Actions of the hyperoctahedral group to compute minimal contractors

Luc Jaulin

Lab-Sticc, ENSTA-Bretagne

Abstract

The hyperoctahedral group B_n is the group of symmetries of the hypercube $[-1,1]^n$ of $\mathbb{R}^n$. For instance permutations, or symmetries along each of the n canonical planes of $\mathbb{R}^n$ all belong to B_n. Now, many sets of equations contain symmetries in B_n. This is the case of the addition constraint : $x_1 + x_2 = x_3$ or the multiplication $x_1 \cdot x_2 = x_3$. In robotics, many specific geometrical constraints such as for instance constraints involving distances or angles used for localization also have these symmetries. This paper shows the fundamental role of the hyperoctahedral group for interval-based methods. These methods use operators, called *contractors*, which contract axis-aligned boxes, without removing any point of the solution set defined by a conjunction of constraints (typically equations, or inequalities). More precisely, the paper presents an algorithm which allows us to build minimal contractors associated to constraints with symmetries in B_n. As an application, we will consider the geometrical constraint associated to the angle between vectors. The corresponding contractor will then be used in a constraint propagation framework in order to localize a robot using several radars.

1 Introduction

Contractor programming [7] is an efficient tool to solve rigorously complex non-linear problems involving bounded uncertainties [6] [26] [35] . It is based on the notion of *contractor* which is an operator which shrinks an axis-aligned box $[\mathbf{x}]$ of $\mathbb{R}^n$ without removing any point of the subset $\mathbb{X}$ of $\mathbb{R}^n$ to which it is associated. The set $\mathbb{X}$ is assumed to be defined by equations or inequalities involving the components $x_1, \ldots, x_n$ of $\mathbf{x}$.

As a result, combined with a paver [37] which bisects boxes, the contractor will allow us to build an outer approximation of the set $\mathbb{X}$. This outer approximation is represented as a list of boxes $\mathcal{L}$ with sides parallel to the coordinate axes. This list $\mathcal{L}$ contains as few elements not from $\mathbb{X}$ as possible. The resulting methodology can be applied in several domains of engineering such as identification [33], localization [23] [32], SLAM [31] [36], vision [18], reachability [22] [28], control [34] [41], calibration [17], motion planning [13], etc.

Contractor programming relies on a catalog of efficient elementary contractors which are usually built using interval arithmetic [30]. Basically, interval computation allows us to compute the range of a function the input of which are known to be inside intervals. Take for instance, $f(x) = x^2 - x$ where $x \in [x] = [x^-, x^+]$. We have $f(x) = a - x$, where $a = x^2$. To enclose $f(x)$, we first compute the smallest interval $[a]$ which contains all feasible $a = x^2$, $x \in [x]$. Then, we compute the smallest interval $[y]$ which encloses the set of feasible $a-x$, assuming that $a \in [a]$ and $x \in [x]$. Interval computation can be used to build contractors for elementary equations. There contractors shrink intervals of feasible values for the variables, without removing a single feasible value. Note that straightforward (naive) interval arithmetic which simply replaces all arithmetic operations with corresponding operations of interval arithmetic leads to pessimistic contractors. As explained in [29] more accurate enclosure can be obtained using centered form, monotonicity checking, and other ideas now used in effective interval packages.

Combining all available elementary contractors, we can construct a more sophisticated one consistent with the solution set of the problem we want to solve. This operation introduces a pessimism which has to be balanced by additional bisections performed by the paver [40]. For more efficiency, it is important to extend the catalog by adding some new specific contractors.

In this paper, we propose to use the hyperoctahedral group B_n of symmetries [2] to build optimal contractors for different types of constraints used in the field of robotics [14][25][27]. The fundamental role of this group in the domain of interval computation is here demonstrated for the first time. The important role of symmetries in constraint programming has already been underlined by several authors in the case where the solution set is symmetric (see, *e.g.,* [20] or [21]). In our applications, the problem is not symmetric but involves constraints that contain symmetries.

To illustrate our approach, we consider the *rotate* constraint which links two vectors $\mathbf{p} = (p_1, p_2)$ and $\mathbf{y} = (y_1, y_2)$ of the plane and the angle θ between them. The corresponding contractor aims to contract the intervals $[p_1]$, $[p_2]$, $[\theta]$, $[y_1]$, $[y_2]$, for p_1, p_2, θ, y_1, y_2, without loosing any feasible value.

This *rotate* constraint can be used for localization of mobile robots when bearing measurements are collected [10, 16]. A test case will show that our approach is able to obtain an outer approximation of the solution set in a much more efficient manner than simply composing elementary interval contractors.

This paper is organized as follows. Section 2 presents some notions of the theory of hyperoctahedral groups. Section 3 shows how these groups can be used as an operator to manipulate sets of $\mathbb{R}^n$. More precisely, we introduce the semigroup of actions (or acts) to create complex sets using the symmetry operators. Section 3 presents an algorithm than can be used to compute minimal contractors in the case where hyperoctahedral symmetries exist. The methodology will be illustrated on two elementary constraints: the square $x_2 = x_1^2$ and the multiplication $x_1 \cdot x_2 = x_3$. Section 5 gives an optimal contractor for a constraint which relates two vectors and their angles. This geometrical constraint, ubiquitous in robot localization, cannot be treated optimally without consider-

ing our approach. Section 6 considers the localization problem of a robot with several radars. Section 7 concludes the paper.

2 Theory

2.1 Hyperoctahedral group

The hyperoctahedral group B_n is the group of symmetries of the hypercube $[-1,1]^n$ [12] of $\mathbb{R}^n$. It corresponds to the group of $n \times n$ orthogonal matrices whose entries are integers and contains $2^n \cdot n!$ elements. For instance, for $n = 2$, we have $2^2 \cdot 2! = 8$ elements. If we use the matrix form, the elements of B_2 are

$$\begin{array}{rclrcl} \sigma_0 & = & \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \sigma_1 & = & \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \\ \sigma_2 & = & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & \sigma_3 & = & \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \\ \sigma_4 & = & \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} & \sigma_5 & = & \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\ \sigma_6 & = & \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} & \sigma_7 & = & \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \end{array} \tag{1}$$

Matrices associated to B_n are called *hyperoctahedral matrices.* Each line and each column should contain one and only one non-zero entry which should be 1 or -1. It is trivial to check that B_n is a group with respect to the multiplication, or equivalently, for the composition $\circ$. In this paper, a symmetry σ will be seen both as a linear function from $\mathbb{R}^n$ to $\mathbb{R}^n$ or as a $n \times n$ matrix. For instance, we will write equivalently

$$\sigma_5 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \text{ or } \sigma_5 : \begin{cases} \mathbb{R}^2 & \mapsto & \mathbb{R}^2 \\ (x_1, x_2) & \mapsto & (x_2, -x_1) \end{cases} \tag{2}$$

Figure 1 corresponds to the multiplication table (known as Cayley table [5]).

	σ_0	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7
σ_0	σ_0	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7
σ_1	σ_1	σ_0	σ_6	σ_4	σ_3	σ_7	σ_2	σ_5
σ_2	σ_2	σ_5	σ_0	σ_7	σ_6	σ_1	σ_4	σ_3
σ_3	σ_3	σ_4	σ_7	σ_0	σ_1	σ_6	σ_5	σ_2
σ_4	σ_4	σ_3	σ_5	σ_1	σ_0	σ_2	σ_7	σ_6
σ_5	σ_5	σ_2	σ_4	σ_6	σ_7	σ_3	σ_0	σ_1
σ_6	σ_6	σ_7	σ_1	σ_5	σ_2	σ_0	σ_3	σ_4
σ_7	σ_7	σ_6	σ_3	σ_2	σ_5	σ_4	σ_1	σ_0

Fig. 1: Multiplication table

Now, due to the fact that we have a group, the transitivity and the inversion property should be satisfied. As a consequence, the entries of the table are

dependent. We can easily check that from the second and the third lines, we can reconstruct the whole table. We say that the two elements σ_1, σ_2 are generators of the group B_2 or equivalently, we write $B_2 = \langle \sigma_1, \sigma_2 \rangle$.

Compute for instance $\sigma_6 \circ \sigma_7$ using these two lines (red and blue in the figure). We get:

$$\sigma_6 \circ \sigma_7 \quad = \quad \sigma_1 \circ \sigma_2 \circ \sigma_1 \circ \underbrace{\sigma_2 \circ \sigma_1}_{\sigma_5} = \sigma_4 \qquad \underbrace{\qquad\qquad}_{\sigma_7} \qquad \underbrace{\qquad\qquad\qquad}_{\sigma_3} \tag{3}$$

To prove that $\{\sigma_1, \sigma_2\}$ is a generator pair of B_2, we build the Cayley graph [11] (see Figure 2). We first draw a node for each of the 8 symmetries. Then we draw the arcs by reading the lines of the multiplication table corresponding to σ_1, σ_2. For instance, since we read $\sigma_7 = \sigma_1 \circ \sigma_5$, we draw an arc labeled by σ_1 between σ_5 and σ_7. Since the graph is strongly connected, we conclude that all symmetries can be obtained by compositions of σ_1, σ_2.

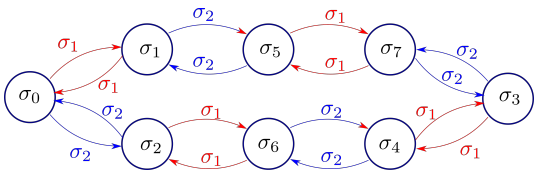

Fig. 2: Cayley graph associated to the hyperoctahedral group B_2

An important property that will be used later for building optimal contractors is that any $\sigma \in B_n$ is *box-conservative* [15], *i.e*, if for all $\mathbb{A} \subseteq \mathbb{R}^n$,

$$\sigma\left([\![\mathbb{A}]\!]\right) = [\![\sigma\left(\mathbb{A}\right)]\!]\,. \tag{4}$$

where $[\![\mathbb{A}]\!]$ is the *hull* operator, *i.e.*, the smallest box which encloses $\mathbb{A}$. An illustration is given by Figure 3.

