

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES

ÉCOLE DOCTORALE N° 648

Sciences pour l'Ingénieur et le Numérique

*Spécialité : Mathématiques et Sciences et Technologies
de l'Information et de la Communication*

Par

Maël GODARD

Safe navigation of a robot in a landmarked environment

Thèse présentée et soutenue à Brest, le TBD

Unité de recherche : Lab-STICC (UMR 6285)

Rapporteur·trice·s avant soutenance :

Michel KIEFFER Professeur des universités, Centrale-Supélec, Gif-sur-Yvette, FR
Matthias ALTHOFF Associate professor, Technische Universität München, Munich, DE

Composition du Jury :

Attention, en cas d'absence d'un·e des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président·e :

Examineur·trice·s :

Dir. de thèse : Luc JAULIN Professeur des universités, Université de Bretagne Occidentale
Co-dir. de thèse : Damien MASSÉ Maître de conférences, Université de Bretagne Occidentale

TABLE OF CONTENTS

1	Introduction	7
1.1	Autonomous underwater exploration	7
1.1.1	Under the sea ?	7
1.1.2	The rise of robotics	7
1.1.3	To locate or not to locate...	8
1.1.4	... that is the question	10
1.1.5	Additionnal use cases	10
1.2	Guaranteed enclosure of a robot's state	11
1.2.1	Dynamic of a robot	11
1.2.2	The need for guarantees	12
1.2.3	How do we enclose a robot state ?	13
1.3	PhD thesis context	14
1.3.1	Thesis objective	14
1.3.2	Image of a set by a function	15
1.3.3	Computation of the reachable set	16
1.3.4	Thesis structure	16
I	Tools and notations	21
2	Set based methods	22
2.1	Introduction	22
2.2	Interval analysis	23
2.2.1	Intervals, Interval vectors and matrices	23
2.2.2	Interval operations	25
2.2.3	Inclusion function	27
2.2.4	Origin of pessimism	29
2.2.5	Pavings and pavers	32
2.2.6	Contractors	35

TABLE OF CONTENTS

2.3	Zonotopic representations	42
2.3.1	Zonotopes	43
2.3.2	Parallelepipeds	46
2.3.3	Limits of paving	50
2.4	Tools and libraries	52
2.5	Conclusion	53
3	Topology and differential geometry	55
3.1	Introduction	55
3.2	Topology	56
3.2.1	Sets: Boundary and Compactness	56
3.2.2	Manifolds: Atlases and Smoothness	59
3.3	Differential geometry	61
3.4	Conclusion	63
4	Reachability of dynamical systems	64
4.1	Introduction	64
4.2	Formalism of dynamical systems	65
4.2.1	Definition	65
4.2.2	Classical problems	66
4.2.3	The flow function	68
4.3	Finite-time reachability analysis	70
4.3.1	Reachable set from an initial state	71
4.3.2	Reachable set from an initial set	73
4.4	Tools	74
4.5	Conclusion	74
II	Contributions	76
5	Inner and outer approximation of an image set with parallelepipeds	77
5.1	Introduction	77
5.2	Parallelepiped inclusion function	78
5.2.1	Approximation theorem	78
5.2.2	Parallelepiped inflation	80
5.2.3	Parallelepiped enclosure of the image of a box	87

5.3	Computing the image of a set by a nonlinear $\mathbb{R}^m \rightarrow \mathbb{R}^m$ function	92
5.3.1	The gnomonic atlas	93
5.3.2	Inner and outer approximation of the image of a set	96
5.3.3	Examples	98
5.4	Extension to ODE	102
5.4.1	Method	102
5.4.2	Examples	103
5.5	Conclusion	105
6	Inner and outer approximation of the reachable set	107
6.1	Introduction	107
6.2	Context	108
6.3	Characterization of the smooth reachable set	109
6.3.1	Preliminary result	109
6.3.2	Main contribution	112
6.3.3	Examples	114
6.4	Extension to non smooth reachable sets	118
6.4.1	Contribution	119
6.4.2	Example	122
6.5	Conclusion	123
7	Constrained adaptive parallelepipedic approximation of the image of a set by a nonlinear function	125
7.1	Introduction	125
7.2	Adaptive parallelepipedic enclosure	126
7.2.1	Enclosure of the image of a box	127
7.2.2	Enclosure of the image of a set	128
7.3	Scalability	129
7.3.1	Ad-PEIB	130
7.3.2	Ad-PEIBOS	131
7.3.3	Discussion	131
7.4	Example	131
7.4.1	3d robotic arm	132
7.4.2	Area reaching	135
7.5	Inner and outer approximation of the unsafe areas	138

TABLE OF CONTENTS

7.6	Conclusion	143
8	Removal of fake boundaries in boundary based approaches	145
8.1	Introduction	145
8.2	Problem presentation	146
8.3	Box chains to detect self intersections	148
8.3.1	Hypothesis	148
8.3.2	Definitions	148
8.3.3	Box Chain decomposition	149
8.3.4	Removing Fake boundaries	156
8.4	To go further	161
8.5	Conclusion	162
9	Implementation	163
9.1	Introduction	163
9.2	Numerical guarantee	163
9.3	Implementations	164
10	Conclusion	167
10.1	Scope of the contributions	167
10.1.1	Image of a set by a nonlinear function	167
10.1.2	Computation of the reachable set of a dynamical system	167
10.1.3	Adaptive enclosure of the image set	168
10.1.4	Fake boundaries removal	168
10.2	Summary of the contributions	168
10.2.1	Papers	168
10.2.2	Implementations	169
10.3	Limitations and prospects	169
	Bibliography	171

INTRODUCTION

1.1 Autonomous underwater exploration

1.1.1 Under the sea ?

According to the National Oceanic and Atmospheric Administration (NOAA), oceans and seas cover 71% of Earth's surface. However, they comparatively are mostly unexplored. Still according to the NOAA, more than 80% of oceans and seas remain unmapped.

This knowledge which is vastly missing is crucial. Anthropogenic activities (maritime traffic, offshore installations, fishing ...) have an effect on the surrounding ecosystems. Major changes in the environment can drastically modify the living conditions of the local species and endanger them. However, these effects are difficult to predict and require a regular monitoring of the designated areas.

In addition, some areas require a regular mapping to ensure maritime safety. Harbor approaches, straits and maritime corridors are examples of critical maritime areas with a high traffic which require a constant monitoring. According to the World Trade Organization (WTO), maritime transport represents more than 80% of the volume of world trade. Since the topology of the seabed is subject to constant changes, it is highly important to control its evolution to ensure its continuation.

This leads us to an observation : we need to collect a large amount of data, in a vast area, and with limited resources. This is a challenge where robots bring an efficient solution.

1.1.2 The rise of robotics

ALVIN (see Figure 1.1) was the first crewed submersible for research. It made its first dive in 1964 and could reach up to 6.5km depth. It carried out multiple missions, from

the inspection of Titanic's wreck to the biodiversity observations and is still used to date after numerous updates.



Figure 1.1 – ALVIN submersible [*Credits: Woods Hole Oceanographic Institution*]

However, oceans and seas are difficult environment to explore. Many elements make subsea exploration complex : high pressure, underwater currents, turbidity... Given these harsh conditions, the use of Remotely Operated Vehicles (ROVs) in the first place followed by Autonomous Underwater Vehicles (AUVs) increased. The objective was to perform the same tasks as a crewed submersible without endangering human lives.

An additional advantage of taking the crew out of the vehicle is that it saves space. Indeed, whether it is a ROV or an AUV, an underwater robot does not require the presence of humans on board to function. The whole robot space can then be dedicated to the robot's mission : battery storage, sensors, embedded electronics.

Figure 1.2 shows an example of AUV named *Daurade*. This AUV was build in 2005 by the ECA group and has been used for years by DGA-TN Brest and the SHOM (Service Hydrographique et Océanographique de la Marine) for surveying or mine hunting missions. This AUV possessed a large range of acoustic sensors along with an Inertial Navigation System (INS) which made it the cutting edge of technology at the time.

1.1.3 To locate or not to locate...

A major challenge in underwater robotics is the absence of GNSS positioning. Indeed, the signals from the GNSS satellites reflect on the water surface which make GNSS receptors unable to see them after only a few centimeters.

If the integration of inertial measurements gives an idea of the evolution of a robot, it drifts more and more as the duration of the mission increases. A regular correction of



Figure 1.2 – The Daurade AUV [Credits: Simon ROHOU]

the estimate is then required for both the robot safety and to assert that it completes its objective.

If electromagnetic waves struggle to propagate underwater, it is not the case of mechanical ones. Acoustic systems are then widely used underwater to sense the environment and provide information to the robot. In particular, USBL (Ultra Short BaseLine) and LBL (Long BaseLine) systems are widely use to locate underwater vehicles. As displayed Figure 1.3 one requires a surface vessel and the other one requires the installation of underwater acoustic beacons to function. These solutions are then difficult to use for large scale exploration.

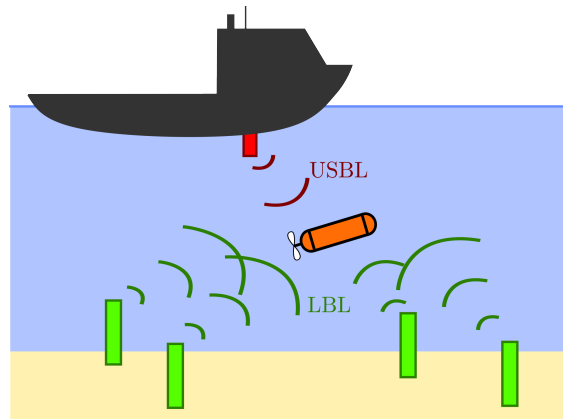


Figure 1.3 – USBL and LBL for underwater positioning

If the environment is known prior to the mission, it is possible to use a remarkable structure (rock, wreck...) as a landmark to locate the robot when it is sensed.

Otherwise, a solution is to use Simultaneous Localization And Mapping (SLAM) methods [83]. As the name suggests, the objective is to create a map of then environment on

the fly while taking into account the uncertainties of the carrier. When the robot senses again an area previously mapped, it is able to both relocate itself and improve its map.

1.1.4 ... that is the question

A direct consequence is that the robot may not always be able to locate itself. Consider a perfectly flat seabed with only a few remarkable rocks as depicted Figure 1.4. Between two rocks, the robot can only rely on inertial measurements to estimate its position in order to reach the next rock and relocate itself.

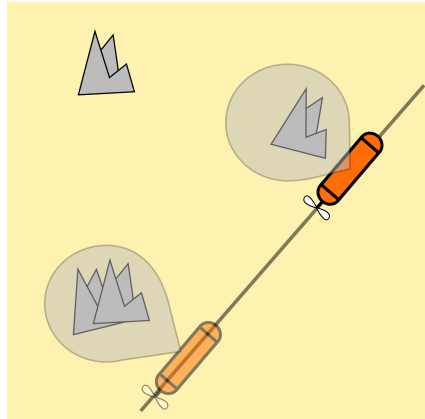


Figure 1.4 – Robot sensing remarkable objects

However, as previously stated, this inertial estimation drifts with time. There is then a risk for the robot to miss the localization area (the rock in our example) and to lose itself underwater. Since these robots may be equipped with expensive sensors, it is primordial to be certain that such scenario never occurs. This is the objective of the work presented in this manuscript.

1.1.5 Additionnal use cases

If we presented the issue of GNSS-less navigation in an underwater context, it has to be noted that this applies to a larger scale. The first thing to note is that the GNSS coverage on Earth's surface is unequal and focuses on the areas with an higher population density.

As shown Figure 1.5, GNSS positioning is also highly subject to jamming. GNSS jamming (which includes GPS jamming) is the act of emitting radio signals at the same

frequency as the GNSS satellites to disturb the receivers and make their positioning worse if not impossible.

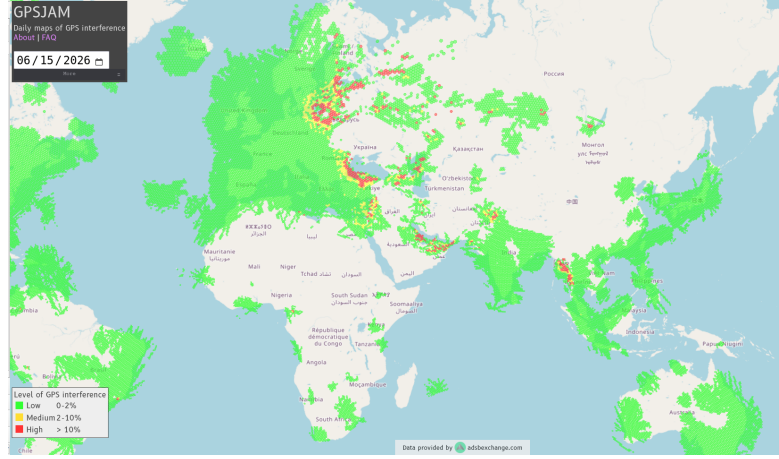


Figure 1.5 – Map of GPS interference the 15th June 2026 [From *gpsjam.org*]

It falls directly that for military applications GNSS localization is not reliable and should be avoided as much as possible. Yet such jamming is not always done on purpose and critical systems should always have another way to locate themselves even in non-military applications. In both cases the work from this manuscript aims to provide a way to ensure the success of a mission prior to its start.

1.2 Guaranteed enclosure of a robot's state

1.2.1 Dynamic of a robot

A mobile robot is a dynamical system. At any time, It can be fully described by its state vector $\mathbf{x}$. This vector can contain information about the robot position, its orientation, its speed etc. The dynamic of a robot can be described by an Ordinary Differential Equation (ODE) of the shape

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t)) \quad (1.1)$$

where $\mathbf{w}$ is a disturbance vector. It represents everything that affects the robot dynamic, from its actuators to its environment (wind, current...). For a given initial condition $\mathbf{x}(0) = \mathbf{x}_0$ and a disturbance trajectory $\mathbf{w}(\cdot)$, one can predict the state of the robot at any time $t > 0$. We will note $\mathbf{x}(t, \mathbf{x}_0, \mathbf{w}(\cdot))$ this state.

1.2.2 The need for guarantees

When we say that we want to be certain of not losing the robot, it has to be understood in terms of guarantees. If the robot is perfect in a perfect world, such guarantee is easy to get since we know the exact disturbance $\mathbf{w}(t)$ applied to the robot at any time $t > 0$.

However real systems are imperfect by nature, and are subject to external disturbances. In Figure 1.6, the robot tries to reach an area (in green) where it will be able to sense a remarkable landmark and locate itself. Due to these uncertainties, the robot's behavior can not be exactly predicted beforehand and it may miss the desired area.

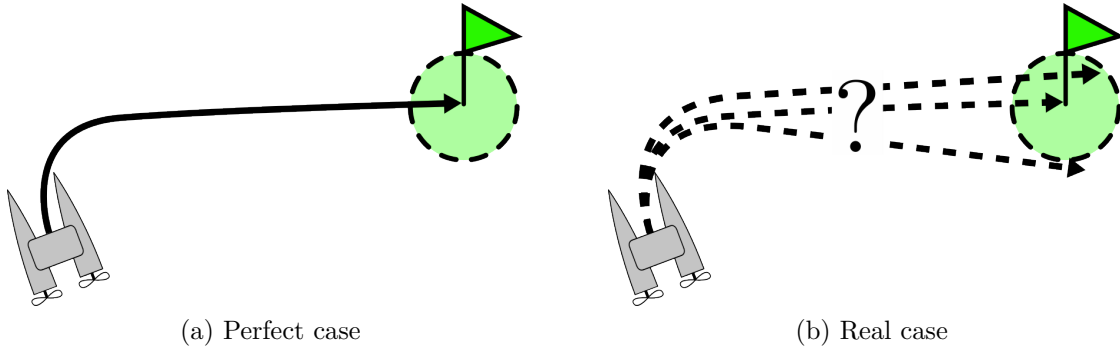


Figure 1.6 – Robot trying to reach the green area

At a given time $t > 0$, $\mathbf{w}(t)$ is not perfectly known and can take multiple values. In addition, the initial state of the robot itself may be uncertain. We then need to compute the possible states of the robot with respect to all its uncertainties and disturbances. If the uncertainties are bounded, these possible states will lie in a set called the reachable set of the system. For this manuscript the initial state will be supposed to lie in a set $\mathbb{X}_0$, and the disturbances will belong to the set $\mathbb{W}$. The reachable set at time $t > 0$ will further be noted $\mathcal{R}_t(\mathbb{X}_0, \mathbb{W})$.

A common way to compute it is to rely on a statistical approach. We pick a random initial state $\mathbf{x}_0$ for the robot, we make it evolve with a random disturbance trajectory $\mathbf{w}(\cdot)$, and we reiterate the process. After a high number of iterations, one can say that most cases have been covered and enclosing all the results in a set gives an approximation of the reachable set of the robot. This set is often an ellipsoid (or at least convex) and is associated with a confidence level.

However, such methods often fail to catch the “worse case scenarios” where the worse disturbance is applied to the worse initial state. Even if such cases have a low chance of

happening, they remain possible and should then be considered in critical applications.

To say that we guarantee a safe mission before its start, every individual step of the a priori validation itself needs to be guaranteed. Set-based methods, and in particular interval-based methods, are able to provide this guarantee. They represent and propagate uncertainties in a guaranteed way under the hypothesis of existing bounds [59]. Hence, they found numerous applications in robotics and especially in maritime robotics where any omitted case can cause the loss of the robot.

1.2.3 How do we enclose a robot state ?

This question does not have a fixed answer. The choice of a representation depends mainly on the use case.

A recent yet intuitive wrapper is the tube [80, 14]. Since a robot is a dynamical system with a continuous evolution (see Equation 1.1), each possible evolution can be represented as a trajectory. A tube is its interval equivalent and encloses all the possible trajectories of a robot over a time horizon $[0, T]$. Figure 1.7 shows how this representation is suited for robotic applications. In this case the initial state is supposed to be known (punctual) and the uncertainty comes from the disturbance set $\mathbb{W}$. The blue tube represents $\mathcal{R}_{[0,T]}(\mathbf{x}_0, \mathbb{W})$.

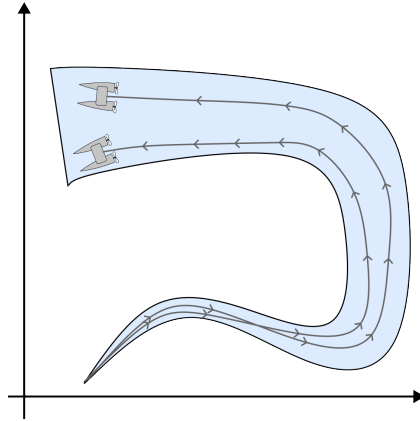


Figure 1.7 – Tube enclosing trajectories of a robot

However, such continuous representation is not always needed. Going back to the example depicted Figure 1.6, proving that there exists a time where the robot is certain to cross the green set is sufficient. To represent the possible states of a robot at a given time, one can choose between many set representations : boxes, ellipsoids, non-convex

sets. It is even possible to use a collection of such objects to cover this reachable set of the robot.

