affine forms with the flatness concept. The flatness property can simplify the computation thanks to a difference parametrization of the system variables.

## 3.1 Flatness

The idea of differential flatness was first introduced by Fliess *et al.* in 1995 [3]. A system is differentially flat if there exists a set of independent variables (equal in number to the dimension of inputs) referred to as flat outputs such that all states and inputs of the system can be expressed in terms of those flat outputs and a finite number of their successive time derivatives (resp. advances) for continuous-time (resp. discrete-time, [13]) systems. In the case of linear systems, a controllable system is a flat system. In the case of nonlinear systems, there is no general theorem. A flat nonlinear system is consequently controllable. The property of flatness is now detailed for nonlinear discrete-time systems.

**Definition 3.1 (Flatness)** *The nonlinear discrete-time system* $(S_d)$ *described in* (6) *is flat if there exists an output* $F_k \in \mathbb{R}^m$ *such that the following relationships are verified for all* $k$*:*

$$x_k = \psi(F_k, F_{k+1}, \ldots, F_{k+r-1}) \tag{11}$$
$$y_k = h(\psi(F_k, F_{k+1}, \ldots, F_{k+r-1})) \tag{12}$$
$$u_k = \varphi(F_k, F_{k+1}, \ldots, F_{k+r}). \tag{13}$$

*The relative degree of the system r is equal to the number of advances (forward-shifts) of the considered output in order to have at least one component of the input vector $u_k$ explicitly appearing.*

The output $F_k \in \mathbb{R}^m$ is then called a flat output, and Equations (11)–(13) are difference parameterizations of the system variables. We denote by $z_k \in \mathbb{R}^{r\times m}$ the vector composed of the flat output $F_k$ and its $r-1$ advances:

$$z_k = \begin{pmatrix} z_{k,1} \\ \vdots \\ z_{k,r} \end{pmatrix} = \begin{pmatrix} F_k \\ \vdots \\ F_{k+r-1} \end{pmatrix}. \tag{14}$$

Equations. (11)-(13) can be reformulated as:

$$x_k = \psi(z_k) \tag{15}$$
$$y_k = h(\psi(z_k)) \tag{16}$$
$$u_k = \varphi(z_k, v_k). \tag{17}$$

The new input $v_k = F_{k+r}$ is the local solution of $u_k = \varphi(z_k, v_k)$, meaning there exists $\theta$ such that $v_k = \theta(z_k, u_k)$. If $\psi$ is a locally invertible function, Eq. (15) can be seen as a coordinate change between $x_k$ and $z_k$. Figure 1 represents the connections between all the variables of a flat system. The local representation of the system with the coordinate change is given by the canonical Brunovsky form

$$z_{k+1} = Az_k + Bv_k, \tag{18}$$

with

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \tag{19}$$

## 3.2 Computation of Admissible Controls for Flat Systems

The flatness property allows us to express the state and input variables in terms of the flat output variable. In our study, it is a twofold advantage. On one hand, we can guarantee the inner computation of the admissible control set (expressed in terms of the flat output). On the other hand, the computational load is reduced (algebraic expression). As shown in Figure 1, we have

$$x_{k+1} = f(x_k, u_k) = \psi(z_{k+1}) = \psi\left(A\psi^{-1}(x_k) + B\theta(\psi^{-1}(x_k), u_k)\right). \tag{20}$$

The right member in Eq. (20) can be used to define an inclusion function for $f(x_k, u_k)$, but is generally not tight due to multiple occurrences of $x_k$ in the expression, leading to larger over-estimates than directly using the expression of $f(x_k, u_k)$:

$$[f]\left([x_k], [u_k]\right) \subseteq [\psi]\left(A\left[\psi^{-1}\right]([x_k]) + B\,[\theta]\left(\left[\psi^{-1}\right]([x_k]), [u_k]\right)\right) \tag{21}$$

with $[f]$, $[\psi]$, $[\theta]$ and $[\varphi]$, the interval extensions of $f$, $\psi$, $\theta$ and $\varphi$, respectively. Thanks to flatness, the set of controls can be evaluated directly by taking into consideration

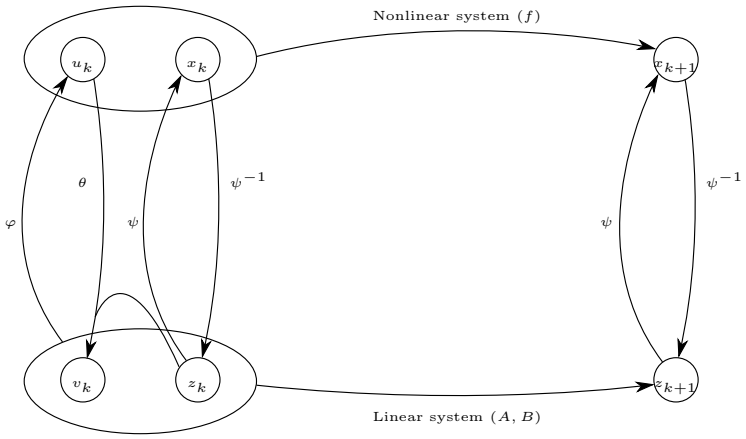


Figure 1: Connection diagram between variables of a flat system.