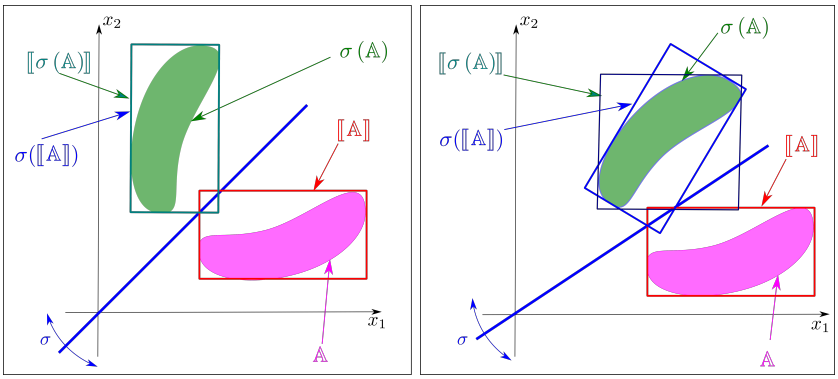

Fig. 3: (Left) σ is hyperoctahedral and the equality $\sigma\,([\![\mathbb{A}]\!]) = [\![\sigma\,(\mathbb{A})]\!]$ is satisfied. (Right) The equality is not anymore satisfied when σ is not hyperoctahedral

2.2 Hyperoctahedral symmetries for a set

Consider a set $\mathbb{X}$ of $\mathbb{R}^n$. We define the *stabilizer subgroup* of B_n with respect to $\mathbb{X}$ as

$$B_n(\mathbb{X}) = \{\sigma \in B_n \,|\, \sigma(\mathbb{X}) = \mathbb{X}\}. \tag{5}$$

The elements of $B_n(\mathbb{X})$ are called *hyperoctahedral stabilizers* for $\mathbb{X}$. Equivalently, we say that $B_n(\mathbb{X})$ corresponds to the hyperoctahedral symmetries of $\mathbb{X}$. It can easily be checked that $B_n(\mathbb{X})$ is a subgroup of B_n.

Figure 4 represents a set $\mathbb{X}$ with exactly 8 hyperoctahedral stabilizers. As seen previously (see Figure 2), the set $B_2(\mathbb{X})$ can be generated from two axis symmetries: σ_1, σ_2 (see (1)). In this situation, $B_n(\mathbb{X}) = B_n$. Now, very often in practice, the set $\mathbb{X}$ has less symmetries and we only have the inclusion $B_n(\mathbb{X}) \subset B_n$.

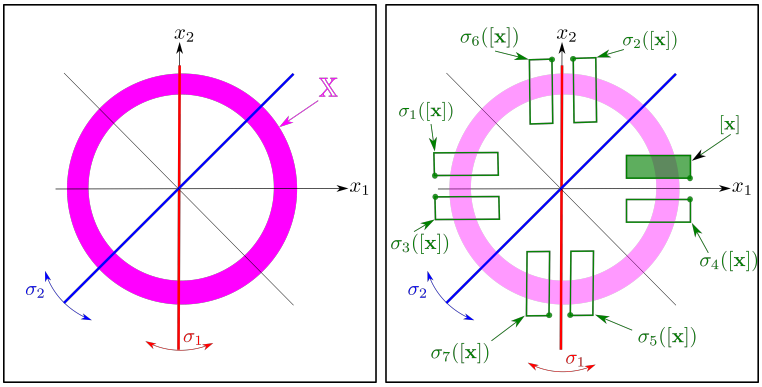

Fig. 4: We count 8 hyperoctahedral stabilizers for the set $\mathbb{X}$. On the right, we applied the corresponding symmetries to the box $[\mathbf{x}]$

2.3 Checking that a symmetry is a stabilizer: the polynomial case

Assume that the set $\mathbb{X} \subset \mathbb{R}^n$ is described by a set of polynomial equations. Checking that $\sigma \in B_n$ is a stabilizer of $\mathbb{X}$, *i.e.*, $\sigma(\mathbb{X}) = \mathbb{X}$ amounts to checking that two polynomial equations are equivalent. For instance, assume that $\mathbb{X}$ is described by the following equations:

$$\mathbb{X} : \left\{ \begin{array}{rcl} x_3x_1 - x_4x_2 - x_5 & = & 0 \\ x_4x_1 + x_3x_2 - x_6 & = & 0 \\ x_3^2 + x_4^2 - 1 & = & 0 \end{array} \right. \tag{6}$$

For

$$\sigma = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

the equality $\sigma(\mathbb{X}) = \mathbb{X}$ holds. Indeed

$$\left\{ \begin{array}{rcl} x_3x_1 - x_4x_2 - x_5 & = & 0 \\ x_4x_1 + x_3x_2 - x_6 & = & 0 \\ x_3^2 + x_4^2 - 1 & = & 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcl} x_3x_5 + x_4x_6 - x_1 & = & 0 \\ -x_4x_5 + x_3x_6 - x_2 & = & 0 \\ x_3^2 + (-x_4)^2 - 1 & = & 0 \end{array} \right. \tag{8}$$

The equality has the form

$$\left\{ \begin{array}{rcl} P_1(\mathbf{x}) & = & 0 \\ P_2(\mathbf{x}) & = & 0 \\ P_3(\mathbf{x}) & = & 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcl} Q_1(\mathbf{x}) & = & 0 \\ Q_2(\mathbf{x}) & = & 0 \\ Q_3(\mathbf{x}) & = & 0 \end{array} \right. \tag{9}$$

where $\mathbf{x} = (x_1, \ldots, x_6)$. To check the equivalence, it suffices to check that the Gröbner bases, computed by the Buchberger's algorithm [4], for $\{P_1, P_2, P_3\}$ and for $\{Q_1, Q_2, Q_3\}$ are the same. This can easily be checked using symbolic calculus [24].

2.4 Finding the subgroup of stabilizers

Consider a set $\mathbb{X} \subset \mathbb{R}^n$ described by a set of polynomial equations. The set $B_n(\mathbb{X})$ is a subgroup of B_n. To find this subgroup, a naive method [11] is to check all $2^n \cdot n!$, which is a tedious work. To find the generators for $B_n(\mathbb{X})$, we could take randomly elements of B_n and check if they can generate the whole subgroup $B_n(\mathbb{X})$.

3 Hyperoctahedral acts

In the previous section, we have introduced the group of hyperoctahedral symmetries for a set $\mathbb{X}$. Such a group can be used to move a box from one zone on

$\mathbb{R}^n$ to another and in a reversible way, without changing its status with respect to $\mathbb{X}$. In this section, we will still use the symmetries as a way to move boxes, but we will adapt the composition in a way to build a set $\mathbb{X}$. For this purpose, we will introduce the notion of *act* to build a set. Analogously, the set $\mathbb{X}$ can be seen as a house to be built with bricks. The elements of the group of symmetries can be used to move one brick from one place to another in a *cut-paste* manner. Instead, the act operator works in a *copy-paste* manner. It uses a symmetry to copy and move the brick, making the operation monotonic (the house inflates) and non invertible (a brick cannot be removed). This describes the behavior of a *semigroup* instead of a group used in the previous section. This construction process will now be explained and used for building contractors for sets with symmetries.

3.1 Act for sets

For $\sigma \in B_n$, we define the *act* operator:

$$\sigma \bullet \mathbb{X} = \mathbb{X} \cup \sigma(\mathbb{X}). \tag{10}$$

We say that σ *acts* on the set $\mathbb{X}$. This operation produces a larger set (see Figure 5).

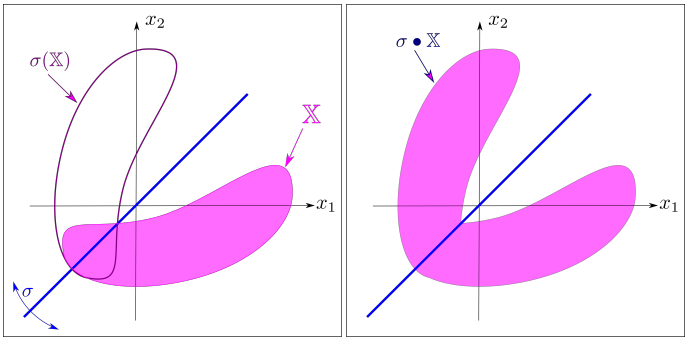

Fig. 5: σ acts on the set $\mathbb{X}$ to generate $\sigma \bullet \mathbb{X} = \mathbb{X} \cup \sigma(\mathbb{X})$

Given a set $\mathbb{X} \subset \mathbb{R}^n$, we define the set A_n as the set of all operators $\sigma : \mathcal{P}(\mathbb{R}^n) \mapsto \mathcal{P}(\mathbb{R}^n)$ which can be written as

$$\sigma(\mathbb{X}) = \sigma \bullet \mathbb{X} = \alpha_m \bullet (\cdots \bullet (\alpha_2 \bullet (\alpha_1 \bullet \mathbb{X})) \dots) \tag{11}$$

where $\alpha_1, \dots, \alpha_m$ all belong to B_n. If σ_1 and σ_2 belong to A_n, we define the operator $\star$ as follows:

$$(\sigma_2 \star \sigma_1) \bullet \mathbb{X} = \sigma_2 \bullet (\sigma_1 \bullet \mathbb{X}). \tag{12}$$

It can be shown that the structure $(A_n, \star, \bullet)$ is a semigroup action [9]. It satisfies:

$$\begin{array}{rcl} \sigma \in B_n & \Rightarrow & \sigma \in A_n \\ \sigma \in B_n, \alpha \in A_n & \Rightarrow & (\alpha \star \sigma) \in A_n \text{ and } (\alpha \star \sigma) \bullet \mathbb{X} = \alpha \bullet (\sigma \bullet \mathbb{X}). \end{array} \tag{13}$$

From this definition we see that each element of B_n is associated to an element of A_n, but inverse is not true. The second condition in (13) is called the *compatibility* condition. It is trivial to prove that $(A_n, \star)$ is a semigroup. For instance, the associativity is checked as follows:

$$(\sigma_3 \star (\sigma_2 \star \sigma_1)) \bullet \mathbb{X} = ((\sigma_3 \star \sigma_2) \star \sigma_1) \bullet \mathbb{X} = \sigma_3 \bullet (\sigma_2 \bullet (\sigma_1 \bullet \mathbb{X})). \tag{14}$$

Figure 6 provides an illustration of the $\star$ operator in the case where two symmetries σ_1, σ_2 of B_2 are involved.

