Inner and Outer Approximations of Existentially Quantified Equality Constraints

Alexandre Goldsztejn[1] and Luc Jaulin[2]

[1] Computer Science Department, University of Central Arkansas, Conway, Arkansas, USA. Alexandre@Goldsztejn.com

[2] Luc Jaulin: E3I2, ENSIETA, 2 rue F. Verny, 29806 Brest Cedex 09
Luc.Jaulin@ensieta.fr

Abstract. We propose a branch and prune algorithm that is able to compute inner and outer approximations of the solution set of an existentially quantified constraint where existential parameters are shared between several equations. While other techniques that handle such constraints need some preliminary formal simplification of the problem or only work on simpler special cases, our algorithm is the first pure numerical algorithm that can approximate the solution set of such constraints in the general case. Hence this new algorithm allows computing inner approximations that were out of reach until today.

1 Introduction

Many problems in computer science amount to characterizing an inner and an outer approximation of a set defined by nonlinear constraints where quantifiers may be involved. We address here the case where existential quantifiers are involved (which actually corresponds to the projection of a manifold defined by equalities). When these constraints are polynomial, symbolic methods have been shown to be able to solve the problem (see e.g., [1]). However, these techniques are restricted to very small systems. When the constraints are defined by inequalities, interval methods make it possible to characterize the solution set (see e.g., [2]). When equality constraints are involved, the problem is much more difficult and no general method seems to be available to compute an inner approximation of a set defined by nonlinear equalities. Some works were already proposed to deal with some specific subclasses of these problems: [3, 4] are restricted to linear systems, [5] is restricted to cases where the different constraints do not share any existentially quantified parameters and [6] is more general but still suffers from strong restrictions.

The paper is dedicated to the approximation of the graph of an existentially quantified constraint $c(x_1, \cdots, x_{n_x})$ defined by

$$\big(\exists y_1 \in \mathbf{y}_1\big) \cdots \big(\exists y_{n_y} \in \mathbf{y}_{n_y}\big)$$
$$\Big(f_1(x_1, \cdots, x_{n_x}, y_1, \cdots, y_{n_y}) = 0 \wedge \cdots \wedge f_m(x_1, \cdots, x_{n_x}, y_1, \cdots, y_{n_y}) = 0\Big),$$

where $\mathbf{y}_k$ for $k \in [1..n_y]$ are some bounded and nonempty intervals. Using vectorial notations, this constraint is written $c_{f,\mathbf{y}}(x)$ and defined by

$$c_{f,\mathbf{y}}(x) \iff \big(\exists y \in \mathbf{y}\big)\big(f(x,y) = 0\big), \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ and $y \in \mathbb{R}^{n_y}$ and $\mathbf{y}$ is a bounded and nonempty box of dimension n_y and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \longrightarrow \mathbb{R}^m$. The graph of $c_{f,\mathbf{y}}$ is denoted by $\Sigma(f,\mathbf{y}) := \{x \in \mathbb{R}^{n_x} \mid c_{f,\mathbf{y}}(x)\}$. It is likely to have a non-zero volume if $n_y \geq m$, i.e. if there are at least as many existentially quantified variables as equations. Therefore, both inner and outer approximations are relevant. Many practical problems can be formulated as the characterization of such a set. Let us quote two of them:

– **Control**. Most dynamical systems can be described by the following state equation $\dot{x}(t) = f\big(x(t), u(t)\big) \wedge g\big(x(t),\ y(t), u(t)\big) = 0$. where the vector u is the input vector, x is the state vector and y is the output vector. The *feasible output set* $\mathbb{O}$ is the set of all $\bar{y}$ such that one can find a control u such that $g(t)$ converges to $\bar{y}$. As proved in [7], $\mathbb{O}$ satisfies

$$\mathbb{O} = \{\bar{y} \mid \exists \bar{u}, \exists \bar{x}, f(\bar{x}, \bar{u}) = 0 \wedge g(\bar{x}, \bar{y}, \bar{u}) = 0\}, \tag{2}$$

and its characterization is therefore an instance of the problem we propose to solve.

– **Robotic.** The geometric model of a robot can often be described by the relation $f(u,x) = 0$, where u is the articulation vector and x is the coordinate vector of the tool (see e.g., [8]). For serial robots, the relation becomes $x = g(u)$ and for parallel robots, it is $u = g(x)$. However for more general robots, neither u nor x can be isolated in the relation. The workspace $\mathbb{W}$ of the robot is defined by

$$\mathbb{W} = \{x \mid \exists u \in \mathbf{u}, f(x,u) = 0\}, \tag{3}$$

where $\mathbf{u}$ is the box of all feasible configuration vectors for the robot. Characterizing the workspace of a general robot can thus be cast into is a projection-equality problem.

2 Interval Analysis

The modern interval analysis was born in the 60's with [9]. Since, it has been widely developed and is today one central tool in the resolution of constraints acting over continuous domains (see [10] and extensive references). We now present the main concepts of interval analysis that will be used in the sequel.

Intervals are denoted by boldface symbols. The set of intervals is denoted by $\mathbb{IR}$ and contains, by convention, the empty set. The union between intervals is not an interval in general. The join between intervals (also called interval hull) is introduced to correct this bad behavior of the union. Let E be a set of intervals. The join of E, denoted by $\vee E$, is the smallest interval that contains each interval of E. When E contains only two elements, i.e. $E = \{\mathbf{x}, \mathbf{y}\}$, the join of E is also denoted by $\mathbf{x} \vee \mathbf{y}$.

The elementary functions are extended to intervals in the following way: let $\circ \in \{+, -, \times, /\}$ then $\mathbf{x} \circ \mathbf{y} = \{x \circ y \mid x \in \mathbf{x}, y \in \mathbf{y}\}$. Due to the monotony properties of these simple functions, formal expressions for the interval arithmetic are available. E.g. $[a, b]+[c, d] = [a+c, b+d]$. Also, continuous one variable functions $f(x)$ are extended to intervals using the same definition: $f(\mathbf{x}) = \{f(x) \mid x \in \mathbf{x}\}$, which is an interval because f is continuous. When one represents numbers using finite precision, the previous operations cannot be computed in general. The outer rounding is then used so as to keep valid the interpretations. For example, $[1, 2] + [2, 3]$ could be equal to $[2.999, 5.001]$ when rounded with a three decimal accuracy.

When one considers more complicated functions that are compounded of elementary functions, he will compute the interval evaluation of the function: this consists of replacing all real operations by their interval counterpart. A very basic result from interval analysis proves that the interval evaluation computes intervals that contain the range of the function. For example, $\mathbf{x} \times (\mathbf{y} - \mathbf{x}) \supseteq \{x(y - x) \mid x \in \mathbf{x}, y \in \mathbf{y}\}$.