A common compromise in set-based methods is the balance between the quality of the enclosure and the complexity to handle it. The easier a set is, the faster the computation but the more pessimistic the result. On the other hand, a really complex representation allows for a fine approximation of the set but is harder to propagate.

If one wants to do such computations on the fly with a limited computational capacity, an easy set representation like a box or an ellipsoid is often preferred. If the computations are done prior to the mission or in post-process, more refined set like polytopes and zonotopes can be used.

1.3 PhD thesis context

1.3.1 Thesis objective

In this manuscript, we want to ensure that a robot is safely able to navigate from landmark to landmark. Let us consider l landmarks and denote $\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_l$ the sets such that the robot is able to sense the associated landmark. If at a time $t > 0$, the state of the robot satisfies

$$\exists i \in \llbracket 1, l \rrbracket, \mathbf{x}(t) \in \mathbb{S}_i \quad (1.2)$$

, the robot senses at least one landmark and is able to relocate itself.

As stated earlier, the dynamic of the robot can not be exactly predicted beforehand due to uncertainties and external disturbances. Under the hypothesis of bounded uncertainties, *i.e.* the initial state and the disturbances are bounded, the robot state at time $t > 0$ lies in the reachable set $\mathcal{R}_t(\mathbb{X}_0, \mathbb{W})$. Ensuring a safe mission of the robot comes down to finding a time $t > 0$ such that the robot senses a landmark, *i.e.*

$$\exists t > 0, \exists i \in \llbracket 1, l \rrbracket, \mathcal{R}_t(\mathbb{X}_0, \mathbb{W}) \subset \mathbb{S}_i \quad (1.3)$$

The objective of the thesis is then to compute this reachable set, and to show how it can be used in a context of validation.

1.3.2 Image of a set by a function

If we take a step back, the problem we are trying to solve comes down to the computation of the image of a set by a function. In our case it is the image of the initial set of the robot by the function which describes its evolution. The main contribution of this manuscript is a method to perform such computation.

The initial reflection which led to the contributions presented here is that most methods yield an over approximation of the image set. In our context, it means that we are able to compute a set such that the robot state is guaranteed to lie in it. Some parts of the set may in fact never be reached, but we are certain that the robot can not end up outside of it. Such over approximation is useful in critical cases such as obstacle avoidance and risk assessment. In such cases, we want to ensure the safety of the robot and its environment no matter what. It is then acceptable to be “too pessimistic” if we get a guaranteed safety criterion.

However, inner approximations have comparatively received little interest. In our case, an inner approximation of the *reachable set* of the robot is a set where all points are guaranteed to be reachable. The robot may end up somewhere else, but given the disturbances we are sure that these points are possible states of the system. Per definition, the actuators of a robot disrupt its natural behavior and may be considered as disturbances. Such inner approximation can then be used to validate a control law since it describes the points for which there is an input leading the robot there.

Since both approximations have a significant interest, we wanted to design a method to get both at the same time. The idea presented throughout this manuscript and summarized Figure 1.8 is that if we are able to compute an over approximation of the image set boundary, we should be able to deduce both approximations.

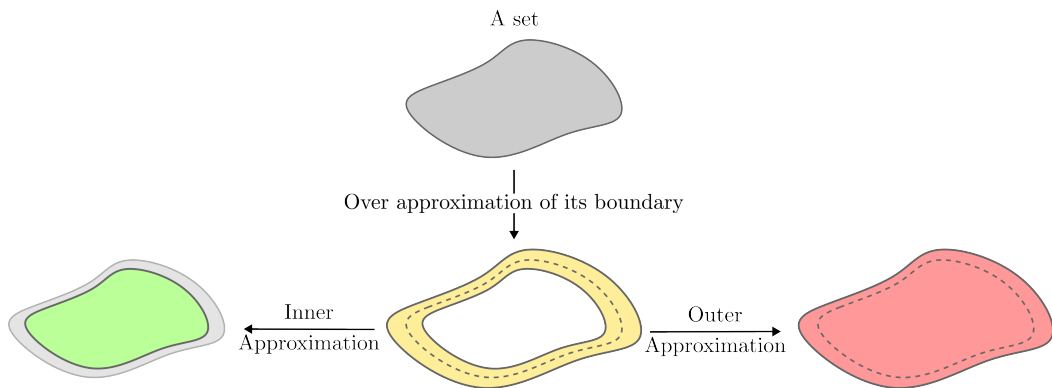


Figure 1.8 – Inner and outer approximation of a set from its boundary

1.3.3 Computation of the reachable set

Going back to our initial problem of computing the reachable set of a mobile robot, we need to account for the disturbances which affect it. Two main possibilities arise : either we consider all the disturbances at every time, or we need to pick the worst disturbances for each possible state.

In the first case, the whole disturbance can be considered in the evolution function of the system. In the second case, it implies that we need to make the disturbance evolve with the system's state to be able to continuously apply the worse one. At some point, the disturbance will have to be considered as an additional state of the system.

The first thing that we tried to reduce in this PhD thesis was the pessimism in the resulting inner and outer approximations. Since both come from an outer approximation of the boundary, it directly implies that we needed to reduce as much as possible the pessimism of this approximation. Out of the two presented possibilities it comes naturally that considering every disturbance at every time will yield a worse result, hence the choice of the second one.

As implied earlier, this choice necessarily results in an increase of the computational complexity of the method. In terms of implementation, it translates as an increasing computation time. However, one should keep in mind that we are computing the reachable set of the robot with an objective in mind : asserting that it will be able to locate itself. With this idea in mind, we will present how our method can be adapted to limit the computational effort as much as possible.

1.3.4 Thesis structure

Figure 1.9 shows the organization and the links between the different chapters of the manuscript. Part I of the manuscript is dedicated to present the notions and definitions that will be used later in the different contributions of Part II.

Part I is divided into three main chapters. Chapter 2 defines the different set representations used in the manuscript and their properties. A brief introduction to both topology and differential geometry is done in Chapter 3 for such notions appear multiple times in the contributions. Chapter 4 formalizes the notion of reachable set of a dynamical system and shows an overview of the existing methods to compute it.

For the contributions, the first and main contribution of the manuscript is in Chapter 5. We display a method to compute both an inner and an outer approximation of a set

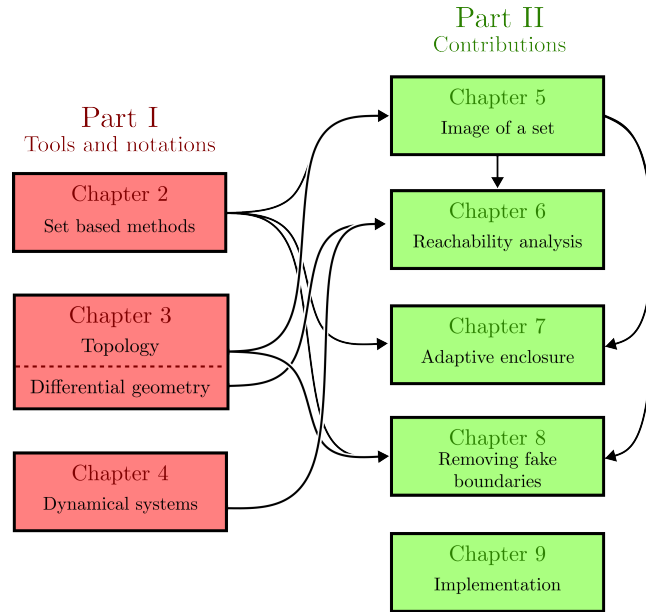


Figure 1.9 – Thesis structure

by a nonlinear function. We also show that an analytic definition of the function is not required, meaning that the method can be used in our context of reachability analysis. This is what is done in Chapter 6 where we present a method to compute the boundary of the reachable set of a dynamical system. We apply the previous method to get an inner and outer approximation of this set, and display an example on a real robotic system. Both contributions raise two main issues : the computation time and the appearance of fake boundaries. Chapters 7 and 8 aim to provide a solution to both problematics while generalizing the findings. Finally a discussion about how the presented methods can be implemented is available in Chapter 9.

NOTATIONS

Sets

$\mathbb{X}$	Set
$\mathcal{P}(\mathbb{X})$	Power set of $\mathbb{X}$
$\text{Cl}(\mathbb{X})$	Closure of $\mathbb{X}$
$\text{Int}(\mathbb{X})$	Interior of $\mathbb{X}$
$\partial\mathbb{X}$	Boundary of $\mathbb{X}$
$\mathbb{X}^+$	Outer approximation of $\mathbb{X}$, $\mathbb{X}^+ \supset \mathbb{X}$
$\mathbb{X}^-$	Inner approximation of $\mathbb{X}$, $\mathbb{X}^- \subset \mathbb{X}$
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of real vectors of size $n > 1$
$\mathbb{PR}^n$	Set of parallelepipeds of $\mathbb{R}^n$
$\mathbb{X}_0$	Initial set
$\mathbb{W}$	Disturbance set

Intervals

$[x]$	Interval
$[\mathbf{x}]$	Interval vector, or box
$\mathbb{IR}$	Set of intervals of $\mathbb{R}$
$\mathbb{IR}^n$	Set of boxes of $\mathbb{R}^n$
$\mathbb{IR}^{n \times m}$	Set of interval matrices of $\mathbb{R}^{n \times m}$
x^+	Upper bound of $[x]$
x^-	Lower bound of $[x]$
$\bar{x}$	Center of $[x]$
$w([\mathbf{x}])$	Width of $[\mathbf{x}]$
$\bar{\mathbf{x}}$	Middle point of $[\mathbf{x}]$

Miscellaneous

f	Scalar function
∇f	gradient of f
$\mathbf{f}$	Vector function
$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$	Jacobian of $\mathbf{f}$

Robotics

$\mathbf{x}$	State vector
$\mathbf{w}$	Disturbance vector
$\dot{\mathbf{x}}$	Derivative of the state with respect to time, $\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt}$
$\mathbf{f}$	Evolution function

PART I

Tools and notations

SET BASED METHODS

2.1 Introduction

By their design, robotic systems are uncertain. This uncertainty comes from multiple factors : actuators are not perfect, the localisation has is uncertain, the robot is subject to environmental perturbations... It is then impossible to estimate exactly the state of the robot and how it will behave. These uncertainties can be of two natures.

Epistemic uncertainties come from a lack of knowledge. They can be reduced with complementary information about the system state. Consider a boat on a lake with no mean to locate itself. The position of the boat is uncertain, i.e. it is “somewhere on the lake”. Yet with some sensors (GNSS, camera...) it could be possible to reduce this uncertainty.

Ontic uncertainties represent the irreducible uncertainty due to the randomness of the real world. If the previous boat on the lake is equipped with a GNSS, it is able to locate itself. As the measurements from the sensor are necessarily noisy, the position provided is not perfect. This uncertainty can be quantified, but can not be reduced.

Set based methods have proven to be efficient to handle such uncertainties. They originated from the work of R. E. Moore who introduced interval analysis in 1966 [92]. Interval analysis aims to represent numbers as intervals instead of single values. Uncertainties can then be represented and propagated to provide a guaranteed result.

This guarantee makes interval analysis suited for critical applications. It notably found numerous applications in the field of robotics where guarantee is required for the safety of both the robot and its environment. Localization [68], state estimation [103] and control [105] are classical topics in robotics where interval analysis has shown to be efficient. Even the problematic of Simultaneous Localization And Mapping (SLAM) [97] has been addressed with intervals to ensure a guaranteed navigation while mapping the environment.

The result of set based methods is often pessimistic in order to be guaranteed. This

pessimism can originate from the formulation of the problem itself, but also from computational limitations. The set representation itself plays a critical role in the reduction of the pessimism.

Since R.E. Moore's results on intervals, other set representations have been considered to handle uncertainties while reducing the conservatism of the result. Ellipsoids [34, 104], polytopes [65] and zonotopes [24] have been explored as they allow for a better wrapping of the solutions. However these objects are more complex to handle, thus increasing the computational and time complexity of the methods. For example, two disks are two ellipsoids, but their intersection is not an ellipsoid. If one wants to work with ellipsoids, this intersection needs to be enclosed in one which increases the computational effort. Choosing a set representation is then a compromise between computation time and pessimism of the result.

2.2 Interval analysis

Intervals are at the root of set based approaches since their apparition in R.E. Moore's work [92]. They allow for a simple and efficient way to represent and propagate uncertainties. The following section provides the definitions and concepts that will be needed later on.

2.2.1 Intervals, Interval vectors and matrices

Let us define the basic objects used in interval analysis.

Definition 1. An *interval* $[x] \in \mathbb{IR}$ is a closed set of real numbers. It is defined by its lower and upper bound, respectively x^- and x^+ such that

$$[x] = [x^-, x^+] = \{x \in \mathbb{R} \mid x^- \leq x \leq x^+\} \quad (2.1)$$

Remark 1. Note that both $-\infty$ and $+\infty$ are possible lower and upper bounds for an interval.

Remark 2. An interval can be punctual, or *degenerate*, when $x^- = x^+$. Degenerate intervals are often noted with braces and behave like reals.

Example 1. Some valid intervals are

— $[-2, 3]$

- $[4] = \{4\}$
- $[-\infty, \infty]$
- $\emptyset$

The last example is the empty set $\emptyset$ and can be considered as a degenerate interval.

Definition 2. An *interval vector*, or *box*, $[\mathbf{x}] \in \mathbb{IR}^n$ is the cartesian product of n intervals

$$[\mathbf{x}] = [x_1] \times \cdots [x_n] \quad (2.2)$$

Definition 3. An *interval matrix* $[\mathbf{A}] \in \mathbb{IR}^{n \times m}$ is a matrix whose elements are intervals.

$$[\mathbf{A}] = \left\{ \mathbf{B} \in \mathbb{R}^{n \times m} \mid \forall (i, j) \in \llbracket 1, n \rrbracket \times \llbracket 1, m \rrbracket, B_{i,j} \in [A_{i,j}] \right\} \quad (2.3)$$

Where $B_{i,j}$ denotes the j -th component of the i -th row of $\mathbf{B}$.

Remark 3. If one component of an interval vector or matrix is degenerate, the interval vector or matrix itself is degenerate.

Remark 4. If one component is the empty set, the box or interval matrix itself is the empty set $\emptyset$.

An interval can be fully described by its lower and upper. Yet for the sake of practicality we can define its center and width.

Definition 4. The *center* of an interval $[x]$ is noted $\bar{x}$ and defined as

$$\bar{x} = \frac{x^+ - x^-}{2} \quad (2.4)$$

Definition 5. The *width* of an interval $[x]$ is noted w and defined as

$$w([x]) = x^+ - x^- \quad (2.5)$$

The width and center definitions on intervals can be extended to interval vectors and matrices.

Definition 6. The *center* of a box $[\mathbf{x}]$ is noted $\bar{\mathbf{x}}$ and defined as

$$\bar{\mathbf{x}} = \begin{pmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_n \end{pmatrix} \quad (2.6)$$

Definition 7. The *width* of a box $[\mathbf{x}]$ is noted w and defined as the length of its largest side

$$w([\mathbf{x}]) = \max_{i \in \llbracket 1, n \rrbracket} (w([x_i])) \quad (2.7)$$

Definition 8. The *center* of an interval matrix $[\mathbf{A}]$ is noted $\bar{\mathbf{A}}$ and defined as

$$\bar{\mathbf{A}} = \begin{pmatrix} \overline{A_{1,1}} & \cdots & \overline{A_{1,m}} \\ \vdots & \ddots & \vdots \\ \overline{A_{n,1}} & \cdots & \overline{A_{n,m}} \end{pmatrix} \quad (2.8)$$

Definition 9. The *width* of an interval matrix $[\mathbf{A}]$ is noted w and defined as the length of its largest side

$$w([\mathbf{A}]) = \max_{(i,j) \in \llbracket 1, n \rrbracket \times \llbracket 1, m \rrbracket} (w([A_{i,j}])) \quad (2.9)$$

2.2.2 Interval operations

Set operations on intervals

Classical set operations can be applied on intervals

Definition 10. The *intersection* of two intervals $[x]$ and $[y]$ is the interval

$$[x] \cap [y] = \{z \in \mathbb{R} \mid z \in [x] \wedge z \in [y]\} \quad (2.10)$$

Definition 11. The *union* of two intervals $[x]$ and $[y]$ is defined as

$$[x] \cup [y] = \{z \in \mathbb{R} \mid z \in [x] \vee z \in [y]\} \quad (2.11)$$

The union of two intervals may not be an interval. We then define the operation of interval union

Definition 12. The *interval union* of two intervals $[x]$ and $[y]$ is defined as the smallest interval enclosing $[x] \cup [y]$

$$[x] \sqcup [y] = [[x] \cup [y]] \quad (2.12)$$

Example 2. Below is an example of set operations applied on intervals

- $[1, 3] \cap [2, 4] = [2, 3]$
- $[-10, 2] \cap [2, \infty] = \{2\}$
- $[1, 6] \cap \emptyset = \emptyset$
- $[-1, 2] \sqcup [5, 7] = [-1, 7]$
- $[-\infty, -2] \sqcup [2, \infty] = [-\infty, \infty]$
- $[2, 5] \sqcup \emptyset = [2, 5]$

These operations can be extended to interval vectors and interval matrices by applying the operator component-wise.

Example 3. Below is an example of set operations applied on interval vectors

- $\begin{pmatrix} [1, 3] \\ [-1, 2] \end{pmatrix} \cap \begin{pmatrix} [2, 4] \\ [-0.5, 1] \end{pmatrix} = \begin{pmatrix} [2, 3] \\ [-0.5, 1] \end{pmatrix}$
- $\begin{pmatrix} [1, \infty] \\ [-2, 4] \end{pmatrix} \cap \emptyset = \emptyset$
- $\begin{pmatrix} [-1, 2] \\ [-\infty, -2] \end{pmatrix} \sqcup \begin{pmatrix} [5, 7] \\ [2, \infty] \end{pmatrix} = \begin{pmatrix} [-1, 7] \\ [-\infty, \infty] \end{pmatrix}$
- $\begin{pmatrix} [1, 3] \\ [-1, 2] \end{pmatrix} \sqcup \emptyset = \begin{pmatrix} [1, 3] \\ [-1, 2] \end{pmatrix}$

Interval arithmetic

An interval arithmetic is defined to extend classical arithmetic operators on reals to intervals.

