Constrained adaptive parallelepipedic approximation of the image of a set by a nonlinear function*

Maël Godard[a,*], Luc Jaulin[a], Damien Massé[b]

[a] *Lab-STICC, ROBEX Team, ENSTA, 2 rue Francois Verny, Brest, 29200, France*
[b] *Lab-STICC, ROBEX Team, Université de Bretagne Occidentale, 20 avenue Le Gorgeu, Brest, 29200, France*

Abstract

This paper proposes a method to compute an adaptive outer approximation of the image of a set by a function. This approximation relies on a cover of the boundary of the image set with parallelepipeds.

Since the boundary is generally not a parallelepiped, covering it comes down to enclosing it in an union of parallelepipeds. The fewer parallelepipeds used, the more pessimistic the approximation becomes. If we want to validate that the image set respects some constraints, excessive pessimism in the cover can lead to incorrect conclusions. Increasing the number of parallelepipeds significantly reduces the pessimism, but also increases the computationnal complexity.

The method presented here constructs an adaptive covering of the boundary of the image set. The objective is to verify that the image set satisfies given constraints while saving computationnal effort where possible. In the cases where the constraints can not be validated, we are also able to provide a precise detection of the parts of the initial and image sets boundaries responsible for it.

Keywords: Parallelepipeds, Boundary approach, Adaptive method, Constraint Validation

*This work was supported by the Defense Innovation Agency (AID) and the Brittany Region.

*Corresponding author. E-mail addresses: mgodard00@outlook.com, mael.godard@ensta.fr

1. Introduction

Sets are a common way to represent and propagate uncertainties[1, 2]. One of the basic problems that can be solved by using sets is the *direct problem*. Let us consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $\mathbf{x} \in \mathbb{R}^n$. If $\mathbf{x}$ is known with a certain uncertainty we can define a set $\mathbb{X} \subset \mathbb{R}^n$ such that $\mathbf{x}$ is guaranteed to lie inside $\mathbb{X}$. The objective of the *direct problem* is to characterize $\mathbb{Y}$, the image of the set $\mathbb{X}$ by the function $\mathbf{f}$, i.e. $\mathbb{Y} = \mathbf{f}(\mathbb{X})$. By doing so, we can assert that the real image vector $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is inside $\mathbb{Y}$.

Guaranteed ODE integration [3, 4], nonlinear observer design [5, 6] and reachability analysis [7, 8] are common *direct problems* for which set methods have proven to be efficient. These set methods provide an outer - and sometime an inner - approximation of the image set. In particular, the outer approximation of $\mathbb{Y}$ is a set larger than $\mathbb{Y}$ that is guaranteed to contain it.

Interval methods [9, 10] offer an easy way to compute an outer approximation of $\mathbb{Y}$. However, interval representations are subject to high conservatism due to their simplicity. . In a validation context, i.e. if we want to ensure that the set $\mathbb{Y}$ satifsies a given constraint, this large pessimism can be an issue.

A first solution to palliate this issue is to subdivide the initial set $\mathbb{X}$ into smaller sets $\mathbb{X}_i$. For each of these sets we can compute an outer approximation of its image $\mathbb{Y}_i^+$. Finally an outer approximation of $\mathbb{Y}$ can be obtained by computing the union $\bigcup_i \mathbb{Y}_i^+$. This idea is for example used in branch and bound algorithms [11] and for the computation of global attractors [12]. The result is more accurate than with a single computation, but the computation time can increase significantly.

Other types of sets can then be considered such as polytopes [13], zonotopes [14] or ellipsoids [15]. There is a compromise in set methods between the simplicity of the representation and the pessimism of the result. For example polytopes are more complex than zonotopes, but they are less pessimistic. This is why some works [16, 17] rely on both to benefit from their respective advantages.

Parallelepipeds, or parallelotopes, can be seen as simple instances of zonotopes. In the litterature [18], they are presented as a compromise between the simple but overly pessimistic boxes, and tight but computationnaly more expensive zonotopes. In [19], parallelotope bundles are introduced in a context of rechability analysis to improve the tightness of the enclosure at low computationnal cost. More recently, [20] proposed an adaptive method that

dynamically adjusts the template direction of such bundles for an even tighter enclosure.

In contrast, the contribution of this work is not to compute the tightest enclosure possible using parallelepipeds, but to develop an adaptive scheme to validate a set of constraints. As a result, the parallelepiped covering is refined only where needed to ensure that the constraints are satisfied. In addition, when the constraints can not be verified by our method, we are able to provide a precise enclosure of the parts of the initial set boundary responsible for it.

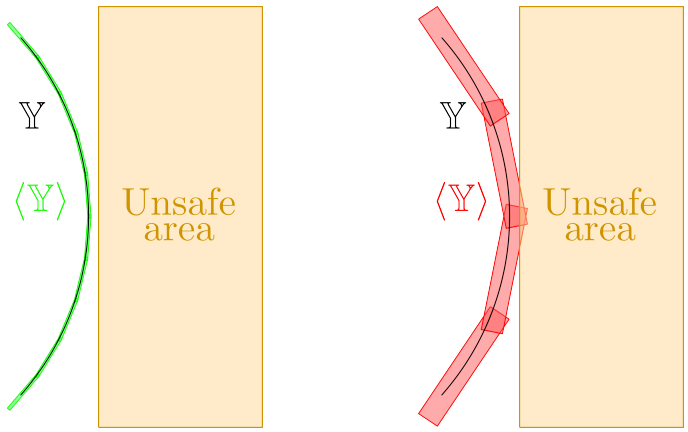
We propose a method to compute the direct image of a set by a nonlinear function while asserting that it satisfies a given constraint. A boundary approach will be used to limit as much as possible the pessimism in the computations. In addition, parallelepipeds will be used to cover the boundary of the image set. Finally an adaptive slicing of the initial set will help to refine the outer approximation of the image set only where it is needed to save computation time.

Our main contributions are the Ad-PEIB and Ad-PEIBOS algorithms. The Ad-PEIB algorithm computes the parallelepiped enclosure of the image of a box. In the case where symmetries exist in the initial set, the Ad-PEIBOS algorithm adds the notion of gnomonic atlas [21] to simplify the analysis. Thanks to the gnomonic atlas, the Ad-PEIB algorithm can be used.

The paper is organized as follows. Section 2 exposes the motivation of this work. Section 3 introduces the notions that will be used later on. In particular it defines the differents sets, zonotopes and parallelepipeds, and introduces the notion of contractor to handle the constraints. Section 4 introduces the main contribution of this work in the form of two algorithms. One is the general algorithm for the computation of the image of a set. The other one relies on a gnomonic atlas to get a better result when the initial set possesses symmetries. Section 5 shows how the parallelepiped inclusion function from [21] can be extended to dynamical systems with an input trajectory. Two example will be displayed to show the effectiveness of the Ad-PEIBOS algorithm. Section 6 concludes the paper.

2. Motivation

Consider an initial set $\mathbb{X} \subset \mathbb{R}^n$ and a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. We will denote by $\mathbb{Y} = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathbb{X}\}$ the image set. Let us consider the case depticted



(a) No false alarm (b) False alarm

Figure 1: Outer approximation of $\mathbb{Y}$ next to an unsafe area

in Figure 1 where we want to ensure that the image set $\mathbb{Y}$ does not cross an unsafe area. This could for example be the computation of the workspace of a robotic arm which operates next to a wall.

As the exact computation of $\mathbb{Y}$ is not possible, set-based methods can be used to get an outer approximation of this set noted $\mathbb{Y}^+$. For this paper we will focus on the use of parallelepipeds. In the remainder of this section, we will denote by $\langle\mathbb{Y}\rangle$ the parallelepiped outer approximation of $\mathbb{Y}$.

Figure 1 shows two possible parallelepipedic enclosures of $\mathbb{Y}$ for two different resolutions. In the perfect case (in black on both sides) we can see that the set does not cross the unsafe area. On Figure 1a the pessimism of the parallelepipedic approximation (in green) is small enough to not cross the unsafe area. On Figure 1b the parallelepipedic approximation (in red) crosses the unsafe area, thus raising a false alarm.

A solution to avoid these false alarms would be to use the best resolution everywhere. However, this solution implies an increase in the computation time. This increase gets even worse as the number of dimensions increases. In addition, some areas far from the unsafe area do not need a fine resolution as they are less likely to raise a false alarm.

The idea presented in this paper is then to use a low resolution everywhere, and to refine the approximation where needed. This is depicted on Figure 2 where two different resolutions are used to get an adaptive parallelepipedic enclosure of $\mathbb{Y}$. In this case we can see that $\langle\mathbb{Y}\rangle$ does not cross the unsafe area and so no "false alarm" is raised.