This will be useful to use some vectorial notations. The variables $x_1, \ldots, x_n$ are denoted by the vector $x = (x_1, \ldots, x_n)$. The domains of the variables $x_1, \ldots, x_n$ are then denoted by the n-dimensional box $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, meaning that the domain of x_k is $\mathbf{x}_k$. It will also be useful to denote the vector $(x_1, \ldots, x_{n_x}, y_1, \ldots, y_{n_y})$ by (x, y) and therefore the box $(\mathbf{x}_1, \ldots, \mathbf{x}_{n_x}, \mathbf{y}_1, \ldots, \mathbf{y}_{n_y})$ by $(\mathbf{x}, \mathbf{y})$.

3 General Description of the Algorithm

In the sequel, we consider two initial boxes $\mathbf{x}^{Init} \in \mathbb{IR}^{n_x}$ and $\mathbf{y}^{Init} \in \mathbb{IR}^{n_y}$, both bounded and nonempty. The inner and outer approximations of $\Sigma(f, \mathbf{y}^{Init}) \cap \mathbf{x}^{Init}$ will be studied. The algorithm is decomposed into three phases:

1. We compute a set of boxes $\mathcal{F}$ (called *boundary Free boxes*) that are proved not to intersect the boundary of $\Sigma(f, \mathbf{y}^{Init})$ (Section 4). The remaining boxes (called *Weak boundary boxes*) are put in the list $\mathcal{W}$.
2. We classify the *boundary free boxes* into outer and inner boxes (Section 5). Unknown boxes can appear here but this is very unlikely.
3. We focus on the *weak boundary boxes* and classify them into inner boxes or unknown boxes (Section 6).

This is summarized in Algorithm 1 which is built from functions that are described in the next sections.

4 Computation of Boundary Free Boxes

The first phase consists of computing two finite sets of boxes $\mathcal{F}$ and $\mathcal{W}$ (all subsets of $\mathbf{x}^{Init}$) such that:

1. $\cup(\mathcal{F} \cup \mathcal{W}) = \mathbf{x}^{Init}$ and the boxes of $\mathcal{F} \cup \mathcal{W}$ do not overlap except perhaps over their boundaries;

```
Algorithm 1: Approximate($f, \mathbf{x}^{Init}, \mathbf{y}^{Init}, \epsilon$)
    Input: $f$ (from $\mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ to $\mathbb{R}^m$), $\mathbf{x}^{Init} \in \mathbb{IR}^{n_x}$, $\mathbf{y}^{Init} \in \mathbb{IR}^{n_y}$, $\epsilon \in \mathbb{R}^+$
    Output: ($\mathcal{I}$, $\mathcal{O}$, $\mathcal{U}$) (triplet of finite sets of boxes in $\mathbb{IR}^{n_x}$)
1   $(\mathcal{F}, \mathcal{W})$ = BoundaryFreeBoxes($f, \mathbf{x}^{Init}, \mathbf{y}^{Init}, \epsilon$);
2   $(\mathcal{I}', \mathcal{O}, \mathcal{W}')$ = ClassifyBoundaryFreeBoxes($f, \mathcal{F}, \mathbf{y}^{Init}, \epsilon$);
3   $(\mathcal{I}'', \mathcal{U})$ = ClassifyWeakBoundaryBoxes($f, \mathcal{W} \cup \mathcal{W}', \mathbf{y}^{Init}, \epsilon$);
4   $\mathcal{I} = \mathcal{I}' \cup \mathcal{I}''$;
5   return ($\mathcal{I}$, $\mathcal{O}$, $\mathcal{U}$);
```

2. Each box of $\mathcal{F}$ does not intersect $\mathbf{x}^{Init} \cap \partial\Sigma(f, \mathbf{y}^{Init})$ (these boxes are called *boundary free boxes*); therefore, $\mathbf{x}^{Init} \cap \partial\Sigma(f, \mathbf{y}^{Init})$ is included in $\cup\mathcal{W}$.

The boxes from $\mathcal{F}$ will be classified into inner or outer boxes in Section 5 while the boxes from $\mathcal{W}$ will be classified into inner or unknown boxes in Section 6.

Algorithm 2 computes the wanted sets of boxes. The two functions that are used in Algorithm 2 are described in the rest of the section. First of all,

```
Algorithm 2: BoundaryFreeBoxes($f, \mathbf{x}^{Init}, \mathbf{y}^{Init}, \epsilon$)
    Input: $f$ (from $\mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ to $\mathbb{R}^m$), $\mathbf{x}^{Init} \in \mathbb{IR}^{n_x}$, $\mathbf{y}^{Init} \in \mathbb{IR}^{n_y}$, $\epsilon \in \mathbb{R}^+$
    Output: ($\mathcal{F}$, $\mathcal{W}$) (couple of finite sets of boxes in $\mathbb{IR}^{n_x}$)
1   $\mathcal{U} \leftarrow$ BranchAndPrune( $(\mathbf{x}^{Init}, \mathbf{y}^{Init}), \mathcal{C}, \epsilon$) where $\mathcal{C}$ is given by (6);
2   $(\mathcal{F}, \mathcal{W}) \leftarrow$ Projection($\mathbf{x}^{Init}, \mathcal{U}$);
3   return $(\mathcal{F}, \mathcal{W})$;
```

Subsection 4.1 presents the basic test that will be used to characterize boundary free boxes. The computations performed at Line 1 are described in Subsection 4.2 while the computations performed at Line 2 are described in Subsection 4.3.

4.1 Basic Test

Our algorithm is based on the study of the relative position of boxes w.r.t. the boundary of $\Sigma(f, \mathbf{y}^{Init})$. The following theorem will play a key role in this approach.

Theorem 1. *Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \longrightarrow \mathbb{R}^m$ be a continuously differentiable function and $\tilde{x} \in \mathbb{R}^{n_x}$ be an arbitrary vector and $\mathbf{y} \in \mathbb{IR}^{n_y}$ bounded and nonempty. For $x \in \mathbb{R}^{n_x}$ and $y \in \mathbb{R}^{n_y}$ define the matrix $M_{f,\mathbf{y}}(x, y)$ in the following way:*

$$\Big(M_{f,\mathbf{y}}(x,y)\Big)_{ij} := \begin{cases} \frac{\partial f_i}{\partial y_j}(x,y) \text{ if } y_j \in \operatorname{int} \mathbf{y}_j \\ 0 \quad \text{otherwise.} \end{cases} \tag{4}$$

Then $\tilde{x} \in \partial\Sigma(f, \mathbf{y})$ implies

$$\big(\exists y \in \mathbf{y}\big)\big(f(\tilde{x}, y) = 0 \wedge \operatorname{rank} M_{f,\mathbf{y}}(\tilde{x}, y) < m\big). \tag{5}$$

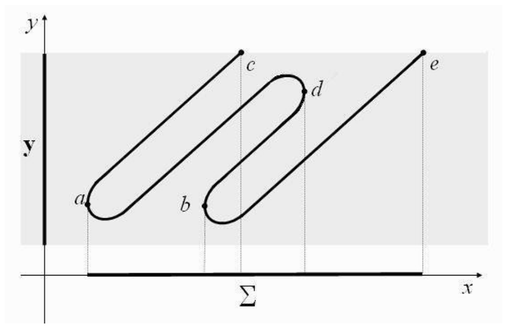

Fig. 1.