Definition 13. Consider two intervals $[x]$ and $[y]$ and a binary operator $\diamond \in \{+, -, \cdot, /\}$. The result of $[x] \diamond [y]$ is defined as the smallest interval enclosing $x \diamond y$ for every $x \in [x]$ and $y \in [y]$ [91]

$$[x] \diamond [y] = [\{x \diamond y \mid x \in [x] \text{ and } y \in [y]\}] \quad (2.13)$$

Example 4. Below are examples of interval computations

- $[-1, 2] + [4, 7] = [-3, 9]$
- $[2, 5] - [1, 3] = [-1, 4]$
- $[-2, 3] \cdot [2, 3] = [-6, 9]$
- $[-4, -2] / [1, 2] = [-4, -1]$

As for set operations, binary operations can be extended to interval vectors and interval matrices by applying them component-wise.

Example 5. Below are examples of binary operators applied on interval vectors and matrices

$$\begin{aligned} & - \begin{pmatrix} [1, 3] \\ [-1, 2] \end{pmatrix} + \begin{pmatrix} [2, 4] \\ [-3, 1] \end{pmatrix} = \begin{pmatrix} [3, 7] \\ [-4, 3] \end{pmatrix} \\ & - \begin{pmatrix} [-2, 3] \\ [-3, -2] \end{pmatrix} \cdot \begin{pmatrix} [2, 3] \\ [3, 4] \end{pmatrix} = \begin{pmatrix} [-6, 9] \\ [-12, -6] \end{pmatrix} \\ & - \begin{pmatrix} [-1, 5] & [1, 4] \\ [-4, -2] & [2, 3] \end{pmatrix} \cdot \begin{pmatrix} [1, 2] \\ [-1, 2] \end{pmatrix} = \begin{pmatrix} [-6, 18] \\ [-11, 4] \end{pmatrix} \end{aligned}$$

2.2.3 Inclusion function

Inclusion functions are at the root of interval based methods. Considering a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a n -dimensional box $[\mathbf{x}]$, the image of $[\mathbf{x}]$ by $\mathbf{f}$ is not necessarily a box. Depending on the properties of $\mathbf{f}$, the image set may even be non-connected as depicted Figure 2.1 where $n = m = 2$.

For easier handling, an inclusion function of $\mathbf{f}$ encapsulating the image $\mathbf{f}([\mathbf{x}])$ in a single box can be defined.

Definition 14. Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. An inclusion function of $\mathbf{f}$ is a function $[\mathbf{f}] : \mathbb{IR}^n \rightarrow \mathbb{IR}^m$ such that

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subseteq [\mathbf{f}]([\mathbf{x}]) \quad (2.14)$$

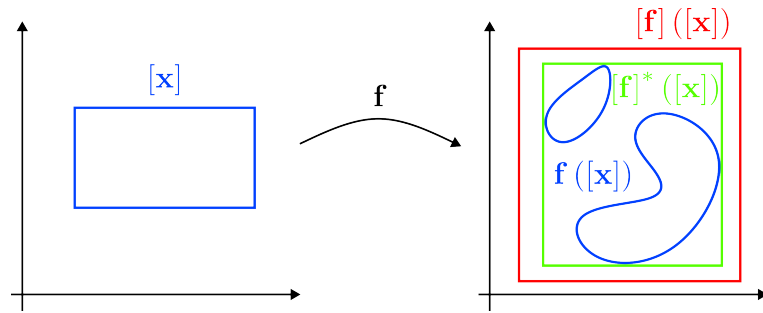


Figure 2.1 – Inclusion function

For a given function $\mathbf{f}$, there exists an infinite number of inclusion functions $[\mathbf{f}]$. Yet only one of them is minimal, i.e. it returns the smallest box enclosing the image set. This minimal inclusion function is often noted $[\mathbf{f}]^*$ as in green shown Figure 2.1.

Remark 5. The minimal inclusion function of $\mathbf{f}$ can easily be computed in the case where the function is monotonic. In this case, evaluating the function $\mathbf{f}$ on the lower and upper bounds of the input box $[\mathbf{x}]$ is sufficient.

Example 6. Consider the function

$$\begin{aligned} f : \mathbb{R} &\rightarrow \mathbb{R} \\ x &\mapsto \log_{10}(x) \end{aligned} \quad (2.15)$$

This function is increasing, and its minimal inclusion function is simply

$$\begin{aligned} [f]^* : \mathbb{IR} &\rightarrow \mathbb{IR} \\ [x] &\mapsto [\log_{10}(x^-), \log_{10}(x^+)] \end{aligned} \quad (2.16)$$

For classical elementary functions such as $\cos$ or $\exp$, their minimal inclusion function is simply noted with square braces $[\cos]$ and $[\exp]$.

Notable inclusion functions in interval analysis are the natural inclusion function and the centered form.

Definition 15. The *natural inclusion function*, also called *natural interval extension*, of a function $\mathbf{f}$ is an inclusion function of $\mathbf{f}$. It is obtained by replacing all the variables and elementary functions ($\sin, \exp, \sqrt{\cdot}, \dots$) of $\mathbf{f}$ by their interval counterpart.

Example 7. Consider the function

$$\begin{aligned} \mathbf{f} : \mathbb{R} &\rightarrow \mathbb{R}^2 \\ x &\mapsto \begin{pmatrix} 2 \cdot \sin(x) \\ \exp(x) + 3 \end{pmatrix} \end{aligned} \quad (2.17)$$

Its natural inclusion function is

$$\begin{aligned} [\mathbf{f}] : \mathbb{IR} &\rightarrow \mathbb{IR}^2 \\ [x] &\mapsto \begin{pmatrix} 2 \cdot [\sin]([x]) \\ [\exp]([x]) + 3 \end{pmatrix} \end{aligned} \quad (2.18)$$

Definition 16. Consider a C^1 function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and note $\mathbf{J}_{\mathbf{f}} : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$ its Jacobian. The centered form of $\mathbf{f}$ is $[\mathbf{f}_c]$ and is defined as

$$\begin{aligned} [\mathbf{f}_c] : \mathbb{IR}^n &\rightarrow \mathbb{IR}^m \\ [\mathbf{x}] &\mapsto \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_{\mathbf{f}}]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \end{aligned} \quad (2.19)$$

Where $[\mathbf{J}_f]$ is the natural extension of $\mathbf{J}_f$. In addition, $[\mathbf{f}_c]$ is an inclusion function of $\mathbf{f}$ [93].

2.2.4 Origin of pessimism

Set-based methods are inherently pessimistic. This property is directly due to the fact that they provide a guaranteed enclosure of the result while trying to wrap it in a given object (box, ellipsoid ...). Consider a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, then the set-valued function

$$\begin{aligned} [\mathbf{f}] : \mathbb{IR}^n &\rightarrow \mathbb{IR}^m \\ [\mathbf{x}] &\mapsto [-\infty, \infty]^m \end{aligned} \quad (2.20)$$

is a valid inclusion function for $\mathbf{f}$. However, the result is way too pessimistic to be able to conclude anything. This shows that the pessimism (sometimes called conservatism) of such methods needs to be understood to avoid unusable results. Pessimism in interval analysis has two main origins.

Wrapping effect

The first noticeable source of pessimism lies in the set representation itself. In interval analysis, the boxes that we consider are necessarily axis-aligned. Then representing any non axis-aligned set by a box leads to a pessimistic enclosure representation. This phenomenon is called the *wrapping effect*. Figure 2.2 shows this phenomenon in the two dimensional case.

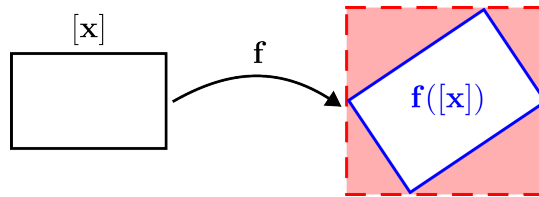


Figure 2.2 – Wrapping effect

In this example the blue set is not axis-aligned. The red box is chosen as its bounding box, i.e. it is the best interval approximation that we can get for this set. Yet the red area shows that we add pessimism with our representation.

This wrapping effect accumulates at each computation, thus making the result more pessimistic each time.

Example 8. An easy way to visualize this, and the one considered in [92], is to consider consecutive rotations of a box. Consider the initial box $[\mathbf{x}_0] = \begin{pmatrix} [-0.1, 0.1] \\ [-0.05, 0.05] \end{pmatrix}$ and a rotation matrix $\mathbf{R} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$. This matrix corresponds to a rotation of $\frac{\pi}{4}$ rad.

Figure 2.3 shows in blue $[\mathbf{x}_0]$, $\mathbf{x}_1 = \mathbf{R} \cdot [\mathbf{x}_0]$ and $\mathbf{x}_2 = \mathbf{R}^2 \cdot [\mathbf{x}_0]$

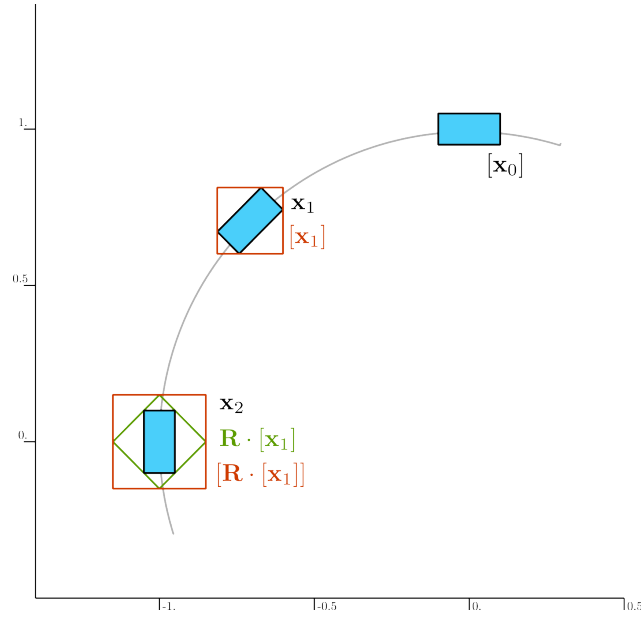


Figure 2.3 – Accumulation of pessimism

Computing $\mathbf{R} \cdot [\mathbf{x}_1]$, in green Figure 2.3 gives the green parallelepiped. Enclosing it in the red box gives a box even larger due to the accumulation of pessimism.

Quality of the inclusion function

Another source of pessimism comes from the inclusion function itself. As mentioned earlier, each function has an infinite number of inclusion function. Only one of them is minimal, but some give better results than other.

One explanation for this lies in the analytical expression chosen for the function. When a variable appears several times in the expression, the associated natural inclusion function gets more pessimistic.

Example 9. Consider two real-valued functions

$$\begin{cases} f_1(x) = x^2 \\ f_2(x) = x \cdot x \end{cases}$$

For any real value $x \in \mathbb{R}$, these two functions are completely equivalent. However if we consider their natural interval counterparts

$$\begin{cases} [f_1]([x]) = [x]^2 \\ [f_2]([x]) = [x] \cdot [x] \end{cases}$$

the expression where $[x] \in \mathbb{IR}$ appears only once gives a better enclosure. If we consider $[x] = [-2, 4]$, we have

$$\begin{aligned} [f_1]([-2, 4]) &= [-2, 4]^2 \\ &= [0, 16] \end{aligned} \tag{2.21}$$

and

$$\begin{aligned} [f_2]([-2, 4]) &= [-2, 4] \cdot [-2, 4] \\ &= [-8, 16] \end{aligned} \tag{2.22}$$

In addition, multiple methods exist to construct an inclusion function. The natural extension, the centered form and the mean value form are examples of it. To compare the quality of the approximation provided by an inclusion function, we introduce the convergence order.

Definition 17. Given a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the inclusion function $[\mathbf{f}]$ converges with order k if for any small box $[\mathbf{x}] \in \mathbb{IR}^n$ the excess width satisfies

$$w([\mathbf{f}]([\mathbf{x}])) - w(\mathbf{f}([\mathbf{x}])) = O(w([\mathbf{x}])^k) \tag{2.23}$$

The *convergence order* of $[\mathbf{f}]$ is then the largest k such that $[\mathbf{f}]$ converges with order k . Figure 2.4 shows this excess width in the case where $n = m = 1$.

A standard result in interval analysis is that the natural extension of a function has a convergence order of 1 [94]. In [50], Hansen has shown that the excess width of the centered form is of order $w([\mathbf{x}])^2$. According to our definition, the centered form then has a convergence order of 2.

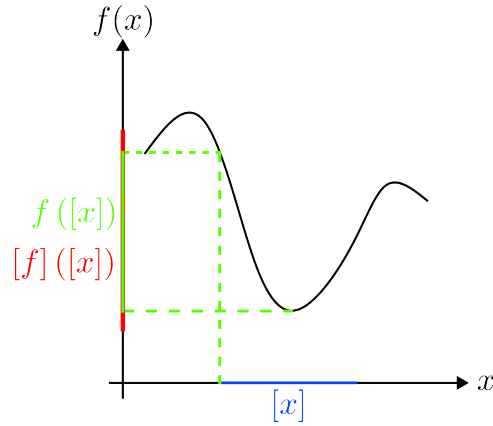


Figure 2.4 – Excess width of the inclusion

2.2.5 Pavings and pavers

Principle

Approximating a set with a single box can lead to high pessimism due to the wrapping effect presented in Subsection 2.2.4. To paliate this issue, one can use a collection of boxes instead of a single box.

Definition 18. A *paving* $\mathbb{K}$ is a finite union of boxes

$$\mathbb{K} = \bigcup_{i=1}^k [\mathbf{x}_i] \quad (2.24)$$

such that the interiors of all the boxes are disjoint, i.e. they are non-overlapping.

As depicted Figure 2.5, pavings allow for a tighter enclosure of any set.

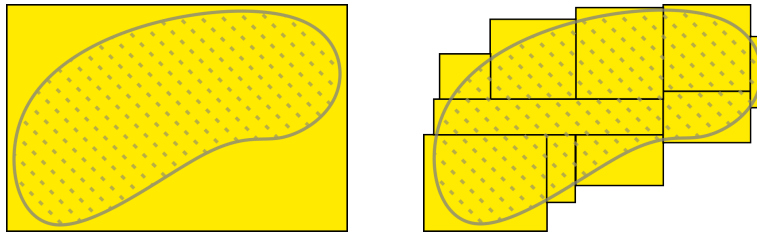


Figure 2.5 – Approximation of a set with a box (left) and a paving (right)

An even better representation of a set can be obtained with pavings. Considering a set $\mathbb{X}$, it can be enclosed between an inner and an outer approximation.

Definition 19. An *inner approximation* of a set $\mathbb{X}$ is a set $\mathbb{X}^-$ such that

$$\mathbb{X}^- \subseteq \mathbb{X} \quad (2.25)$$

Definition 20. An *outer approximation* of a set $\mathbb{X}$ is a set $\mathbb{X}^+$ such that

$$\mathbb{X} \subseteq \mathbb{X}^+ \quad (2.26)$$

Pavings are especially suited to get both approximations. In this context, both $\mathbb{X}^-$ and $\mathbb{X}^+$ are called *subpavings*.

Figure 2.6 shows how the set $\mathbb{X}$ can be approximated with two pavings. The green paving $\mathbb{X}^-$ is an inner approximation of $\mathbb{X}$. The paving $\mathbb{X}^+$ is an outer approximation of $\mathbb{X}$ and is composed of both the green and yellow boxes. We can already note that the yellow boxes provide an outer approximation of the boundary of $\mathbb{X}$. This notion of boundary will be discussed later on in Section 3.2.1.

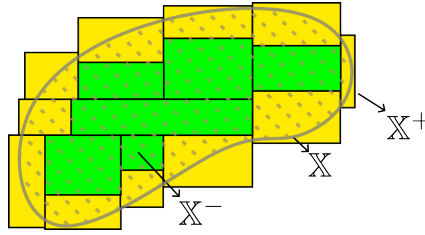


Figure 2.6 – Inner (green) and outer (green + yellow) approximation of a set

The SIVIA algorithm

A classical paving algorithm is the Set Inversion Via Interval Analysis (SIVIA) algorithm [61]. As mentioned in its name, this algorithm relies on interval analysis to solve a set-inversion problem. Consider a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a set $\mathbb{Y} \subset \mathbb{R}^m$, the objective of a set inversion problem is to characterize the set

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} \quad (2.27)$$

The SIVIA algorithm requires an initial box $[\mathbf{x}]$ and an inclusion function $[\mathbf{f}]$ for $\mathbf{f}$. It returns two pavings $\mathbb{X}^-$ and $\mathbb{X}^+$ representing an inner and outer approximation of $\mathbb{X}$.

To do so, it evaluates $[\mathbf{f}]([\mathbf{x}])$ and checks if it belongs to $\mathbb{Y}$ or not. Three results are possible :

1. $[f]([x]) \cap Y = \emptyset$: we are certain that the box $[x]$ does not belong to X .
2. $[f]([x]) \subseteq Y$: we are certain that the box $[x]$ belongs to X , it is added to both X^- and X^+ .
3. $[f]([x]) \cap Y \neq \emptyset$ and $[f]([x]) \not\subseteq Y$: the box $[x]$ may or may not belong to X .

This last case is critical in the SIVIA algorithm since we can not conclude on whether or not $[x]$ belongs to X . In this case, the box $[x]$ is bisected in two boxes, and the test is runned again on both. If one of the boxes is still undecisive, it is also bisected and so on.

Due to the pessimism in the computation of $[f]([x])$, it is most of the time impossible to stop the algorithm as is. Indeed, even for a tiny box $[x]$ close to the boundary of X , the test $[f]([x]) \subseteq Y$ may fail. A limit size ϵ is then introduced, and the boxes smaller than this criterium are added to the outer approximation X^+ .

A full recursive version of the SIVIA algorithm is proposed Algorithm 1

Algorithm 1 Recursive SIVIA for the function f

```

1: Inputs:  $[f] : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $[x] \in \mathbb{R}^n$ ,  $Y \subset \mathbb{R}^m$ ,  $\epsilon > 0$ 
2: Outputs:  $(X^-, X^+)$ 
3:
4: function SIVIA( $[f]$ ,  $[x]$ ,  $Y$ ,  $\epsilon$ ,  $X^- = \{\}$ ,  $X^+ = \{\}$ )
5:   if  $[f]([x]) \cap Y \neq \emptyset$  then
6:     if  $w([x]) < \epsilon$  then
7:        $X^+ \leftarrow X^+ \cup [x]$ 
8:     else if  $[f]([x]) \subseteq Y$  then
9:        $X^- \leftarrow X^- \cup [x]$ 
10:       $X^+ \leftarrow X^+ \cup [x]$ 
11:     else
12:        $([x_1], [x_2]) \leftarrow \text{bisect}([x])$ 
13:       SIVIA( $[f]$ ,  $[x_1]$ ,  $Y$ ,  $\epsilon$ ,  $X^-$ ,  $X^+$ )
14:       SIVIA( $[f]$ ,  $[x_2]$ ,  $Y$ ,  $\epsilon$ ,  $X^-$ ,  $X^+$ )
15:     end if
16:   end if
17: end function

```

Remark 6. In paving algorithms, we often choose to bisect the box along its largest dimension. Since symmetries may arise from the expression of our problem, the bisection is regularly not done in the middle of the dimension to break these potential symmetries. Depending on the library used, the box is often splitted with a 45/55 or 49/51 ratio.