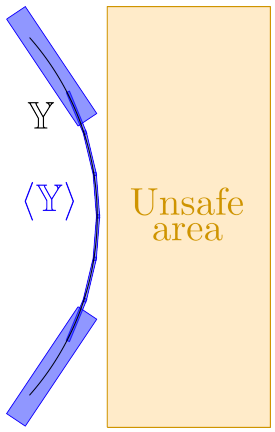


Figure 2: Adaptive parallelepipedic enclosure

3. Notations and definitions

This section aims to provide the notation and definitions needed for the remaining of the article. Parallelepipeds and zonotopes are defined, and the notion of contractors is introduced to handle the desired constraints.

3.1. Parallelepiped inclusion function

Considering a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, computing the exact image of a set by $\mathbf{f}$ is not always possible. To address this issue, set-based methods are often used to give an inner and outer approximation of the image set.

Definition 1. Considering a set $\mathbb{Y} \subset \mathbb{R}^m$, an inner and an outer approximation of $\mathbb{Y}$ are respectively two sets $\mathbb{Y}^- \subset \mathbb{R}^m$ and $\mathbb{Y}^+ \subset \mathbb{R}^m$ such that $\mathbb{Y}^- \subseteq \mathbb{Y} \subseteq \mathbb{Y}^+$.

Intervals have often been used to perform such tasks. The set of intervals is denoted $\mathbb{IR}$, and the cartesian product of n intervals is an interval vector, or a box. The set of n-dimensional boxes is denoted $\mathbb{IR}^n$. Interval arithmetics offers an easy way to compute the image of a set. Yet the result obtained by such methods is pessimistic, notably due to the so called *wrapping effect* as depicted Figure 3.

To limit this wrapping effect, other types of sets have been considered.

Definition 2. A zonotope is a convex and symmetric polytope. A zonotope $\mathcal{Z}$ of $\mathbb{R}^m$ can be described by its center $\mathbf{p} \in \mathbb{R}^m$ and its shape matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ as

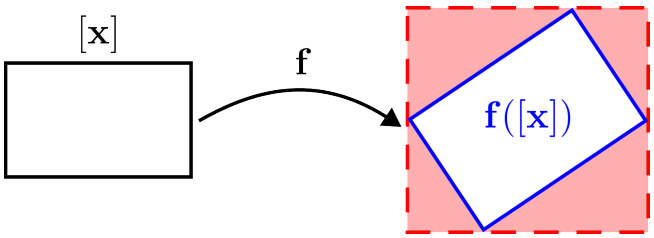


Figure 3: Wrapping effect : The red box containing $\mathbf{f}([\mathbf{x}])$ is larger

$$\mathcal{Z} = \mathbf{p} + \mathbf{A} \cdot [-1, 1]^n \tag{1}$$

The columns of the shape matrix $\mathbf{A}$ are called the generators of the zonotope.

The propagation of these zonotopes can be expensive due to the increase in the number of generators at each step [22]. In the rest of this paper, parallelepipeds will be considered as a compromise between boxes and zonotopes. In what follows the set of parallelepipeds of $\mathbb{R}^m$ will be denoted $\mathbb{PR}^m$.

Definition 3. A parallelepiped $\langle \mathbf{y} \rangle \in \mathbb{PR}^m$ is a zonotope in the case where $n \leq m$. It can be described by its center $\bar{\mathbf{y}} \in \mathbb{R}^m$ and its shape matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, such that

$$\langle \mathbf{y} \rangle = \bar{\mathbf{y}} + \mathbf{A} \cdot [-1, 1]^n \tag{2}$$

Note that in the case where $n < m$ the parallelepiped is sometimes referred as a *flat parallelepiped*.

We can then define the notion of parallelepiped inclusion function.

Definition 4. Consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. A *parallelepiped inclusion function* of $\mathbf{f}$ noted $\langle \mathbf{f} \rangle$ is a function :

$$\langle \mathbf{f} \rangle : \begin{array}{ccc} \mathbb{IR}^n & \to & \mathbb{PR}^m \\ [\mathbf{x}] & \to & \langle \mathbf{f} \rangle([\mathbf{x}]) \end{array} \tag{3}$$

such that

$$\mathbf{f}([\mathbf{x}]) \subset \langle \mathbf{f} \rangle([\mathbf{x}]). \tag{4}$$

Zonotopes and parallelepipeds can easily be projected as follows

Proposition 1. *Consider a zonotope* $\mathcal{Z} = \mathbf{p} + \mathbf{A} \cdot [-1, 1]^n$ *of* $\mathbb{R}^m$*. It is defined by its center* $\mathbf{p} \in \mathbb{R}^m$ *and its shape matrix* $\mathbf{A} \in \mathbb{R}^{m \times n}$*. Let* $(x_1, \cdots, x_m)$ *be the standard basis of* $\mathbb{R}^m$ *and* $\{i_1, \cdots, i_p\} \in [\![1, m]\!]^p$ *be* p *different indexes. The projection of* $\mathcal{Z}$ *on the* p*-dimensional subspace of* $\mathbb{R}^m$ *defined by the* p *vectors* $\left(x_{i_1}, \cdots, x_{i_p}\right)$ *is a zonotope* $\mathcal{Z}_p$ *defined by* $\mathbf{z} \in \mathbb{R}^p$ *and* $\mathbf{B} \in \mathbb{R}^{p \times n}$*:*

$$\begin{cases} \mathcal{Z}_p = \mathbf{z} + \mathbf{B} \cdot [-1, 1]^n \\ \qquad \mathbf{z} = \begin{pmatrix} \mathbf{p}_{i_1} \\ \vdots \\ \mathbf{p}_{i_p} \end{pmatrix} \\ \qquad \mathbf{B} = \begin{pmatrix} \mathbf{A}_{i_1} \\ \vdots \\ \mathbf{A}_{i_p} \end{pmatrix} \end{cases} \tag{5}$$

Where for $j \in [\![1, n]\!]$*,* $\mathbf{p}_j$ *and* $\mathbf{A}_j$ *are respectively the* j*-th row of* $\mathbf{p}$ *and* $\mathbf{A}$*.*

In other words, projecting a zonotope comes down to projecting its center and each of its generators on the desired subspace.

Remark 1. As a parallelepiped is a zonotope, Proposition 1 also applies to them. In addition the projection of a parallelepiped is a zonotope.

3.2. Constraints

The objective of the method presented later in this paper is to compute an adaptive enclosure of a set to ensure that it respects some criteria. These criteria can be formalised in the form of a set of constraints.

Definition 5. A constraint $\mathcal{L}$ is a boolean function defined by :

$$\begin{array}{rccc} \mathcal{L}: & \mathbb{R}^m & \to & \{0, 1\} \\ & \mathbf{y} & \mapsto & \begin{cases} 1 & \text{if } \mathbf{y} \text{ satisfies the constraint} \\ 0 & \text{Otherwise} \end{cases} \end{array} \tag{6}$$

Remark 2. This definition can be extended to parallelepipeds (or any other set). A constraint is validated on a set if it is validated in every point of it.

$$\begin{array}{rccc} \mathcal{L}: & \mathcal{P}\left(\mathbb{R}^m\right) & \to & \{0, 1\} \\ & \mathbb{Y} & \mapsto & \begin{cases} 1 & \text{if } \forall \mathbf{y} \in \mathbb{Y},\ \mathcal{L}(\mathbf{y}) = 1 \\ 0 & \text{otherwise} \end{cases} \end{array} \tag{7}$$

Two constraints or more can be combined. If we consider two constraints $\mathcal{L}_1$ and $\mathcal{L}_2$, we can define their intersection $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$ as

$$\mathcal{L}(\mathbf{y}) = \begin{cases} 1 & \text{if } (\mathcal{L}_1(\mathbf{y}) \wedge \mathcal{L}_2(\mathbf{y})) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

We can also define the complementary $\bar{\mathcal{L}}$ of the constraint $\mathcal{L}$

$$\begin{array}{rccl} \bar{\mathcal{L}}: & \mathbb{R}^m & \to & \{0,1\} \\ & \mathbf{y} & \mapsto & \left\{ \begin{array}{ll} 1 & \text{if } \mathcal{L}(\mathbf{y}) = 0 \\ 0 & \text{Otherwise} \end{array} \right. \end{array} \tag{9}$$

Definition 6. A contractor [23] $\mathcal{C}_{\mathcal{L}}$ associated to a constraint $\mathcal{L}$ is a function $\mathcal{C}_{\mathcal{L}} : \mathbb{IR}^m \to \mathbb{IR}^m$ contracting a box with respect to the constraint $\mathcal{L}$. It satisfies

$$\forall\, [\mathbf{x}] \in \mathbb{IR}^m,\, \mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}] \qquad \text{(Contractance)} \tag{10}$$

$$\forall \mathbf{x} \in [\mathbf{x}]\,,\, \mathcal{L}(\mathbf{x}) \implies \mathbf{x} \in \mathcal{C}_{\mathcal{L}}\,([\mathbf{x}]) \qquad \text{(Consistency)} \tag{11}$$

Remark 3. This definition can be extended to other sets such as zonotope and parallelepiped.