the definition of the control in terms of the flat output $u_k = \varphi(z_k, v_k)$. The reference trajectory is an admissible trajectory for the flat output. The set of admissible controls $\mathcal{U}_k^*$ at each time $k$ is

$$\mathcal{U}_k^* = \{\varphi(z_k, v_k) \in \mathbb{R}^m \mid z_k \in \mathbb{Y}_{k..k+r-1}^{\text{ref}}, v_k \in \mathbb{Y}_{k+r}^{\text{ref}}\}. \tag{22}$$

**Property 3.1** *Consider the sets $\mathcal{U}_k^*$ described in Eq.* (22) *and $\mathcal{U}_{k..k+n_p-1}^*$ described in Eq.* (10) *with $n_p \geqslant r+1$. We have:*

$$\mathcal{U}_k^* = \underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*) \tag{23}$$

*with $\underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*)$, the projection of the set $\mathcal{U}_{k..k+n_p-1}^*$ onto the control $u_k$.*

*Proof: i) Let us prove that $\underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*) \subseteq \mathcal{U}_k^*$. We assume that a control $u_k \in \underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*)$. Then there exist $(u_{k+1}, \ldots, u_{k+n_p-1})$ such that $(u_k, \ldots, u_{k+n_p-1}) \in \mathcal{U}_{k..k+n_p-1}^*$ is an admissible control sequence. The restriction to the $r$ first components of this control sequence satisfies*

$$(g_k(u_k), g_{k+1}(u_{k..k+1}), \ldots, g_{k+r-1}(u_{k..k+r-1})) = (y_{k+1}, \ldots, y_{k+r}).$$

*Thanks to the flatness, there exists $y_k$ such that the difference parameterization of the control gives $u_k = \varphi(y_k, y_{k+1}, \ldots, y_{k+r}) \in \mathcal{U}_k^*$.*

*ii) We now prove that $\mathcal{U}_k^* \subseteq \underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*)$. We assume that a control $u_k \in \mathcal{U}_k^*$. Consequently, There exists $(y_k, \ldots, y_{k+r})$ such that $u_k = \varphi(y_k, \ldots, y_{k+r}) \in \mathcal{U}_k^*$. We can extract $u_{k+1..k+i} = (u_{k+1}, \ldots, u_{k+i})$ such that $g_{k+i}(u_k, u_{k+1..k+i}) = y_{k+i+1}$ for all $i = 1$ to $r-1$. It implies that $u_k \in \underset{u_k}{proj}(\mathcal{U}_{k..k+r}^*)$. The outputs $g_{k+i}$ do not depend on the control $u_k$ according to the definition of the relative degree. Thus $u_k \in \underset{u_k}{proj}(\mathcal{U}_{k..k+n_p-1}^*)$.* □

Characterizing the set of admissible controls defined in Eq. (22) corresponds to the problem of computing the image $S$ of a set $\mathbb{X}$ by a function $\mathcal{F}$

$$S = \{\mathcal{F}(x) \mid x \in \mathbb{X}\}, \tag{24}$$

where $S$ corresponds to the set $\mathcal{U}_k^*$, $\mathcal{F}$ to the function $\varphi$ and $\mathbb{X}$ to the reference trajectory set $\mathbb{Y}^{\text{ref}}$. The computation of such a set $S$ has already been addressed in [4, 16] with two different methods. In [4], the set $S$ considered is of dimension 1, and in [16], the dimension can be greater than 1.

### 3.2.1 Generic case: $m \geqslant 1$

For SISO and MIMO systems ($u_k \in \mathbb{R}^m$, $m \geqslant 1$), the method proposed in [16] combines the B&P algorithm with a specific property $P$ for Algorithm 1. This property to be checked is a test based on the Hansen-Sengupta operator (see Corollary 3.1 in [16]). It consists of constructing a pseudo-inverse inclusion function of $\mathcal{F}$, noted $\left[\mathcal{F}^{\dagger}\right]$ and comparing the initial box $[x]$ with the resulting box $\left[x^{\dagger}\right] = \left[\mathcal{F}^{\dagger}\right]([y])$, where $[y] = [\mathcal{F}]([x])$. In the sequel, this test will be referred to as the *inner test*. Applied to our problem, the function $\mathcal{F}$ corresponds to the function $\varphi$ from $\mathbb{R}^{m\times(r+1)}$ to $\mathbb{R}^m$. If the rank of $\varphi$ is $m$ and given its inclusion function $[\varphi]$, any box $[y_{k..k+r}] \subseteq \mathbb{Y}^{\text{ref}}_{k..k+r}$ can be used to compute an inner approximation of $\mathcal{U}_k^*$.