Example 10. Consider the set $Y = [1, 3]$ and the 2D distance function

$$\mathbf{f} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \sqrt{x_1^2 + x_2^2} \quad (2.28)$$

The inverse set $\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y})$ is a ring of inner radius 1 and outer radius 3. For this example, we will consider the centered form of $\mathbf{f}$ noted $[\mathbf{f}_c]$. Figure 2.7 shows the result of $\text{SIVIA}([\mathbf{f}_c], [-4, 4]^2, \mathbb{Y}, \epsilon)$ for two different values of ϵ .

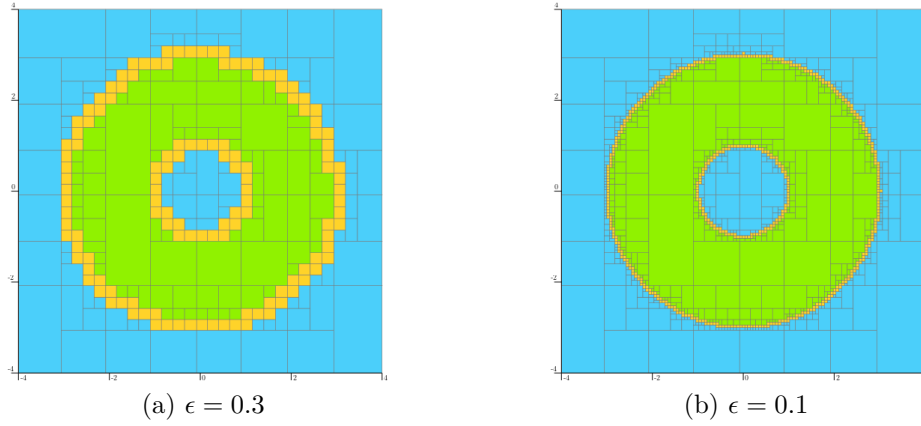


Figure 2.7 – Outputs of the SIVIA algorithm

Note that in this example the boxes outside of $\mathbb{X}$, i.e. the boxes $[\mathbf{x}]$ such that $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$ are in blue.

2.2.6 Contractors

Constraints

In Interval Analysis, contractors are tools to compute the part of a box which satisfies a given constraint. These constraints can be formalized as boolean functions.

Definition 21. A *constraint* $\mathcal{L}$ is a boolean function defined by :

$$\begin{aligned} \mathcal{L} : \mathbb{R}^n &\rightarrow \{0, 1\} \\ \mathbf{x} &\mapsto \begin{cases} 1 & \text{if the constraint is satisfied} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (2.29)$$

Remark 7. This definition of constraints can be extended to sets. A constraint is validated on a set if it is validated in every point of it, i.e.

$$\begin{aligned}
 \mathcal{L} : \mathcal{P}(\mathbb{R}^n) &\rightarrow \{0, 1\} \\
 \mathbb{X} &\mapsto \begin{cases} 1 & \text{if } \forall \mathbf{x} \in \mathbb{X}, \mathcal{L}(\mathbf{x}) = 1 \\ 0 & \text{if } \forall \mathbf{x} \in \mathbb{X}, \mathcal{L}(\mathbf{x}) = 0 \\ \{0, 1\} & \text{otherwise} \end{cases}
 \end{aligned} \tag{2.30}$$

Since constraints are boolean functions, logical operations can be applied to combine two or more constraints.

Definition 22. Consider two constraints $\mathcal{L}_1 : \mathbb{R}^n \rightarrow \{0, 1\}$ and $\mathcal{L}_2 : \mathbb{R}^n \rightarrow \{0, 1\}$ and a logical operator $\diamond = \{\wedge, \vee, \bar{}, \underline{}\}$. The constraint $\mathcal{L} = \mathcal{L}_1 \diamond \mathcal{L}_2$ is defined as

$$\begin{aligned}
 \mathcal{L} : \mathbb{R}^n &\rightarrow \{0, 1\} \\
 \mathbf{x} &\mapsto \begin{cases} 1 & \text{if } \mathcal{L}_1(\mathbf{x}) \diamond \mathcal{L}_2(\mathbf{x}) = 1 \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{2.31}$$

The complementary of a constraint can also be defined.

Definition 23. Consider a constraint $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$. Its *complementary* $\bar{\mathcal{L}}$ is defined as

$$\begin{aligned}
 \bar{\mathcal{L}} : \mathbb{R}^n &\rightarrow \{0, 1\} \\
 \mathbf{x} &\mapsto \begin{cases} 1 & \text{if } \mathcal{L}(\mathbf{x}) = 0 \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{2.32}$$

Contractors

A contractor associated with a constraint $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$ is a function from $\mathbb{IR}^n$ to $\mathbb{IR}^n$. Given a box, it returns the part of it which satisfies the constraint.

Definition 24. A *contractor* $\mathcal{C}_{\mathcal{L}}$ associated with the constraint $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$ is a function $\mathcal{C}_{\mathcal{L}} : \mathbb{IR}^n \rightarrow \mathbb{IR}^n$ which satisfies the following properties

- $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}]$ (Contractance)
- $\forall \mathbf{x} \in [\mathbf{x}], \mathcal{L}(\mathbf{x}) = 1 \implies \mathbf{x} \in \mathcal{C}_{\mathcal{L}}([\mathbf{x}])$ (Consistency)

A contractor is said to be optimal when it outputs the smallest box which satisfies the desired constraint.

Example 11. Consider Himmelblau's function from [53]

$$\mathbf{f} : \begin{array}{ccc} \mathbb{R}^2 & \rightarrow & \mathbb{R} \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \mapsto & (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \end{array} \quad (2.33)$$

From this function, we can define the constraint

$$\mathcal{L} : \mathbb{R}^2 \rightarrow \begin{array}{ccc} & \{0, 1\} \\ \mathbf{x} & \mapsto & \begin{cases} 1 & \text{if } \mathbf{f}(\mathbf{x}) = 80 \\ 0 & \text{otherwise} \end{cases} \end{array} \quad (2.34)$$

Such constraint can be referred to as an *inverse constraint* since it comes down to an inversion problem. Figure 2.8 shows in black the solution set $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{f}(\mathbf{x}) = 80\}$. Five initial boxes $[\mathbf{x}]$ are considered and drawn in blue, and the results after contraction $\mathcal{C}([\mathbf{x}])$ are drawn in red.

In the case where no red box is displayed, we can assert that no solution was contained in the considered box. This is directly deduced from the consistency property of contractors.

Remark 8. In the expression of $\mathbf{f}$, both x_1 and x_2 appear twice. As mentionned in Section 2.2.4 this multi-occurency generates pessimism. The contractor associated with the function is then not minimal.

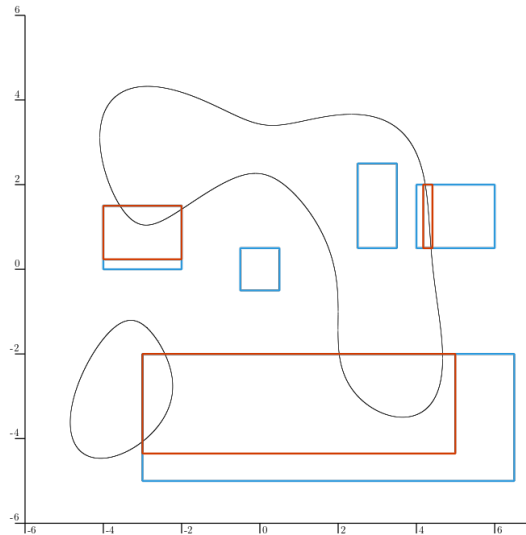


Figure 2.8 – Contractor on the inverse constraint

Contractor programming

Contractors are powerful tools when it comes to handling a single constraint. Yet for more complex cases, it may be hard to formalize the problem with a single constraint. A single contractor may then be hard to find.

The contractor programming introduced in [20] proposes to consider contractors not as functions from $\mathbb{R}^n$ to $\mathbb{R}^n$, but as subsets of $\mathbb{R}^n$. Classical operations on sets can then be extended to contractors.

Definition 25. Consider two contractors $\mathcal{C}_1 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $\mathcal{C}_2 : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Then for any box $[\mathbf{x}] \in \mathbb{IR}^n$ we define

$$\begin{aligned}
 (\mathcal{C}_1 \cap \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \cap \mathcal{C}_2([\mathbf{x}]) && \text{(intersection)} \\
 (\mathcal{C}_1 \cup \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1([\mathbf{x}]) \cup \mathcal{C}_2([\mathbf{x}]) && \text{(union)} \\
 (\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) &:= \mathcal{C}_1(\mathcal{C}_2([\mathbf{x}])) && \text{(composition)} \\
 \mathcal{C}_1^\infty &:= \mathcal{C}_1 \circ \mathcal{C}_1 \circ \mathcal{C}_1 \circ \dots && \text{(iterated composition)}
 \end{aligned} \tag{2.35}$$

This formalism allows for a simpler handling of contractors. A complex problem can be decomposed into a set of constraint, with an associated contractor for each constraint. These elementary contractors can then be composed, intersected etc... to construct a single contractor which would have been hard to write analytically.

It has allowed for a wider diffusion of contractor-based approaches. Indeed, once the main elementary contractors are implemented in a library, end-users can easily use them to solve their own problematic. A discussion about interval analysis libraries and their scope is available in Section 2.4.

Example 12. Say that we want to compute a contractor for the intersection of two pies. The first pie has a center $\mathbf{a}_1 = \begin{pmatrix} 2 & 3 \end{pmatrix}^T$, a radius $[r_1] = [2, 3]$ and an angle $[\theta_1] = \left[-\pi, -\frac{\pi}{2}\right]$. The second pie has a center $\mathbf{a}_2 = \begin{pmatrix} -2 & -2 \end{pmatrix}^T$, a radius $[r_2] = [3, 4.1]$ and an angle $[\theta_2] = \left[0, \frac{\pi}{2}\right]$. We will denote $a_{1,1}$ the first element of $\mathbf{a}_1$, and $a_{1,2}$ its second element. The same stands for $\mathbf{a}_2$.

This constraint could be written from a single function providing both ranges and angles

$$\begin{aligned} \mathbf{f} : \mathbb{R}^2 &\rightarrow \mathbb{R}^4 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &\mapsto \begin{pmatrix} \sqrt{(x_1 - a_{1,1})^2 + (x_2 - a_{1,2})^2} \\ \text{atan2}(x_2 - a_{1,2}, x_1 - a_{1,1}) \\ \sqrt{(x_1 - a_{2,1})^2 + (x_2 - a_{2,2})^2} \\ \text{atan2}(x_2 - a_{2,2}, x_1 - a_{2,1}) \end{pmatrix} \end{aligned} \quad (2.36)$$

The constraint can then be expressed as an inverse constraint

$$\begin{aligned} \mathcal{L}_{inv} : \mathbb{R}^2 &\rightarrow \{0, 1\} \\ \mathbf{x} &\mapsto \begin{cases} 1 & \text{if } \mathbf{f}(\mathbf{x}) \in \begin{pmatrix} [r_1] \\ [\theta_1] \\ [r_2] \\ [\theta_2] \end{pmatrix} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (2.37)$$

We will denote $\mathcal{C}_{inv}$ the contractor associated with this inverse constraint. Another way to express this same problem is to rely on an elementary constraint : the polar constraint. if we define

$$\pi \begin{pmatrix} \rho \\ \theta \end{pmatrix} = \begin{pmatrix} \rho \cdot \cos(\theta) \\ \rho \cdot \sin(\theta) \end{pmatrix} \quad (2.38)$$

The polar constraint is

$$\begin{aligned} \mathcal{L}_{pol} : \mathbb{R}^4 &\rightarrow \{0, 1\} \\ \begin{pmatrix} x_1 \\ x_2 \\ \rho \\ \theta \end{pmatrix} &\mapsto \begin{cases} 1 & \text{if } \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \pi \begin{pmatrix} \rho \\ \theta \end{pmatrix} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (2.39)$$

A minimal contractor $\mathcal{C}_{pol}$ for this polar constraint was built in [31] and will be used for our problem. The contractor for the intersection of both pies is then the intersection of two polar contractors $\mathcal{C} = \mathcal{C}_{pol,1} \cap \mathcal{C}_{pol,2}$. The output of both contractors for the initial box $\mathbf{x}_0 = \begin{pmatrix} [0.5, 1.5] \\ [1, 2.5] \end{pmatrix}$ is displayed on Figure 2.9.

We can notice that both contractors output the same result. Yet the approach based on elementary constraints is more general and can even be used in way more complex

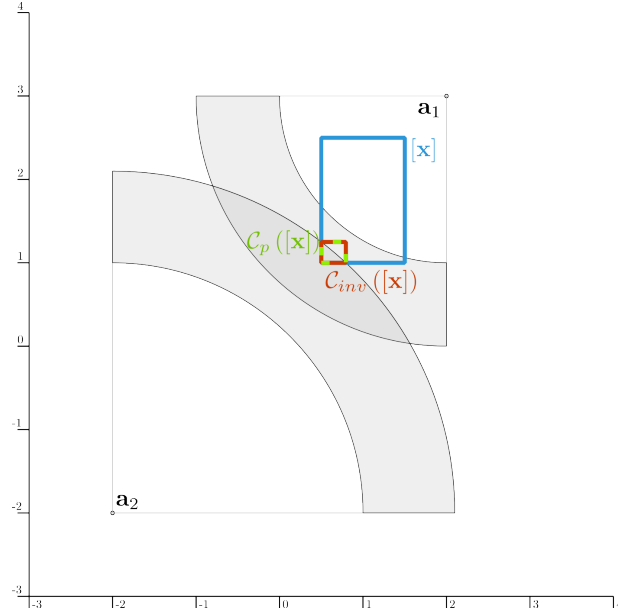


Figure 2.9 – Contractor on the intersection of two pies

cases. Sometimes, it may even not be possible to express the problem with a single function to inverse as we did there.

Paving with contractors

Contractor can be used in paving algorithms to make them more efficient. Consider a constraint $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$ and an box $[x] \in \mathbb{IR}^n$. If we want to characterise an outer approximation of the set

$$\{x \in [x] \mid \mathcal{L}(x) = 1\} \quad (2.40)$$

A first approach could replicate the principle of the SIVIA algorithm with a simple bisect and test logic. This paving algorithm is described in Algorithm 2. Its inputs are the constraint $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$ and the initial box $[x] \in \mathbb{IR}^n$. As for the SIVIA algorithm, a limit size ϵ also needs to be introduced to ensure that the algorithm ends. It outputs a paving $\mathbb{X}^+$ enclosing the solution set described Equation 2.40.

The efficiency of this algorithm can be improved if we are able to provide a contractor $\mathcal{C}_{\mathcal{L}}$ for the constraint $\mathcal{L}$. Such contractor can easily be constructed thanks to the contractor programming introduced in Section 2.2.6. Algorithm 3 presents how the previous algorithm can be adapted to benefit from the contractors.

Algorithm 2 Recursive paving without contractor

```
1: Inputs:  $\mathcal{L} : \mathbb{R}^n \rightarrow \{0, 1\}$ ,  $[\mathbf{x}] \in \mathbb{IR}^n$ ,  $\epsilon > 0$ 
2: Output:  $\mathbb{X}^+$ 
3:
4: function PAVE( $\mathcal{L}$ ,  $[\mathbf{x}]$ ,  $\epsilon$ ,  $\mathbb{X}^+ = \{\}$ )
5:   if  $\mathcal{L}([\mathbf{x}]) \neq 0$  then
6:     if  $w([\mathbf{x}]) < \epsilon$  then
7:        $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ 
8:     else
9:        $([\mathbf{x}_1], [\mathbf{x}_2]) \leftarrow \text{bisect}([\mathbf{x}])$ 
10:      pave( $\mathcal{L}$ ,  $[\mathbf{x}_1]$ ,  $\epsilon$ ,  $\mathbb{X}^+$ )
11:      pave( $\mathcal{L}$ ,  $[\mathbf{x}_2]$ ,  $\epsilon$ ,  $\mathbb{X}^+$ )
12:    end if
13:  end if
14: end function
```

Algorithm 3 Recursive paving with contractor

```
1: Inputs:  $\mathcal{C}_{\mathcal{L}} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,  $[\mathbf{x}] \in \mathbb{IR}^n$ ,  $\epsilon > 0$ 
2: Output:  $\mathbb{X}^+$ 
3:
4: function PAVE( $\mathcal{C}_{\mathcal{L}}$ ,  $[\mathbf{x}]$ ,  $\epsilon$ ,  $\mathbb{X}^+ = \{\}$ )
5:    $[\mathbf{x}] \leftarrow \mathcal{C}_{\mathcal{L}}([\mathbf{x}])$ 
6:   if  $[\mathbf{x}] \neq \emptyset$  then
7:     if  $w([\mathbf{x}]) < \epsilon$  then
8:        $\mathbb{X}^+ \leftarrow \mathbb{X}^+ \cup [\mathbf{x}]$ 
9:     else
10:       $([\mathbf{x}_1], [\mathbf{x}_2]) \leftarrow \text{bisect}([\mathbf{x}])$ 
11:      pave( $\mathcal{C}_{\mathcal{L}}$ ,  $[\mathbf{x}_1]$ ,  $\epsilon$ ,  $\mathbb{X}^+$ )
12:      pave( $\mathcal{C}_{\mathcal{L}}$ ,  $[\mathbf{x}_2]$ ,  $\epsilon$ ,  $\mathbb{X}^+$ )
13:    end if
14:  end if
15: end function
```

Example 13. The Tryskell is a geometrical shape composed of multiple circular arcs. A polar constraint can be associated to each arc, and the polar contractor from [31] can be used. Thanks to contractor programming, we are then able to construct a contractor for the Tryskell. Figure 2.10 shows two pavings for the Tryskell, both with an initial box $[\mathbf{x}] = [-10, 10]^2$ and a limit size $\epsilon = 0.25$. Figures 2.10a and 2.10b respectively show the result of the paving algorithm 2 (without contractions) and 3 (with contractions).