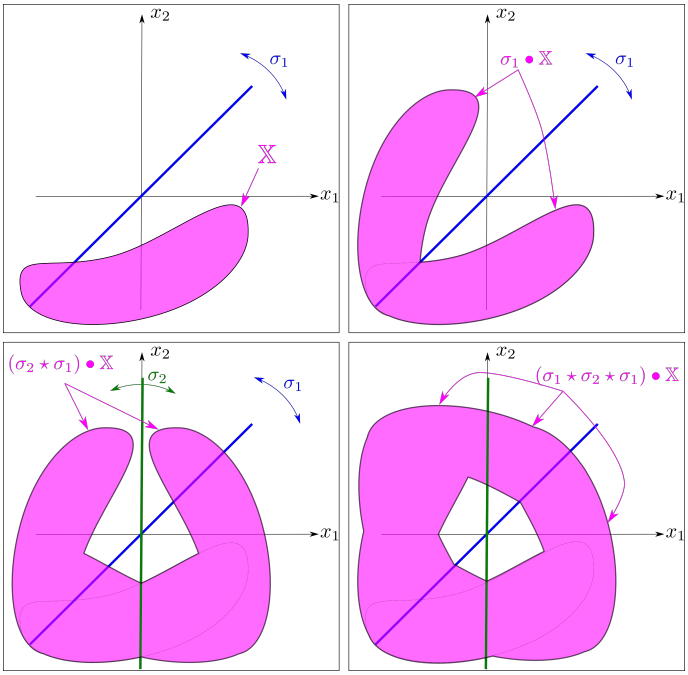

Fig. 6: Illustration of the $\star$ operator

3.2 Contractors

We define a box of $\mathbb{R}^n$ as the Cartesian product of n intervals. A box will be written as follows:

$$[\mathbf{x}] = [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+] = [\mathbf{x}^-, \mathbf{x}^+]. \tag{15}$$

Denote by $\mathbb{IR}^n$ the set of boxes of $\mathbb{R}^n$. A *contractor* $\mathcal{C}$ for a set $\mathbb{X} \subseteq \mathbb{R}^n$ is an operator $\mathbb{IR}^n \mapsto \mathbb{IR}^n$ such that

$$\begin{array}{ll} \mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}] & \text{(contractance)} \\ [\mathbf{x}] \subseteq [\mathbf{y}] \;\Rightarrow\; \mathcal{C}([\mathbf{x}]) \subseteq \mathcal{C}([\mathbf{y}]). & \text{(monotonicity)} \\ \mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X} & \text{(consistency)} \end{array} \tag{16}$$

We define the inclusion between two contractors $\mathcal{C}_1$ and $\mathcal{C}_2$ as follows:

$$\mathcal{C}_1 \subseteq \mathcal{C}_2 \Leftrightarrow \forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}_1([\mathbf{x}]) \subseteq \mathcal{C}_2([\mathbf{x}]). \tag{17}$$

There exists a unique minimal contractor for $\mathbb{X}$, given by

$$\mathcal{C}([\mathbf{x}]) = [\![[\mathbf{x}] \cap \mathbb{X}]\!] \tag{18}$$

The quantity $\mathcal{C}([\mathbf{x}])$ corresponds to the smallest box that can be obtained by a contraction of $[\mathbf{x}]$ without removing a single point of $\mathbb{X}$, as illustrated by Figure 7.

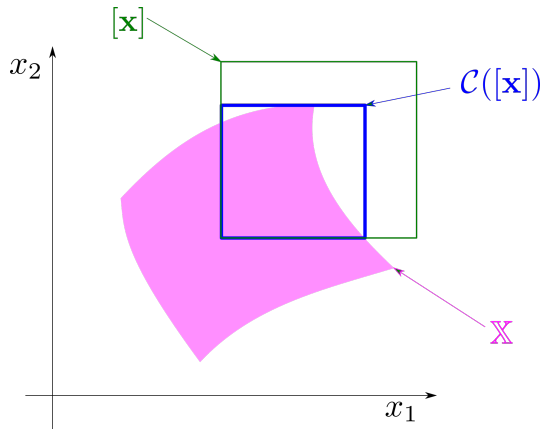

Fig. 7: Minimal contractor $\mathcal{C}$ for the set $\mathbb{X}$

Define the box union of two boxes $[\mathbf{x}] \sqcup [\mathbf{y}]$ as the smallest box which encloses both $[\mathbf{x}]$ and $[\mathbf{y}]$, *i.e.*,

$$[\mathbf{x}] \sqcup [\mathbf{y}] = [\![[\mathbf{x}] \cup [\mathbf{y}]]\!]. \tag{19}$$

Given $\mathcal{C}_1$ a contractor for the set $\mathbb{X}_1 \subset \mathbb{R}^n$ and if $\mathcal{C}_2$ is contractor for the set $\mathbb{X}_2 \subset \mathbb{R}^n$, we define the union of the two contractors as

$$(\mathcal{C}_1 \sqcup \mathcal{C}_2)([\mathbf{x}]) = \mathcal{C}_1([\mathbf{x}]) \sqcup \mathcal{C}_2([\mathbf{x}]). \tag{20}$$

In [15], it has been proved that if $\mathcal{C}_1$ and $\mathcal{C}_2$ minimal, then $\mathcal{C}_1 \sqcup \mathcal{C}_2$ is a minimal contractor for $\mathbb{X}_1 \cup \mathbb{X}_2$.

The following proposition shows that the minimality is also true for the projection. This result, which seems intuitive, is new to my knowledge. Since it will be used later in our application, we formalize this claim as a proposition and we prove it.

Proposition 1. *Define the set*

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \exists \mathbf{y} \in [\mathbf{y}], \mathbf{z} = (\mathbf{x}, \mathbf{y}) \in \mathbb{Z}\} \tag{21}$$

which corresponds to the projection of the set $\mathbb{Z} \cap (\mathbb{R}^n \times [\mathbf{y}])$ *onto* $\mathbb{R}^n$*. If* $\mathcal{C}_{\mathbb{Z}}$ *is a minimal contractor for* $\mathbb{Z}$ *then*

$$\mathcal{C}_{\mathbb{X}}([\mathbf{x}]) = proj_{\mathbf{x}}\mathcal{C}_{\mathbb{Z}}([\mathbf{x}], [\mathbf{y}]) \tag{22}$$

is a minimal contractor for $\mathbb{X}$*.*

Proof. We have

$$\begin{array}{rcl} \mathcal{C}_{\mathbb{X}}([\mathbf{x}]) & = & \text{proj}_{\mathbf{x}}\mathcal{C}_{\mathbb{Z}}([\mathbf{x}], [\mathbf{y}]) \\ & = & \text{proj}_{\mathbf{x}}\left[\!\left[([\mathbf{x}] \times [\mathbf{y}]) \cap \mathbb{Z}\right]\!\right] \\ & = & \left[\!\left[\text{proj}_{\mathbf{x}}(([\mathbf{x}] \times [\mathbf{y}]) \cap \mathbb{Z})\right]\!\right] \\ & = & \left[\!\left[[\mathbf{x}] \cap \mathbb{X}\right]\!\right] \end{array} \tag{23}$$

Thus, $\mathcal{C}_{\mathbb{X}}([\mathbf{x}]) = \text{proj}_{\mathbf{x}}\mathcal{C}_{\mathbb{Z}}([\mathbf{x}], \mathbb{R}^m)$ is the minimal contractor for the set $\mathbb{X}$. □

Example. Consider the set

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \exists y \in [1, 5], \mathbf{z} = (\mathbf{x}, y) \in \mathbb{Z}\} \tag{24}$$

where

$$\mathbb{Z} = \left\{(x_1, x_2, y) | x_1^2 + x_2^2 + y^2 \in [0, 16]\right\}. \tag{25}$$

Take $[\mathbf{x}] = [1, 5] \times [1, 5]$. The minimal contractor associated to $\mathbb{Z}$ yields

$$\left[\!\left[([\mathbf{x}] \times [y]) \cap \mathbb{Z}\right]\!\right] = [1, \sqrt{14}] \times [1, \sqrt{14}] \times [1, \sqrt{14}] \tag{26}$$

which corresponds to the red three dimensional box in Figure 8. The projection of this box on the $\mathbf{x}$ space is

$$\left[\!\left[\text{proj}_{\mathbf{x}}(([\mathbf{x}] \times [\mathbf{y}]) \cap \mathbb{Z})\right]\!\right] = [1, \sqrt{14}] \times [1, \sqrt{14}] \tag{27}$$

which corresponds to the optimal contraction with respect to $\mathbb{X}$ (see the red two dimensional box).

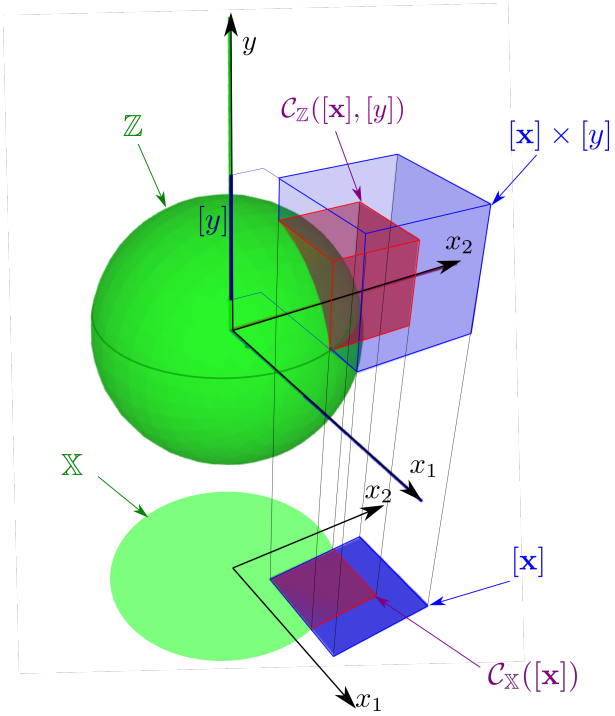

Fig. 8: The minimality of a contractor is conserved by projection

3.3 Contractor Acts

If $\mathcal{C}$ is a contractor in $\mathbb{R}^n$, and $\sigma \in B_n$, we define the *contractor act* of σ on $\mathcal{C}$ as

$$\sigma \bullet \mathcal{C} = \mathcal{C} \sqcup (\sigma \circ \mathcal{C} \circ \sigma^{-1}). \tag{28}$$

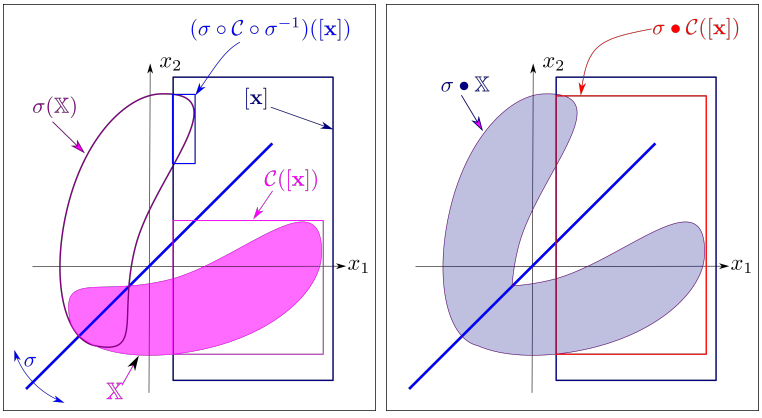

Fig. 9: Minimal contractor $\sigma \bullet \mathcal{C}$ for the set $\sigma \bullet \mathbb{X}$

Proposition 2. *If $\mathcal{C}$ is a minimal contractor for $\mathbb{X}$ and if $\sigma \in B_n$ then $\sigma \bullet \mathcal{C}$ is a minimal contractor for $\sigma \bullet \mathbb{X}$, i.e.,*

$$\sigma \bullet \mathcal{C}([\mathbf{x}]) = [\![[\mathbf{x}] \cap \sigma \bullet \mathbb{X}]\!]. \tag{29}$$

An illustration of the proposition is given by Figure 9.