A contractor is said to be optimal when it outputs the smallest set which satifies the desired constraint. Some optimal contractors exist for classical constraints [24] but such contractor can not always be found.

4. Adaptive parallelepipedic approximation

This section presents the main contribution of this paper in the form of an algorithm *Ad-PEIBOS*. This algorithm provides an efficient way to compute an outer approximation of the image of the boundary of a set. This approximation is adaptive with respect to a given constraint or set of constraints.

4.1. Adaptive parallelepiped enclosure

Consider a C^1 function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, and a box $[\mathbf{x}]$. We would like to ensure that the image of $[\mathbf{x}] \in \mathbb{R}^n$ by $\mathbf{f}$ satisfies a given constraint $\mathcal{L} : \mathbb{R}^m \to \{0, 1\}$, i.e. we want to ensure that:

$$\forall \mathbf{x} \in [\mathbf{x}], \mathcal{L}(\mathbf{f}(\mathbf{x})) = 1 \tag{12}$$

Figure 4 shows a case where we consider a box $[\mathbf{x}]$ (in blue) and the constraint $\mathcal{L}$ is "be in the green circle". The red area is the part of the initial box which does not satisfy $\mathcal{L}$. The points in this area then satisfy $\bar{\mathcal{L}}$. Due to the consistency property of contractors (see 11) the contractor $\mathcal{C}_{\bar{\mathcal{L}}}$ applied on $[\mathbf{x}]$ will necessarily return a non-empty box if $[\mathbf{x}]$ does not fully satisfies the constraint $\mathcal{L}$.

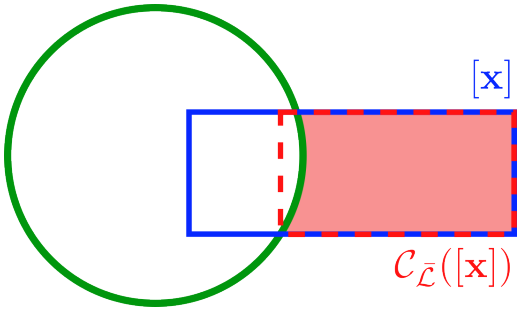


Figure 4: Contractor on the complementary

A classical method consists in directly computing the box $[\mathbf{y}] = [\mathbf{f}]([\mathbf{x}])$ where $[\mathbf{f}]$ is an interval inclusion function of $\mathbf{f}$ [2]. Then a contractor $\mathcal{C}_{\bar{\mathcal{L}}}$ can be constructed on the complementary of the desired constraint $\mathcal{L}$. If $\mathcal{C}_{\bar{\mathcal{L}}}([\mathbf{y}]) = \emptyset$, then we can be certain that the constraint is validated on the whole box $[\mathbf{x}]$.

As the computation of such box is subject to high conservatism it can be impossible to validate the constraint directly. A method is then to split the box $[\mathbf{x}]$ into smaller boxes and to test the contraction on the image of each small box. If the constraint is not validated the small boxes can be split recursively until a limit size is reached. When the limit is reached we say that the constraint can not be validated. This principle is often used in branch-and-bound [25] and branch-and-prune [26, 11] algorithms.

The method presented here relies on the same principle. Algorithm 1 presents a step of what will be the *Ad-PEIBOS* algorithm. Its name *Ad-PEIB* stands for "**Ad**aptive **P**arallelepipedic **E**nlcosure of the **I**mage of a

Box". It takes a function $\mathbf{f}$, a box $[\mathbf{x}_0]$ and a contractor $\mathcal{C}_{\bar{\mathcal{L}}}$ as inputs. As presented earlier by Equation 12 the objective is to assert that the constraint $\mathcal{L}$ is satisfied on the whole image of $[\mathbf{x}_0]$ by $\mathbf{f}$. To do so the box $[\mathbf{x}_0]$ is split recursively when needed until the constraint is verified everywhere or a limit size ϵ_{lim} is reached. An initial resolution ϵ and the limit resolution ϵ_{lim} are then also needed as inputs.

Note the algorithm requires a parallelepiped inclusion function $\langle \mathbf{f} \rangle$. Given a function $\mathbf{f}$ and a box $[\mathbf{x}]$, it must returns a parallelepiped enclosing $\mathbf{f}([\mathbf{x}])$ as defined in Definition 4.

Algorithm 1: Adaptive Parallelepiped Enclosure (Ad-PEIB)

Input:
A parallelepiped inclusion function of $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, noted $\langle \mathbf{f} \rangle$
A box $[\mathbf{x}_0] \in \mathbb{IR}^n$
A contractor $\mathcal{C}_{\bar{\mathcal{L}}} : \mathbb{R}^m \to \mathbb{R}^m$ associated with the complement of $\mathcal{L} : \mathbb{R}^m \to \{0, 1\}$
A resolution $\epsilon > 0$ with lower bound $\epsilon_{\lim} > 0$
A list of parallelepipeds L_P

Output: L_P completed in place

```
if ε < ε_lim then
    Raise an alarm ;        // Constraint could not be verified
else
    Split [x_0] into a list of boxes L_x of diameter ≤ ε;
    for [x] ∈ L_x do
        ⟨y⟩ ← ⟨f⟩([x]);
        if C_L̄(⟨y⟩) = ∅;                    // L is satisfied on ⟨y⟩
         then
            Store ⟨y⟩ in L_P;
        else
            Ad-PEIB(f, [x], C_L̄, ε/2, ε_lim, L_P) ;     // Recursive call
```

At the end of Algorithm 1 if no alarm was raised we obtain a list of parallelepipeds L_P such that

$$\begin{cases} \forall \langle \mathbf{y} \rangle \in L_P,\ \mathcal{L}\left(\langle \mathbf{y} \rangle\right) = 1 \\ \mathbf{f}\left([\mathbf{x}_0]\right) \subseteq \bigcup\limits_{\langle \mathbf{y} \rangle \in L_P} \langle \mathbf{y} \rangle \end{cases} \tag{13}$$

4.2. Ad-PEIBOS algorithm

Computing the image of a whole set $\mathbb{X} \subset \mathbb{R}^m$ requires to choose between the quality of the result and the computation time. Indeed as the initial set gets larger, the pessimism in the computation of its image also gets larger. As explained in Subsection 4.1 a solution is to subdivide this initial set into smaller ones. This solution reduces by a lot the conservatism, but it also increases the computation time.

Another solution is to compute the image of the boundary of the set noted $\partial\mathbb{X}$. The boundary considered here is the *topological boundary*. For a set $\mathbb{X}$, its topological boundary can be defined as

$$\partial\mathbb{X} = \overline{\mathbb{X}} \backslash \text{int}(\mathbb{X}) \tag{14}$$

where $\overline{\mathbb{X}}$ is the closure of $\mathbb{X}$ and $\text{int}(\mathbb{X})$ is its interior. See for example [27] for more detailed information on the topic.

Let us consider a function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ and suppose that the set $\mathbb{X}$ is compact. Then in the case where the function $\mathbf{f}$ is locally injective in a neighborhood of every $\mathbf{x} \in \mathbb{X}$, we have [28]:

$$\partial\left(\mathbf{f}\left(\mathbb{X}\right)\right) \subseteq \mathbf{f}\left(\partial\mathbb{X}\right) \tag{15}$$

This means that in this case the computation of the image of the boundary $\mathbf{f}\left(\partial\mathbb{X}\right)$ is sufficient to get the boundary of the image set $\partial\left(\mathbf{f}\left(\mathbb{X}\right)\right)$. Then from this boundary it is possible to get both an inner and an outer approximation of the image set. This is an interesting property because the computation of the image of the boundary is subject to less pessimism. Indeed as $\partial\mathbb{X} \subseteq \mathbb{X}$, the set is smaller and so less pessimism is induced in the computations.

As suggested in [29] box atlases can be used to make this computation easier. Section 4 of [21] introduces the notion of gnomonic atlas. Gnomonic atlases are an extension of box atlases. The idea is to rely on the properties of symmetry of the initial set to construct the atlas.