**Remark 3.1** *If additional constraints on the controls have to be satisfied, meaning that $u_k$ must belong to a particular set $\mathcal{U}$, the set of admissible controls $\mathcal{U}_k^*$ becomes:*

$$\mathcal{U}_k^* = \{\varphi(z_k, v_k) \in \mathcal{U} \mid z_k \in \mathbb{Y}^{ref}_{k..k+r-1}, v_k \in \mathbb{Y}^{ref}_{k+r}\}. \tag{25}$$

*To take into consideration this modification, the property $P$ in the B&P algorithm is supplemented with a test based on the inclusion function $[\varphi]$: for any candidate $[y_{k..k+r}] \subseteq \mathbb{Y}^{ref}_{k..k+r}$, $[\varphi]([y_{k..k+r}]) \in \mathcal{U}$. If constraints on state variables ($x_k \in \mathbb{X}_k$) have to be handled, the set $\mathcal{U}_k^*$ becomes:*

$$\mathcal{U}_k^* = \{\varphi(z_k, v_k) \in \mathbb{R}^m \mid z_k \in \mathbb{Y}^{ref}_{k..k+r-1},\ v_k \in \mathbb{Y}^{ref}_{k+r},\ \psi(z_k) \in \mathbb{X}_k\}. \tag{26}$$

*Using an inclusion function $[\psi]$ and add a test in $P$ verifying that $[\psi]([y_{k..k+r}]) \subseteq \mathbb{X}$ for a given box $[y_{k..k+r}] \subseteq \mathbb{Y}^{ref}_{k..k+r}$.*

To summarize, constraints on states and/or controls can be handled easily by the proposed method without increasing its time complexity.

### 3.2.2 Special case: $m = 1$

For the special case of a SISO system, another method based on generalized intervals [11] and affine forms [20] is developed. A generalized interval is an interval $[\underline{x}, \overline{x}]$ without the constraint $\underline{x} \leqslant \overline{x}$. Generalized intervals allow us to compute inner approximations in specific cases instead of outer approximations using classical intervals. For an interval $[x] = [\underline{x}, \overline{x}]$, if $\underline{x} \leqslant \overline{x}$ (classical interval), $[x]$ is said to be proper. Otherwise, if $\underline{x} \geqslant \overline{x}$, $[x]$ is improper. An operator pro returns a proper interval: $\text{pro}([\underline{x}, \overline{x}]) = [\overline{x}, \underline{x}]$ if $\underline{x} \geqslant \overline{x}$, and $\text{pro}([\underline{x}, \overline{x}]) = [\underline{x}, \overline{x}]$ if $\underline{x} \leqslant \overline{x}$.

Affine forms are designed to retain the linear dependencies between variables occurring in computation. An affine form $\hat{x}$ representing the set of values taken by a variable $x$ is denoted by

$$\hat{x} = \alpha_0^x + \sum_{i=1}^{n} \alpha_i^x \varepsilon_i, \tag{27}$$

with $\alpha_i^x \in \mathbb{R}$ for all $i$. Each noise symbol $\varepsilon_i \in [-1,1]$ is unknown. It represents an independent component of the global uncertainty on $\hat{x}$. An interval $[x] = [\underline{x}, \overline{x}]$ can be converted easily to an affine form $\hat{x}$ by

$$\hat{x} = \frac{\overline{x} + \underline{x}}{2} + \frac{\overline{x} - \underline{x}}{2}\varepsilon_1. \tag{28}$$

From an affine form, the associated interval can be computed by replacing each noise symbol by the interval $[-1,1]$:

$$[\hat{x}] = \alpha_0^x + \sum_{i=1}^{n} \alpha_i^x \, [-1,1]\,, \tag{29}$$

which is an outer approximation of the values $x$ can take.