Proof. We have

$$\begin{array}{lcll}
(\sigma \bullet \mathcal{C})([\mathbf{x}]) & \overset{(28)}{=} & \mathcal{C}([\mathbf{x}]) \sqcup (\sigma \circ \mathcal{C} \circ \sigma^{-1})([\mathbf{x}]) & \\
& = & \mathcal{C}([\mathbf{x}]) \sqcup (\sigma(\mathcal{C}(\sigma^{-1}([\mathbf{x}])))) & \\
& = & \mathcal{C}([\mathbf{x}]) \sqcup \sigma([\![\sigma^{-1}([\mathbf{x}]) \cap \mathbb{X}]\!]) & \text{since } \mathcal{C} \text{ is minimal} \\
& \overset{(4)}{=} & \mathcal{C}([\mathbf{x}]) \sqcup [\![\sigma(\sigma^{-1}([\mathbf{x}]) \cap \mathbb{X})]\!] & \\
& = & \mathcal{C}([\mathbf{x}]) \sqcup [\![\sigma(\sigma^{-1}([\mathbf{x}])) \cap \sigma(\mathbb{X})]\!] & \text{since } \sigma \text{ bijective} \\
& = & \mathcal{C}([\mathbf{x}]) \sqcup [\![[\mathbf{x}] \cap \sigma(\mathbb{X})]\!] & \\
& = & [\![[\mathbf{x}] \cap \mathbb{X}]\!] \sqcup [\![[\mathbf{x}] \cap \sigma(\mathbb{X})]\!] & \text{since } \mathcal{C} \text{ is minimal} \\
& = & [\![([\mathbf{x}] \cap \mathbb{X}) \cup ([\mathbf{x}] \cap \sigma(\mathbb{X}))]\!] & \\
& = & [\![[\mathbf{x}] \cap (\mathbb{X} \cup \sigma(\mathbb{X}))]\!] & \\
& = & [\![[\mathbf{x}] \cap \sigma \bullet \mathbb{X}]\!] &
\end{array}$$

□

3.4 Expansion theorem

In Subsection 3.1, we have shown how the $\star$ operator can be used to produce minimal contractors. In this section, we provide a new theorem (called *expansion theorem*) which can be used to build a minimal contractor for a specific set $\mathbb{X}$ using an expression involving symmetries connected by the $\star$ operator.

Theorem 3. *Consider a set $\mathbb{X} \subset \mathbb{R}^n$, a box $[\mathbf{a}] \in \mathbb{IR}^n$ and a sequence $\{\sigma_1, \ldots, \sigma_m\}$ of $B_n(\mathbb{X})$. Define the two sequences*

$$\begin{array}{rcl}
\mathbb{X}(0) & = & [\mathbf{a}] \cap \mathbb{X} \\
\mathbb{X}(k+1) & = & \sigma_{k+1} \bullet \mathbb{X}(k)
\end{array} \tag{30}$$

and

$$\begin{array}{rcl}
\mathbb{A}(0) & = & [\mathbf{a}] \\
\mathbb{A}(k+1) & = & \sigma_{k+1} \bullet \mathbb{A}(k)
\end{array} \tag{31}$$

We have

$$\mathbb{X} \subseteq \mathbb{A}(m) \Rightarrow \mathbb{X}(m) = \mathbb{X}. \tag{32}$$

Moreover, if $\mathbb{X} \subseteq \mathbb{A}(m)$, the contractor $\mathcal{C}_m$ defined by the sequence

$$\begin{array}{rcl}
\mathcal{C}_0([\mathbf{x}]) & = & [\![[\mathbf{x}] \cap \mathbb{X}(0)]\!] \\
\mathcal{C}_{k+1} & = & \sigma_{k+1} \bullet \mathcal{C}_k
\end{array} \tag{33}$$

is the minimal contractor for $\mathbb{X}$.

Proof. (i) In order to prove (32), we first prove by induction that

$$\mathbb{X}(k) = \mathbb{A}(k) \cap \mathbb{X}. \tag{34}$$

For $k = 0$, the equality holds. Assume that (34) is true for k. Since

$$\begin{array}{rcll}
\mathbb{X}(k+1) & = & \sigma_{k+1} \bullet \mathbb{X}(k) & \\
& \overset{(10)}{=} & \mathbb{X}(k) \cup \sigma_{k+1}(\mathbb{X}(k)) & \\
& \overset{(34)}{=} & (\mathbb{A}(k) \cap \mathbb{X}) \cup \sigma_{k+1}(\mathbb{A}(k) \cap \mathbb{X}) & \\
& = & (\mathbb{A}(k) \cap \mathbb{X}) \cup (\sigma_{k+1}(\mathbb{A}(k)) \cap \sigma_{k+1}(\mathbb{X})) & \text{(since } \sigma_{k+1} \text{ is injective)} \\
& = & (\mathbb{A}(k) \cap \mathbb{X}) \cup (\sigma_{k+1}(\mathbb{A}(k)) \cap \mathbb{X}) & \text{(since } \sigma_{k+1} \in B_n(\mathbb{X})) \\
& = & (\mathbb{A}(k) \cup \sigma_{k+1}(\mathbb{A}(k))) \cap \mathbb{X} & \\
& \overset{(10)}{=} & (\sigma_{k+1} \bullet \mathbb{A}(k)) \cap \mathbb{X} & \\
& \overset{(31)}{=} & \mathbb{A}(k+1) \cap \mathbb{X} &
\end{array}$$

the property (34) is also true for $k + 1$.

(ii) Assume now that for $k = m$, we have $\mathbb{X} \subseteq \mathbb{A}(m)$. From (34), $\mathbb{X}(m) = \mathbb{A}(m) \cap \mathbb{X}$. And thus, $\mathbb{X}(m) = \mathbb{X}$.

(iii) We prove by induction that for all k,

$$\mathcal{C}_k([\mathbf{x}]) = [\![[\mathbf{x}] \cap \mathbb{X}(k)]\!]. \tag{35}$$

For $k = 0$, this relation is $\mathcal{C}_0([\mathbf{x}]) = [\![[\mathbf{x}] \cap \mathbb{X}(0)]\!]$ which is true from (33). Assume that (35) is true for k. We have

$$\begin{array}{rcl}
\mathcal{C}_{k+1}([\mathbf{x}]) & \overset{(33)}{=} & (\sigma_{k+1} \bullet \mathcal{C}_k)([\mathbf{x}]) \\
& \overset{(29)}{=} & [\![[\mathbf{x}] \cap \sigma_{k+1} \bullet \mathbb{X}(k)]\!] \\
& = & [\![[\mathbf{x}] \cap \mathbb{X}(k+1)]\!]
\end{array} \tag{36}$$

□

Figure 10 illustrates the two first steps of the sequence for $\mathbb{X}(k)$.

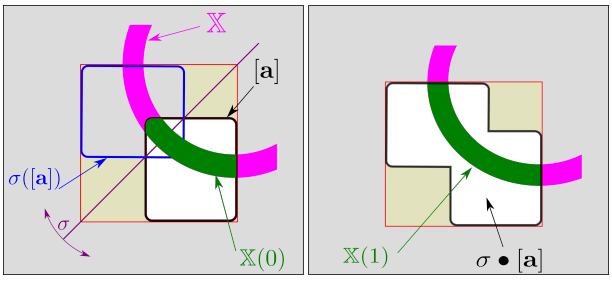

Fig. 10: Construction of the sequence $\mathbb{X}(k)$ using the symmetries

Figure 11 shows how the minimal contractor can be obtained for $\mathbb{X}(1)$.

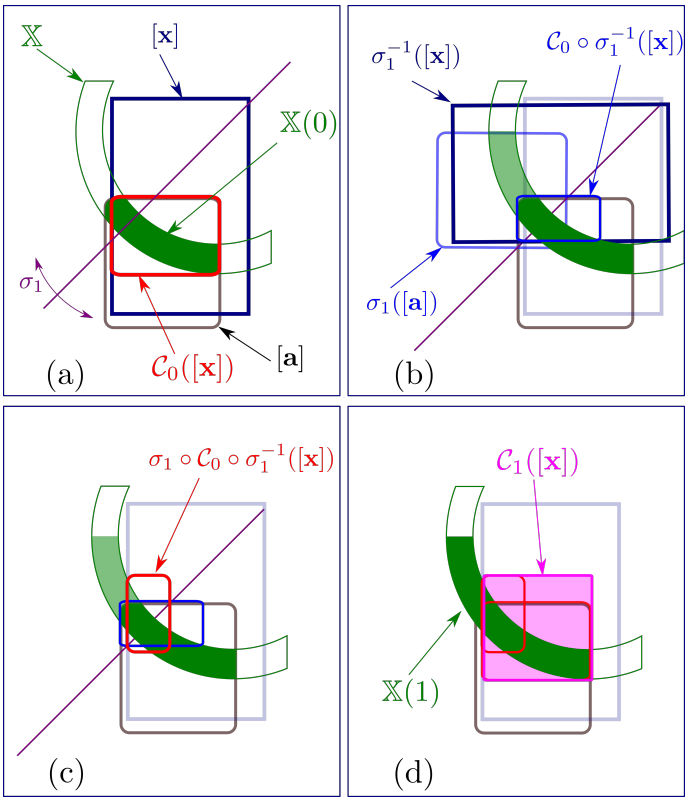

Fig. 11: Construction of the optimal contractor $\mathcal{C}_1$ for $\mathbb{X}(1)$

Corollary 4. *If $\mathbb{X} \not\subseteq \mathbb{A}(m)$ then $\mathcal{C}_m$ is not a contractor for $\mathbb{X}$.*

Proof. The operator $\mathcal{C}_m$ is an optimal contractor for $\mathbb{X}(m)$. Now, $\mathbb{X}(m) \subseteq \mathbb{A}(m) \not\supseteq \mathbb{X}$. If $[\mathbf{x}]$ is box in $\mathbb{X} \backslash \mathbb{X}(m)$, we will have $\mathcal{C}_m([\mathbf{x}]) = \emptyset$. □

4 Algorithm

Consider a set $\mathbb{X}$ of $\mathbb{R}^n$ described by a set of polynomial equations with hyperoctahedral symmetries described by $B_n(\mathbb{X})$. Assume that we have a box $[\mathbf{a}]$ such that an optimal contractor $\mathcal{C}_0([\mathbf{x}]) = [\![[\mathbf{x}] \cap [\mathbf{a}] \cap \mathbb{X}]\!]$ is available for $\mathbb{X} \cap [\mathbf{a}]$. Take a sequence of symmetries $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ for $\mathbb{X}$. Algorithm 1 checks if we can generate from $[\mathbf{a}]$ using Σ, a set which encloses $\mathbb{X}$.

Algorithm 1 ValidSequence

	Input: $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}, [\mathbf{a}]$
1	$\mathbb{A}(0) = [\mathbf{a}]$
2	For $k \in \{0, \ldots, m-1\}$
3	$\quad \mathbb{A}(k+1) = \sigma_{k+1} \bullet \mathbb{A}(k)$,
4	If $\mathbb{X} \not\subseteq \mathbb{A}$ return "Fail: not enough symmetries".
5	Else return Success

If the algorithm terminates, we know that we can reach any point of $\mathbb{X}$ from a point of $[\mathbf{a}]$ by a sequence of acts made with symmetries. We want a valid sequence which is as small as possible. This sequence can be found by the following algorithm.