Proof. Provided in Appendix A.1.

Theorem 1 is a generalization of Equation (6) in [11]. Theorem 1 is more efficient in a constraint framework: it can deal with bounded domains directly. The technique proposed in [11] needs a change of variables that introduces sin and cos functions, hence leading to less efficient computations.

Example 1. Consider a function $f : \mathbb{R}\times\mathbb{R} \longrightarrow \mathbb{R}$ whose implicit graph $f(x, y) = 0$ is plotted on Figure 1. The graph is restricted to $\mathbb{R} \times \mathbf{y}$ so its projection on the x-axis equals $\Sigma(f, \mathbf{y})$. We displayed the vectors a, b, c, d and e which satisfy the condition $\mathsf{rank}\, M_{f,\mathbf{y}}(x, y) < 1$: first, in the case of a vector $(x, y) \in \{a, b, d\}$, we have $\frac{\partial f}{\partial y}(x, y) = 0$ and therefore $\mathsf{rank}\, M_{f,\mathbf{y}}(x, y) = 0$. Second, in the case of a vector $(x, y) \in \{c, e\}$, the component y is on the boundary of $\mathbf{y}$ and therefore $M_{f,\mathbf{y}}(x, y)$ is set to zero by definition and finally $\mathsf{rank}\, M_{f,\mathbf{y}}(x, y) = 0$. As one can see on Figure 1, the boundary of $\Sigma(f, \mathbf{y})$ is included in the projection of $\{a, b, c, d, e\}$.

Definition 1. *With the notations introduced in Theorem 1, vectors* $(x, y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ *that satisfy* $f(x, y) = 0$ *and* $\mathsf{rank}\, M_{f,\mathbf{y}}(x, y) < m$ *are called* singular vectors of f in $\mathbf{y}$ *(or simply singular vectors as no confusion is possible here). Then, the* weak boundary *of* $\Sigma(f, \mathbf{y})$ *is defined as the projection of the set of singular vectors into the* x*-space.*

With these definitions, Theorem 1 is simply stated saying that the boundary of $\Sigma(f, \mathbf{y})$ is included inside its weak boundary. In Example 1, the singular vectors are $\{a, b, c, d, e\}$. The projection of these vectors is the weak boundary of $\Sigma(f, \mathbf{y})$, and it actually contains $\partial\Sigma(f, \mathbf{y})$.

The computation of boundary free boxes $\mathcal{F}$ is done in two steps: a branch and prune algorithm is used to construct an outer approximation of the set of singular vectors (Subsection 4.2), and then this set is projected in order to provide a rigorous outer approximation of the weak boundary of $\Sigma(f, \mathbf{y})$ (Subsection 4.3).

4.2 Outer Approximation of the Set of Singular Vectors

An outer approximation of the set of singular vectors is computed using a basic branch and prune algorithm. This algorithm is described in Algorithm 3. The function $\mathsf{prune}\big(\tilde{\mathbf{u}}, c(u)\big)$ is often called a contractor and returns a new box $\tilde{\mathbf{u}}' \subseteq \tilde{\mathbf{u}}$ such that $\big(\forall u \in \mathbf{u}\big)\big(c(u) \Rightarrow u \in \mathbf{u}'\big)$. This algorithm is well known and $\cup\mathsf{BranchAndPrune}\big((\mathbf{x}^{Init}, \mathbf{y}^{Init}), \mathcal{C}, \epsilon\big)$, where

$$\mathcal{C} = \{\ f(x,y) = 0\ ,\ \mathsf{rank}\, M_{f,\mathbf{y}^{Init}}(x,y) < m\ \}, \tag{6}$$

is obviously an outer approximation of the set of singular vectors of f in $(\mathbf{x}^{Init}, \mathbf{y}^{Init})$.

Algorithm 3: $\mathsf{BranchAndPrune}(\mathbf{u}, \mathcal{C}, \epsilon)$

```
   Input: $\mathbf{u} \in \mathbb{IR}^n$ , $\mathcal{C}$ (finite set of n-ary constraints) , $\epsilon \in \mathbb{R}^+$
   Output: $\mathcal{U} \subseteq \mathbb{IR}^n$
 1 $\mathcal{L} \leftarrow \{\mathbf{u}\}$;
 2 while $\mathcal{L} \neq \emptyset$ do
 3     $\tilde{\mathbf{u}} \leftarrow \mathsf{Extract}(\mathcal{L})$;
 4     if $||\mathsf{wid}\,\tilde{\mathbf{u}}|| \geq \epsilon$ then
 5         foreach $c \in \mathcal{C}$ do
 6             $\tilde{\mathbf{u}} \leftarrow \mathsf{Prune}(\tilde{\mathbf{u}}, c(u))$;
 7         end
 8         if $\tilde{\mathbf{u}} \neq \emptyset$ then
 9             $\mathcal{L} \leftarrow \mathcal{L} \cup \mathsf{Bisect}(\tilde{\mathbf{u}})$;
10         end
11     else
12         $\mathcal{U} \leftarrow \mathcal{U} \cup \{\tilde{\mathbf{u}}\}$;
13     end
14 end
15 return $\mathcal{U}$;
```

Usually, the function $\mathsf{extract}(\mathcal{L})$ extracts the box that has the largest $||\mathsf{wid}\,\tilde{\mathbf{u}}||$. This presents the advantage that the search is performed uniformly in the search space. Extracting the box that has the smallest $||\mathsf{wid}\,\tilde{\mathbf{u}}||$ leads to a deep-first algorithm. This latter algorithm is well suited for a quick search of one approximate solution and will be used in Section 5. The function bisect must bisect fairly, meaning that each component is regularly bisected. A widely used bisection strategy is to bisect the largest component of the box, hence ensuring convergence.