When nonlinear operations occur on affine forms, the nonlinear dependencies are represented by a new noise symbol $\eta$ with its associated partial deviation. They represent an outer approximation of the associated nonlinear dependency. For example, one way to compute the multiplication between two affine forms $\hat{x} = \alpha_0^x + \sum_{i=1}^n \alpha_i^x \varepsilon_i$ and $\hat{y} = \alpha_0^y + \sum_{i=1}^n \alpha_i^y \varepsilon_i$ is

$$\hat{x} \times \hat{y} = \alpha_0^x \alpha_0^y + \frac{1}{2}\sum_{i=1}^{n} \alpha_i^x \alpha_i^y + \sum_{i=1}^{n} (\alpha_0^x \alpha_i^y + \alpha_0^y \alpha_i^x)\varepsilon_i + \left( \frac{1}{2}\left| \sum_{i=1}^{n} \alpha_i^x \alpha_i^y \right| + \left| \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \alpha_i^x \alpha_j^y \right| \right) \eta. \tag{30}$$

In [4], the authors dealt with the computation of inner approximations by combining generalized intervals and affine forms.

**Definition 3.2 (First-order generalized affine vectors)** *A first-order generalized affine vector from $\mathbb{R}^n$ to $\mathbb{R}^p$ is a triple $(\hat{Z}, c, \hat{J})$ such that $\hat{Z} \in \mathcal{M}(n+m+1, p)$ is a vector of affine forms, $c \in \mathbb{R}^p$ a vector, and $\hat{J} \in (\mathcal{M}(n,p))^n$ is a matrix of affine forms.*

The generalized affine form $(\hat{Z}, c, \hat{J})$ associated to a function $\mathcal{F}$ and a set $\mathbb{X}$ have the properties

- the affine vector $\hat{Z}$ represents an outer approximation of $\{\mathcal{F}(x) \mid x \in \mathbb{X}\}$,
- the vector $c$ corresponds to the center of $\{\mathcal{F}(x) \mid x \in \mathbb{X}\}$, and
- the matrix of affine forms $\hat{J}$ is an outer approximation of the values the Jacobian of $\mathcal{F}$ can take for $x \in \mathbb{X}$.

From a generalized affine form $\breve{x} = (\hat{Z}_x, c_x, \hat{J}_x)$ representing the set of values a variable $x$ can take, an outer approximation of $[\hat{x}]$ can be computed using $\hat{Z}_x$ in Eq. (29). An interval considered as an inner approximation of this set can be computed as

$$[\breve{x}] = \mathrm{pro}(c_x + \left[\hat{J}_x\right] [1,-1]). \tag{31}$$

However, the formulation into an inner approximation using intervals is only possible for affine forms ($m = 1$, SISO sytem where $u_k \in \mathbb{R}$) and not for affine vectors. To implement this method, we first define the generalized affine forms corresponding to the values of the reference trajectory $\mathbb{Y}^{\mathrm{ref}}_{k..k+r}$, and then we compute the generalized affine form $\breve{u}_k$ thanks to $\varphi$.

# 4 Simulations

The methods we developed are tested and compared with the existing method [10] in two different cases: a SISO system (m=1) and a MIMO system (m=2). Simulations highlight a significant reduction of computation time, while keeping a satisfactory precision for the resulting inner approximations.

## 4.1 Case of a SISO System

Consider the nonlinear discrete-time SISO system

$$\begin{cases} x_1(k+1) = x_1(k)x_2(k) \\ x_2(k+1) = x_2(k) + u(k) \\ y(k) = x_1(k). \end{cases} \tag{32}$$

The system is controllable. The objective is to determine the set of admissible controls to track the reference trajectory over a prediction horizon $n_p = 2$ to be able to compare the results with the existing approach. The first step is to verify that the output variable $x_1(k)$ is a flat output of the system. Let us set $F(k) = x_1(k)$ and express the state and the input variables in terms of the flat output $F(k)$ and its advances,

$$\begin{aligned} x_1(k) &= F(k) \\ x_2(k) &= \frac{x_1(k+1)}{x_1(k)} = \frac{F(k+1)}{F(k)} \\ u(k) &= x_2(k+1) - x_2(k) = \frac{F(k+2)}{F(k+1)} - \frac{F(k+1)}{F(k)}. \end{aligned}$$

By setting $z(k) = (F(k)\ F(k+1))^T$ and $v(k) = F(k+2)$, the input $u_k$ is

$$u(k) = \frac{v(k)}{z_2(k)} - \frac{z_2(k)}{z_1(k)} = \varphi(z(k), v(k)). \tag{33}$$

The function $\varphi$ requires the knowledge of the flat output and its two successive advances ($r = 2$). It means that to compute the set of admissible controls ($u(0)$ and $u(1)$) over the prediction horizon $n_p$ ($n_p = 2$), $n_p + r$ values are necessary: the value of the flat output at time $k = 0$ and three values of the reference trajectory from $k = 1$ to $n_p + 1$. The set of the reference trajectory $\mathbb{Y}^{\text{ref}}_{1..3}$ is

$$\mathbb{Y}^{\text{ref}}_{1..3} = (\mathbb{Y}^{\text{ref}}_1, \mathbb{Y}^{\text{ref}}_2, \mathbb{Y}^{\text{ref}}_3) = ([1,2], [1,2], [5,7]). \tag{34}$$