Algorithm 2 FindSequence

	Input: $\mathcal{S}$,$[\mathbf{a}]$
1	For $m \in \{0, 1, 2, 3, \ldots\}$
2	$\quad$ For all sequences $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ taken in $\mathcal{S}$
3	$\quad\quad$ If ValidSequence(Σ) successes return Σ

If Algorithm 2 returns the sequence Σ, then from Theorem 3, the minimal contractor for $\mathbb{X}$ is

$$\mathcal{C} = (\sigma_m \star \cdots \star \sigma_1) \bullet \mathcal{C}_0. \tag{37}$$

Moreover, from Corollary 4, we also know that there is no sequence with a length smaller than m which leads to a contractor for $\mathbb{X}$.

4.1 Square constraint

Denote by $\mathbb{X}$ the set of all $\mathbf{x} = (x_1, x_2)$ which satisfy the constraint.

$$\varphi(x_1, x_2) = x_1^2 - x_2 = 0. \tag{38}$$

Due to the monotony of φ, on $[\mathbf{a}] = \mathbb{R}^+ \times \mathbb{R}^+$, the minimal contractor on $[\mathbf{a}]$ for the constraint $\varphi(x_1, x_2) = 0$ is

$$\mathcal{C}_0 \begin{pmatrix} [x_1] \\ [x_2] \end{pmatrix} = \begin{pmatrix} [x_1] \cap \left[\sqrt{x_2^-}, \sqrt{x_2^+}\right] \\ [x_2] \cap \left[x_1^{-2}, x_1^{+2}\right] \end{pmatrix}. \tag{39}$$

For $n = 2$, the set B_2 has $2^2 * 2! = 8$ elements given by (1). The symmetries for $\mathbb{X}$ are $B_2(\mathbb{X}) = \{\sigma_0, \sigma_1\}$. If we apply σ_1 to (38), the equation remains satisfied.

$$x_1^2 - x_2 = 0 \Leftrightarrow (-x_1)^2 - x_2 = 0. \tag{40}$$

The algorithm ValidSequence($\{\sigma_1\}, [\mathbf{a}]$) returns True, which means that $\sigma_1 \bullet \mathcal{C}_0$ is the minimal contractor for $\mathbb{X}$.

4.2 Product constraint

Denote by $\mathbb{X}$ the set of all $\mathbf{x} = (x_1, x_2, x_3)$ which satisfy the constraint:

$$\varphi(x_1, x_2, x_3) = x_1 x_2 - x_3 = 0, \tag{41}$$

as illustrated by Figure 12. Take

$$[\mathbf{a}] \quad = \quad \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+. \tag{42}$$

The part of $\mathbb{X}$ which is inside $[\mathbf{a}]$ is painted red.

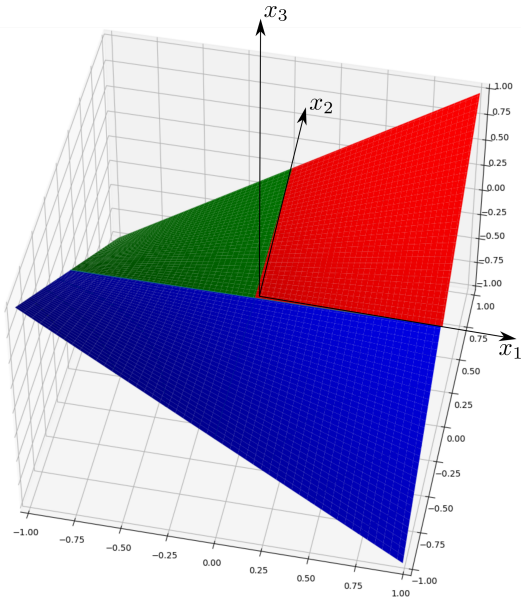

Fig. 12: Product constraint: $x_1 x_2 = x_3$

Due to the monotonicity, a minimal contractor on $[\mathbf{a}]$ for the constraint $\varphi(x_1, x_2, x_3) = 0$ is

$$\mathcal{C}_0 \begin{pmatrix} [x_1] \\ [x_2] \\ [x_3] \end{pmatrix} = \begin{pmatrix} [x_1] \cap \left[\frac{x_3^-}{x_2^+}, \frac{x_3^+}{x_2^-}\right] \\ [x_2] \cap \left[\frac{x_3^-}{x_1^+}, \frac{x_3^+}{x_1^-}\right] \\ [x_3] \cap [x_1^- \cdot x_2^-, x_1^+ \cdot x_2^+] \end{pmatrix}. \tag{43}$$

For $n = 3$, the set B_3 has $2^3 * 3! = 48$ elements. One of them is

$$\sigma_1 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{44}$$

Among all hyperoctahedral matrices, some of them are stabilizers for $\mathbb{X}$, some not. To test if $\sigma_1 \in B_3(\mathbb{X})$, we apply σ_1 to (41) and we check if the equation remains satisfied:

$$x_1x_2 - x_3 = 0 \Leftrightarrow (-x_1)\cdot(-x_2) - x_3 = 0. \tag{45}$$

The symmetry σ_1 allows us to build the green part of $\mathbb{X}$. Other symmetry matrices could be found. At least two of them should be added to generate $B_3(\mathbb{X})$. For instance, we can take

$$\sigma_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \sigma_3 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}. \tag{46}$$

Note that $\langle \sigma_2, \sigma_3 \rangle = B_3(\mathbb{X}) \subset B_3$. This means that two symmetries are sufficient to generate $B_3(\mathbb{X})$, but if we restrict the symmetries to σ_2, σ_3, the length of the sequence will not be optimal. The algorithm FindSequence$(\{\sigma_1, \sigma_2, \sigma_3\}, [\mathbf{a}])$ finds two expressions of minimal length for the minimal contractor for $\mathbb{X}$: $(\sigma_3 \star \sigma_1) \bullet \mathcal{C}_0$ and $(\sigma_1 \star \sigma_3) \bullet \mathcal{C}_0$. This means that σ_3 allows us to complete the construction of $\mathbb{X}$, as illustrated by the blue part of $\mathbb{X}$ on Figure 12. Note that even if σ_2 is needed to generate B_3 we do not need it to build $\mathbb{X}$ from $\mathbb{X} \cap [\mathbf{a}]$.

As an illustration, consider the set $\mathbb{P}$ of all $\mathbf{p} = (x_1, x_2) \in \mathbb{R}^2$, such that $x_1 \cdot x_2 = x_3$, where $x_3 \in [-9, -2]$. We choose here to characterize the two dimensional set $\mathbb{P}$ to visualize the minimality of the contractor $\mathcal{C} = (\sigma_2 \star \sigma_0) \bullet \mathcal{C}_0$, which could not be possible in a three dimensional representation. If we use a paver which bisects only on the (x_1, x_2)-space, we get Figure 13, where the frame box is $[\mathbf{p}](0) = [x_1](0) \times [x_2](0) = [-20, 20] \times [-20, 20]$. Note that $\mathcal{C}$ is a three dimensional contractor, but since $\mathbb{P}$ is two dimensional, we only select the two first components of the contractor. We thus build the projection of the contractor $\mathcal{C}_{\mathbf{p}}([\mathbf{p}]) = \text{proj}_{\mathbf{p}}\mathcal{C}([\mathbf{p}], [-9, -2])$. From Proposition 1, $\mathcal{C}_{\mathbf{p}}$ is a minimal contractor for $\mathbb{P}$.

The principle of the procedure is a branch and prune method which takes subboxes of $[\mathbf{p}](0)$ stored inside a list $\mathcal{L}$. For each of $[\mathbf{p}]$ in $\mathcal{L}$, the minimal contractor $\mathcal{C}$ is called, with $[x_3] = [-9, -2]$. The part of $[\mathbf{p}]$ that has been pruned by $\mathcal{C}$ is painted blue. If the width of $\mathcal{C}_{\mathbf{p}}([\mathbf{p}])$ is smaller than a given $\varepsilon > 0$, $[\mathbf{p}]$ is painted yellow. Otherwise, $[\mathbf{p}]$ is bisected and the two resulting boxes are stored in $\mathcal{L}$ to be treated later. At the initialization of the procedure, $\mathcal{L}$ contains a single element corresponding to $[\mathbf{p}](0)$. We observe that we do not have any isolated blue boxes that do not touch any tiny yellow box. This is consistent with the fact that the contractor is minimal.

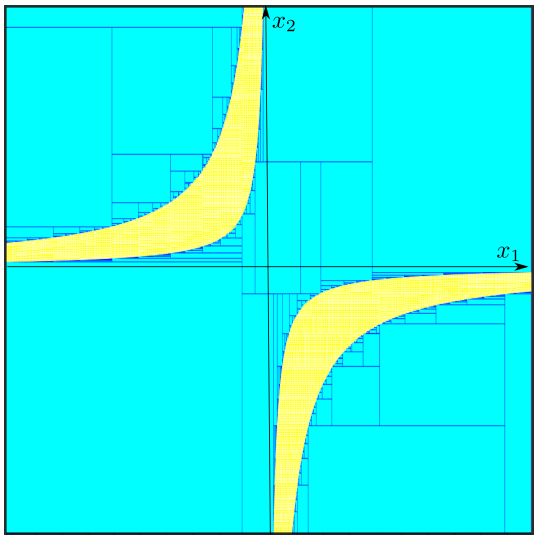

Fig. 13: Paving obtained using the minimal contractor for $x_1 \cdot x_2 \in [-9, -2]$

5 Rotate constraint

In this section, we consider the constraint *rotate* which tells us that the vector $\mathbf{y} = (y_1, y_2)$ of $\mathbb{R}^2$ can be obtained from the rotation of angle θ of the vector $\mathbf{p} = (p_1, p_2)$. This constraint occurs for instance when we want to localize a robot from bearing measurements. Assume that $p_1 \in [p_1]$, $p_2 \in [p_2]$, $\theta \in [\theta]$, $y_1 \in [y_1]$, $y_2 \in [y_2]$, we want a minimal contractor which is able to contract all five intervals as much as possible, without removing any value consistent with the *rotate* constraint.

Many hyperoctahedral symmetries exist for the *rotate* constraint and this is the reason why it has been chosen to illustrate the methodology developed in this paper. In the following Subsection 5.1, we rewrite the *rotate* constraint into polynomial equations to fit with the formalism developed previously. In order to find a minimal contractor for our constraint, we consider in Subsection 5.2 a simpler constraint called *angle*, which can be interpreted as a two-dimensional projection of *rotate*. Taking into account the monotonicity of the *angle* constraint, we build a minimal contractor on a restricted domain. We then explain in Subsection 5.3 how the restricted *angle* contractor can be used to build a restricted contractor for *rotate*. An extension of this restricted contractor, valid for any interval entries, is built in Subsection 5.4. This extension uses the symmetries to cover the whole space.