Definition 7. [21] Let us consider a manifold $\mathbb{M} \subset \mathbb{R}^m$ of dimension $n \leq m$. A *gnomonic atlas* $\{\boldsymbol{\psi}_0, \Sigma\}$ for $\mathbb{M}$ is composed of a function $\boldsymbol{\psi}_0$, called the inverse chart, and a set of symmetries Σ of $\mathbb{R}^m$ such that

$$\mathbb{M} = \bigcup_{\sigma \in \Sigma} \sigma \circ \boldsymbol{\psi}_0([-1,1]^n) \tag{16}$$

where $\sigma \circ \psi_0$ denotes the composition of σ and ψ_0, i.e.

$$\forall \mathbf{x} \in \mathbb{R}^n,\ \sigma \circ \psi_0\left(\mathbf{x}\right) = \sigma\left(\psi_0\left(\mathbf{x}\right)\right) \tag{17}$$

Figure 5 shows that the cube can be constructed from a box $[-1,1]^2 \in \mathbb{R}^2$, a function $\psi_0 : \mathbb{R}^2 \to \mathbb{R}^3$ and six symmetries. The corresponding gnomonic atlas is $\left\{\psi_0, \left\{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\right\}\right\}$.

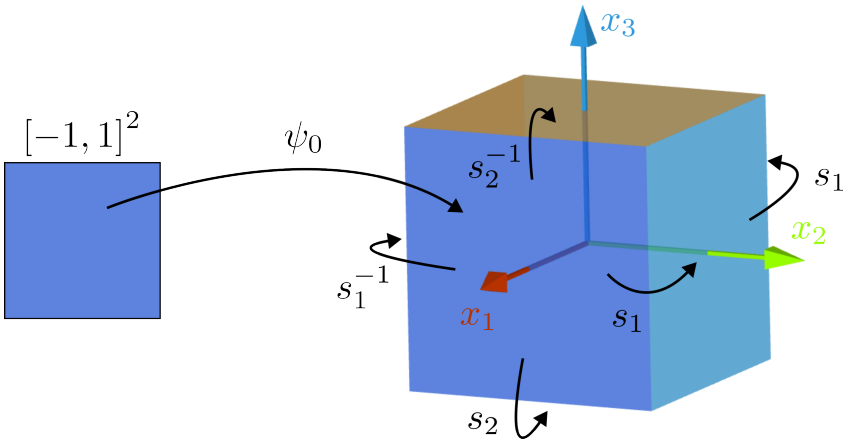


Figure 5: Gnomonic atlas on the cube

In the case where we can define a gnomonic atlas $\{\boldsymbol{\psi}_0, \Sigma\}$ for the boundary of the set $\partial\mathbb{X}$, Equations 15 and 16 then yield

$$\partial\left(\mathbf{f}\left(\mathbb{X}\right)\right) \subseteq \bigcup_{\sigma\in\Sigma} \mathbf{f}\left(\sigma \circ \boldsymbol{\psi}_0([-1,1]^n)\right) \tag{18}$$

This is the idea behind Algorithm 2.

Algorithm 2 combines the notion of gnomonic atlas and the adaptive parallelepiped enclosure of Algorithm 1. Let us denote by

$\mathbb{X} \subset \mathbb{R}^m$ the studied initial set and $\partial\mathbb{X}$ its boundary. In the case where $\partial\mathbb{X}$ is a manifold,

this algorithm computes the image of $\partial\mathbb{X}$ by a function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ while asserting that a constraint $\mathcal{L} : \mathbb{R}^m \to \{0,1\}$ is satisfied. To do so, the algorithm relies on a gnomonic atlas $\{\boldsymbol{\psi}_0, \Sigma\}$ which satisfies Equation 16 to cover $\partial\mathbb{X}$.

Let us denote $\sigma \in \Sigma$ a symmetry from the gnomonic atlas. If the studied manifold is of dimension n, then ψ_0 is a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. Each function $\mathbf{f} \circ \sigma \circ \psi_0$ is then also a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. Algorithm 1 can then be used on each of these functions. Finally from Equation 18 we know that we will compute an outer approximation of $\partial(\mathbf{f}(\mathbb{X}))$.

Algorithm 2: Ad-PEIBOS

Input: Function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$
Gnomonic atlas $\{\boldsymbol{\psi}_0, \Sigma\}$, with $\boldsymbol{\psi}_0 : \mathbb{R}^n \to \mathbb{R}^m$
Contractor $\mathcal{C}_{\bar{\mathcal{L}}} : \mathbb{R}^m \to \mathbb{R}^m$ associated with the complement of $\mathcal{L} : \mathbb{R}^m \to \{0, 1\}$
Resolution $\epsilon > 0$ with lower bound $\epsilon_{\text{lim}} > 0$

Output: List L_p of parallelepipeds enclosing the image set

$L_p \leftarrow \varnothing$;
for $\sigma \in \Sigma$ **do**
  Ad-PEIB$\Big(\mathbf{f} \circ \sigma \circ \boldsymbol{\psi}_0, [-1, 1]^n, \epsilon, \epsilon_{\text{lim}}, \mathcal{C}_{\bar{\mathcal{L}}}, L_p\Big)$;
return L_p

4.3. Scalability

Let us study how our algorithms scale with the dimension of the problem. To do so, we will study their time complexity. Since the method is adaptive, a global time complexity can not be determined. However, we can study the best and worst case scenarios for each algorithm. We will consider that each bisection, evaluation of the inclusion function and test has a unit cost.

4.3.1. Ad-PEIB

Let us consider an initial box $[\mathbf{x}] \in \mathbb{IR}^n$. We denote $w([\mathbf{x}])$ its witdh, defined as the length of its largest side.

The best case for the Ad-PEIB occurs when no bisection is needed and only one evaluation and two tests are enough to terminate the algorithm. In this best case, the Ad-PEIB algorithm then has a time complexity in $O(1)$.

The worst case occurs when

- The initial box has the same width on all its dimensions, i.e. $\forall i \in [\![1, n]\!]$, $w([x_i]) = w([\mathbf{x}])$,
- The box is initially not bisected, i.e. at the first iteration $\epsilon = w([\mathbf{x}])$,
- The constraint can not be validated, i.e. $\forall \mathbf{x} \in [\mathbf{x}]$, $\mathcal{L}(\mathbf{x}) = 0$.

In this case, the initial box needs to be splitted until all the boxes have a size inferior to ϵ_{lim}. As in the best case, the first iteration comes down to

just an evaluation and two tests. Then at each recursive call of the function, the width of the box is divided by two.

Let us denote by m the smallest integer such that

$$\frac{w\left([\mathbf{x}]\right)}{2^m} \leq \epsilon_{\text{lim}} \tag{19}$$

Then the value of m is given by

$$m = \left\lceil \log_2 \left(\frac{w\left([\mathbf{x}]\right)}{\epsilon_{\text{lim}}} \right) \right\rceil \tag{20}$$

where $\lceil x \rceil$ is the ceiling operator, giving the smallest integer greater or equal to x. This means that our recursion tree has a height m.

At the first level of the recursion tree, the initial box is tested for the limit criterion and its width is then divided by two. This operation comes down to n bisections in the n dimensions of the box. For each of the resulting n boxes, we perform an evaluation and a test. This level has a time complexity in $O(n)$.

At the second level, each of the n boxes is again tested, and then bisected n times. It generates n^2 boxes with an evaluation and a test for each. The i-th level of the tree then has a time complexity in $O\left(n^i\right)$.

This is performed until the level m of the tree is reached, where we bisect each of the n^{m-1} boxes n times for a total of n^m bisections, and each resulting box is tested and raises the alarm. The global time complexity of the algorithm in the worst case is then $O\left(n^m\right)$.

4.3.2. Ad-PEIBOS

The best and worst cases for the Ad-PEIBOS algorithm are the same as the ones for the Ad-PEIB algorithm. Let us denote p the number of symmetries in the gnomonic atlas.

In the best case, the Ad-PEIB algorithm is called p times and has a time complexity in $O(1)$. The Ad-PEIBOS algorithm then has a time complexity in $O(p)$.

In the worst case, the Ad-PEIB algorithm is called p times and has a time complexity in $O(n^m)$. The Ad-PEIBOS algorithm then has a time complexity in $O(p \cdot n^m)$.