Remaining is to describe the contractors that will be used for each of the two involved constraints. The contractor $\mathsf{prune}\big((\mathbf{x},\mathbf{y}), f(x,y) = 0\big)$ can be implemented using the usual constraint satisfaction techniques (cf. [10, 12, 13]). The contractor $\mathsf{prune}\big((\mathbf{x},\mathbf{y}), \mathsf{rank}\, M_{f,\mathbf{y}}(x,y) < m\big)$ is implemented using the interval Gauss elimination algorithm (cf. [14]). First, we need an interval evaluation of

$M_{f,\mathbf{y}}(x,y)$. For $\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}} \in \mathbb{IR}^n$, $\mathbf{y} \subseteq \tilde{\mathbf{y}}$, let us define $\mathbf{M}_{f,\tilde{\mathbf{y}}}(\mathbf{x},\mathbf{y})$ in the following way:

$$\Big(\mathbf{M}_{f,\tilde{\mathbf{y}}}(\mathbf{x},\mathbf{y})\Big)_{ij} := \begin{cases} \frac{\partial \mathbf{f}_i}{\partial y_j}(\mathbf{x},\mathbf{y}) \text{ if } \ \mathbf{y}_j \subseteq \mathsf{int}\,\tilde{\mathbf{y}}_j \\ \quad 0 \qquad \text{otherwise,} \end{cases} \tag{7}$$

where $\frac{\partial \mathbf{f}_i}{\partial y_j}(\mathbf{x},\mathbf{y})$ is an interval evaluation of $\frac{\partial f_i}{\partial y_j}(x,y)$. With this definition, we obviously have $M_{f,\tilde{\mathbf{y}}}(x,y) \in \mathbf{M}_{f,\tilde{\mathbf{y}}}(\mathbf{x},\mathbf{y})$ for all $(x,y) \in (\mathbf{x},\mathbf{y})$. We can therefore define

$$\mathsf{prune}\big((\mathbf{x},\mathbf{y}), \mathsf{rank}\, M_{f,\tilde{\mathbf{y}}}(x,y) < m\big) := \begin{cases} \emptyset & \text{if } \mathsf{GaussElim}\big(\mathbf{M}_{f,\tilde{\mathbf{y}}}(\mathbf{x},\mathbf{y})\big) \\ (\mathbf{x},\mathbf{y}) & \text{otherwise,} \end{cases}$$

where the function $\mathsf{GaussElim}(\mathbf{M})$ returns true if and only if the interval Gauss elimination algorithm succeeds in proving that $\mathbf{M}$ has full rank.

4.3 Projection in the x-Space

Algorithm 4 computes the sets of boxes $\mathcal{F}$ (boundary free boxes) and $\mathcal{W}$ (weak boundary boxes) using the outer approximation $\mathcal{U}$ of the singular vectors computed in the previous subsection. We can display two points:

– Line 6: a box $\tilde{\mathbf{x}}$ is put in $\mathcal{F}$ only if $\big(\forall (\tilde{\mathbf{x}}', \tilde{\mathbf{y}}') \in \mathcal{U}\big)\big(\tilde{\mathbf{x}} \cap \tilde{\mathbf{x}}' = \emptyset\big)$. Because $\cup\mathcal{U}$ is an outer approximation of the set of singular vectors of $\Sigma(f, \mathbf{y}^{Init})$, this proves that $\tilde{\mathbf{x}}$ does not contain any projection of some singular vectors. Hence, $\tilde{\mathbf{x}}$ does not intersect the weak boundary of $\Sigma(f, \mathbf{y}^{Init})$ and finally does not intersect $\partial\Sigma(f, \mathbf{y}^{Init})$ neither.
– A box $\tilde{\mathbf{x}}$ is either put in $\mathcal{F}$ (Line 6) or in $\mathcal{W}$ (Line 8) or bisected (Line 11). Therefore, we have $\cup(\mathcal{F} \cup \mathcal{W}) = \tilde{\mathbf{x}}$, and hence $\partial\Sigma(f, \mathbf{y}^{Init}) \subseteq (\cup\mathcal{W})$.

The efficiency of Algorithm 4 can be drastically improved by keeping track of the tests performed at Line 5 in order to avoid useless comparisons, but the details are not presented here.

5 Classification of Boundary Free Boxes

In this section, we consider a finite set of boundary free boxes $\mathcal{F}$, i.e. boxes that are proved not to intersect $\partial\Sigma(f, \mathbf{y}^{Init})$. We aim to classify these boxes $\mathbf{x}$ into inner boxes, i.e. $\mathbf{x} \subseteq \Sigma(f, \mathbf{y}^{Init})$, and outer boxes, i.e. $\mathbf{x} \cap \Sigma(f, \mathbf{y}^{Init}) = \emptyset$.

As $\mathbf{x} \in \mathcal{F}$ does not intersect the boundary of $\Sigma(f, \mathbf{y}^{Init})$, we can study a simpler problem focusing on one arbitrary vector inside the box $\mathbf{x}$. This is formalized by Proposition 1.

Proposition 1. *Let $\mathbf{x} \in \mathbb{IR}^n$ and E be a closed subset of $\mathbb{R}^n$ such that $\mathbf{x} \cap \partial E = \emptyset$. Then $\mathbf{x} \cap E \neq \emptyset \Longrightarrow \mathbf{x} \subseteq E$, or equivalently $\mathbf{x} \not\subseteq E \Longrightarrow \mathbf{x} \cap E = \emptyset$.*

Proof. Provided in Appendix A.2.

Algorithm 4: $\mathsf{Projection}(\mathbf{x},\mathcal{U},\epsilon)$

```
Input: $\mathbf{x} \in \mathbb{IR}^{n_x}$, $\mathcal{U}$ (finite set of boxes in $\mathbb{IR}^{n_x} \times \mathbb{IR}^{n_y}$), $\epsilon \in \mathbb{R}^+$
Output: $(\mathcal{F}, \mathcal{W})$ (couple of finite sets of boxes in $\mathbb{IR}^{n_x}$)
1  $\mathcal{L} \leftarrow \{\mathbf{x}\}$;
2  while $\mathcal{L} \neq \emptyset$ do
3      $\tilde{\mathbf{x}} \leftarrow \mathsf{Extract}(\mathcal{L})$;
4      if $||\mathsf{wid}\,\tilde{\mathbf{x}}|| \geq \epsilon$ then
5          if $\big(\forall(\tilde{\mathbf{x}}',\tilde{\mathbf{y}}') \in \mathcal{U}\big)\big(\tilde{\mathbf{x}} \cap \tilde{\mathbf{x}}' = \emptyset\big)$ then
6              $\mathcal{F} \leftarrow \mathcal{F} \cup \{\tilde{\mathbf{x}}\}$;
7          else
8              $\mathcal{L} \leftarrow \mathcal{L} \cup \mathsf{Bisect}(\tilde{\mathbf{x}})$;
9          end
10     else
11         $\mathcal{W} \leftarrow \mathcal{W} \cup \{\tilde{\mathbf{x}}\}$;
12     end
13 end
14 return $(\mathcal{F},\mathcal{W})$;
```

This proposition has two interesting consequences. First, given a box $\mathbf{x} \in \mathcal{F}$, we can now focus on one arbitrary vector inside $\mathbf{x}$. The problem will now be to decide if $\big(\exists y \in \mathbf{y}^{Init}\big)\big(f(\mathsf{mid}\,\mathbf{x}, y) = 0\big)$ is true or not, instead of having to decide if $\big(\forall x \in \mathbf{x}\big)\big(\exists y \in \mathbf{y}^{Init}\big)\big(f(x, y) = 0\big)$ is true or not. Second, when a box $\mathbf{x}$ is proved to be inside or outside $\Sigma(f, \mathbf{y}^{Init})$, the same property holds for all boxes $\mathbf{x}' \in \mathcal{F}$ such that $\mathbf{x} \cap \mathbf{x}' \neq \emptyset$. This last remark is able to strongly accelerate the computations but it is not explicitly described in Algorithm 5.