5.1 Definition

Consider the *rotate* constraint

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}}_{\mathbf{R}_\theta} \underbrace{\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}}_{\mathbf{p}} \tag{47}$$

Note that if $p_2 = 0$ then, we get the *Polar* constraint [15] given by

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} p_1 \cos\theta \\ p_1 \sin\theta \end{pmatrix}. \tag{48}$$

To fit with the formalism of the previous section, we rewrite the constraint (47) as

$$\begin{cases} x_3x_1 - x_4x_2 - x_5 &= 0 \\ x_4x_1 + x_3x_2 - x_6 &= 0 \\ x_3^2 + x_4^2 - 1 &= 0 \end{cases} \tag{49}$$

where x_1, x_2, x_3, x_4, x_5, x_6 stand for p_1, $p_2, \cos\theta$, $\sin\theta$, y_1, y_2, respectively. Define by $\mathbb{X}$ the solution set of (49). This constraint can be expressed into the form

$$\varphi(x_1, x_2, x_3, x_4, x_5, x_6) = \mathbf{0}. \tag{50}$$

To my knowledge, the minimal contractor for *rotate* constraint (47) has never been proposed in the literature and it would be very difficult to get it without using the tools derived from the semigroup action of hyperoctahedral symmetries presented in this paper.

5.2 Angle contractor

Consider the constraint $\mathbf{y} = \mathbf{R}_\theta \cdot \mathbf{p}$ given by (47) and assume that $[\mathbf{y}] \subset \mathbb{R}^+ \times \mathbb{R}^+$ and $[\theta] = [\theta^-, \theta^+] \subset [0, \frac{\pi}{2}]$. The set of all feasible $\mathbf{p} = (p_1, p_2)$ is given by

$$\begin{array}{rcl} \mathbb{P} &=& \{\mathbf{p} = (p_1, p_2) \,|\, \exists\theta \in [\theta], \exists\mathbf{y} \in [\mathbf{y}], \mathbf{p} = \mathbf{f}(\theta, \mathbf{y})\} \\ &=& \mathbf{f}([\theta], [\mathbf{y}]) \end{array} \tag{51}$$

where

$$\mathbf{f}(\theta, \mathbf{y}) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}. \tag{52}$$

The constraint $\mathbf{p} \in \mathbb{P}$ is called the *angle* constraint, and we would like to find a minimal contractor $\mathcal{C}_\mathbb{P}$ for $\mathbb{P}$. This minimal contractor is given by

$$\mathcal{C}_\mathbb{P} : [\mathbf{p}] \to [\![[\mathbf{p}] \cap \mathbf{f}([\theta], [\mathbf{y}])]\!]. \tag{53}$$

We thus need to characterize the set $\mathbb{P} = \mathbf{f}([\theta], [\mathbf{y}])$. Since $[\theta], [y_1], [y_2]$ are all positive, the set $\mathbb{P}$ has a boundary $\partial\mathbb{P}$ delimited by edges (arcs and segments) as illustrated by Figure 14(a). The intersection between two adjacent edges is

called a *vertex* and are denoted by $\mathbf{a}_i, i \in \{1, 2, \dots\}$. Denote by $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$ the corners of $[\mathbf{y}]$. The set $\mathbb{P}$ has vertices given by:

$$\begin{array}{lcll} \mathbf{a}_1 & = & \mathbf{f}(\theta^-, \mathbf{m}_1) & \\ \mathbf{a}_2 & = & \mathbf{f}(\theta^-, \mathbf{m}_2) & \\ \mathbf{a}_3 & = & \mathbf{f}(\theta^-, \mathbf{m}_3) & \\ \mathbf{a}_4 & = & \mathbf{f}(\text{proj}_{[\theta]}\theta_R, \mathbf{m}_3), & \text{where } \theta_R = \text{atan}(\frac{m_{3y}}{m_{3x}}) \\ \mathbf{a}_5 & = & \mathbf{f}(\theta^+, \mathbf{m}_3) & \\ \mathbf{a}_6 & = & \mathbf{f}(\theta^+, \mathbf{m}_4) & \\ \mathbf{a}_7 & = & \mathbf{f}(\theta^+, \mathbf{m}_1) & \\ \mathbf{a}_8 & = & \mathbf{f}(\text{proj}_{[\theta]}\theta_L, \mathbf{m}_1) & \text{where } \theta_L = \text{atan}(\frac{m_{1y}}{m_{1x}}) \end{array}$$

where $\text{proj}_{[\theta]}\alpha$ is the element in $[\theta]$ which is the nearest to α.

The corresponding edges are

$$\begin{array}{ll} \text{segm}(\mathbf{a}_1, \mathbf{a}_2) & \\ \text{segm}(\mathbf{a}_2, \mathbf{a}_3) & \\ \text{arc}(\mathbf{a}_3, \mathbf{a}_4) & \text{if } \theta^- < \theta_R \\ \text{arc}(\mathbf{a}_4, \mathbf{a}_5) & \text{if } \theta_R < \theta^+ \\ \text{segm}(\mathbf{a}_5, \mathbf{a}_6) & \\ \text{segm}(\mathbf{a}_6, \mathbf{a}_7) & \\ \text{arc}(\mathbf{a}_7, \mathbf{a}_8) & \text{if } \theta_L < \theta^+ \\ \text{arc}(\mathbf{a}_4, \mathbf{a}_5) & \text{if } \theta^- < \theta_L \end{array} \tag{54}$$

where $\text{segm}(\mathbf{a}_i, \mathbf{a}_j)$ is the segment which links the two vertices $\mathbf{a}_i$ and $\mathbf{a}_j$. Note that some arcs may not exist when the corresponding conditions (see the right hand side of (54)) are not satisfied. Take a box $[\mathbf{p}]$ as shown in Figure 14(b). We want to compute the contracted box $\mathcal{C}_{\mathbb{P}}([\mathbf{p}]) = [\![[\mathbf{p}] \cap \mathbb{P}]\!]$ painted red. For this purpose, we compute the contracted boxes associated to each existing arc or segment. We get all small boxes represented in Figure 14(c). The contracted box $\mathcal{C}_{\mathbb{P}}([\mathbf{p}])$ is is obtained by taking the interval hull of all these boxes, as shown in Figure 14(d). To be more correct, the procedure does not compute $\mathcal{C}_{\mathbb{P}}([\mathbf{p}])$. Instead, it computes the optimal contractor $\mathcal{C}_{\partial\mathbb{P}}([\mathbf{p}])$ on the boundary $\partial\mathbb{P}$ of $\mathbb{P}$. This is illustrated by Figure 14(e), where we use the contractor $\mathcal{C}_{\partial\mathbb{P}}$ inside a paver. To get the contractor $\mathcal{C}_{\mathbb{P}}([\mathbf{p}])$ for $\mathbb{P}$ or the contractor $\mathcal{C}_{\overline{\mathbb{P}}}([\mathbf{p}])$ for the complementary set $\overline{\mathbb{P}}$ of $\mathbb{P}$, we need the test the membership of corners of $[\mathbf{p}]$. This allows us to get an inner and an outer approximation of the set $\mathbb{P}$ as illustrated by Figure 14(f).

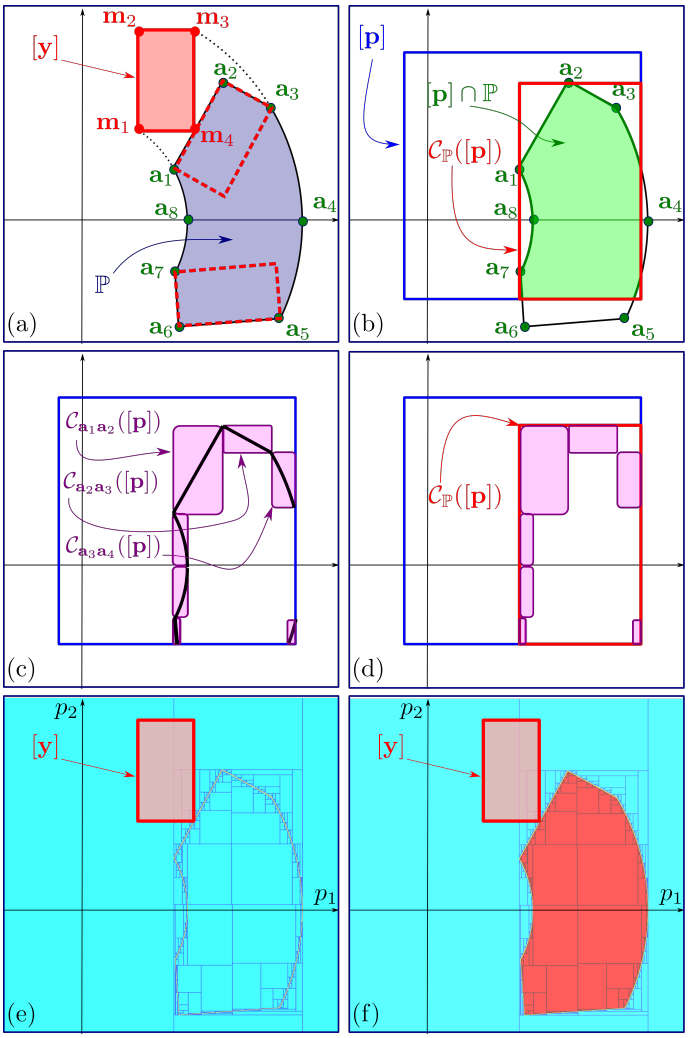

Fig. 14: Angle contractor

5.3 Generator

There is no general procedure to build efficient contractors for monotonic constraints. A possibility is to use the contractor based on the monotonicity given in [1] [8] but the method is limited to the case of a single monotonic constraint. The use of the axis convexity has been been proposed in [38], but the axis property is not always easy to prove and the combination of two axis-convex constraints is not always axis-convex.

Now, for many monotonic set of constraints, the monotonicity property helps to build analytically the minimal contractor, but a rigorous analysis has to be

performed, similar to that presented in Subsection 5.2. Most of the time, we have to proceed componentwise.