Note that in this case the expression of m can be simplified since the initial box has a width equal to 2. We have :

$$\begin{aligned} \log_2\left(\frac{2}{\epsilon_{\lim}}\right) &= \log_2(2) - \log_2(\epsilon_{\lim}) \\ &= 1 - \log_2(\epsilon_{\lim}) \end{aligned} \tag{21}$$

and the expression of m becomes :

$$m = 1 - \lceil \log_2(\epsilon_{\lim}) \rceil \tag{22}$$

4.3.3. Discussion

It should be noted that the worst cases presented here should never occur. Indeed, if a case were critical enough to require subdividing all the boxes down to a size inferior to $\epsilon_{\lim}$, an initial slicing should already have been performed.

The real advantage of such adaptive method lies in the more common situations where only some parts of the approximation need to be refined. The gain in computation time sould be evaluated on a case-by-case basis, but an adaptive approach is often benefitial.

5. Application

In this section we display two use cases of the Ad-PEIBOS algorithm. As the Ad-PEIB algorithm requires a parallelepiped inclusion function we will use the one from [21]. It defines a parallelepiped inclusion function for any C^1 function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ if $n < m$. The CODAC library [30] will be used to perform the interval operations when needed.

5.1. Application 1 : 3-dimensional robotic arm

Let us consider the 3-dimensional robotic arm described Figure 6.

We will study the position of the effector (in orange Figure 6). It is defined by

$$\mathbf{z} = \mathbf{f}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \cos(y_1)(l_1 + l_2\cos(y_2) + l_3\cos(y_2+y_3)) \\ \sin(y_1)(l_1 + l_2\cos(y_2) + l_3\cos(y_2+y_3)) \\ l_2\sin(y_2) + l_3\sin(y_2+y_3)) \end{pmatrix}$$

For numerical applications, we will take $l_1 = 0.45$, $l_2 = l_3 = 0.3$. The studied angles will be limited to $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

We then define

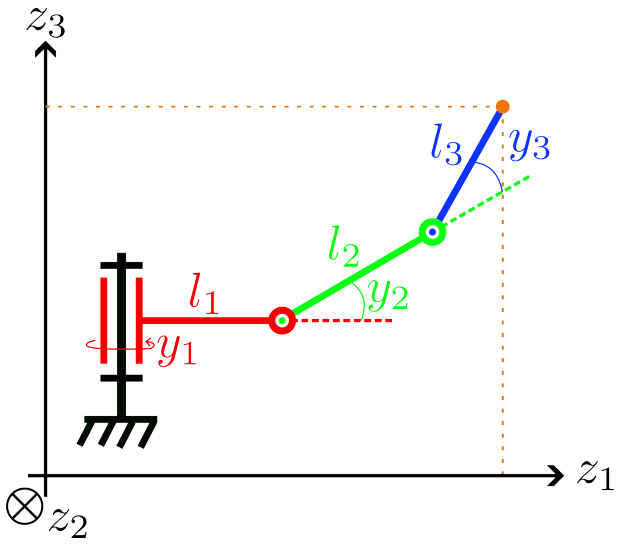


Figure 6: 3D robotic arm

- $\boldsymbol{\psi}_0(\mathbf{x}) = \begin{pmatrix} \frac{\pi}{2} \\ x_1 \cdot \frac{\pi}{2} \\ x_2 \cdot \frac{\pi}{2} \end{pmatrix}$
- $s_1 = e^{\mathbf{k} \cdot \frac{\pi}{2}}$ the rotation of $\frac{\pi}{2}$ with respect to $\mathbf{k}$ (the x_3 axis)
- $s_2 = e^{\mathbf{j} \cdot \frac{\pi}{2}}$ the rotation of $\frac{\pi}{2}$ with respect to $\mathbf{j}$ (the x_2 axis)

For this application we will consider that our robotic arm needs to operate in a cylinder as depicted in Figure 7, without touching it. We then get two constraints :

$$\begin{cases} \mathcal{L}_1(\mathbf{z}) = \left(\sqrt{{z_2}^2 + z_3^2} < 1.2\right) & \text{(radius of the cylinder)} \\ \mathcal{L}_2(\mathbf{z}) = (z_1 < 1.055) & \text{(depth of the cylinder)} \end{cases} \tag{23}$$

The resulting constraint is then $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$ as defined by Equation 8

Note that in this application the criterium from [21] on the local injectivity of $\mathbf{f}$ is not respected. Indeed there is a singularity in the function $\mathbf{f}$ when the two last arms are aligned, i.e. when $y_3 = 0$. This result can be found by searching for solutions to the equation

$$\det\left(\frac{d\mathbf{f}}{d\mathbf{y}}(\mathbf{y})\right) = 0 \tag{24}$$

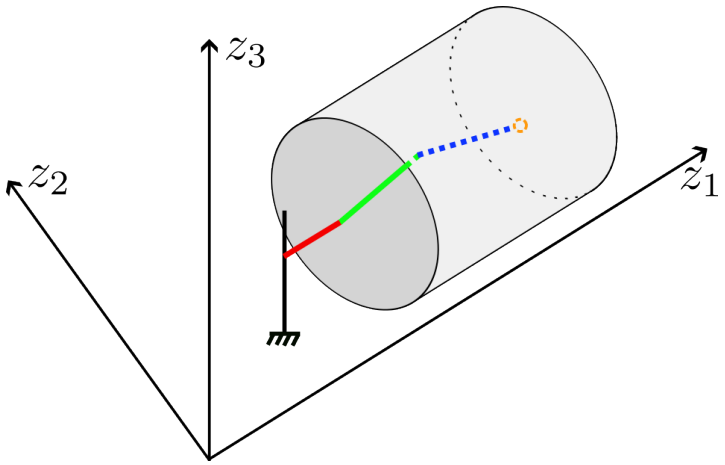
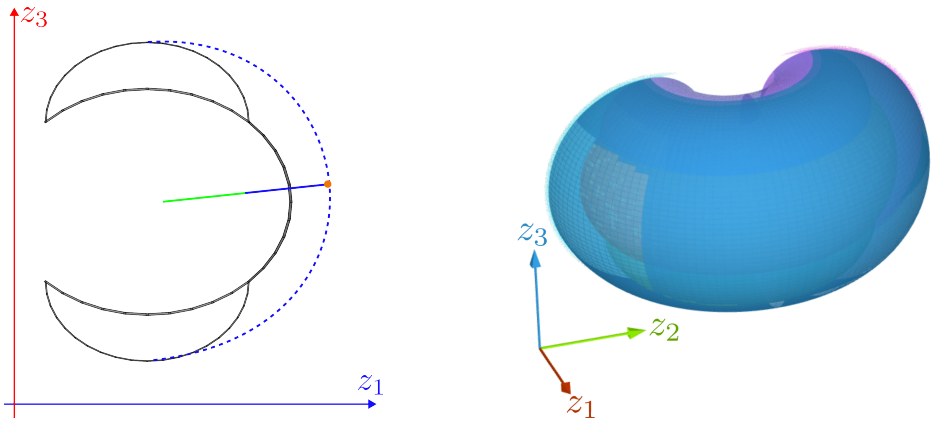


Figure 7: Robotic arm in a cylinder

where det is the determinant operator. This singularity is depicted on Figure 8. Figure 8b shows the singularity in the 3-dimensionnal workspace of the robotic arm. If we cut this volume at $z_2 = 0$ (middle of the volume), we obtain the result from Figure 8a where we can easily visualise how this singularity is generated by the alignement of the two last segments of the arm.



(a) 2D representation

(b) 3D representation

Figure 8: Singularity in blue in the robotic arm workspace

This singularity then needs to be handled apart. To do so we define :

$$\boldsymbol{\psi}(\mathbf{x}) = \begin{pmatrix} x_1 \cdot \frac{\pi}{2} \\ x_2 \cdot \frac{\pi}{2} \\ 0 \end{pmatrix} \tag{25}$$

To get the full workspace of the robotic arm with respect to the constraints, we first perform the algorithm Ad-PEIBOS with the gnomonic atlas $\left\{\psi_0, \left\{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\right\}\right\}$. The result is then completed by using the Ad-PEIB algorithm on the function $\mathbf{f} \circ \boldsymbol{\psi}$.