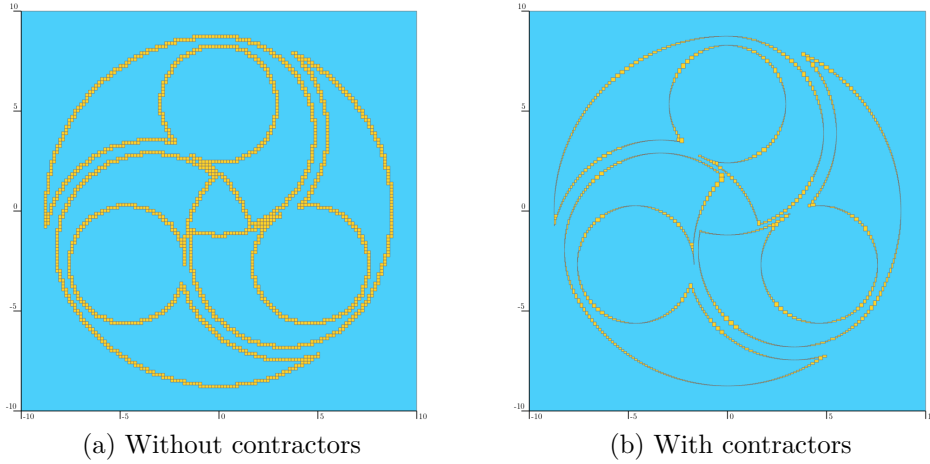


Figure 2.10 – Paving of the Tryskell [38]

Table 2.1 summarizes the performances of both methods. Thanks to the use of contractors, we have been able to get a better result in less time.

	Without contractors	With contractors
Computation time (s)	0.6	0.2
Number of boxes	1414	953
Total area	34.6	11.7
Area per box	0.024	0.012

Table 2.1 – Performance comparison between paving algorithms on the Tryskell

2.3 Zonotopic representations

Since the introduction of Interval Analysis, different kind of sets have been considered to limit the wrapping effect in set-based methods. Between all the possibilities, zonotopic representations have been extensively studied for their interesting properties. Indeed,

they represent a trade off between the simplicity of the representation and the quality of the wrapping.

2.3.1 Zonotopes

Polytopes are the generalization of polyhedrons in higher dimensions. They can be defined as objects with flat faces. Convex polytopes can be represented by either their vertices, or by the intersection of half spaces [132, 46]. Since both these representations are computationally expensive to propagate, special cases of convex polytopes are often preferred. In this context, zonotopes are symmetric and convex polytopes.

Definition 26. A zonotope is a convex and centrally symmetric polytope whose faces are also centrally symmetric. A zonotope $\mathcal{Z}$ of $\mathbb{R}^n$ can be described by its center $\mathbf{c} \in \mathbb{R}^n$ and its shape matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ as

$$\mathcal{Z} = \mathbf{c} + \mathbf{A} \cdot [-1, 1]^m \quad (2.41)$$

The columns of the shape matrix $\mathbf{A}$ are called the generators of the zonotope. Such zonotope can be seen as the Minkowsky sum of m vectors.

Example 14. The 2-dimensional zonotope

$$\mathcal{Z}_1 = \begin{pmatrix} 4 \\ 1.5 \end{pmatrix} + \begin{pmatrix} -0.2 & 0.2 & 0.08 \\ 0.1 & 0.04 & 0.18 \end{pmatrix} \cdot [-1, 1]^3 \quad (2.42)$$

has three generators $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ respectively drawn in red, green and blue Figure 2.11.

Remark 9. A zonotope can be interpreted as an affine image of a box. A connection with affine arithmetic methods [23, 27] can then be established as they rely on the propagation of linear approximations and bounded symbolic noise.

The central symmetry property of zonotopes makes these objects easier to handle. For example, the sum of two zonotopes is a zonotope. Computing it comes down to summing up the center of both zontopes and concatenating their generators.

Example 15. Consider the zonotope $\mathcal{Z}_1$ from Equation 2.42 and the zonotope

$$\mathcal{Z}_2 = \begin{pmatrix} 3 \\ -0.5 \end{pmatrix} + \begin{pmatrix} 0.2 & -0.1 \\ 0.15 & 0.01 \end{pmatrix} \cdot [-1, 1]^2 \quad (2.43)$$

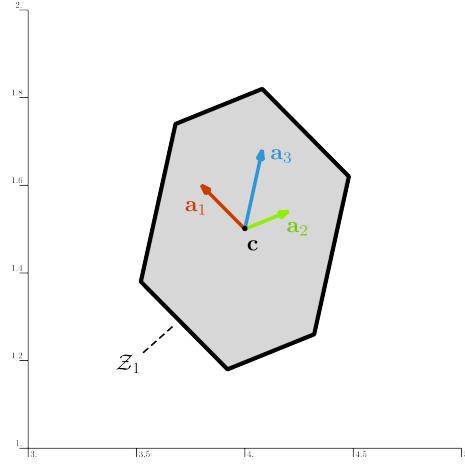


Figure 2.11 – 2-dimensional zonotope

respectively in gray and blue Figure 2.12. Their sum is depicted in red Figure 2.12 and is

$$\mathcal{Z}_1 + \mathcal{Z}_2 = \begin{pmatrix} 7 \\ 1 \end{pmatrix} + \begin{pmatrix} -0.2 & 0.2 & 0.08 & 0.2 & -0.1 \\ 0.1 & 0.04 & 0.18 & 0.15 & 0.01 \end{pmatrix} \cdot [-1, 1]^5 \quad (2.44)$$

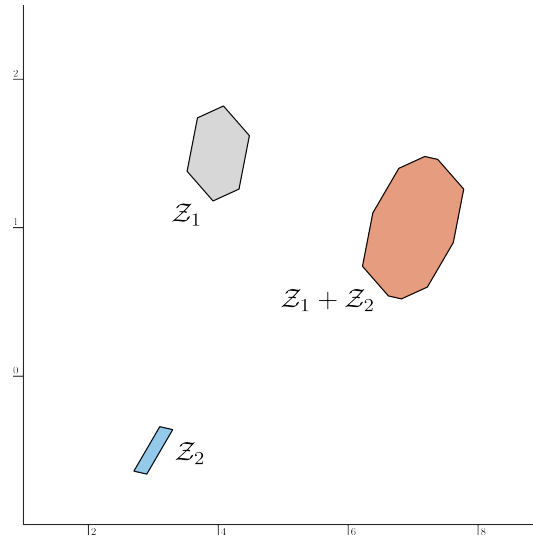


Figure 2.12 – Sum of zonotopes

Thanks to their representation, zonotopes can also easily be projected and the result is also a zonotope.

Definition 27. Consider a zonotope $\mathcal{Z} = \mathbf{c} + \mathbf{A} \cdot [-1, 1]^m$ of $\mathbb{R}^n$. Let $(x_1, \dots, x_n)$ be the

standard basis of $\mathbb{R}^n$ and $\{i_1, \dots, i_p\} \in \llbracket 1, n \rrbracket^p$ be p different indexes. The projection of $\mathcal{Z}$ on the p -dimensional subspace of $\mathbb{R}^n$ defined by the p vectors $(x_{i_1}, \dots, x_{i_p})$ is a zonotope $\mathcal{Z}_p$ defined by :

$$\left\{ \begin{array}{l} \mathcal{Z}_p = \mathbf{p} + \mathbf{B} \cdot [-1, 1]^m \\ \mathbf{p} = \begin{pmatrix} \mathbf{c}_{i_1} \\ \vdots \\ \mathbf{c}_{i_p} \end{pmatrix} \\ \mathbf{B} = \begin{pmatrix} \mathbf{A}_{i_1} \\ \vdots \\ \mathbf{A}_{i_p} \end{pmatrix} \end{array} \right.$$

where for $j \in \llbracket 1, n \rrbracket$, $\mathbf{c}_j$ and $\mathbf{A}_j$ are respectively the j -th row of $\mathbf{c}$ and $\mathbf{A}$.

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

Example 16. Consider the 4-dimensional zonotope

$$\mathcal{Z} = \begin{pmatrix} 4 \\ -5 \\ 1.5 \\ 12 \end{pmatrix} + \begin{pmatrix} -0.2 & 0.2 & 0.08 \\ 0.4 & 5 & 1.2 \\ 0.1 & 0.04 & 0.18 \\ 0.01 & 3 & -4.5 \end{pmatrix} \cdot [-1, 1]^3 \quad (2.45)$$

and denote $(x_1, \dots, x_4)$ the standard basis of $\mathbb{R}^4$. The projection of this zonotope on the plane defined by x_1 and x_3 is the zonotope $\mathcal{Z}_1$ from Equation 2.42.

Since they are not constrained to be axis aligned, zonotopes offer a better approximation of sets than standard boxes. However, the propagation of these zonotopes often comes down to adding a box (or a zonotope) to the current one to account for its evolution or approximation errors [39]. As mentioned earlier in this section, this process increases the number of generators of the zonotope.

This phenomenon can cause an explosion in the number of generators, making the zonotope expensive to handle. The order of the zonotope is a metric used to characterise how complex it is.

Definition 28. Consider a n -dimensional zonotope $\mathcal{Z} \subset \mathbb{R}^n$ and let us note n_g its number of generators (i.e. the number of columns in its shape matrix). The *order* of the zonotope $\mathcal{Z}$ is defined as

$$o(\mathcal{Z}) = \frac{n_g}{n} \quad (2.46)$$

To alleviate this issue of increasing complexity, order reduction techniques [72, 130] are used to simplify zonotopes. This process however has a cost, both in terms of conservatism and computation time. One may then consider parallelepipeds instead as a simpler set representation alternative.

2.3.2 Parallelepipeds

Definitions

Parallelepipeds are zonotopes with a square shape matrix. This makes them easier to use and lighter to compute.

Definition 29. A *parallelepiped* is a zonotope of order 1. As for a zonotope, a parallelepiped $\langle \mathbf{x} \rangle \subset \mathbb{R}^n$ can be defined by its center $\mathbf{c} \in \mathbb{R}^n$ and its shape matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ as

$$\langle \mathbf{x} \rangle = \mathbf{c} + \mathbf{A} \cdot [-1, 1]^n \quad (2.47)$$

The set of parallelepipeds of $\mathbb{R}^n$ will further be denoted by $\mathbb{PR}^n$.

Remark 10. Zonotopes of order inferior to 1 are sometimes referred to as *flat parallelepipeds*.

As parallelepipeds are zonotopes, the Minkowsky sum of zonotopes presented earlier also applies to them. The result of this sum will then be a zonotope. However if two parallelepipeds have a similar shape, this sum can be simplified to provide a parallelepiped. To define this notion of similar shape, we first introduce a decomposition.

Definition 30. Consider a parallelepiped $\langle \mathbf{x} \rangle = \mathbf{c} + \mathbf{A} \cdot [-1, 1]^n$. Its shape matrix $\mathbf{A}$ can be uniquely decomposed into a matrix $\mathbf{N}$ and $\mathbf{D}$ such that

- $\mathbf{A} = \mathbf{N} \cdot \mathbf{D}$
- the columns of $\mathbf{N}$ have unit length
- the matrix $\mathbf{D}$ is diagonal and positive

The matrix $\mathbf{D}$ will further be referenced as the *ratio matrix* of the parallelepiped. The matrix $\mathbf{N}$ will further be referenced as its *normalized shape matrix*. Any further mention of a “decomposition of the shape matrix” will refer to this specific decomposition.

We can then define the similarity of two parallelepipeds.

Definition 31. Two parallelepipeds $\langle \mathbf{x}_1 \rangle$ and $\langle \mathbf{x}_2 \rangle$ have a *similar shape* if they have the same normalized shape matrix, with an eventual permutation of their columns. If we denote $\mathbf{N}_1$ and $\mathbf{N}_2$ their respective normalized shape matrix, the parallelepipeds have a similar shape if there exists a permutation matrix $\mathbf{P}$ such that

$$\mathbf{N}_2 = \mathbf{N}_1 \cdot \mathbf{P} \quad (2.48)$$

This means that two parallelepipeds have the same shape iff the transformation between them is a rescaling with an eventual rearrangement of their generators.

In the case where two parallelepipeds have a similar shape, their Minkowsky sum can be computed without concatenating their generators.

Definition 32. Consider two parallelepipeds of $\mathbb{R}^n$, $\langle \mathbf{x}_1 \rangle = \mathbf{c}_1 + \mathbf{A}_1 \cdot [-1, 1]^n$ and $\langle \mathbf{x}_2 \rangle = \mathbf{c}_2 + \mathbf{A}_2 \cdot [-1, 1]^n$ having the same shape. Both shape matrices can be decomposed as suggested in Definition 30. Then, there exists a permutation matrix $\mathbf{P}$ such that Equation 2.48 is satisfied. The Minkowsky sum of the two parallelepipeds is then the parallelepiped defined as

$$\langle \mathbf{x}_1 \rangle + \langle \mathbf{x}_2 \rangle = (\mathbf{c}_1 + \mathbf{c}_2) + \mathbf{N}_1 \cdot (\mathbf{D}_1 + \mathbf{P} \cdot \mathbf{D}_2) \cdot [-1, 1]^n \quad (2.49)$$

Remark 11. The resulting parallelepiped has the same shape as $\langle \mathbf{x}_1 \rangle$ and $\langle \mathbf{x}_2 \rangle$.

Remark 12. If the two parallelepipeds have the same normalized matrix, their Minkowsky sum is simply

$$\langle \mathbf{x}_1 \rangle + \langle \mathbf{x}_2 \rangle = (\mathbf{c}_1 + \mathbf{c}_2) + (\mathbf{A}_1 + \mathbf{A}_2) \cdot [-1, 1]^n \quad (2.50)$$

The notion of inclusion function presented Section 2.2.3 can be extended to parallelepipeds.

Definition 33. Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. A parallelepiped inclusion function of $\mathbf{f}$ is a function $\langle \mathbf{f} \rangle : \mathbb{IR}^n \rightarrow \mathbb{PR}^m$ such that

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subseteq \langle \mathbf{f} \rangle([\mathbf{x}]) \quad (2.51)$$

This definition is illustrated Figure 2.13.

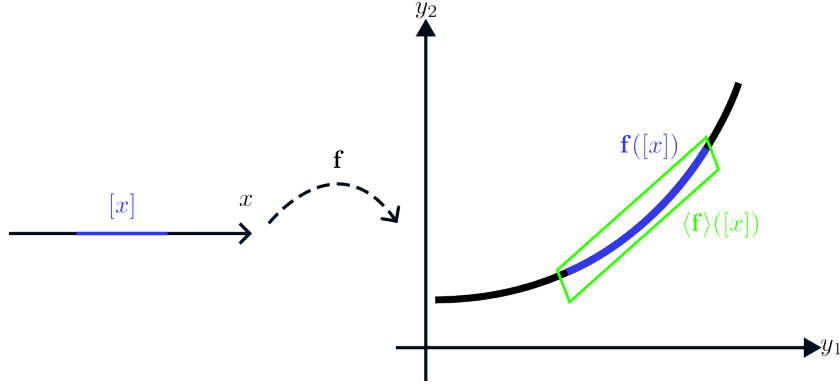


Figure 2.13 – Parallelepiped inclusion function

The classical definition of convergence order is not adapted to such inclusion functions. Indeed, the width of a parallelepiped (or even a zonotope) is not necessarily representative of its excess volume. We then define it in a volumetric way as follows.

Definition 34. Given $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ a C^1 function with $n < m$. The inclusion function $\langle \mathbf{f} \rangle$ converges with an order k if, for any tiny box $[\mathbf{x}]$, we have

$$\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])) = \varepsilon^m \varepsilon^{k(m-n)} \quad (2.52)$$

where we use the asymptotic Bachmann-Landau notation [79]: $\varepsilon = O(w([\mathbf{x}]))$ (see below). The *order of convergence* of $\langle \mathbf{f} \rangle$ is the largest k such that $\langle \mathbf{f} \rangle$ converges with an order k .

To complete this definition, consider the following points.

1. If $n = m$, Definition 34 does not apply. It is important to have $\text{vol}(\mathbf{f}([\mathbf{x}])) = 0$ (*i.e.*, $n < m$) to be able to define the order of convergence. This definition of the order of convergence becomes essential as soon as $m - n > 0$.
2. The notation with ε is used to shorten the mathematical expressions, but should be handled with care. When we write $\phi([\mathbf{x}]) = \varepsilon^i$, it supposes that $\phi([\mathbf{x}]) \in \mathbb{R}$. In addition, it means that if we take a nested sequence $[\mathbf{x}]_k$ converging to $\bar{\mathbf{x}} \in \mathbb{R}$, the ratio $\frac{\phi([\mathbf{x}]_k)}{w([\mathbf{x}]_k)^i}$ remains bounded.
3. If we consider $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ and if $[\mathbf{f}]$ is the natural inclusion function for $\mathbf{f}$, we can write $w([\mathbf{x}]) = \varepsilon$ and $\text{vol}([\mathbf{x}]) = \varepsilon^n$. From [93], we know that $\text{vol}([\mathbf{f}]([\mathbf{x}])) = \varepsilon^m$. Thus with this definition the natural inclusion function has an order 0.
4. A logarithmic view of Equation 2.52 is

$$\log(\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])) = (n + k(m - n)) \log(\varepsilon) \quad (2.53)$$

Thus, if we draw for tiny boxes $\log(\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])))$ as a function of $\log(\varepsilon)$ we should observe an asymptotic behavior when $\varepsilon \rightarrow 0$.

Parallelepiped contractor

A contractor for the parallelepiped can easily be constructed. Let us consider a parallelepiped $\langle \mathbf{x} \rangle \in \mathbb{PR}^n$, the associated constraint is

$$\begin{aligned} \mathcal{L}_{\langle \mathbf{x} \rangle} : \mathbb{R}^n &\rightarrow \{0, 1\} \\ \mathbf{x} &\mapsto \begin{cases} 1 & \text{if } \mathbf{x} \in \langle \mathbf{x} \rangle \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (2.54)$$

Algorithm 4 proposes an implementation of a contractor $\mathcal{C}_{\langle \mathbf{x} \rangle}$ associated to the parallelepiped $\langle \mathbf{x} \rangle = \mathbf{c} + \mathbf{A} \cdot [-1, 1]^n$.

Considering a box $[\mathbf{x}] \in \mathbb{IR}^n$, the idea is to first project the box in the parallelepiped referential. This projection noted $[\mathbf{x}_p]$ is then intersected with the n-dimensional box defining the parallelepiped $[-1, 1]^n$. Finally, the contracted box $[\mathbf{x}_c]$ is constructed by reprojecting $[\mathbf{x}_p]$ into the initial referential. To ensure the contractance property of the contractor as defined Definition 24, the contracted box is intersected with the initial one.