The next proposition is an existence test that allows to check the existence of a solution to the system of equations $g(y) = 0$. In our context it will be used with $g(y) = f(\mathsf{mid}\,\mathbf{x}, y)$. This existence test is new, and should be compared to the usual existence tests (e.g. Moore-Kioustelidis). Proposition 2 presents two advantages over the usual existence tests: first it does not need any preconditioning. Second, it can be applied to under-constrained systems of equations. We use it here for these two reasons.

Proposition 2. *Let $g : \mathbb{IR}^n \longrightarrow \mathbb{IR}^m$ be a continuously differentiable function and $\mathbf{y} \in \mathbb{IR}^n$ be a bounded nonempty box. Consider a box $\mathbf{z}$ such that $0 \in \mathbf{z}$ and $g^{-1}(\mathbf{z}) \cap \mathbf{y} \neq \emptyset$.*[3] *Suppose that*

$$\{(y, z) \in (\mathbf{y}, \mathbf{z}) \mid g(y) = z \ \wedge\ \mathsf{rank}\, M_{g,\mathbf{y}}(y) < m\} = \emptyset, \tag{8}$$

where

$$\Big(M_{g,\mathbf{y}}(y)\Big)_{ij} := \begin{cases} \frac{\partial g_i}{\partial y_j}(y) \text{ if } y_j \in \mathsf{int}\,\mathbf{y}_j \\ 0 \quad \text{otherwise.} \end{cases}$$

Then there exists $y \in \mathbf{y}$ such that $g(y) = 0$.

[3] The box $\mathbf{z} := 0 \vee g(\tilde{y})$, where $\tilde{y} \in \mathbf{y}$ is an approximate solution of $g(y) = 0$, is both efficient and easy to compute.

Proof. Provided in Appendix A.2.

Being given a box $\mathbf{x}$ in addition to the initial box $\mathbf{y}^{Init}$, we define a set of constraints that correspond to the statement of Proposition 2:

$$\mathcal{C} = \{\ f(\mathsf{mid}\,\mathbf{x}, y) = z\ ,\ \mathsf{rank}\, M_{f,\mathbf{y}^{Init}}(y) < m\ \}. \tag{9}$$

We can now propose Algorithm 5 that classifies boundary free boxes into inner boxes and outer boxes (and possibly unknown boxes). Lines preceded by "Com." are commented bellow:

Algorithm 5: ClassifyBoundaryFreeBoxes($f, \mathcal{F}, \mathbf{y}, \epsilon$)

```
        Input: $f$ (from $\mathbb{R}^{n_x} \times \mathbb{R}^{n_y}$ to $\mathbb{R}^m$), $\mathcal{F}$ (finite sets of boxes in $\mathbb{IR}^{n_x}$), $\mathbf{y} \in \mathbb{IR}^{n_y}$, $\epsilon \in \mathbb{R}^+$
        Output: ($\mathcal{I}$, $\mathcal{O}$,$\mathcal{U}$) (triplet of finite sets of boxes in $\mathbb{IR}^{n_x}$)
      1 while $\mathcal{F} \neq \emptyset$ do
      2     $\mathbf{x} \leftarrow$ Extract($\mathcal{F}$);
 Com. 3     $\mathcal{L} \leftarrow$ BranchAndPrune( $\mathbf{y}$ , $\{f(\mathsf{mid}\,\mathbf{x}, y) = 0\}$ , $\epsilon$);
 Com. 4     if $\mathcal{L} = \emptyset$ then
      5         $\mathcal{O} = \mathcal{O} \cup \{\mathbf{x}\}$;
      6     else
      7         $\mathbf{y}$ = Extract($\mathcal{L}$);
 Com. 8         $\mathbf{z} \leftarrow 0 \vee f(\mathsf{mid}\,\mathbf{x}, \mathsf{mid}\,\mathbf{y})$;
      9         $\mathcal{L}' \leftarrow$ BranchAndPrune( $(\mathbf{y}, \mathbf{z})$ , $\mathcal{C}$ , $\epsilon$);
     10         if $\mathcal{L}' = \emptyset$ then
     11             $\mathcal{I} = \mathcal{I} \cup \{\mathbf{x}\}$;
     12         else
     13             $\mathcal{U} = \mathcal{U} \cup \{\mathbf{x}\}$;
     14         end
     15     end
     16 end
     17 return ($\mathcal{I}, \mathcal{O}, \mathcal{U}$);
```

– Line 3: the branch and prune algorithm must either prove the emptiness or provide one approximate solution; therefore, it is modified to a deep-first search algorithm.
– Line 4: if $\mathcal{L}$ is empty then the branch and prune algorithm has proved $\mathsf{mid}\,\mathbf{x} \notin \Sigma(f, \mathbf{y}^{Init})$. Because $\mathbf{x}$ is supposed to be a boundary free box, Proposition 1 then proves that $\mathbf{x} \cap \Sigma(f, \mathbf{y}^{Init}) = \emptyset$.
– Line 8: $f(\mathsf{mid}\,\mathbf{x}, \mathsf{mid}\,\mathbf{y})$ is computed using interval arithmetic and therefore leads to an interval that rigorously contains the image of $(\mathsf{mid}\,\mathbf{x}, \mathsf{mid}\,\mathbf{y})$. The interval vector $\mathbf{z}$ is the join (or interval hull) of the latter interval vector and 0, and it therefore contains both 0 and the image of $(\mathsf{mid}\,\mathbf{x}, \mathsf{mid}\,\mathbf{y})$. Hence, $\mathbf{z}$ is a good interval vector to use in Proposition 2.

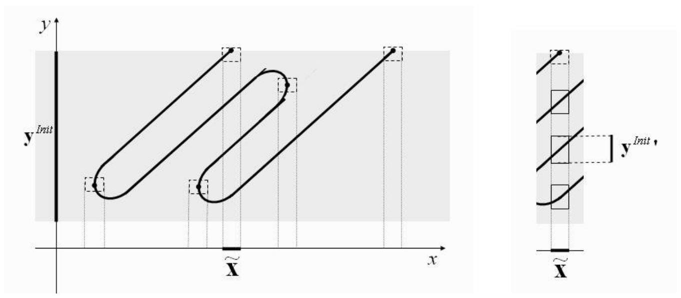

Fig. 2.