Figure 9 shows the complete result, with a color code corresponding to the resolution used to get each parallelepiped. To do so we concatenated the results of Ad-PEIBOS($\mathbf{f}$, $\left\{\psi_0, \left\{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\right\}\right\}$, $\mathcal{C}_{\tilde{\mathcal{L}}}$, 1.0, 0.01) and Ad-PEIB($\mathbf{f} \circ \boldsymbol{\psi}$, $[-1,1]^2$,$\mathcal{C}_{\tilde{\mathcal{L}}}$, 1.0, 0.01,{})

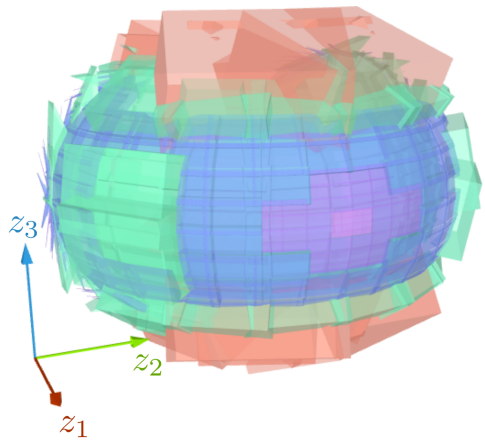
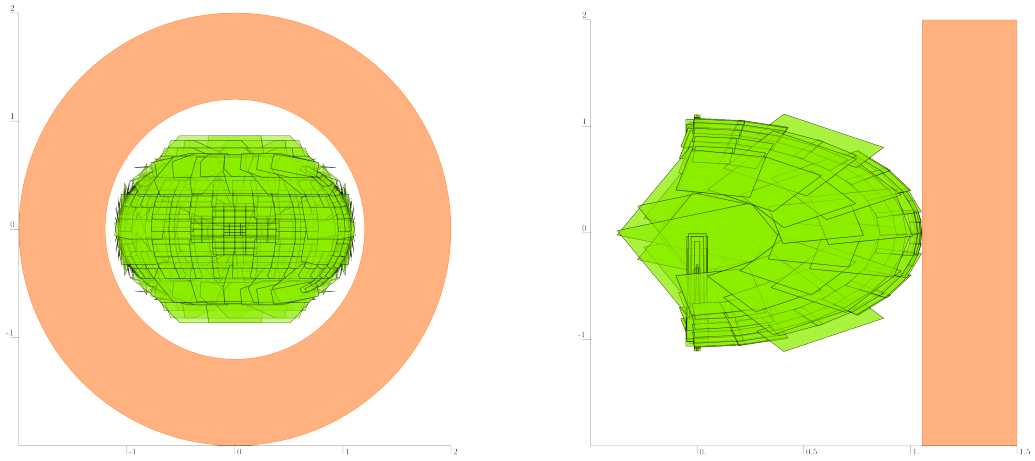


Figure 9: Adaptive outer approximation of the arm workspace

On Figures 10 and 11, the constraints are represented by a orange ring of inner radius 1.2 in the (z_2, z_3) plane, and an orange box starting at $z_1 = 1.055$ in the (z_1, z_3) plane.

With the adaptive slicing, we can see on Figure 10 that we are able to satisfy both constraints, while refining only the necessary areas. The smallest resolution used to get this result is $\epsilon_{min} = 0.03125$.

To compare with the standard implementation, Figure 11 shows the result obtained when using the fine resolution ϵ_{min} everywhere. In this case the outer approximation is also fine enough to verify both constraints. However, the computation time increased from $0.1s$ to $4s$. This means that our method allowed us to divide the computation time by a factor 40.



(a) (z_2, z_3) plane (b) (z_1, z_2) plane

Figure 10: Constraints validated thanks to an adaptive slicing

In addition, our algorithm can be used to perform a precise detection of the unsafe configurations. To illustrate this, let us reduce the diameter of the cylinder to 1.05 instead of the initial 1.2. Figure 12 shows that we are able to detect the unsafe arm positions, in red Figure 12b. Note that since the problem is symmetric, red parallelepiepds also appear on the other side of the volume (not visible here). In Figure 12a, we can see in red the configurations that led to the unsafe positions of the robotic arm.

5.2. Extension to dynamical systems with input trajectory

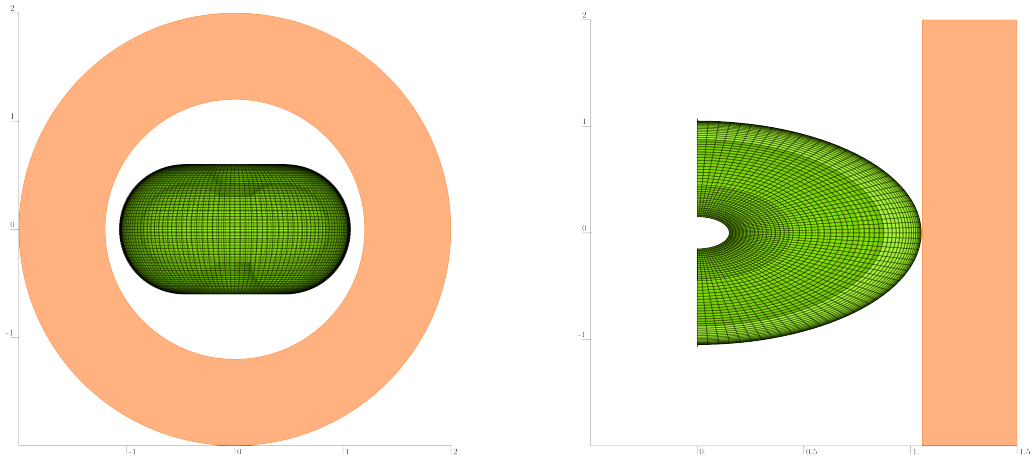
Let us consider a dynamical system defined by its state $\mathbf{x} \in \mathbb{R}^n$. Starting from an initial set $\mathbb{X}_0$ we want to compute the set of possible states at time $t \in \mathbb{R}^+$ noted $\mathbb{X}_t$. The method from [21] was limited to the study of dynamical systems of the form :

$$\dot{\mathbf{x}} = \alpha(\mathbf{x}) \tag{26}$$

If we add an input trajectory that can be expressed as a function of the state $\mathbf{x}$ and the time $t \in \mathbb{R}^+$, the system then follows an equation of the form :

$$\dot{\mathbf{x}} = \gamma(t, \mathbf{x}) \tag{27}$$

We can then define a new system state as :



(a) (z_2, z_3) plane (b) (z_1, z_3) plane

Figure 11: Fine workspace

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ t \end{pmatrix} \tag{28}$$

This new system follows an equation of the form :

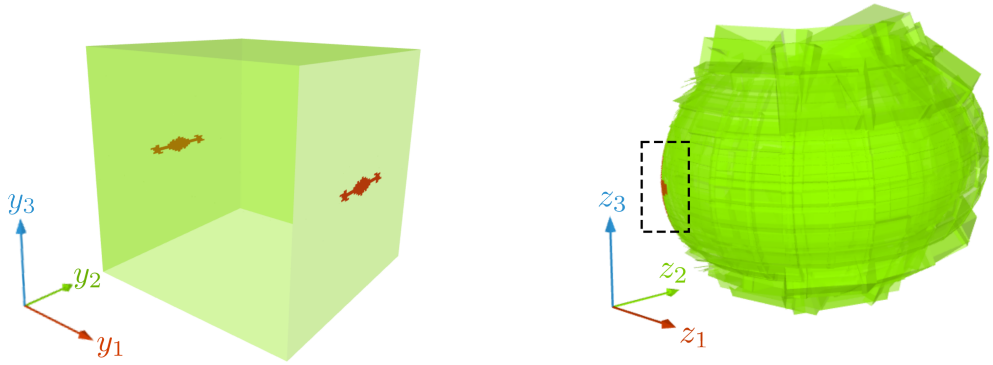
$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) = \begin{pmatrix} \gamma(t, \mathbf{x}) \\ 1 \end{pmatrix} \tag{29}$$

Let us assume that γ is continuous and Lipschitz continuous in $\mathbf{x}$ with a time-independent Lipschitz constant. The Picard-Lindelöf theorem [31] states that for any initial condition $\mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{Y}_0$ this equation admits a unique solution, called the flow function and noted $\phi : \mathbb{R} \times \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ which satisfies :

$$\forall t \in \mathbb{R}^+, \phi(t, \mathbf{y}_0) = \mathbf{y}(t) \tag{30}$$

We will denote by ϕ_i the function giving the i^{th} component of ϕ. Then for any initial condition $\mathbf{y}(0) = \mathbf{y}_0$:

$$\left.\begin{matrix} \phi_1(t, \mathbf{y}_0) \\ \vdots \\ \phi_n(t, \mathbf{y}_0) \end{matrix}\right\} = \mathbf{x}(t)$$
$$\phi_{n+1}(t, \mathbf{y}_0) = t \tag{31}$$



(a) In the initial set (b) In the image set

Figure 12: Detection of the unsafe configurations

If we denote $\mathbf{A} = \frac{\partial \phi(t,\mathbf{y}_0)}{\partial \mathbf{y}_0}$, the flow function also satisfies the variationnal equation :

$$\dot{\mathbf{A}} = \frac{d\mathbf{g}(\mathbf{y})}{d\mathbf{y}} \cdot \mathbf{A} \tag{32}$$

Integrating the augmented ODE

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) \\ \dot{\mathbf{A}} = \frac{d\mathbf{g}(\mathbf{y})}{d\mathbf{y}} \cdot \mathbf{A} \\ (\mathbf{y}(0), \mathbf{A}(0)) = (\mathbf{x}_0, 0)^T, I_{n+1} \end{cases} \tag{33}$$

over a time t for an initial condition $\mathbf{y}_0$ outputs both $\phi(t, \mathbf{y}_0)$ and its Jacobian.