Algorithm 4 Parallelepiped contractor

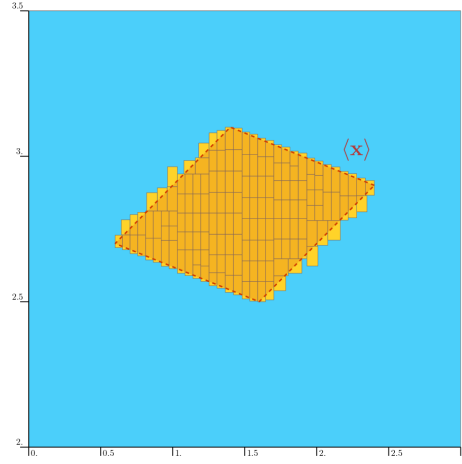
```

1: Input:  $[\mathbf{x}] \in \mathbb{IR}^n$ 
2: Output:  $[\mathbf{x}_c] \in \mathbb{IR}^n$ 
3:
4: function  $\mathcal{C}_{\langle \mathbf{x} \rangle}([\mathbf{x}])$ 
5:    $[\mathbf{x}_p] \leftarrow \mathbf{A}^{-1} \cdot ([\mathbf{x}] - \mathbf{c})$ 
6:    $[\mathbf{x}_p] \leftarrow [\mathbf{x}_p] \cap [-1, 1]^n$ 
7:    $[\mathbf{x}_c] \leftarrow \mathbf{c} + \mathbf{A} \cdot [\mathbf{x}_p]$ 
8:    $[\mathbf{x}_c] \leftarrow [\mathbf{x}_c] \cap [\mathbf{x}]$ 
9: end function
```

Example 17. Let us consider the 2-dimensional parallelepiped

$$\langle \mathbf{x} \rangle = \begin{pmatrix} 1.5 \\ 2.8 \end{pmatrix} + \begin{pmatrix} 0.5 & 0.4 \\ -0.1 & 0.2 \end{pmatrix} \cdot [-1, 1]^2 \quad (2.55)$$

Figure 2.14 shows the output of the paving Algorithm 3 with the initial box $[\mathbf{x}] = \begin{pmatrix} [0, 3] \\ [2, 3.5] \end{pmatrix}$ and $\epsilon = 0.1$. For reference, the parallelepiped $\langle \mathbf{x} \rangle$ is drawn in red.

Figure 2.14 – Paving of $\mathcal{C}_{\langle \mathbf{x} \rangle}$

We are indeed able to get an outer-approximation of the parallelepiped.

2.3.3 Limits of paving

As already visible in Figure 2.14, using a paver to handle non-axis aligned objects gives a quite pessimistic result. This is due to the wrapping effect described in Subection 2.2.4.

To illustrate this, consider that we have n parallelepipeds converging the unit circle. Thanks to the parallelepiped contractor of Section 2.3.2 and the use of contractor programming from Section 2.2.6, we are able to construct a contractor for the union of parallelepipeds.

Figure 2.15 shows the result of the paving Algorithm 3 with $n = 15$ parallelepipeds (in red), $[\mathbf{x}] = [-1.5, 1.5]^2$ and $\epsilon = 0.15$. By computing the difference between the initial box $[\mathbf{x}]$ and the union of the yellow boxes, we are able to get the blue boxes.

With a simple injectivity criterion, we can then sort the blue boxes between the boxes that are inside of the unit circle, and the ones that are outside. The result of this sorting is shown on Figure 2.16. As expected, we get an inner (green) and outer (green and yellow) approximations of the unit disk.

However, the paving step added some unwanted conservatism on the boundary, in yellow. Since the union of our n parallelepipeds forms a polygon, we can use a triangulation [13] on it. In particular, we will here perform a Delaunay triangulation as first introduced in [30].

The idea there is to consider the union of parallelepipeds as a polygon with a hole

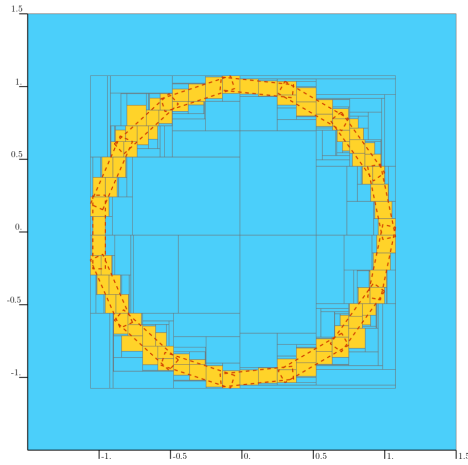
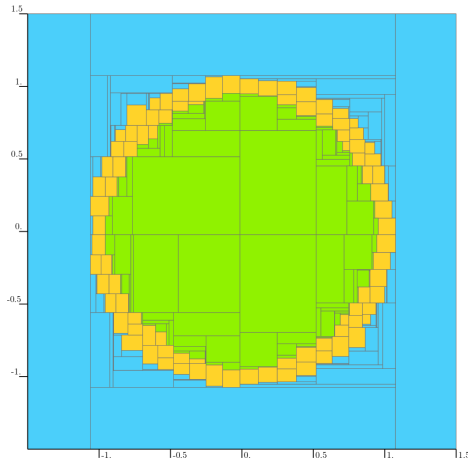
Figure 2.15 – Paving of $n = 15$ parallelepipeds

Figure 2.16 – Inner (green) and outer (green+yellow) approximations of the unit disk with a paving

in it. The Delaunay triangulation will then rely on the points of the hole to fill it with triangles. The same idea can be applied on the exterior of the polygon by also considering the initial box as a polygon with a hole. Figure 2.17 shows both polygons in red, with the hole in grey.

Figure 2.17a shows the first polygon used. In this case, the hole will be triangulated to get a set of triangles inside the unit disk.

Figure 2.17b shows the second polygon used. In this case, the polygon except the hole will be triangulated to get a set of triangles outside the unit disk.

The concatenation of both triangulations gives the result of Figure 2.18. On this Figure, the green triangulation is an inner approximation of the unit disk. If we add the

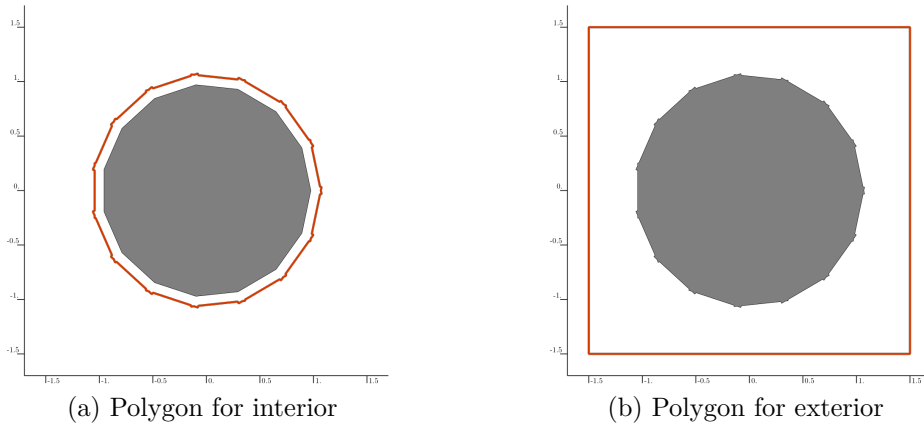


Figure 2.17 – Polygons with holes used for the Delaunay triangulate

parallelepipeds, in yellow, we get an outer approximation of it.

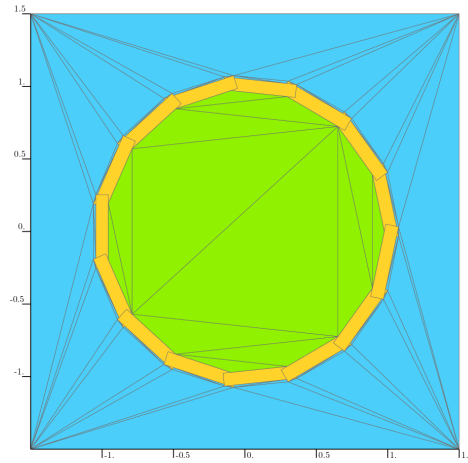


Figure 2.18 – Inner (green) and outer (green+yellow) approximation of the unit disk with a triangulation

We can observe that thanks to this approach we limited the pessimism in both the inner and the outer approximations. However, it has to be noted that if further computations are needed, such triangulation is harder to handle and propagate than a regular paving.

2.4 Tools and libraries

Numerous libraries have been developed to handle interval arithmetics. For example, MPFI [106] and flib++ [84] are some of the standard libraries developed for low-level in-

terval computations. The book [45] extensively describes and compares the performances of several of these libraries.

Modern set-based techniques now involve different set representations and constraint-based approaches, including polytopes, zonotopes and contractors. Higher-level libraries have therefore been developed to handle these new objects and provide dedicated tools. Depending on the programming language and the use case, the user can choose between several frameworks. For example, IBEX [19] is a standard C++ library for set-based approaches, used for constraint solving, while MATLAB users can rely on INTLAB [112] for validated computations. Readers interested in the use of interval or set-based libraries are invited to read Section 9.2 for discussions about their numerical guarantee.

In this work, we will rely on the version 2 of the CODAC library [109]. CODAC is a C++ library which is also available in both Python and MATLAB. It relies on GAOL [43] for its low-level interval computations. The name CODAC comes directly from the addition that the library proposes : Catalog Of Domains And Contractors. It is a tool dedicated to interval computations and constraint programming over real numbers, trajectories and sets. In particular, it provides standard inclusion functions, contractors and paving algorithms.

When needed, we will rely on the CGAL library [124] to perform the Delaunay triangulations mentioned in Section 2.3.3.

2.5 Conclusion

Interval arithmetic has shown to be an efficient and guaranteed tool to represent and propagate uncertainties. This ability to handle uncertain systems has shown to be efficient to tackle many classical problems in robotics.

The cornerstone of interval-based methods are the inclusion functions. Given an input interval or interval vector, inclusion functions provide an interval or interval vector which is guaranteed to enclose the true image of the input by the considered function. However, having a guaranteed enclosure means that it is pessimistic.

This pessimism can originate from multiple factors. First, the choice of an axis-aligned representation creates a wrapping effect. This wrapping is propagated when the output of the inclusion function is further used, making the result even more pessimistic after each step. Another source of pessimism is the inclusion function itself. A given function has an infinite number of inclusion functions with different properties. Notably, multi-

occurency in the expression of the inclusion function can degrade its performances. To be able to compare how well they perform, the convergence order indicates how fast inclusion functions converge to the true solution when the input gets smaller.

A first solution to limit this pessimism is to use a set of boxes to represent the desired sets instead of a single one. These sets are called pavings, and are produced by paving algorithms such as the SIVIA algorithm. Contractors and constraint programming can be used to improve the computational performances of the existing paving algorithms. They allow for both a faster and a thinner enclosure of the desired sets.

Another solution is to consider sets that are not axis-aligned boxes. Ellipsoids, polytopes and zonotopes are set representations that have gained increasing interest in recent set-based works. In particular, parallelepipeds offer a compromise between the simplicity of boxes and the low wrapping effect of zonotopes.

In a lot of cases, we can observe that computing the boundary of a set is enough to characterize it. Since this reduces the dimension of the problem, it is an efficient way to reduce the conservatism of the resulting enclosure. In addition, lighter computations are needed for robots to be able to perform the desired calculations on the fly. Notions of topology and differential geometry can then be combined to the classical set-based methods to improve their performances.

TOPOLOGY AND DIFFERENTIAL GEOMETRY

3.1 Introduction

Topology and differential geometry are two different yet strongly connected mathematical domains. While topology studies the general properties of geometrical objects (connectedness, holes...), differential geometry specifically studies smooth shapes and spaces.

Both domains find numerous applications in robotics. Topological notions often appear in robotics when a configuration space is considered. This is for example the case in motion planning [22, 35]. Topological tools such as the topological degree may also prove to be useful when estimating the coverage measure [28, 29] and the area explored by a sensor [127].

On the other hand, differential geometry can also be used when smooth manifolds arise from the expression of the problem. For example, Lie groups [2] are groups, but also differentiable (smooth) manifolds. Several works, see for example [120, 116, 101], have shown that even if they are abstract objects, Lie groups find applications in robotics. They can for example be used to handle robot dynamics or to perform a state estimation.

Since both topics have been extensively studied, this chapter aims to present the basic notions that will be used later on in this manuscript. Especially, we will focus on how we can define and handle the boundary of a set with topological tools. Under the presented assumptions, we will see that differential geometry may also be used on this boundary. In our case, the set that we want to characterize from its boundary is the reachable set of the robot. Especially, the tools presented here will be used to compute the boundary of the reachable set itself, and to properly deduce its interior.

3.2 Topology

Topology studies the geometrical properties of objects and how they evolve under continuous deformations. The compactness, connectedness and dimension are examples of topological properties that can be used to describe a geometrical object. In particular, topology defines the notion of boundary of a set (or an object), as well as the notion of interior. As already suggested in Section 2.3.3, both notions can be used in set-based methods.

3.2.1 Sets: Boundary and Compactness

Topological references often use the term *object* to describe what they study. In our context, we will prefer the term *set* as it will make the parallel between the topological notions presented here and the set-based methods from Chapter 2 easier. The content of this section will mainly rely on the book *General topology* by Stephen Willard ([129]) which broadly covers the subject of topology. Most topological definitions take place in a topological space. In our case we will limit our definitions to $\mathbb{R}^n$ which is a standard topological space.

A major classification of sets in topology is the distinction between open and closed sets.

Definition 35. A set $\mathbb{X} \subset \mathbb{R}^n$ is *open* if for each $\mathbf{x} \in \mathbb{X}$, there exists a ball centered on $\mathbf{x}$ with a radius $\epsilon > 0$ such that the whole disk is contained in $\mathbb{X}$. It is *closed* if its complementary $\mathbb{R}^n - \mathbb{X}$ is open.

Example 18. Consider a positive real $\sigma > 0$. The set

$$\left\{ \mathbf{x} \in \mathbb{R}^2 \mid \sqrt{x_1^2 + x_2^2} < \sigma \right\} \quad (3.1)$$

is an open set. The set

$$\left\{ \mathbf{x} \in \mathbb{R}^2 \mid \sqrt{x_1^2 + x_2^2} \leq \sigma \right\} \quad (3.2)$$

is a closed set.

Remark 13. Per construction, all the sets from Chapter 2 (intervals, boxes, polytopes...) are closed sets.

Remark 14. A set can be both closed and open, it is for example the case of $\mathbb{R}^n$.

This definition can be completed with the definitions of the closure and interior of a set. As their name suggests, these operations turn an open set into a closed set and vice-versa.

Definition 36. Consider a set $\mathbb{X} \subset \mathbb{R}^n$, the *closure* of $\mathbb{X}$ in $\mathbb{R}^n$ is the intersection of all closed supersets of $\mathbb{X}$. It is defined as

$$\text{Cl}(\mathbb{X}) = \bigcap \{ \mathbb{K} \subset \mathbb{R}^n \mid \mathbb{K} \text{ is closed and } \mathbb{X} \subset \mathbb{K} \} \quad (3.3)$$

Definition 37. Consider a set $\mathbb{X} \subset \mathbb{R}^n$, the *interior* of $\mathbb{X}$ in $\mathbb{R}^n$ is the union of all open subsets of $\mathbb{X}$. It is defined as

$$\text{Int}(\mathbb{X}) = \bigcup \{ \mathbb{G} \subset \mathbb{R}^n \mid \mathbb{G} \text{ is open and } \mathbb{G} \subset \mathbb{X} \} \quad (3.4)$$

Remark 15. The closure of a closed set is itself, and the interior of an open set is also itself.

Thanks to both notions, we are able to formally define the topological boundary of a set, later only referred to as boundary.

Definition 38. The *boundary*, or *frontier*, of a set $\mathbb{X} \subset \mathbb{R}^n$ is denoted $\partial\mathbb{X}$ and can be defined as

$$\begin{aligned} \partial\mathbb{X} &= \text{Cl}(\mathbb{X}) \cap \text{Cl}(\mathbb{R}^n - \mathbb{X}) \\ &= \text{Cl}(\mathbb{X}) - \text{Int}(\mathbb{X}) \end{aligned} \quad (3.5)$$

In this manuscript, the considered sets will always be compact. A set is said to be *compact* if it is both closed and bounded.

Remark 16. It immediatly falls from this definition that bounded intervals, boxes and zonotopes are compacts.

Instead of representing a set as a single block, it can be subdivided with a cover.

Definition 39. A *cover* (or *covering*) of a set $\mathbb{X}$ is a collection of subsets of $\mathbb{X}$ whose union is $\mathbb{X}$. A collection of sets $\{\mathbb{U}_\alpha\}_{\alpha \in A}$ is then a cover of $\mathbb{X}$ if

$$\begin{cases} \forall \alpha \in A, \mathbb{U}_\alpha \subseteq \mathbb{X} \\ \bigcup_{\alpha \in A} \mathbb{U}_\alpha = \mathbb{X} \end{cases} \quad (3.6)$$

If the $\mathbb{U}_\alpha$ are open, the cover is referred to as an *open cover*. In the same way, a *closed cover* is a collection of closed sets.

Such subdivision will be useful later on to limit the pessimism of the used set-based methods. This is closely related to the way paving algorithms from Chapter 2 work.

The considered sets may need to be projected. Only orthogonal projections will be considered in this manuscript.

Definition 40. Let us consider $p \in \llbracket 1, n \rrbracket$ and a point $\mathbf{x} \in \mathbb{R}^n$. We denote $\Pi_p : \mathbb{R}^n \rightarrow \mathbb{R}^p$ the orthogonal projection of $\mathbf{x}$ on its p first dimensions, i.e.

$$\Pi_p(\mathbf{x}) = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} \quad (3.7)$$

For any unit-length vector $\mathbf{v} \in \mathbb{R}^n$, we also define the projection operator onto the hyperplane orthogonal to $\mathbf{v}$ as in [85]

$$\begin{aligned} \text{Proj}_{\mathbf{v}} : \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ \mathbf{u} &\mapsto \mathbf{u} - (\mathbf{v}^\top \mathbf{u}) \mathbf{v} \end{aligned} \quad (3.8)$$

where the product is the Euclidian inner product.

Both projections are depicted in Figure 3.1. In this Figure, the vector $\mathbf{e}_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$ is considered for the projection.

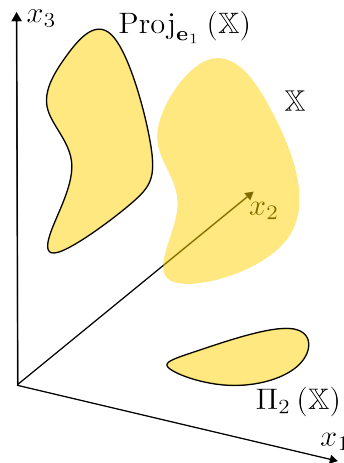


Figure 3.1 – Orthogonal projections of a set

3.2.2 Manifolds: Atlases and Smoothness

In addition to classical sets, manifolds are topological objects that can be used to describe complex objects in a simpler way.

Definition 41. [54] A n -manifold is a topological space which is locally homeomorphic to an open subset of $\mathbb{R}^n$. This means that if we consider a small portion of the manifold, it can be seen as an open subset of dimension n .

Example 19. The unit disk D^2 is a two-dimensional object. At any point of it, this disk is homeomorphic to an open subset of $\mathbb{R}^2$. The unit disk is then a 2-manifold. Its boundary, the unit circle S^1 , is a 1-manifold embedded in a two-dimensional space. If a small portion of the circle is considered, it can be seen as a straight segment.