6 Classification of Weak Boundary Boxes

We consider a box $\mathbf{x}$ that was not proved to be a boundary free box. The idea is to consider a stronger problem $\Sigma(f, \mathbf{y}^{Init\prime})$ with $\mathbf{y}^{Init\prime} \subseteq \mathbf{y}^{Init}$. Then, the weak boundary of $\Sigma(f, \mathbf{y}^{Init\prime})$ is certainly different from the one of $\Sigma(f, \mathbf{y}^{Init})$, and the box $\mathbf{x}$ is hopefully a boundary free box for the new problem. So, we will use the algorithms presented in the previous sections to handle the new problem $\Sigma(f, \mathbf{y}^{Init\prime})$.

Let us illustrate this technique with an example. Consider the quantified constraint represented on Figure 1. The left-hand side graphic of Figure 2 displays the unknown boxes generated at Line 1 of Algorithm 2. These boxes contain the singular vectors of $\Sigma(f, \mathbf{y}^{Init})$. The projection of these boxes forms the weak boundary. The right-hand side graphic focuses on one box $\tilde{\mathbf{x}}$ among the weak boundary box. The four boxes of the right hand side graphic are obtained using the branch and prune algorithm to prune the constraint $f(x, y) = 0$ with $x \in \tilde{\mathbf{x}}$ and $y \in \mathbf{y}^{Init}$. We can now easily pick up a box $\mathbf{y}^{Init\prime}$ where no singularity occurs, and the algorithms presented in the previous section are likely to succeed in proving that it is an inner box.

In practice, however, the difference between singular boxes and nonsingular boxes is not as clearly identified as on Figure 2. Although finding an efficient heuristic to compute a new initial box $\mathbf{y}^{Init\prime}$ is one important forthcoming work, the experimentations presented in the next section show that this simple heuristic is already useful.

The process described in this section leads to a function

$$\mathsf{ClassifyWeakBoundaryBoxes}(f, \mathcal{W}, \mathbf{y}^{Init}, \epsilon)$$

that returns a couple $(\mathcal{I}, \mathcal{U})$. The set of boxes $\mathcal{I}$ contains the boxes that where proved to be inside $\Sigma(f, \mathbf{y}^{Init\prime}) \subseteq \Sigma(f, \mathbf{y})$. The set of boxes $\mathcal{U}$ contains the boxes we were not able to prove anything about.

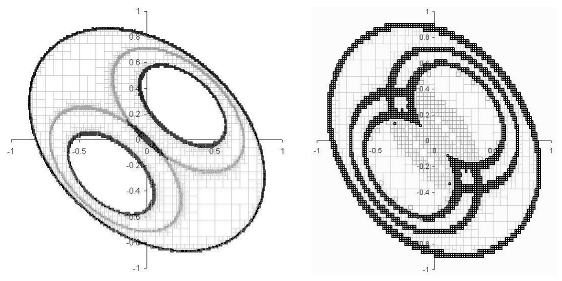

Fig. 3.

7 Experimentations

The application of the algorithm to three examples is now presented. No comparison is provided as our algorithm is the first numerical algorithm that is able to compute the inner approximations proposed in this section. Formal quantifier elimination (cf. [1]) can be applied to the first two examples. The results obtained by our algorithm are similar to the one obtained by formal quantifier elimination.

7.1 Well-Constrained Academic Problem

The first problem is defined by

$$f(x,y) = \begin{pmatrix} x_1^2 + x_2^2 + y_1^2 + y_2^2 - 1 \\ x_1 + x_2 + y_1 + y_2 \end{pmatrix} \; ; \; \mathbf{x}^{Init} = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \; ; \; \mathbf{y}^{Init} = \begin{pmatrix} [-0.7,0.7] \\ [-0.8,0.8] \end{pmatrix}.$$

The pavings plotted in the left hand side graphic of Figure 3 are obtained after one minute using a precision $\epsilon = 0.01$. Inner boxes are in gray (light gray for boundary free boxes and dark gray for weak boundary boxes that have been proved to be inner boxes) and unknown boxes are in black. The algorithm behaves very well with this example and provides a good inner approximation.

7.2 Under-Constrained Academic Problem

The second problem is defined by

$$f(x,y) = \begin{pmatrix} x_1^2 + x_2^2 + y_1^2 + y_2^2 + y_3^2 - 1 \\ x_1 + x_2 + y_1 + y_2 + y_3 \end{pmatrix} \; ; \; \mathbf{x}^{Init} = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix} \; ; \; \mathbf{y}^{Init} = \begin{pmatrix} [-0.7,0.7] \\ [-0.8,0.8] \\ [-2,2] \end{pmatrix}.$$

The pavings plotted in right hand side graphic of Figure 3 are obtained after 15 minutes using a precision $\epsilon = 0.02$. The heuristic presented in Section 6 for

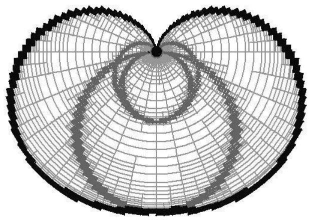

Fig. 4.

the classification of weak boundary boxes is clearly not efficient for this example: it is missing some parts of the weak boundary while 99% of the computations are spent working on the weak boundary. This will have to be investigated; however, the algorithm presents a good behavior for the computation and classification of boundary free boxes.

7.3 Speed Diagram of a Sailboat

The third problem was proposed in [5]. It is defined by

$$f(v,\theta,\delta_r,\delta_s) = \begin{pmatrix} \alpha_s(V\cos(\theta+\delta_s) - v\sin\delta_s)\sin\delta_s - \alpha_r v\sin^2\delta_r - \alpha_f v \\ (l - r_s\cos\delta_s)\alpha_s(V\cos(\theta+\delta_s) - v\sin\delta_s) - r_r\alpha_r v\sin\delta_r\cos\delta_r \end{pmatrix},$$

where the following values are chosen for parameters: $\alpha_s = 500$, $\alpha_r = 300$, $\alpha_f = 60$, $V = 10$, $r_s = 1$, $r_r = 2$, and $l = 1$. The initial domains are $\mathsf{Dom}(v) = [0,20]$, $\mathsf{Dom}(\theta) = [-\pi,\pi]$, $\mathsf{Dom}(\delta_r) = [-\pi/2,\pi/2]$ and $\mathsf{Dom}(\delta_r) = [-\pi/2,\pi/2]$. The graph to be approximated is therefore

$$\Big\{(v,\theta) \in ([0,20],[-\pi,\pi]) \mid \big(\exists\delta_r \in [-\frac{\pi}{2},\frac{\pi}{2}]\big)\big(\exists\delta_s \in [-\frac{\pi}{2},\frac{\pi}{2}]\big)\big(f(v,\theta,\delta_r,\delta_s) = 0\big)\Big\}.$$

This set corresponds to the speed v and angle θ w.r.t. the wind that can be reached for some command δ_r and δ_s in their domains. An inner approximation was computed in [5] after a specific formal simplification of the problem. The pavings plotted in Figure 4 are obtained after 10 minutes using a precision $\epsilon = 0.01$. We obtain the same results as in [5] but without any preliminary formal simplification. Our algorithm is slower, but it works in a 4 dimensional space (while the simplification used in [5] decreases the dimension by one), and it was not yet optimized.