### 4.1.1 Generalized affine forms (Method 1)

We have to compute the generalized affine form associated with $u(0)$ and $u(1)$ using the function $\varphi$ and the reference trajectory $\mathbb{Y}^{\text{ref}}_{1..3}$. The first step is to compute the

generalized affine forms associated with $F(0) = x_1(0), \mathbb{Y}_1^{\text{ref}}, \mathbb{Y}_2^{\text{ref}}, \mathbb{Y}_3^{\text{ref}}$ by using Eq. (28):

$$\check{x}_0 = \begin{pmatrix} \hat{x}_0 & = 1 \\ c_{x_0} & = 1 \\ \hat{J}_{x_0} & = (0\ 0\ 0) \end{pmatrix} \tag{35}$$

$$\check{y}_1 = \begin{pmatrix} \hat{y}_1 & = \frac{3}{2} + \frac{1}{2}\varepsilon_1 \\ c_{y_1} & = \frac{3}{2} \\ \hat{J}_{y_1} & = (\frac{1}{2}\ 0\ 0) \end{pmatrix} \tag{36}$$

$$\check{y}_2 = \begin{pmatrix} \hat{y}_2 & = \frac{3}{2} + \frac{1}{2}\varepsilon_2 \\ c_{y_2} & = \frac{3}{2} \\ \hat{J}_{y_2} & = (0\ \frac{1}{2}\ 0) \end{pmatrix} \tag{37}$$

$$\check{y}_3 = \begin{pmatrix} \hat{y}_3 & = 6 + \varepsilon_3 \\ c_{y_3} & = 6 \\ \hat{J}_{y_3} & = (0\ 0\ 1) \end{pmatrix}. \tag{38}$$

The quantities $\varepsilon_2, \varepsilon_3 \in [-1, 1]$ are noise symbols. Thanks to Eq. (33), we can compute the generalized affine form associated with $\check{u}_0$ and $\check{u}_1$:

$$\check{u}(0) = \frac{\check{y}_2}{\check{y}_1} - \frac{\check{y}_1}{\check{x}_0}, \qquad \check{u}(1) = \frac{\check{y}_3}{\check{y}_2} - \frac{\check{y}_2}{\check{y}_1}. \tag{39}$$

The computation of the inverse of a generalized affine form was not previously defined. For this purpose, we extended the Min-Range approximation defined for affine forms in Corollary 1 in [19] to a generalized affine form inverse. The resulting generalized affine forms are

$$\check{u}_0 = \begin{pmatrix} \hat{u}_0 & = -\frac{3}{8} - \frac{11}{16}\varepsilon_1 + \frac{3}{8}\varepsilon_2 + \frac{5}{16}\eta \\ c_{u_0} & = -\frac{1}{2} \\ \hat{J}_{u_0} & = \big((-\frac{35}{32} + \frac{9}{128}\varepsilon_1 - 0.197917\varepsilon_2 + 0.471354\eta) \\ & \quad (\frac{3}{8} - \frac{1}{16}\varepsilon_1 + \frac{1}{16}\eta)\ (0)\big) \end{pmatrix}, \text{ and} \tag{40}$$

$$\check{u}_1 = \begin{pmatrix} \hat{u}_1 & = 3.375 + 0.1875\varepsilon_1 + -1.125\varepsilon_2 + 0.75\varepsilon_3 + 1.3125\eta \\ c_{u_1} & = 3 \\ \hat{J}_{u_1} & = ((0.59375 - 0.0703125\varepsilon_1 + 0.197917\varepsilon_2 + 0.471354\eta) \\ & \quad (-\frac{11}{4} + \frac{1}{16}\varepsilon_1 + \frac{9}{32}\varepsilon_2 - 0.395833\varepsilon_3 + 1.67708\eta) \\ & \quad (\frac{3}{4} - \frac{1}{8}\varepsilon_2 + \frac{1}{8}\eta)) \end{pmatrix}. \tag{41}$$

Replacing the quantities $\varepsilon_2, \varepsilon_3, \eta$ by their interval counterparts $[-1, 1]$, we obtain the intervals $\mathcal{U}_0^*$ and $\mathcal{U}_1^*$ thanks to Eq. (31).