To illustrate the procedure, consider the constraint (49) and take as a generator, the box

$$[\mathbf{a}] = \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+. \tag{55}$$

To get a minimal contractor on $[\mathbf{a}]$, we need (see Theorem 3) an expression for the contractor

$$\mathcal{C}_0 : [\mathbf{x}] \to [\![[\mathbf{x}] \cap [\mathbf{a}] \cap \mathbb{X}]\!], \tag{56}$$

for instance under the form of an algorithm. Assume that we want to get the two first components of $\mathcal{C}_0$. We rewrite the *rotate constraint* as

$$\begin{array}{ll} (i) & \begin{pmatrix} x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ (ii) & \begin{cases} 0 &= x_3 - x_4 \cdot \tan\theta \\ 0 &= x_3^2 + x_4^2 - 1 \end{cases} \end{array} \tag{57}$$

Since the only variable shared by *(i)* and *(ii)* is θ, the combination of *(i)* and *(ii)* will not generate a wrapping effect. Equivalently, if we have a minimal contraction $[\theta_i]$ for *(i)* and a minimal contraction $[\theta_{ii}]$ for *(ii)*, then the intersection $[\theta_i] \cap [\theta_{ii}]$ corresponds to the minimal contraction for the conjunction of *(i)* and *(ii)*. As a consequence, the interval hull of the set

$$\{(x_1, x_2) \in [x_1], [x_2] \,|\, \exists x_i \in [x_i] \subset [a_i], i \in \{3, \ldots, 6\}, \text{rotate}(\mathbf{x})\}$$

is computed by the following algorithm:

Input: $([x_1], \ldots, [x_6]) \subset [\mathbf{a}]$
Step 1. $[x_3] = [x_3] \cap \sqrt{1 - [x_4]^2}$
Step 2. $[x_4] = [x_4] \cap \sqrt{1 - [x_3]^2}$
Step 3. $[\theta] = \left[\text{atan}\frac{x_4^-}{x_3^+}, \text{atan}\frac{x_4^+}{x_3^-}\right]$
Step 4. $[\mathbf{p}] = [x_1] \times [x_2]$; $[\mathbf{y}] = [x_5] \times [x_6]$
Step 5. $[x_1], [x_2] = \mathcal{C}_{\text{angle}}^{[\theta],[\mathbf{y}]}([\mathbf{p}])$
Return $([x_1], [x_2])$

In this algorithm, $\mathcal{C}_{\text{angle}}^{[\theta],[\mathbf{y}]}([\mathbf{p}])$ is the *angle* contractor developed in Section 5.2.

A similar work has to be developed for the contraction of the four other component of C_0:x_3, x_4, x_5, x_6.

5.4 Computing the symmetries of the rotate constraint

Since the *rotate* constraint involves 6 variables, we have to consider the hyperoctahedral group B_6 which contains $2^6 \cdot 6! = 46080$ elements. Take one of

them:

$$\sigma_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{58}$$

It has been shown in Section 2.3 that σ_1 is a stabilizer for (49), *i.e.*,

$$\left\{ \begin{array}{rcl} x_3x_1 - x_4x_2 - x_5 & = & 0 \\ x_4x_1 + x_3x_2 - x_6 & = & 0 \\ x_3^2 + x_4^2 - 1 & = & 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcl} x_3x_5 + x_4x_6 - x_1 & = & 0 \\ -x_4x_5 + x_3x_6 - x_2 & = & 0 \\ x_3^2 + (-x_4)^2 - 1 & = & 0 \end{array} \right. \tag{59}$$

This equivalence is illustrated geometrically by Figure 15. The associated symmetry means that if we permute $\mathbf{u} = (x_1, x_2)$ and $\mathbf{v} = (x_5, x_6)$ then the angle θ between the two vectors changes its sign. The red vector corresponds to the angle θ on the trigonometric circle.

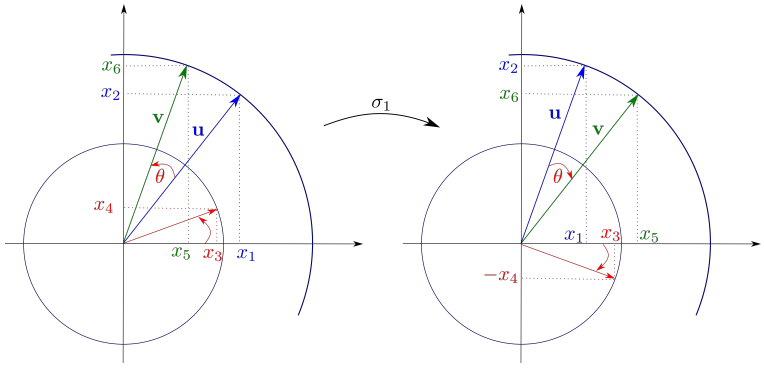

Fig. 15: One possible symmetry for the *rotate* constraint

Other elements of $B_6(\mathbb{X})$ could be found. If we add the four following symmetries

$$\begin{array}{rrcl} \sigma_2: & (x_1, x_2, x_3, x_4, x_5, x_6) & \mapsto & (x_2, -x_1, -x_4, x_3, x_5, x_6) \\ \sigma_3: & (x_1, x_2, x_3, x_4, x_5, x_6) & \mapsto & (x_1, -x_2, x_3, -x_4, x_5, -x_6) \\ \sigma_4: & (x_1, x_2, x_3, x_4, x_5, x_6) & \mapsto & (-x_1, -x_2, -x_3, -x_4, x_5, x_6) \\ \sigma_5: & (x_1, x_2, x_3, x_4, x_5, x_6) & \mapsto & (-x_1, x_2, x_3, -x_4, -x_5, x_6) \end{array} \tag{60}$$

we can check that we are able to generate $B_6(\mathbb{X})$. If we run the algorithm FindSequence, we get that it is impossible to validate any sequence with a length strictly smaller than 5. We were able to validate 54 sequences of length 5 among the 5!=720 existing ones. One of them is $\sigma_5 \star \sigma_4 \star \sigma_3 \star \sigma_2 \star \sigma_1$. As a consequence, a minimal contractor is given by $\mathcal{C}^* = (\sigma_5 \star \sigma_4 \star \sigma_3 \star \sigma_2 \star \sigma_1) \bullet \mathcal{C}_0$.

5.5 Illustration

Consider the set of all $\mathbf{p} \in \mathbb{R}^2$, such that Constraint (47) is satisfied with $\theta \in [\theta] = [1, 5]$, $\mathbf{y} \in [\mathbf{y}] = [-4, -2] \times [10, 14]$. The angle θ is expressed in radian. More precisely, we want to characterize the set

$$\mathbb{P} = \{(p_1, p_2) \,|\, \exists \theta \in [\theta], \exists \mathbf{y} \in [\mathbf{y}],\ \mathbf{y} = \mathbf{R}_\theta \cdot \mathbf{p}\}\,. \tag{61}$$

To apply the minimal contractor $\mathcal{C}^* = (\sigma_5 \star \sigma_4 \star \sigma_3 \star \sigma_2 \star \sigma_1) \bullet \mathcal{C}_0$ developed previously, we set $[x_3] = \cos([\theta])$ and $[x_4] = \sin([\theta])$ and we make the transformation

$$(p_1, p_2, y_1, y_2) \mapsto (x_1, x_2, x_5, x_6). \tag{62}$$

For a comparison, we tested several algorithms and the results are depicted on Figure 16. The frame box is $[\mathbf{p}] = [-20, 20] \times [-20, 20]$. The blue boxes are all proved to have an empty intersection with $\mathbb{P}$. All yellow boxes have a width smaller than 0.5.

Method 1 (Figure 16, Left). We bisect on the (p_1, p_2)-space and we used the HC4-revised contractor [3] which can be considered as the state of the art. Since the contractor is not minimal, the approximation is poor.

Method 2 (Figure 16, Center). We still use the HC4-revised contractor, but we also bisect on the (x_3, x_4, x_5, x_5)-space to control the precision. Due to the fact that the projection algorithm bisects in $\mathbb{R}^6$, the computing time is large (more than three minutes).

Method 3. (Figure 16, Right). We bisect on the (p_1, p_2)-space and we call the minimal contractor $\mathcal{C}^*$. We obtain the approximation in less than 2 sec. Since the projection of a minimal contractor is minimal (see Proposition 1), we get a minimal contractor for $\mathbb{P}$. This can be checked on the figure by the fact that rejected blue boxes touch at least one tiny yellow box. Moreover, due to the minimality of the contractor, we know that each of the yellow boxes contains at least one point of $\mathbb{P}$. Equivalently, we can claim that the Hausdorff distance between our approximation (yellow) and $\mathbb{P}$ is less than the diameter of the yellow boxes (here 0.5).

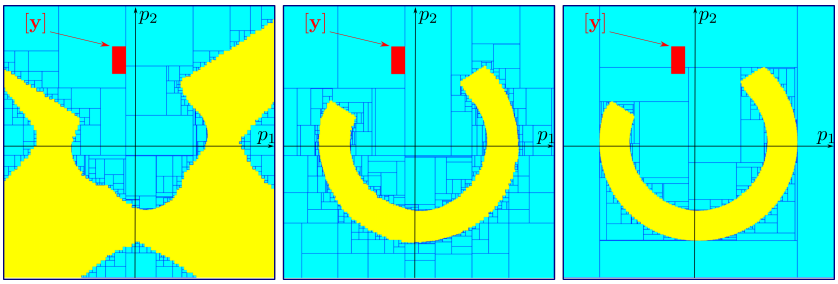

Fig. 16: Approximation of $\mathbb{P}$ obtained by different methods. Left: the classical HC4-revised contractor, Center: Using a projection-based paver, Right: the minimal contractor for the *rotate* constraint

6 Application

Consider three radars at locations $\mathbf{a}_i$ observing a robot at position $\mathbf{x}$ in a 2 dimensional world. The ith radar is able to measure its distance d_i to the robot using the time of flight. Using the Doppler effect, it is also able to collect a measure of $\dot{d}_i$, the time derivative of d_i. The angle α_i is measured with a poor accuracy (here 1 rad) and corresponds to the direction of arrival of the echo of the radio-wave on the robot. The data collected at a given time instant are given by the following table.

i	1	2	3
$\mathbf{a}(i)$	$(-10, 0)$	$(5, 0)$	$(10, 0)$
$d(i) \in$	$[28, 30]$	$[20, 22]$	$[19, 21]$
$\dot{d}(i) \in$	$[42, 44]$	$[29, 31]$	$[20, 22]$
$\alpha(i) \in$	$[0, 1]$	$[1, 2]$	$[1, 2]$

Figure 17 represents the robot (red) with the 3 radars (on the same horizontal line). In yellow is represented the set of all feasible positions for the robot that we want to compute. The frame box is $[-30, 30] \times [-5, 30]$.