8 Conclusion

We have presented the first numerical algorithm that is able to compute an inner approximation (and obviously an outer approximation) of the graph of an existentially quantified constraint with an arbitrary number of equality constraints.

Although some previous works were dedicated to the inner approximation of such constraints in some special cases, the algorithm we proposed can be applied for arbitrary numbers of equalities and existentially quantified parameters.

The idea consisting of using a branch and prune algorithm to approximate the boundary of the constraint graph instead of the constraint graph itself seems to be new. Not only does it allow simplification of the problem to be solved, but it should also make the algorithm accumulate on this boundary and therefore lead to efficient computations. Timings on presented examples are reasonable but not yet good. However, no optimization has been done in order to present the concepts clearly. We expect some strong efficiency improvements in the next implementations of the algorithm. Finally, one advantage of the proposed method is that it relies only on a simple standard branch and prune algorithm. Therefore, any future improvement for pruning operators will be useful for our algorithm.

Convergence of the algorithm remains to be studied. This convergence strongly depends on the heuristic used to deal with weak boundary boxes. Actually, the simple heuristic proposed in Section 6 showed is usefulness but can be certainly improved. In particular, experimentations showed it was not very efficient for under-constrained problems. Therefore, a heuristic that makes the algorithm convergent will have to be found.

A Proofs

A.1 Proof of Theorem 1

Lemma 1. *Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \longrightarrow \mathbb{R}^m$ be a continuous function and $\mathbf{y} \in \mathbb{IR}^{n_y}$ be bounded and nonempty. Then $\Sigma(f, \mathbf{y})$ is closed in $\mathbb{R}^{n_x}$.*

Proof. Consider a sequence $x_n \in \Sigma(f, \mathbf{y})$ that converges to $\tilde{x}$. We have to prove $\tilde{x} \in \Sigma(f, \mathbf{y})$. As $x_n \in \Sigma(f, \mathbf{y})$, there exists $y_n \in \mathbf{y}$ such that $f(x_n, y_n) = 0$. As $\mathbf{y}$ is bounded, the Bolzano-Weierstrass theorem proves that the sequence y_n has at least one accumulation point in $\mathbf{y}$. Let us denote one of these accumulation points $\tilde{y}$. We can pick up a subsequence $y_{\pi(n)}$ that converges to $\tilde{y}$. The sequence $x_{\pi(n)}$ obviously converges to $\tilde{x}$. Therefore, $\lim_{n\to\infty} f(x_{\pi(n)}, y_{\pi(n)}) = f(\tilde{x}, \tilde{y}) = 0$. We proved that there exists $\tilde{y} \in \mathbf{y}$ such that $f(\tilde{x}, \tilde{y}) = 0$ and hence $\tilde{x} \in \Sigma(f, \mathbf{y})$. □

Lemma 2. *Let $\mathbf{y} \in \mathbb{IR}^n$. If $y \in \partial\mathbf{y}$ then there exists $i \in [1..n]$ such that $y_i \in \partial\mathbf{y}_i$.*

Proof. We prove the contrapose. Suppose for all $i \in [1..n]$ we have $y_i \in \operatorname{int} \mathbf{y}_i$. That is, for all $i \in [1..n]$ we have $\inf \mathbf{y}_i < y_i < \sup \mathbf{y}_i$. This proves $y \in \operatorname{int} \mathbf{y}$. □

Proof of Theorem 1. We prove the contrapositive of the statement. There are only two possible cases: $\big(\forall y \in \mathbf{y}\big)\big(f(\tilde{x}, y) \neq 0\big)$ or $\big(\exists y \in \mathbf{y}\big)\big(f(\tilde{x}, y) = 0\big)$. Let us consider both cases. In the first case, we have $\tilde{x} \notin \Sigma(f, \mathbf{y})$, and because $\Sigma(f, \mathbf{y})$ is closed in $\mathbb{R}^{n_x}$ by Lemma 1, we have $\tilde{x} \notin \partial\Sigma(f, \mathbf{y})$. The second case relies on the implicit function theorem. Consider $\tilde{y} \in \mathbf{y}$ such that $f(\tilde{x}, \tilde{y}) = 0$. By hypothesis, we have $\mathsf{rank}\, M_{f,\mathbf{y}}(\tilde{x},\tilde{y}) = m$ (which implies $n_y \geq m$). Therefore, there exists a set of indices $\mathcal{E} := \{e_1, \cdots, e_m\}$ such that $\det M \neq 0$ where $M \in \mathbb{R}^{m\times m}$ is defined by $M_{ij} := \big(M_{f,\mathbf{y}}(\tilde{x},\tilde{y})\big)_{ie_j}$.

Claim: $\tilde{y}_{\mathcal{E}} \in \mathsf{int}\ \mathbf{y}_{\mathcal{E}}$. The claim is proved by contradiction. Suppose that $\tilde{y}_{\mathcal{E}} \in \partial\mathbf{y}_{\mathcal{E}}$, therefore by Lemma 2 there exists $e_j \in \mathcal{E}$ such that $\tilde{y}_{e_j} \in \partial\mathbf{y}_{e_j}$. Then by definition of $M_{f,\mathbf{y}}(\tilde{x},\tilde{y})$ we have $M_{ij} = 0$ for all $i \in [1..n]$. Therefore M is singular which is absurd because $\det M \neq 0$ by hypothesis.

Now define $g : \mathbb{R}^{n_x}\times\mathbb{R}^m \longrightarrow \mathbb{R}^m$ by $g(x, y_{\mathcal{E}}) = f(x, y)$ where $y_{[1..n]\setminus\mathcal{E}}$ is fixed to $\tilde{y}_{[1..n]\setminus\mathcal{E}}$. With this definition, we have $\frac{\partial g_i}{\partial y_{e_j}}(\tilde{x},\tilde{y}_{\mathcal{E}}) = M_{ij}$. As M is nonsingular, we can apply the implicit function theorem that proves the existence of

(i) some open sets $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ and $\mathbb{Y} \subseteq \mathbb{R}^m$ that contain respectively $\tilde{x}$ and $\tilde{y}_{\mathcal{E}}$;
(ii) a continuously differentiable function $\phi : \mathbb{X} \longrightarrow \mathbb{Y}$ such that $\phi(\tilde{x}) = \tilde{y}_{\mathcal{E}}$;
(iii) $y_{\mathcal{E}} = \phi(x)$ implies $g(x, y_{\mathcal{E}}) = 0$ for any $x \in \mathbb{X}$.