### 4.1.2 B&P algorithm with an inner test (Method 2)

For this method, only the function $\varphi$ and its Jacobian are necessary for the computation,

$$J_\varphi = \begin{pmatrix} \frac{z_2(k)}{z_1(k)^2} & -\frac{v(k)}{z_2(k)^2} - \frac{1}{z_1(k)} & \frac{1}{z_2(k)} \end{pmatrix}.$$

### 4.1.3 Results

To highlight the improvement in terms of computation time, the results of Method 1 and 2 are compared with the method described in [10] referred to as Method 3, where

Table 1: Results for the computation of $\mathcal{U}_0^*$.

| Method | Method 1 | Method 2 | Method 3 |
|---|---|---|---|
| Results | $[-1.375, 0.375]$ | $[-1.4375, 0.8125]$ | $[-1.2696, 0.7812]$ |
| Computation time | $1.37672 \times 10^{-6}$s | 0.009s | 0.030s$^{\S}$ |

Table 2: Results for the computation of $\mathcal{U}_1^*$.

| Method | Method 1 | Method 2 | Method 3 |
|---|---|---|---|
| Results | $[1.5, 4.5]$ | $[0.5938, 5.687]$ | $[1.3672, 4.4921]$ |
| Computation time | $2.86311 \times 10^{-6}$s | 0.013656s | 0.030s$^{\S}$ |

the set $\mathcal{U}_{0,1}^*$ is considered. Results for the computation of $\mathcal{U}_0^*$ and $\mathcal{U}_1^*$ are shown in Table 1 and Table 2, respectively. The parameter $\epsilon$ giving the minimum width of the considered boxes is equal to 0.1 for Methods 2 and 3. ($^{\S}$) : the computation time of Method 3 is the time required for the computation of the set $\mathcal{U}_{0,1}^*$. Method 1 provides a single interval which cannot be improved. By contrast, the computational time is faster than the B&P-based methods. For the computational time required by Method 1, Methods 2 and 3 cannot provide any result. On the other hand, they provide a more precise result with respect to the parameter $\epsilon$ chosen. Method 2 gives larger intervals than Method 3.

The total computation times for both $\mathcal{U}_0^*$ and $\mathcal{U}_1^*$ are $0.0042ms, 13.85ms$ and $30ms$ for Methods $1, 2$ and 3, respectively. The number of bisections for methods 2 and 3 is 165 and 8507 respectively. The developed methods (Methods 1 and 2) show a significant reduction of the computation time, while maintaining a satisfactory precision.

Figure 2 corresponds to an iterated computation of the controls from time $t = 0$ to 4. At each time instant $t$, an inner approximation of the set of possible controls $u(t)$ is provided using our method, and a particular control is chosen (here the center of the interval). The control is then applied to the system, and the computation starts again at the time instant $t + 1$. Figure 2 shows that the computed set of admissible controls computed guarantees the reference tracking by ensuring the process output to belong to the reference interval.

## 4.2 Case of a MIMO System

Consider the nonlinear MIMO system ($x(k) \in \mathbb{R}^4$, $u(k) \in \mathbb{R}^2$, $y(k) \in \mathbb{R}^2$):

$$\begin{cases} x_1(k+1) = x_1(k)x_2(k) \\ x_2(k+1) = x_3(k)(2 - x_4(k)) \\ x_3(k+1) = x_1(k)u_1(k) \\ x_4(k+1) = u_2(k) \\ y(k) = \begin{pmatrix} x_1(k) \\ x_4(k) \end{pmatrix} \end{cases} \tag{42}$$

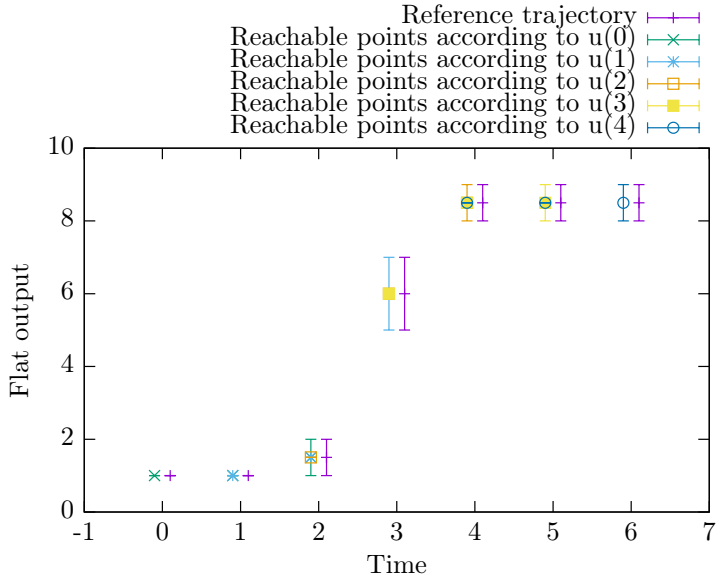


Figure 2: Set of admissible reachable sets according the computed controls.

Table 3: Reference trajectory for $\mathbb{Y}^{\text{ref}}_{1..7}$ for the MIMO example ($\delta = [-0.2, 0.2]$).