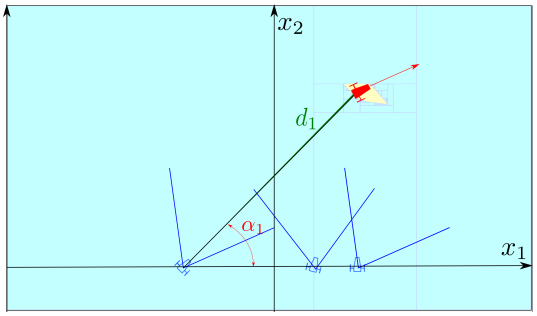

Fig. 17: Three radars observing a robot (red) in the 2D world

For the ith radar measurements, we have

$$\mathbf{x} - \mathbf{a}_i = d_i \cdot \begin{pmatrix} \cos\alpha_i \\ \sin\alpha_i \end{pmatrix} \tag{63}$$

and

$$\dot{\mathbf{x}} = \begin{pmatrix} \cos\alpha_i & -\sin\alpha_i \\ \sin\alpha_i & \cos\alpha_i \end{pmatrix} \cdot \begin{pmatrix} \dot{d}_i \\ d_i \cdot \dot{\alpha}_i \end{pmatrix}. \tag{64}$$

We thus get the system of constraints

$$\begin{aligned} \mathbf{p}_i &= \mathbf{x} - \mathbf{a}_i && \forall i \in \{1, \ldots, 3\} \\ c_i &= \cos\alpha_i \\ s_i &= \sin\alpha_i \\ &\text{rotate}(d_i, 0, c_i, s_i, p_{i1}, p_{i2}) \\ &\text{rotate}(\dot{d}_i, w_i, c_i, s_i, \dot{x}_1, \dot{x}_2) \end{aligned} \tag{65}$$

In the previous equation, $w_i = \dot{\alpha}_i$ is not measured. Thus, its domain will be set to $[w_i] = [-\infty, \infty]$.

We apply the corresponding contractors inside a paver and we get Figure 18, Left, in the x_1, x_2-space in less than 5 sec. The tiny yellow boxes enclose the solution set. All these boxes have a width smaller than 0.5. The frame box is $[-30, 30] \times [-5, 30]$. Note that due to the fact that the projection of minimal contractors is a minimal contractor (see Proposition 1), there is no need to bisect with respect to the variables $c_i, s_i, \mathbf{p}_i, d_i$, *i.e.*, the procedure will characterize the solution set $\mathbb{X}$ with an arbitrary precision as soon as we accept to bisect in the $\mathbf{x}$-space sufficiently. It is not the case with a classical interval method which yields Figure 18, Right with a similar computing time. We could improve the accuracy of the characterization but for this, the classical interval method will have to bisect with respect to variables $c_i, s_i, \mathbf{p}_i, d_i$. In such a case, we could obtain a result comparable to the left figure, but in a time longer than 10 minutes.

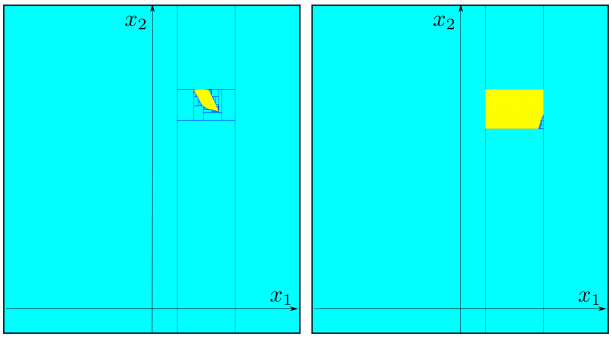

Fig. 18: Left: set obtained using the symmetry-based contractor; Right using classical interval contractors

7 Conclusion

Contractor methods are particularly attractive when solving engineering applications, due to the fact that they can handle and propagate uncertainties. The results are guaranteed even if the problem is non-linear and non-convex. Now, the performances of paving methods are extremely sensitive to the accuracy of the contractors.

In this paper, we proposed a new approach that could help to build more easily minimal contractors associated with some specific constraints containing hyperoctahedral symmetries. The resulting minimal contractors could not have been built using existing interval approaches.

The main contribution of this paper is to build a bridge between the hyperoctahedral group of symmetries and interval analysis. To our knowledge, this link has not been done before and is proven here to be efficient to compute outer approximation of the solution set of non linear problems. This efficiency has

been illustrated on the problem of the bearing localization of a robot.

A perspective of the work would be to build minimal contractors for ubiquitous constraints which contain hyperoctahedral symmetries such as the matrix multiplication $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$. This could drastically improve the efficiency of existing interval solvers such as [19] or [39].

Note. The `Python` programs associated with all examples are given in [24].

References

[1] I. Araya, G. Trombettoni, and B. Neveu. Exploiting monotonicity in interval constraint propagation. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010. 5.3

[2] M. Baake. Structure and representations of the hyperoctahedral group. *J. Math. Phys.*, 25(11), 1984. 1

[3] F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising Hull and Box Consistency. In *ICLP*, pages 230–244, 1999. 5.5

[4] B. Buchberger. Theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976. 2.3

[5] A. Cayley. On the theory of groups, as depending on the symbolic equation $\theta^n = 1$. *Philosophical Magazine*, 7:40–47, 1854. 2.1

[6] M. Cébério and L. Granvilliers. Solving nonlinear systems by constraint inversion and interval arithmetic. In *Artificial Intelligence and Symbolic Computation*, volume 1930, pages 127–141, LNCS 5202, 2001. 1

[7] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009. 1

[8] G. Chabert and L. Jaulin. Hull consistency under monotonicity. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 188–195. Springer, 2009. 5.3

[9] A. H. Clifford and G. B. Preston. The algebraic theory of semigroups. *American Mathematical Society*, 1961. 3.1

[10] E. Colle and S. Galerne. Mobile robot localization by multiangulation using set inversion. *Robotics and Autonomous Systems*, 61(1):39–48, 2013. 1

[11] H. Coxeter. The complete enumeration of finite groups of the form $r_i^2 = (r_i \cdot r_j)^{k_{ij}} = 1$. *Journal of the London Mathematical Society*, 10(1):21–25, 1935. 2.1, 2.4

[12] H. Coxeter. *The Beauty of Geometry: Twelve Essays*. Dover Books on Mathematics, 1999. 2.1

[13] P.A. Crépon, A. Panchea, and A. Chapoutot. Reliable motion plannning for a mobile robot. In *IEEE International Conference on Robotic Computing*, 2018. 1

[14] D. Daney, N. Andreff, G. Chabert, and Y. Papegay. Interval Method for Calibration of Parallel Robots: Vision-based Experiments. *Mechanism and Machine Theory, Elsevier*, 41:926–944, 2006. 1

[15] B. Desrochers and L. Jaulin. A minimal contractor for the polar equation; application to robot localization. *Engineering Applications of Artificial Intelligence*, 55:83–92, 2016. 2.1, 3.2, 5.1

[16] M. Di Marco, A. Garulli, S. Lacroix, and A. Vicino. Set membership localization and mapping for autonomous navigation. *International Journal of Robust and Nonlinear Control*, 7(11):709–734, 2001. 1

[17] J. Alexandre dit Sandretto, G. Trombettoni, D. Daney, and G. Chabert. Certified calibration of a cable-driven robot using interval contractor programming. In F. Thomas and A. Pérez Gracia, editors, *Computational Kinematics, Mechanisms and Machine Science*, Spinger, 2014. 1

[18] A. Ehambram, R. Voges, and B. Wagner. Stereo-visual-lidar sensor fusion using set-membership methods. In *17th IEEE International Conference on Automation Science and Engineering, CASE 2021, Lyon, France, August 23-27, 2021*, pages 1132–1139. IEEE, 2021. 1

[19] S. Rohou et. al. *Codac (Catalog Of Domains And Contractors), available at* `http://codac.io/`. Robex, Lab-STICC, ENSTA-Bretagne, 2021. 7

[20] I.P. Gent, K.E. Petrie, and J.F. Puget. Symmetry in constraint programming. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 329–376. Elsevier, 2006. 1

[21] A. Goldsztejn, C. Jermann, V. Ruiz de Angulo, and C. Torras. Variable symmetry breaking in numerical constraint problems. *Artif. Intell.*, 229:105–125, 2015. 1

[22] E. Goubault and S. Putot. Robust under-approximations and application to reachability of non-linear control systems with disturbances. *IEEE Control. Syst. Lett.*, 4(4):928–933, 2020. 1

[23] R. Guyonneau, S. Lagrange, and L. Hardouin. A visibility information for multi-robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. 1

[24] L. Jaulin. Codes associated to the paper on hyperoctohedral symmetries. `www.ensta-bretagne.fr/jaulin/hyperoctohedral.html`, 2022. 2.3, 7

[25] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Guaranteed mobile robot tracking using interval analysis. In *Proceedings of the MISC'99 Workshop on Applications of Interval Analysis to Systems and Control*, pages 347–359, Girona, Spain, 1999. 1

[26] V. Kreinovich, A.V. Lakeyev, J. Rohn, and P.T. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Springer, 1997. 1

[27] M. Langerwisch and B. Wagner. Guaranteed mobile robot tracking using robust interval constraint propagation. *Intelligent Robotics and Applications, Springer*, 7507:354–365, 2012. 1

[28] T. Le Mézo, L. Jaulin, and B. Zerr. Bracketing backward reach sets of a dynamical system. *International Journal of Control*, 2020. 1

[29] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966. 1

[30] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009. 1

[31] M. Mustafa, A. Stancu, N. Delanoue, and E. Codres. Guaranteed SLAM; An Interval Approach. *Robotics and Autonomous Systems*, 100:160–170, 2018. 1

[32] R. Neuland, J. Nicola, R. Maffei, L. Jaulin, E. Prestes, and M. Kolberg. Hybridization of monte carlo and set-membership methods for the global localization of underwater robots. In *IROS 2014*, pages 199–204, 2014. 1

[33] N. Ramdani and P. Poignet. Robust dynamic experimental identification of robots with set membership uncertainty. *IEEE/ASME Transactions on Mechatronics*, 10(2):253–256, 2005. 1

[34] A. Rauh and E. Auer. Interval approaches to reliable control of dynamical systems. In B.M. Brown, E. Kaltofen, S. Oishi, and S.M. Rump, editors, *Computer-assisted proofs - tools, methods and applications, 15.11. - 20.11.2009*, volume 09471 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. 1

[35] A. Rauh and E. Auer (ed.). *Modeling, Design, and Simulation of Systems with Uncertainties*. Springer, 2011. 1

[36] S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, and S. Veres. *Reliable robot localization; A Constraint-Programming Approach Over Dynamical Systems*. ISTE, WILEY, 2019. 1

[37] R. Sainudiin. *Machine Interval Experiments: Accounting for the Physical Limits on Empirical and Numerical Resolutions*. LAP Academic Publishers, Köln, Germany, 2010. 1

[38] D. Sam-Haroud. *Constraint consistency techniques for continuous domains*. PhD dissertation 1423, Swiss Federal Institute of Technology in Lausanne, Switzerland, 1995. 5.3

[39] J. Alexandre Dit Sandretto and A. Chapoutot. Dynibex: a differential constraint library for studying dynamical systems. In *Conference on Hybrid Systems: Computation and Control*, Vienne, Austria, 2016. 7

[40] P. van Hentenryck, Y. Deville, and L. Michel. *Numerica: A Modeling Language for Global Optimization*. MIT Press, Boston, MA, 1997. 1

[41] J. Wan. *Computationally reliable approaches of contractive model predictive control for discrete-time systems*. PhD dissertation, Universitat de Girona, Girona, Spain, 2007. 1