Now define $\mathbb{Y}' = \mathbb{Y} \cap (\mathsf{int}\,\mathbf{y}_{\mathcal{E}})$ which is open because it is the intersection of two open sets. As $\tilde{y}_{\mathcal{E}} \in \mathsf{int}\,\mathbf{y}_{\mathcal{E}}$ and $\tilde{y}_{\mathcal{E}} \in \mathbb{Y}$ we have $\tilde{y}_{\mathcal{E}} \in \mathbb{Y}'$. As ϕ is continuous the preimage $\mathbb{X}' := \phi^{-1}(\mathbb{Y}')$ is also open. Furthermore $\tilde{x} \in \mathbb{X}'$ because $\tilde{y}_{\mathcal{E}} \in \mathbb{Y}'$ and $\phi(\tilde{x}) = \tilde{y}_{\mathcal{E}}$.

For all $x \in \mathbb{X}'$ define $y \in \mathbf{y}$ by $y_{\mathcal{E}} := \phi(x)$ and $y_{[1..n]\setminus\mathcal{E}} = \tilde{y}_{[1..n]\setminus\mathcal{E}}$. Using (iii), we have $g(x, y_{\mathcal{E}}) = 0$ which implies $f(x, y) = 0$ by definition of g. We have therefore proved $\mathbb{X}' \subseteq \Sigma(f,\mathbf{y})$ which eventually proves that $\tilde{x} \in \mathsf{int}\ \Sigma(f,\mathbf{y})$. Therefore $\tilde{x} \notin \partial\Sigma(f,\mathbf{y})$.

A.2 Proofs of Proposition 1 and Proposition 2

Lemma 3. *Let E be closed in $\mathbb{R}^n$ and $x \in \mathsf{int}\ E$ and $x' \notin E$. Any continuous path connecting x to x' intersects ∂E.*

Proof. Cf. [15]. □

Proof of Proposition 1 It is sufficient to prove that the box $\mathbf{x}$ is either inside or outside of E. This is proved by contradiction: let us suppose that $\mathbf{x}$ is neither inside nor outside, i.e. there exist $x, x' \in \mathbf{x}$ such that $x \in E$ and $x' \notin E$. As $\mathbf{x}$ does not intersect ∂E, we have $x \in \mathsf{int} E$. Furthermore, $\mathbf{x}$ being path-connected, there exists a path w that is contained in $\mathbf{x}$ and which connects x and x'. Applying Lemma 3, we prove that w intersects ∂E, and therefore that $\mathbf{x}$ intersects ∂E, which is eventually absurd because we supposed $\mathbf{x} \cap \partial E = \emptyset$.

Proof of Proposition 2 We define $h(z, y) := g(y) - z$ and by definition we have $\Sigma(h,\mathbf{y}) = \big\{z \in \mathbb{R}^m \mid (\exists y \in \mathbf{y})\big(h(z, y) = 0\big)\big\}$. We will prove that $\mathbf{z} \subseteq \Sigma(h,\mathbf{y})$, which will conclude the proof because by hypothesis $0 \in \mathbf{z}$ and by definition of h, $h(0, y) = 0 \implies g(y) = 0$. As $g^{-1}(\mathbf{z}) \cap \mathbf{y} \neq \emptyset$, there exists $\tilde{y} \in \mathbf{y}$ such that $g(\tilde{y}) \in \mathbf{z}$. We have $g(\tilde{y}) \in \Sigma(h,\mathbf{y})$ because $h(g(\tilde{y}), \tilde{y}) = 0$ by definition of h. As $g(\tilde{y}) \in \mathbf{z}$ by hypothesis, we have $\mathbf{z} \cap \Sigma(h,\mathbf{y}) \neq \emptyset$. Therefore, thanks to Proposition 1, we just have to prove that $\mathbf{z} \cap \partial\Sigma(h,\mathbf{y}) = \emptyset$.

By definition of h and $M_{g,\mathbf{y}}(y)$ and $M_{h,\mathbf{y}}(z,y)$, the condition (8) is equivalent to $\{(y,z) \in (\mathbf{y},\mathbf{z}) \mid h(z,y)=0 \ \wedge\ \mathsf{rank}\, M_{h,\mathbf{y}}(z,y) < m\} = \emptyset$. As a direct consequence we obtain

$$\big\{z \in \mathbf{z} \mid \big(\exists y \in \mathbf{y}\big)\big(h(z,y)=0 \ \wedge\ \mathsf{rank}\, M_{h,\mathbf{y}}(z,y) < m\big)\big\} = \emptyset.$$

Finally, this condition validates the hypothesis of Theorem 1 which proves that $\mathbf{z} \cap \partial\Sigma(h,\mathbf{y}) = \emptyset$, hence concluding the proof.

References

1. Collins, G.: Quantifier elimination by cylindrical algebraic decomposition–twenty years of progress. In Quantifier Elimination and Cylindrical Algebraic Decomposition (1998) 8–23
2. Ratschan, S.: Uncertainty propagation in heterogeneous algebras for approximate quantified constraint solving. Journal of Universal Computer Science **6**(9) (2000) 861–880
3. Shary, S.: A new technique in systems analysis under interval uncertainty and ambiguity. Reliable computing **8** (2002) 321–418
4. Goldsztejn, A.: A Right-Preconditioning Process for the Formal-Algebraic Approach to Inner and Outer Estimation of AE-solution Sets. Reliable Computing **11**(6) (2005) 443–478
5. Herrero, P., Jaulin, L., Vehi, J., Sainz, M.: Inner and outer approximation of the polar diagram of a sailboat. Interval analysis, constraint propagation, applications, Held in conjunction with the Eleventh International Conference on Principles and Practice of Constraint Programming (CP 2005), Sitges, Spain (2005)
6. Goldsztejn, A.: A branch and prune algorithm for the approximation of non-linear ae-solution sets. In: Proceedings of the 21st ACM Symposium on Applied Computing track Reliable Computations and their Applications, Dijon, France, April 2006 (SAC 2006). (2006)
7. Khalil, H.: Nonlinear Systems, Third Edition. Prentice Hall (2002)
8. Reboulet, C.: Modélisation des robots parallèles. In Boissonat, J.D., Faverjon, B., Merlet, J.P., eds.: Techniques de la robotique, architecture et commande. Herms, Paris, France (1988) 257–284
9. Moore, R.: Interval analysis. Prentice-Hall (1966)
10. Benhamou, F., Older, W.: Applying Interval Arithmetic to Real, Integer and Boolean Constraints. Journal of Logic Programming **32**(1) (1997) 1–24
11. Haug, E., Luh, C., Adkins, F., Wang, J.: Numerical algorithms for mapping boundaries of manipulator workspaces. ASME Journal of Mechanical Design **118** (1996) 228–234
12. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer-Verlag (2001)
13. Lebbah, Y., Michel, C., Rueher, M., Daney, D., Merlet, J.: Efficient and Safe Global Constraints for handling Numerical Constraint Systems. SIAM Journal on Numerical Analysis **42**(5) (2005) 2076–2097
14. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge Univ. Press, Cambridge (1990)
15. Goldsztejn, A., Jaulin, L.: Inner Approximation of the Range of Vector-Valued Functions. (Submitted to Reliable Computing)