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $z_1(k)$ | $1.7+\delta$ | $1.5+\delta$ | $-0.8+\delta$ | $-4.9+\delta$ | $-4.1+\delta$ | $1.6+\delta$ | $-1.6+\delta$ |
| $z_2(k)$ | $1.2+\delta$ | $-1+\delta$ | $1.5+\delta$ | $1+\delta$ | $1.5+\delta$ | $1.2+\delta$ | $0.8+\delta$ |

This system is flat with the flat output $y(k) = (x_1(k), x_4(k))^T$. We can rewrite $u(k)$ in terms of the flat output and its advances:

$$u(k) = \varphi(\underbrace{z_1(k), z_2(k), z_1(k+2)}_{z(k)}, \underbrace{z_1(k+3)}_{v(k)}) = \begin{pmatrix} \frac{z_1(k+3)}{z_1(k)z_1(k+2)(2-z_2(k))} \\ z_2(k+1) \end{pmatrix}. \tag{43}$$

The prediction horizon $n_p = 5$ and the relative degrees are $r_1 = 3$ and $r_2 = 1$ for $y_1(k)$ and $y_2(k)$, respectively. The reference trajectory $\mathbb{Y}^{\text{ref}}_{1..7}$ is a sequence of intervals with $\delta = [-0.2, 0.2]$ given in Table 3. We can compute $u(k)$ from $k = 0$ to 4 by taking into consideration the advances of the flat output required for the computation.

### 4.2.1 Results

The B&P algorithm with the inner test is applied with a parameter $\epsilon = 0.05$. Figure 3 and Figure 4 represent the set of admissible controls for $u(0)$ and $u(4)$, respectively. The set of blue boxes is an inner approximation, and the union of blue and red boxes is an outer approximation. Computation times for each control set are given in Table 4. The existing method cannot provide any result. Indeed, due to the dimension of

Table 4: Time needed to compute an inner approximation of $u(0)$ to $u(4)$.

| u(k) | u(0) | u(1) | u(2) | u(3) | u(4) |
|---|---|---|---|---|---|
| time (s) | 0.32 | 0.14 | 0.14 | 0.32 | 0.67 |

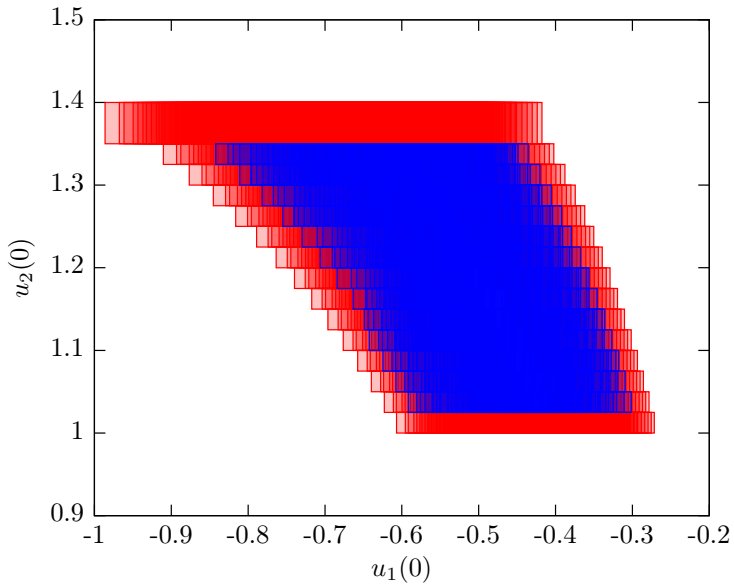


Figure 3: Set of admissible controls for $u(0)$.

the control box ($m \times n_p = 2 \times 5 = 10$), the computational time largely exceeds an acceptable time.

# 5 Conclusion

In this paper, two methods are proposed for the computation of admissible control set for an uncertain trajectory tracking over a prediction horizon. Based on set-membership computation combined with the flatness property of dynamical systems, the control problem can be viewed as the computation of the image of a set by a function. Two methods for the computation of inner approximations have been developed according to the dimension of the control input. A Branch & Prune algorithm with an inner test can be used whatever the dimension considered. Furthermore, constraints on states and controls can be handled easily in this case. For the special case of SISO systems, generalized affine forms can easily provide inner approximations with a very fast computational time. For both cases, simulations show the computational time and precision efficiency of the methods.