Since they are locally homeomorphic to an open subset of lower dimension, manifolds can be studied in this dimension instead. Atlases are used to do so.

Definition 42. [126] Let $\mathbb{M}$ be a n -manifold. An *atlas* of $\mathbb{M}$ is a collection $\{(U_i, \varphi_i)\}_{i \in I}$ of charts such that the U_i are open sets and $\mathbb{M} = \bigcup_{i \in I} U_i$, and the $\varphi_i : U_i \rightarrow \varphi_i(U_i)$ are homeomorphisms onto open subsets of $\mathbb{R}^n$.

Example 20. Let us consider the unit circle S^1 . As depicted Figure 3.2, we define two open sets $U_1 = S^1 \setminus \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $U_2 = S^1 \setminus \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ and the two homeomorphisms

$$\begin{aligned} \varphi_1 : U_1 &\rightarrow]-1, 1[\\ \mathbf{x} &\mapsto \frac{x_1}{1-x_2} \end{aligned} \quad (3.9)$$

$$\begin{aligned} \varphi_2 : U_2 &\rightarrow]-1, 1[\\ \mathbf{x} &\mapsto \frac{x_1}{1+x_2} \end{aligned} \quad (3.10)$$

The collection $\{(U_i, \varphi_i)\}_{i \in \{1,2\}}$ is an atlas of the unit circle.

When there is an overlap between the open sets of an atlas, we can define a transition between them.

Definition 43. Let $\mathbb{M}$ be a manifold with an atlas $\{(U_i, \varphi_i)\}_{i \in I}$. If two charts i and j overlap, i.e. if $U_i \cap U_j \neq \emptyset$, we define the *transition map* from φ_i to φ_j as in [82]

$$\begin{aligned} \varphi_j \circ \varphi_i^{-1} : \varphi_i(U_i \cap U_j) &\rightarrow \varphi_j(U_i \cap U_j) \\ \mathbf{x} &\mapsto \varphi_j(\varphi_i^{-1}(\mathbf{x})) \end{aligned} \quad (3.11)$$

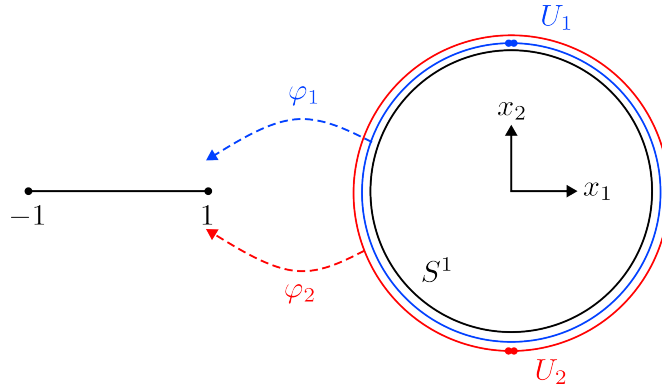


Figure 3.2 – Atlas of the unit circle made with 2 charts

These transition maps are used to define the smoothness of a manifold, which is the basis of differential geometry.

Definition 44. Let $\mathbb{M}$ be a manifold. If there exists an atlas for $\mathbb{M}$ such that all its transition maps are smooth (or C^∞), the manifold $\mathbb{M}$ itself is *smooth*.

Example 21. Consider the atlas of the unit circle defined in Example 20. We can define the inverses of φ_1 and φ_2 as

$$\begin{aligned} \varphi_1^{-1} :]-1, 1[&\rightarrow U_1 \\ t &\mapsto \begin{pmatrix} \frac{2t}{1+t^2} \\ \frac{t^2-1}{1+t^2} \end{pmatrix} \end{aligned} \quad (3.12)$$

$$\begin{aligned} \varphi_2^{-1} :]-1, 1[&\rightarrow U_2 \\ t &\mapsto \begin{pmatrix} \frac{2t}{1+t^2} \\ \frac{1-t^2}{1+t^2} \end{pmatrix} \end{aligned} \quad (3.13)$$

We can easily verify that

$$\forall t \in]-1, 1[, \varphi_2 \left(\varphi_1^{-1}(t) \right) = \varphi_1 \left(\varphi_2^{-1}(t) \right) = t \quad (3.14)$$

Which is smooth, so the unit circle is a smooth manifold.

Compact sets and smooth manifolds are closely related. If a set $\mathbb{X} \subset \mathbb{R}^n$ can be defined by a C^1 value function $v : \mathbb{R}^n \rightarrow \mathbb{R}$ as its zero-sublevel, i.e.

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid v(\mathbf{x}) \leq 0\} \quad (3.15)$$

the boundary of the set is simply $\partial\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid v(\mathbf{x}) = 0\}$. We can then apply the regular level set theorem on this boundary. As stated in [126], if all the points of the boundary are regular, i.e.

$$\forall \mathbf{x} \in \partial\mathbb{X}, \frac{dv}{d\mathbf{x}}(\mathbf{x}) \neq 0 \quad (3.16)$$

then $\partial\mathbb{X}$ is a smooth $(n - 1)$ -manifold.

Recent works have extended atlases to closed sets. In particular [57] defines the box atlases as follows

Definition 45. Let $\mathbb{M}$ be a compact manifold, a *box atlas* of $\mathbb{M}$ is a collection $\{(U_i, \varphi_i)\}_{i \in I}$, with I finite, of charts $V_i = \varphi_i(U_i)$ such that

- U_i are closed subsets of $\mathbb{M}$,
- $\forall i, j \in I, (i \neq j) : \dim(U_i \cap U_j) < \dim(U_i)$,
- $\mathbb{M} = \bigcup_{i \in I} U_i$,
- the charts V_i are boxes,
- $\varphi_i : U_i \rightarrow \varphi_i(U_i)$ are homeomorphisms.

Since the V_i are boxes, interval analysis tools can be applied on them. Such atlas can be used to cover the torus, the sphere...

3.3 Differential geometry

Differential geometry studies the properties of smooth manifolds. For this section, we will consider a smooth manifold $\mathbb{M}$ embedded in $\mathbb{R}^n$, $n \geq 1$.

The *tangent space* of $\mathbb{M}$ at $\mathbf{x} \in \mathbb{M}$ is denoted $T_{\mathbf{x}}\mathbb{M}$. See for example *An Introduction to Manifolds* from Loring W. Tu ([126]) for a formal definition. An example in the case where $\mathbb{M}$ is embedded in $\mathbb{R}^3$ is depicted Figure 3.3.

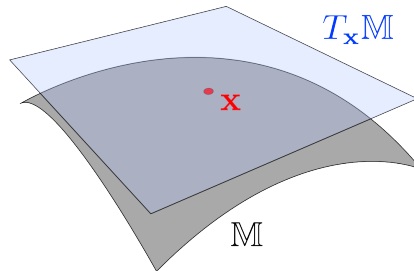


Figure 3.3 – Tangent space

An interesting property of differential geometry is the pushforward on tangent spaces.

Definition 46. Consider a smooth map $\mathbf{f} : \mathbb{N} \rightarrow \mathbb{M}$ between two smooth manifolds and any $\mathbf{x} \in \mathbb{N}$. The *pushforward* states that the differential of $\mathbf{f}$ is a function $d\mathbf{f} : T_{\mathbf{x}}\mathbb{N} \rightarrow T_{\mathbf{f}(\mathbf{x})}\mathbb{M}$. If we consider a vector $\mathbf{v} \in T_{\mathbf{x}}\mathbb{N}$ tangential to $\mathbb{N}$ at $\mathbf{x}$, $d\mathbf{f}(\mathbf{v}) \in T_{\mathbf{f}(\mathbf{x})}\mathbb{M}$ is then tangential to $\mathbb{M}$ at $\mathbf{f}(\mathbf{x})$.

We can also define the normal space from the tangent space.

Definition 47. The *normal space* of $\mathbb{M}$ at $\mathbf{x} \in \mathbb{M}$ is denoted $N_{\mathbf{x}}\mathbb{M}$ and is defined as

$$N_{\mathbf{x}}\mathbb{M} = \left\{ \mathbf{w} \in \mathbb{R}^n \mid \mathbf{w}^T \cdot \mathbf{v} = 0 \text{ for all } \mathbf{v} \in T_{\mathbf{x}}\mathbb{M} \right\} \quad (3.17)$$

If in addition the manifold is orientable (see [126]), we can define the Gauss map of the manifold.

Definition 48. Let $\mathbb{M} \subset \mathbb{R}^n$ be a smooth oriented $n - 1$ manifold. The *Gauss map* maps each point of $\mathbf{x} \in \mathbb{M}$ to the unique unit vector, normal to $\mathbb{M}$ at $\mathbf{x}$ compatible with the chosen orientation. Formally,

$$\mathbf{n}^{\mathbb{M}} : \mathbb{M} \rightarrow \mathbb{S}^{n-1} \\ \mathbf{x} \mapsto \mathbf{y}, \begin{cases} \mathbf{y} \in N_{\mathbf{x}}\mathbb{M} \\ \|\mathbf{y}\| = 1 \\ \det(\mathbf{v}_1, \dots, \mathbf{v}_{n-1}, \mathbf{y}) > 0 \end{cases}, \quad (3.18)$$

Where $\det$ is the determinant operator, applied on the matrix whose columns are the vectors $\mathbf{v}_1, \dots, \mathbf{v}_{n-1}, \mathbf{y}$. $(\mathbf{v}_1, \dots, \mathbf{v}_{n-1})$ is a positively oriented basis of $T_{\mathbf{x}}\mathbb{M}$. This is depicted Figure 3.4.

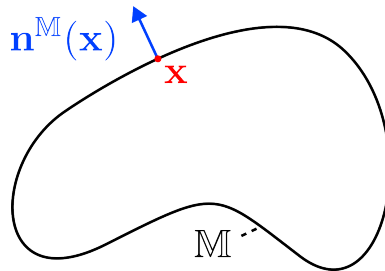


Figure 3.4 – Gauss map

Remark 17. In the case where the considered smooth manifold is defined as the boundary of a compact set, it is orientable and the Gauss map can be defined.

3.4 Conclusion

Topology describes and classifies sets depending on their properties. From the definitions of this section, we can say that boxes and zonotopes are closed and compact sets if bounded.

When possible, the boundary of a set can be interpreted as a manifold. In the case where the manifold is smooth, differential geometry applies to it. Tangent and normal spaces can thus be defined. In particular, the Gauss map can be applied on the boundary of a set to define its orthogonal and outward pointing vectors. This map is then closely related to the topological definition of the interior of a set. This will be used later on to deduced the interior of the reachable set from its boundary.

Another major interest of a manifold is that it can be studied in a space of lower dimension. Atlases are used to do so, but their classical formulation only applies to open sets. Recent works have extended the notion of atlas to boxes, which are closed sets. This definition of box atlases hints that topology can be used to make set-based methods less pessimistic. Since the charts of such atlases are boxes, standard set-based methods can be directly applied to them. As the use of an atlas reduces the dimension of the problem, the conservatism naturally decreases.

In the context of this manuscript, we will see how the concepts introduced in this section can be used in Reachability Analysis to handle uncertainties at a low computational cost. Especially, the notions of differential geometry will be used to compute the boundary of the reachable set. Tools of topology will then be used to get both an inner and an outer approximation of this reachable set from its boundary. Before that, let us define some notions about dynamical systems and reachability analysis.

REACHABILITY OF DYNAMICAL SYSTEMS

4.1 Introduction

Control theory is a major field of research in robotics. The objective of this domain is to characterize how a dynamical system behaves, and to define control laws that are suited for it to fulfill its mission. In this chapter only continuous dynamical systems will be considered.

A classical problem in control theory is the Forward Reachability Analysis. The objective of such analysis is to compute the set of states that a system can reach at a given time or over a time horizon. One can divide Reachability Analysis into two main categories.

- *Infinite-time reachability analysis* : The objective is to characterize the states that may be reached by the system over an unbounded time horizon. This may involve analyzing limit cycles or basins of attractions [48, 123].
- *Finite-time reachability analysis* : The objective is to compute the set of states reachable by the system at a given and bounded time (or range of time) [58, 11]. This is the case in this manuscript.

The objective of this chapter is to properly define the reachable set of a dynamical system, and to give some insights on the existing methods to compute it. To do so we first introduce some formalism of dynamical systems. We then present some existing methods for Reachability Analysis along with their working hypothesis and scope. Building upon one of the presented methods, the notions from Chapters 2 and 3 will be used later on in Chapter 6 for our method of reachable set computation.

4.2 Formalism of dynamical systems

4.2.1 Definition

For this section and for the remainder of the manuscript, the state of the dynamical system will be denoted $\mathbf{x}$.

The behavior of such system can be described by an Ordinary Differential Equation (ODE) of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \quad (4.1)$$

In robotics, the systems are often subject to inputs and to disturbances. For the dynamical system both the input and the disturbances modify its natural behavior. We can then group both in a single vector $\mathbf{w}$, later referred to as the disturbance vector. The ODE from Equation 4.1 can then be rewritten as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t)) \quad (4.2)$$

Remark 18. The special case where $\forall t \geq 0, \mathbf{w}(t) = 0$ is often referred to as the zero-input or free dynamics of the system.

Example 22. The pendulum is a two dimensional dynamical system. As depicted on Figure 4.1, the state of the pendulum can be fully described by its angular position x_1 and its angular velocity x_2 . It follows the ODE

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{g}{L} \sin(x_1) \end{pmatrix} \quad (4.3)$$

Where L and g are two constants, respectively the length of the pendulum and the gravity.

If the pendulum is disturbed by a motor generating a torque, it produces an angular acceleration w and the ODE becomes

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{g}{L} \sin(x_1) + w \end{pmatrix} \quad (4.4)$$

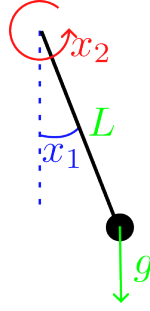


Figure 4.1 – Pendulum

4.2.2 Classical problems

When a dynamical system is studied, we often want to compute its state through time starting from an initial condition.

Definition 49. An *Initial Value Problem*, or IVP, is the combination of an ODE and an initial condition. It can be written as

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (4.5)$$

We can also defined the *disturbed IVP* from the disturbed ODE of Equation 4.2 as

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (4.6)$$

This formalism only holds as long as the behavior of the system can be precisely described, i.e. we have an analytic expression for $\mathbf{f}$. If we do not have such expression, or if it is by nature uncertain, we need another formalism.

Definition 50. *Differential inclusions* are an extension of ODEs in the case where the dynamic of the system is not precisely known. The dynamic can be bounded by an inclusion function $\mathbf{F} : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathcal{P}(\mathbb{R}^n)$ such that

$$\forall t \in \mathbb{R}, \dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), t) \quad (4.7)$$

Let us consider the disturbed ODE from Equation 4.2. If the disturbances are constrained to lie in a set, i.e. we have

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t)) \\ \forall t \in \mathbb{R}^+, \mathbf{w}(t) \in \mathbb{W} \end{cases} \quad (4.8)$$

where $\mathbb{W}$ is the disturbance set. Then the evolution of the system is described by a differential inclusion as defined in Equation 6.1 with

$$\mathbf{F}(\mathbf{x}(t), t) = \mathbf{f}(\mathbf{x}(t), \mathbb{W}) \quad (4.9)$$

In other words, at any time we consider the whole disturbance set instead of a single value.

We can then define an Initial Value Problem for a differential inclusion.

Definition 51. An *IVP for a differential inclusion* is the combination of a differential inclusion and an initial condition. It can be written as

$$\begin{cases} \dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), t) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (4.10)$$

An uncertainty can even be considered on the initial condition. In this case we say that we have a set-valued initial condition.

Definition 52. An *IVP for a differential inclusion with a set-valued initial condition*, is the combination of a differential inclusion and an initial set. It can be written as

$$\begin{cases} \dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), t) \\ \mathbf{x}(0) \in \mathbb{X}_0 \end{cases} \quad (4.11)$$

If we consider a disturbed ODE with a set-valued initial condition and bounded disturbances, i.e.

$$\begin{cases} \dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), \mathbb{W}) \\ \mathbf{x}(0) \in \mathbb{X}_0 \end{cases} \quad (4.12)$$

we get an IVP for a differential inclusion with a set-valued initial condition. This is the IVP that we will be focusing on in this manuscript.

Example 23. The Dubins car is a simple model of a wheeled robot. It models a car moving forward at constant speed, with the possibility to steer it. The state of the car is $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \theta \end{pmatrix}$ where (x_1, x_2) is the position of the car and θ is its heading as shown Figure 4.2.

The dynamic of the car can be described by the disturbed ODE

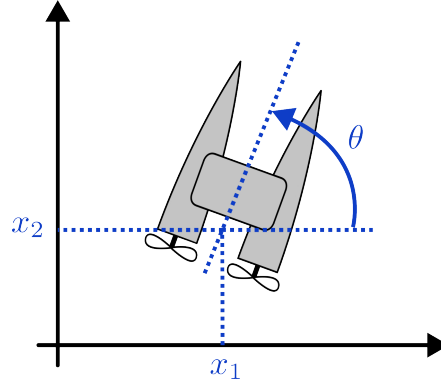


Figure 4.2 – Dubins car

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), w(t)) = \begin{pmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \\ w(t) \end{pmatrix} \quad (4.13)$$

This car can either go straight ($w = 0$), turn left ($w = -1$) or turn right ($w = 1$). If the steering speed is bounded between -1 and 1 , the dynamic of the car can then be described by the differential inclusion

$$\dot{\mathbf{x}}(t) \in \mathbf{F}(\mathbf{x}(t), w(t)) = \begin{pmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \\ [-1, 1] \end{pmatrix} \quad (4.14)$$

If the initial state of the car is known to lie in a set $\mathbb{X}_0$, we get an IVP for a differential inclusion with a set-valued initial condition. This could for example be a case where we know the initial heading, but we are only able to get the initial position of the system with a bounded error, say ρ . The initial set $\mathbb{X}_0$ would then be a disk of radius ρ .

4.2.3 The flow function

When an ODE is used to describe the behavior of a dynamical system, some hypothesis can be done on the evolution function $\mathbf{f}$.

For this manuscript $\mathbf{f}$ will be at least C^1 . It is then continuous, differentiable and its derivative is continuous. In addition, both $\mathbf{f}$ and its Jacobian $\nabla \mathbf{f}$ are Lipschitz.

Definition 53. [119] Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. $\mathbf{f}$ is said to be *globally Lipschitz*, or *Lipschitz continuous*, if there exists a constant $A > 0$ such that

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, \|\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)\| \leq A \cdot \|\mathbf{x}_1 - \mathbf{x}_2\| \quad (4.15)$$

where $\|\cdot\|$ denotes the Euclidian norm. This constant A is referred to as a *Lipschitz constant* of $\mathbf{f}$. Note that the smallest constant A satisfying this inequality is sometimes referred to as the *best Lipschitz constant* of $\mathbf{f}$ [12].