With the method from [21] we can then compute a parallelepipedic outer approximation of $\mathbb{Y}_t$ for all $t \in \mathbb{R}^+$. By projecting the parallelepipeds as suggested in Proposition 1 on the first n dimension we can perform a *time projection*. We then obtain an outer approximation of $\mathbb{X}_t$.

5.3. Application 2 : Area reaching

For this application, we will consider the dynamic of the Dubins car :

$$\dot{\mathbf{x}} = \beta(\mathbf{x}, u) = \begin{pmatrix} \cos(x_3) \\ \sin(x_3) \\ u \end{pmatrix} \tag{34}$$

With $u(t) = \cos(2\pi t)$. We can then rewrite Equation 34

$$\dot{\mathbf{x}} = \gamma(t, \mathbf{x}) = \begin{pmatrix} \cos(x_3) \\ \sin(x_3) \\ \cos(2\pi t) \end{pmatrix} \tag{35}$$

As suggested in Section 5.2, we can use this formulation to compute the set reachable by the system at any time $t \in \mathbb{R}^+$ noted $\mathbb{X}_t$. To do so we define $\mathbf{y} = (\mathbf{x}, t)^T$ and the ODE :

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) = \begin{pmatrix} \gamma(t, \mathbf{x}) \\ 1 \end{pmatrix} \tag{36}$$

We will denote $\phi_{\mathbf{g}}$ the flow function associated to this ODE.

For the initial set, we will consider a centered sphere of radius 0.02 at time $t_0 = 0$.We then define

- $\boldsymbol{\psi}_0(\mathbf{b}) = 0.02 \cdot \begin{pmatrix} \frac{1}{\sqrt{1+b_1^2+b_2^2}} \\ \frac{b_1}{\sqrt{1+b_1^2+b_2^2}} \\ \frac{b_2}{\sqrt{1+b_1^2+b_2^2}} \\ 0 \end{pmatrix}$
- $s_1 = e^{\mathbf{k} \cdot \frac{\pi}{2}}$ the rotation of $\frac{\pi}{2}$ with respect to $\mathbf{k}$ (the x_3 axis)
- $s_2 = e^{\mathbf{j} \cdot \frac{\pi}{2}}$ the rotation of $\frac{\pi}{2}$ with respect to $\mathbf{j}$ (the x_2 axis)

Note that these rotations are defined in the (x_1, x_2, x_3) space and have no effect on the fourth coordinate (the time).

For this application we want to ensure that after $1s$ (i.e. a period of u) the car ends up in a circle of center $\mathbf{a} = (a_1, a_2)$ and radius r as depicted Figure 13. We then define the constraint

$$\mathcal{L}(\mathbf{y}) = \left(\sqrt{(a_1 - y_1)^2 + (a_2 - y_2)^2} < r \right) \tag{37}$$

For numerical applications we will consider the constraint of a circle of center $\mathbf{a} = (1, 0)$ and radius $r = 0.03$. The CAPD library [32] will be used for the guaranteed integration of the ODE.

Figure 14 shows the result of a naïve parallelepipedic approximation of the state of the Dubins car at 8 time points in $[0, 1]$ for $\epsilon = 0.4$ in the (x_1, x_2, x_3) space.

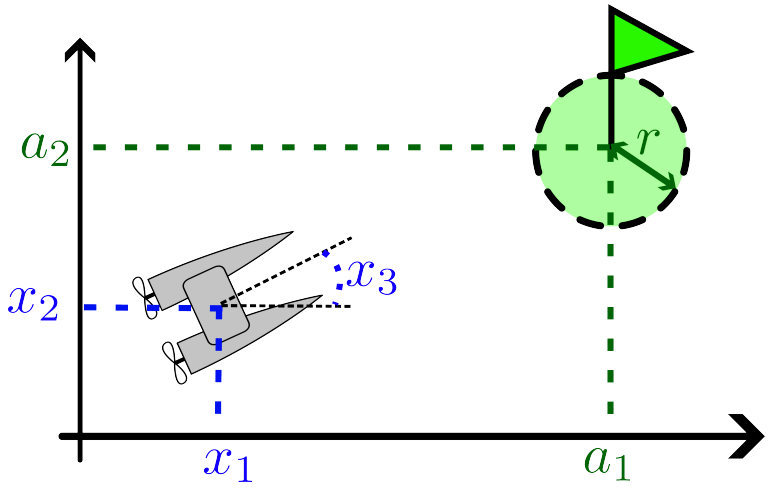


Figure 13: Application 2

To get the set reached by the car after $1s$, noted $\mathbb{X}_1$, we then perform the algorithm Ad-PEIBOS($\phi_{\mathbf{g}}, \left\{\psi_0, \left\{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\right\}\right\}, \mathcal{C}_{\bar{\mathcal{L}}}, 0.5, 0.01$)). Figure 15 shows the complete result projected in the (x_1, x_2, x_3) space. The color code corresponds to the resolution used to get the each parallelepiped.

Figure 16 shows the projection of $\mathbb{X}_1$ on the (x_1, x_2)-plane with the constraint represented as an orange ring of inner radius r. The constraint is satisfied on both side. On the left is the result of the Ad-PEIBOS algorithm. It was obtained in $18.5s$ and the smallest resolution needed was $\epsilon_{min} = 0.03125$. The result on the right was obtained in $166s$ by using the

Figure 14: Evolution of the Dubins car

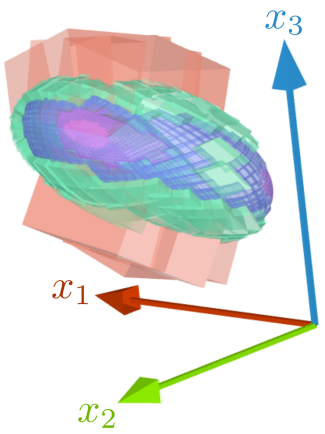


Figure 15: Adaptive outer approximation of $\mathbb{X}_1$

resolution ϵ_{min} everywhere. In this example the use of the adaptive slicing divided the computation time by a factor 9.

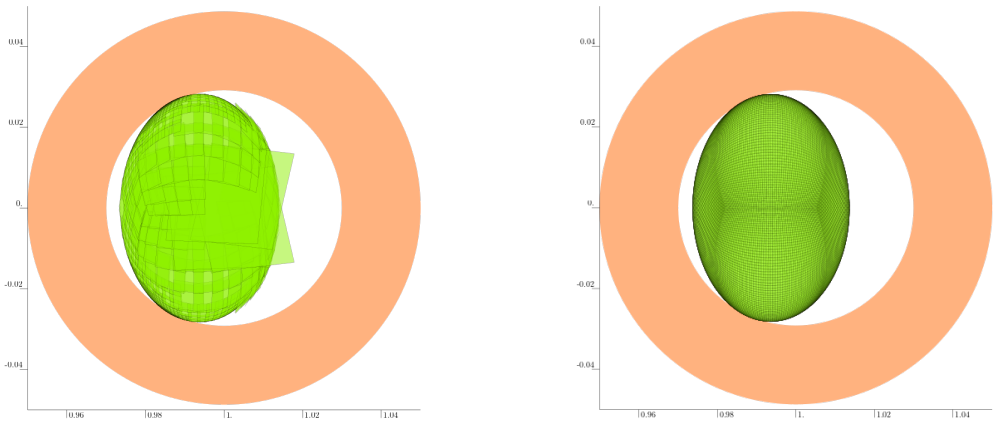
6. Conclusion

In this paper we proposed an algorithm Ad-PEIB to compute a constrained adaptive outer approximation for a set $\mathbb{Y}$ defined as the image of a box $[\mathbf{x}]$ by a nonlinear function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. This method allows us to assert that the set $\mathbb{Y}$ verifies a given constraint, or set of constraints, while refining the outer approximation only when it is needed.