Our methods could be used for online purposes to track robustly an uncertain reference trajectory. The domain of applications concerned by this study is very widespread, from mobile robot navigation to economic applications where an uncer-

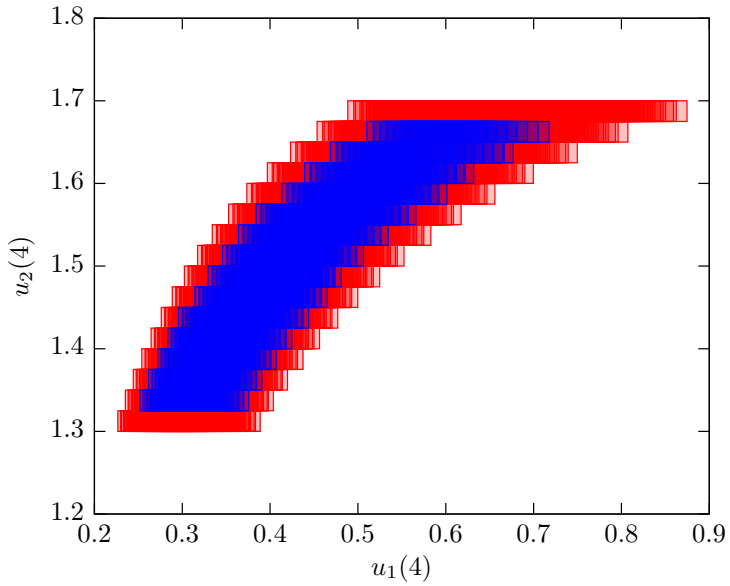


Figure 4: Set of admissible controls for $u(4)$.

tain targeted variable has to be tracked. An extension to nonlinear continuous-time systems is under investigation.

## Acknowledgements

This work is supported by an initiative of the French Ministry of Higher Education and Research "Investissements d'Avenir", through the project VOLTAIRE (ANR-10-LABX-100-01).

## References

[1] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[2] E. Courtial and C. Garrouste. Model predictive control strategy to forecast employability in earth sciences. In *19th World Congress of the International Federation of Automatic Control, Cape Town, South Africa*, pages 10731–10736, 2014.

[3] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995.

[4] Eric Goubault, Olivier Mullier, Sylvie Putot, and Michel Kieffer. Inner approximated reachability analysis. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 163–172, New York, NY, USA, 2014. ACM.

[5] Didier Henrion and Christophe Louembet. Convex inner approximations of nonconvex semialgebraic sets applied to fixed-order controller design. *International Journal of Control*, 85(8):1083–1092, 2012.

[6] Pascal Van Hentenryck, David McAllester, and Deepak Kapur. Solving polynomial systems using a branch and prune approach. *SIAM Journal on Numerical Analysis*, 34(2):797–827, 1997.

[7] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

[8] Luc Jaulin, Michel Kieffer, Isabelle Braems, and Eric Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control*, 74(18):1772–1782, 2001.

[9] Luc Jaulin, Stefan Ratschan, and Laurent Hardouin. Set computation for nonlinear control. *Reliable Computing*, 10(1):1–26, 2004.

[10] Luc Jaulin and Eric Walter. Global numerical approach to nonlinear discrete-time control. *Automatic Control, IEEE Transactions on*, 42(6):872–875, 1997.

[11] E. Kaucher. Interval analysis in the extended interval space IR. In Götz Alefeld and Rolf Dieter Grigorieff, editors, *Fundamentals of Numerical Computation (Computer-Oriented Numerical Analysis)*, pages 33–49. Springer, Vienna, 1980.

[12] Michel Kieffer, Luc Jaulin, Éric Walter, and Dominique Meizel. Robust autonomous robot localization using interval analysis. *Reliable computing*, 6(3):337–362, 2000.

[13] J. B. Mare and J. A. De Donà. A case study in explicit solutions to constrained nonlinear MPC. In *Conference on Nonlinear Model Predictive Control for Fast Systems*, Grenoble, France, 2006.

[14] Jean-Pierre Merlet. Interval analysis and reliability in robotics. *International Journal of Reliability and Safety*, 3(1-3):104–130, 2009.

[15] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[16] O. Mullier, E. Goubault, M. Kieffer, and S. Putot. General inner approximation of vector-valued functions. *Reliable Computing*, 18:117–143, November 2013.

[17] A. Neumaier. *Interval Methods for Systems of Equations (Encyclopedia of Mathematics and its Applications)*. Cambridge University Press, Jan 1991.

[18] Balasaheb M Patre and Pramod J Deore. Robust state feedback for interval systems: An interval analysis approach. *Reliable Computing*, 14(1):46–60, 2010.

[19] Siegfried M. Rump and Masahide Kashiwagi. Implementation and improvements of affine arithmetic. *Nonlinear Theory and Its Applications, IEICE*, 6(3):341–359, 2015.

[20] J Stolfi and LH De Figueiredo. An introduction to affine arithmetic. *Trends in Applied and Computational Mathematics*, 4(3):297–312, 2003.