Our approach assumes that $\mathbf{f}$ is a C^1 function. A first "large" cover of $\mathbb{Y}$ with parallelepipeds is done. Contractors are then used to ensure that each parallelepiped satisfies the desired constraints. Otherwise, the algorithm is called recursively until a limit size is reached where we say that the constraints can not be validated.

Considering a set $\mathbb{X}$, if it can be covered by a gnomonic atlas, the Ad-PEIBOS algorithm can be used. It uses the symmetries of the initial set to reduce the dimensions of the problem, thus reducing the conservatism. If the constraints can not be validated, we can then precisely enclose the responsible parts of the initial set boundary.

An extension of this work could be to improve the performances of the Ad-PEIBOS algorithm. Indeed in the case where the contraction of the parallelepiped outputs a non-empty set, we should be able to detect which



(a) Adaptive slicing (b) Fine resolution

Figure 16: Projected parallelepipedic approximation of $\mathbb{X}_1$ with (left) and without (right) adaptive slicing

part of the initial set is responsible and refine it. This could also be a step towards a parallelepiped-based set inversion algorithm.

References

[1] E. Fogel, Y. F. Huang, On the value of information in system identification - bounded noise case, Automatica 18 (2) (1982) 229–238.

[2] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics, Springer-Verlag, London, 2001. doi:10.1007/978-1-4471-0249-6_2.

[3] Y. Lin, M. A. Stadtherr, Validated solution of initial value problems for odes with interval parameters, Reliable Computing 12 (2006) 1–19, accounts for both interval-valued initial conditions and parameters.
URL https://www3.nd.edu/~markst/lin-stadtherr-rec06-v2.pdf

[4] J. A. D. Sandretto, A. Chapoutot, Validated explicit and implicit runge–kutta methods, Reliable Computing 22, special issue devoted to material presented at SWIM 2015 (2016).
URL https://interval.louisiana.edu/reliable-computing-journal/volume-22/reliable-computing-22-pp-79-103.pdf

[5] A. Rauh, S. S. Butt, H. Aschemann, Nonlinear state observers and extended kalman filters for battery systems, International Journal of Applied Mathematics and Computer Science 23 (3) (2013) 539–556. doi:10.2478/amcs-2013-0041.
URL https://doi.org/10.2478/amcs-2013-0041

[6] M. Zeitz, The extended luenberger observer for nonlinear systems, Systems & Control Letters 9 (2) (1987) 149–156. doi:https://doi.org/10.1016/0167-6911(87)90021-1.
URL https://www.sciencedirect.com/science/article/pii/0167691187900211

[7] M. Althoff, G. Frehse, A. Girard, Set propagation techniques for reachability analysis, Annual Review of Control, Robotics, and Autonomous Systems 4 (1) (2021) 369–395. doi:10.1146/annurev-control-071420-081941.
URL https://doi.org/10.1146/annurev-control-071420-081941

[8] A. Girard, Reachability of uncertain linear systems using zonotopes, in: Hybrid Systems: Computation and Control (HSCC 2005), Lecture Notes in Computer Science, vol. 3414, Springer Berlin Heidelberg, 2005, pp. 291–305. doi:10.1007/978-3-540-31954-2_19.

[9] R. Moore, Methods and Applications of Interval Analysis, Society for Industrial and Applied Mathematics, 1979. doi:10.1137/1.9781611970906.

[10] B. Kearfott, V. Kreinovich, Applications of interval computations: An introduction, in: Applied Optimization, Springer US, 1996, pp. 1–22. doi:10.1007/978-1-4613-3440-8_1.

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

[12] M. Dellnitz, A. Hohmann, A subdivision algorithm for the computation of unstable manifolds and global attractors, Numerische Mathematik 75 (1997) 293–317. doi:10.1007/s002110050240.

[13] M. Khajenejad, F. Shoaib, S. Z. Yong, Guaranteed state estimation via direct polytopic set computation for nonlinear discrete-time sys-

tems, IEEE Control Systems Letters 6 (2022) 2060–2065. doi:10.1109/LCSYS.2021.3138355.

[14] C. Combastel, A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes, in: 44th IEEE Conference on Decision and Control, 2005.

[15] C. Durieu, B. Polyak, E. Walter, Ellipsoidal state outer-bounding for MIMO systems via analytical techniques, in: Proceedings of the IMACS—IEEE–SMC CESA'96 Symposium on Modelling and Simulation, Vol. 2, Lille, France, 1996, pp. 843–848.

[16] M. Althoff, O. Stursberg, M. Buss, Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes, Nonlinear Analysis: Hybrid Systems 4 (2) (2010) 233–249, iFAC World Congress 2008. doi:https://doi.org/10.1016/j.nahs.2009.03.009.
URL https://www.sciencedirect.com/science/article/pii/S1751570X09000442

[17] J. A. dit Sandretto, A. Chapoutot, C. Garion, X. Thirioux, A constraint programming approach for polytopic simulation of ordinary differential equations: A collision detection application, Acta Cybernetica 26 (4) (2024) 755–774, special Issue of SWIM 2022 and 2023. doi:10.14232/actacyb.3007711.

[18] T. Dang, T. Dreossi, C. Piazza, Parameter synthesis using parallelotopic enclosure and applications to epidemic models, in: O. Maler, Á. Halász, T. Dang, C. Piazza (Eds.), Hybrid Systems Biology, Springer International Publishing, Cham, 2015, pp. 67–82.

[19] T. Dreossi, T. Dang, C. Piazza, Parallelotope bundles for polynomial reachability, in: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, 2016, pp. 297–306.

[20] E. Kim, S. Bak, P. S. Duggirala, Automatic dynamic parallelotope bundles for reachability analysis of nonlinear systems, in: C. Dima, M. Shirmohammadi (Eds.), Formal Modeling and Analysis of Timed Systems, Springer International Publishing, Cham, 2021, pp. 50–66.

[21] M. Godard, L. Jaulin, D. Massé, Inner and outer approximation of the image of a set by a nonlinear function, International Journal of Approximate Reasoning 187 (2025) 109574. doi:https://doi.org/10.1016/j.ijar.2025.109574.
URL https://www.sciencedirect.com/science/article/pii/S0888613X25002154

[22] X. Yang, J. K. Scott, A comparison of zonotope order reduction techniques, Automatica 95 (2018) 378–384. doi:https://doi.org/10.1016/j.automatica.2018.06.006.
URL https://www.sciencedirect.com/science/article/pii/S000510981830298X

[23] G. Chabert, L. Jaulin, Contractor programming, Artificial Intelligence 173 (11) (2009) 1079–1100. doi:https://doi.org/10.1016/j.artint.2009.03.002.
URL https://www.sciencedirect.com/science/article/pii/S0004370209000381

[24] B. Desrochers, L. Jaulin, A minimal contractor for the polar equation: Application to robot localization, Engineering Applications of Artificial Intelligence 55 (2016) 83–92. doi:https://doi.org/10.1016/j.engappai.2016.06.005.
URL https://www.sciencedirect.com/science/article/pii/S0952197616301129

[25] A. H. Land, A. G. Doig, An automatic method of solving discrete programming problems, Econometrica 28 (3) (1960) 497–520.
URL http://www.jstor.org/stable/1910129

[26] E. R. Hansen, S. Sengupta, Bounding solutions of systems of equations using interval analysis, BIT Numerical Mathematics 21 (2) (1981) 203–211. doi:10.1007/BF01933165.

[27] S. Willard, General Topology, Addison-Wesley, Reading, MA, 1970.

[28] A. Hatcher, Algebraic Topology, Cambridge University Press, Cambridge, UK, 2001.
URL http://www.math.cornell.edu/~hatcher/AT/ATpage.html

[29] A. Ignazi, R. Guyonneau, S. Lagrange, S. Lahaye, Box atlas: An interval version of atlas, International Journal of Approximate Reasoning 183 (2025). doi:10.1016/j.ijar.2025.109441.
URL https://doi.org/10.1016/j.ijar.2025.109441

[30] S. Rohou, B. Desrochers, F. L. Bars, The codac library, Acta Cybernetica 26 (4) (2024) 871–887. doi:10.14232/ACTACYB.302772.

[31] G. Teschl, Ordinary Differential Equations and Dynamical Systems, Vol. 140 of Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, 2012.

[32] T. Kapela, M. Mrozek, D. Wilczak, P. Zgliczynski, CAPD: dynsys, A flexible C++ toolbox for rigorous numerical analysis of dynamical systems, Communications in Nonlinear Science and Numerical Simulation 101 (2021) 105578. doi:10.1016/j.cnsns.2020.105578.