This definition can be extended to matricial functions by adapting the choosen norm. In our case we will in addition consider that $\mathbf{f}$ has a time-independant Lipschitz constant. Under these hypothesis, we can define the flow function of the IVP as done in [55].

Definition 54. Let us consider an IVP as defined in Equation 4.5. Assume $\mathbf{f}$ is continuous in t and C^1 . The Picard-Lindelöf theorem then states that the IVP has a solution and that it is unique. This function is called the *flow function* and satisfies

$$\begin{aligned} \phi : \mathbb{R}^+ \times \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ (t, \mathbf{x}_0) &\mapsto \mathbf{x}(t) \end{aligned} \quad (4.16)$$

For lighter notations, we will further write $\phi_t : \mathbb{R}^n \rightarrow \mathbb{R}^n$ the flow function given the state at time t .

Example 24. Consider $\lambda \in \mathbb{R}$ and define the IVP

$$\begin{cases} \dot{\mathbf{x}}(t) = \lambda \cdot \mathbf{x} \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (4.17)$$

For any initial state $\mathbf{x}_0 \in \mathbb{R}^n$, the flow function is

$$\begin{aligned} \phi : \mathbb{R}^+ \times \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ (t, \mathbf{x}_0) &\mapsto e^{\lambda \cdot t} \cdot \mathbf{x}_0 \end{aligned} \quad (4.18)$$

Now if we constraint both the disturbances and the initial state to lie in a set, we need to redefine the flow function.

Definition 55. Let us consider an IVP for a differential inclusion as defined in Equation 4.12. Assume that $\mathbf{f}$ is continuous in t and Lipschitz continuous in $\mathbf{x}$ with a time-independant Lipschitz constant. Note that the disturbances are bounded. Under these assumptions, for any disturbance $\mathbf{w} \in L^\infty(\mathbb{R}^+, \mathbb{W})$ the Picard-Lindelöf theorem states that the resulting ODE has a unique solution for any initial condition $\mathbf{x}(0) = \mathbf{x}_0$ called

the *flow function* and noted ϕ such that

$$\begin{cases} \phi : \mathbb{R}^+ \times \mathbb{X}_0 \times L^\infty(\mathbb{R}^+, \mathbb{W}) & \rightarrow \mathbb{R}^n \\ (t, \mathbf{x}_0, \mathbf{w}) & \mapsto \mathbf{x}(t) \end{cases} \quad (4.19)$$

This definition is the key to define the reachable sets.

4.3 Finite-time reachability analysis

As mentionned earlier, this manuscript will focus on finite time reachability analysis. The objective of such analysis is to compute the reachable state of a dynamical system after a given duration. Formally, if we consider the IVP of Equation 4.12, we can define the reachable set as follows

Definition 56. Let us consider an IVP as defined in Equation 4.12 such that the flow function is well defined (see Definition 55). The reachable set $\mathcal{R}_t$ at a time $t > 0$ is the set of all the states that the dynamical system can reach in a duration t starting from $\mathbb{X}_0$. It can be defined by:

$$\mathcal{R}_t = \{\phi(t, \mathbf{x}_0, \mathbf{w}) \mid \mathbf{x}_0 \in \mathbb{X}_0, \mathbf{w} \in L^\infty([0, T], \mathbb{W})\} \quad (4.20)$$

Figure 4.3 shows a two dimensional example of this definition.

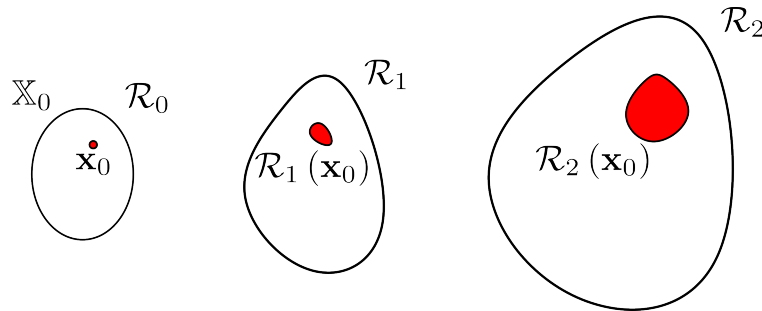


Figure 4.3 – Reachable set through time

Further on in the manuscript, we may write $\mathcal{R}_t(\mathbb{X})$ or $\mathcal{R}_t(\mathbf{x})$ to specify that we consider the reachable set starting from the set $\mathbb{X}$ or the point $\mathbf{x}$. If nothing is specified, i.e. $\mathcal{R}_t$, the initial set is simply $\mathbb{X}_0$. In the following sections, we display some existing methods that can be used to compute such reachable set.

4.3.1 Reachable set from an initial state

A first step to characterize the reachable set from an initial set is to start with an initial point. Such problem can be viewed as an optimal control problem, and associate tools can be used.

Starting from an initial state, and with bounded disturbances, computing the reachable set comes down to finding the disturbances responsible for the maximal variations of the final state. If we find these disturbances, we are able to compute these maximal variations and get an outer approximation of the reachable set.

To do so, the Pontryagin Maximal Principle [102] can be used to perform the reachability analysis. This is the core idea presented in [85, 114].

Let us focus on the work from [85] since it influenced a lot the content of this manuscript. The ODE considered in this paper has the shape $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{w}(t)$. The core idea is to introduce a search direction for the Pontryagin Maximal Principle. The objective is then to find the disturbance trajectory which makes the system evolve as much as possible in this search direction at a given time t . If we consider all directions, we get the convex hull of the reachable set at time t , $\mathcal{R}_t$.

This principle is shown Figure 4.4. On the left we show in black the initial state $\mathbf{x}_0$ and the true reachable set at time t , $\mathcal{R}_t$. On the right we consider three search directions. For each, we show the point of the reachable set “as far as possible” in this search direction. If we do this for all search directions and compute the hull, we get the convex hull of the reachable set noted $H(\mathcal{R}_t)$ in orange on the figure.

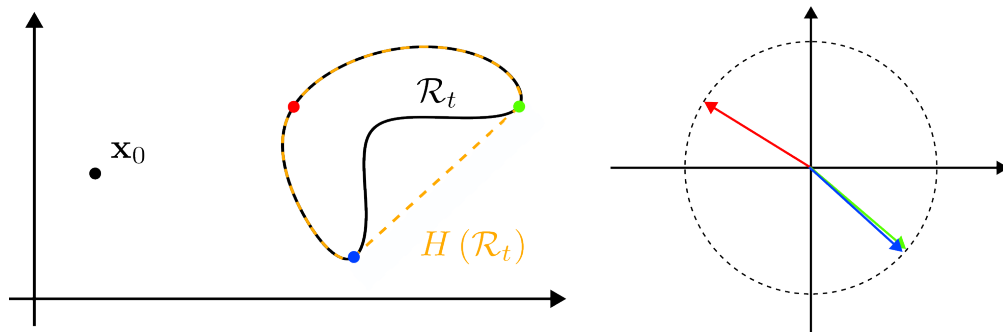


Figure 4.4 – Convex hull of the reachable set

Under the assumptions of this paper, it is shown that these extremal disturbances necessarily start on the boundary of the disturbance set. More specifically, if $\mathbf{q}$ is the desired direction of disturbance, the worse disturbance in the direction of $\mathbf{q}$ is given by

the inverse Gauss map as depicted Figure 4.5. Note that the boundary of $\mathbb{W}$ needs to be an ovaloid for its Gauss map to be invertible.

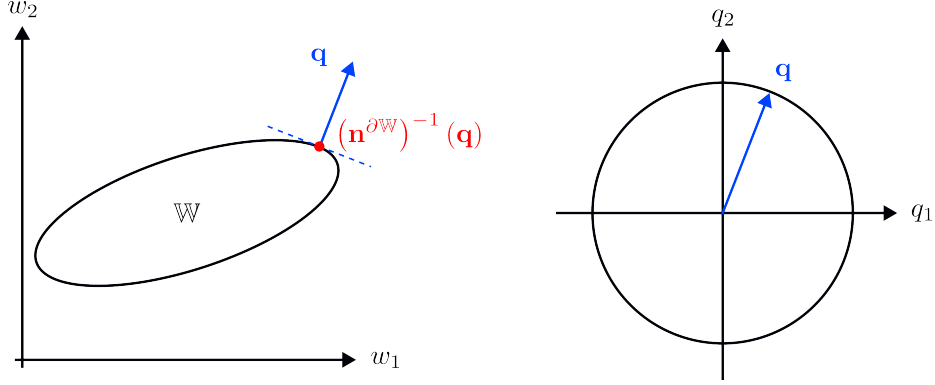


Figure 4.5 – Optimal disturbance in the direction of $\mathbf{q}$

The computation of the reachable set can be then done by integrating the *augmented ODE*

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + (\mathbf{n}^{\partial \mathbb{W}})^{-1}(\mathbf{q}(t)) \\ \dot{\mathbf{q}} = -\text{Proj}_{\mathbf{q}}\left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}(t))^\top \mathbf{q}(t)\right) \\ (\mathbf{x}(0), \mathbf{q}(0)) = \mathbf{x}_0, \mathbf{n}^{\partial \mathbb{W}}(\mathbf{w}(0)) \end{cases} \quad (4.21)$$

for every initial extremal disturbance $\mathbf{w}(0) \in \partial \mathbb{W}$. In practice, the paper proposes to select a certain number of initial disturbances on $\partial \mathbb{W}$ to compute an under approximation of the convex hull. This under approximation is then inflated to account for the fact that the whole boundary $\partial \mathbb{W}$ was not considered but only certain points of it. The result is an over approximation of the convex hull of the reachable set.

Set-based methods from Chapter 2 and the atlases presented in Chapter 3 will be later used in Chapter 6 to avoid this discretisation step.

Example 25. Consider the equation of the pendulum from Equation 4.3. Let $\mathbb{W}$ be a disk of radius r . Its Gauss map is then simply :

$$\begin{aligned} \mathbf{n}^{\partial \mathbb{W}} : \partial \mathbb{W} &\rightarrow S^1 \\ \mathbf{w} &\mapsto \frac{1}{r} \cdot \mathbf{w} \end{aligned} \quad (4.22)$$

It is invertible, and its inverse is :

$$\begin{aligned} (\mathbf{n}^{\partial\mathbb{W}})^{-1} : S^1 &\rightarrow \partial\mathbb{W} \\ \mathbf{q} &\mapsto r \cdot \mathbf{q} \end{aligned} \quad (4.23)$$

The Jacobian matrix of $\mathbf{f}$ is :

$$\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}) = \begin{pmatrix} 0 & 1 \\ -\frac{g}{L} \cos(x_1) & 0 \end{pmatrix} \quad (4.24)$$

Therefore, the augmented ODE to integrate is :

$$\begin{cases} \dot{\mathbf{x}} = \begin{pmatrix} x_2 \\ -\frac{g}{L} \sin(x_1) \end{pmatrix} + r \cdot \mathbf{q} \\ \dot{\mathbf{q}} = \begin{pmatrix} \frac{g}{L} \cos(x_1) q_2 + q_1^2 q_2 \left(1 - \frac{g}{L} \cos(x_1)\right) \\ -q_1 + q_1 q_2^2 \left(1 - \frac{g}{L} \cos(x_1)\right) \end{pmatrix} \end{cases} \quad (4.25)$$

4.3.2 Reachable set from an initial set

The reachable set from a set $\mathbb{X}$ can be defined as

$$\mathcal{R}_t(\mathbb{X}) = \bigcup_{\mathbf{x} \in \mathbb{X}} \mathcal{R}_t(\mathbf{x}) \quad (4.26)$$

If we are able to compute the reachable set from an initial state as mentionned in Section 4.3.1, we can then compute the reachable set from an initial set easily.

A way to compute such reachable set is to select several initial states in $\mathbb{X}$, compute the reachable set from these states, and finally reconstruct $\mathcal{R}_t(\mathbb{X})$ from this. This is the idea behind simulation-based reachability analysis [33, 10].

Another way to do it is to use the set-based methods presented in Chapter 2. The problem of forward reachability analysis can be seen as the propagation of a set through a differential inclusion. The paper [6] offers an overview of the existing set-propagations techniques for reachability analysis. It covers the use of polytopes [9], zonotopes [7, 39, 115] (with their variant polynomial zonotopes [70]), ellipsoids [76] and support functions [81] to represent and propagate the initial set.

Level set methods [117, 90] can also be used for reachability analysis. As suggested in Chapter 3, level sets and boundary methods are closely related. Under sufficient hypothesis, propagating the boundary of the initial set is sufficient to characterize the reachable set.

When possible, the use of a boundary based methods is preferred. As the considered set is *smaller*, the pessimism in the computations is limited. In addition, the use of topological tools such as atlases can make these computations even lighter since they reduce the dimensions of the problem. This motivated our choice of a boundary based approach to compute the reachable set starting from an initial point. This will be presented in Part II of this manuscript.

4.4 Tools

As presented through this section, the computation of a reachable set often requires to compute the solution of an IVP. Standard scheme (Euler, Runge-Kutta) can be used to do so, but the result provided is not guaranteed. While working with set based methods, it is important that each step is guaranteed for the final result itself to be guaranteed.

Extensive work has been performed to develop guaranteed integration scheme, often with the use of interval or other set representations. Historically, Taylor series [92, 98, 87] have been used to do so. An important property of these Taylor series is that it is easy to get their bounds, and so to construct a validated integrator with them.

More recently, validated Runge-Kutta methods [37, 32] have found an increasing interest as a counterpart to Taylor series. One of their interest is that they possess additional stability properties.

With the development of such methods, some tools have been developed to make these validated integration methods available. For example, the work from [32] is available in the DynIbex library. For this manuscript, we will rely on the CAPD library, more precisely on CAPD::DynSys [62], to get solutions of our IVPs when needed.

It has to be noted that library have also been developed to directly perform the reachability analysis of a dynamical system. This is the case of the JuliaReach [16] and CORA [4] libraries.

4.5 Conclusion

In a context of control theory, we introduced the notion of reachability analysis. After defining it, we showed a strong connection between the computation of a reachable set and the use of set-based methods. This is highlighted by the numerous works on the topic which rely on intervals, polytopes or zonotopes.

Indeed, set-based methods allows to handle uncertainties in both the initial state and the dynamic of the system. Additionally, we can see that in some context a boundary approach is sufficient to fully characterize the reachable set. Such approaches allow for less conservative results at a lower computational cost. The notions of topology and differential geometry from Chapter 3 will then be useful to properly define the reachable set from its boundary. The contribution of this manuscript lies in the use of parallelepipeds to cover the boundary of the reachable set, and the different sub-contributions are detailed in the following Part.

PART II

Contributions

INNER AND OUTER APPROXIMATION OF AN IMAGE SET WITH PARALLELEPIPEDS

5.1 Introduction

For the propagation of uncertainties, the two basic problems that have to be solved are the *inverse problem* [56] and the *direct problem* [100, 73]. Let us consider a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ and two vectors $\mathbf{x}$ and $\mathbf{y} = \mathbf{f}(\mathbf{x})$.

- The *direct problem*: Assume that $\mathbf{x}$ is known with some uncertainty, estimate $\mathbf{y} \in \mathbb{R}^m$ and characterize the uncertainty for $\mathbf{y}$
- The *inverse problem*: Assume that $\mathbf{y}$ is known with some uncertainty, estimate $\mathbf{x} \in \mathbb{R}^n$ and characterize the uncertainty for $\mathbf{x}$ [93, 64]

For this section, and the remainder of the manuscript, we will only focus on the direct problem. If $\mathbf{x}$ is known to lie in a set $\mathbb{X}$, its image by the function $\mathbf{f}$ is also a set $\mathbb{Y} = \mathbf{f}(\mathbb{X})$. In the context of this manuscript, the set $\mathbb{X}$ is the initial set of the robot and $\mathbb{Y}$ represents the reachable set that we want to compute. Set-based methods presented in Chapter 2 can be used to perform this computation. However, as mentioned in this same chapter, classical interval methods yield a pessimism which is often too large.

One may then consider zonotopes [24] or ellipsoids [34, 104] instead of intervals to limit the pessimism. However, some conservatism will always remain and can not be quantified. The only way to control the overestimation is to perform bisections, but this process makes the direct problem intractable [74], even if $\mathbf{f}$ is affine [108]. Some efficient algorithms have been proposed to find an outer enclosure for $\mathbb{Y}$ (see *e.g.*, [67]), but to our knowledge, no algorithm had been proposed to find an inner approximation.

The content of this section comes from our paper of [42]. As proposed in [127], the objective of this paper was to propose a new approach to cover the boundary of $\mathbf{f}(\mathbb{X})$, noted $\partial\mathbf{f}(\mathbb{X})$. The difference with [127] is that in our approach we propose to use small parallelepipeds instead of boxes. This creates a link with the literature on geometric

inference, and allows to represent the set $\mathbb{Y} = \mathbf{f}(\mathbb{X})$ using a Delaunay triangulation [17, 1] as suggested in Section 2.3.3.

We will also show how this method can be extended to the study of the flow function of an ODE. This result will be used later on in Chapter 6 to enclose the boundary of a reachable set in a set of parallelepipeds.

5.2 Parallelepiped inclusion function

In this section, we show how to compute an accurate approximation of $\mathbf{f}([\mathbf{x}])$, when $[\mathbf{x}]$ is a tiny box (*i.e.*, sufficiently small) and $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. We consider the case where $m \leq n$. The case $m < n$ will be needed later to compute the image of $\partial\mathbb{X}$, the boundary of the set of interest $\mathbb{X}$. To make notations lighter, we will further note $\mathbb{U} = \{\mathbf{u} \in \mathbb{R}^n \mid \|\mathbf{u}\| \leq 1\}$ the n dimensional unit ball.

To get a parallelepiped inclusion function for $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, we need to compute a parallelepiped which encloses $\mathbf{f}([\mathbf{x}])$, for a given box $[\mathbf{x}]$ of $\mathbb{R}^n$. Our idea is to compute this parallelepiped by inflating a linear approximation of $\mathbf{f}([\mathbf{x}])$. To do so, we first introduce an approximation theorem and a corollary.

5.2.1 Approximation theorem

1. Consider a function $\mathbf{f}$ of class C^1 from $\mathbb{R}^n$ to $\mathbb{R}^m$ and a box $[\mathbf{x}] \in \mathbb{IR}^n$ with center $\bar{\mathbf{x}} \in \mathbb{R}^n$. Define the linear approximation

$$\ell(\mathbf{x}) = \mathbf{f}(\bar{\mathbf{x}}) + \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \cdot (\mathbf{x} - \bar{\mathbf{x}}) \quad (5.1)$$

as illustrated by Figure 5.1. We have

$$\forall \mathbf{x} \in [\mathbf{x}], \|\mathbf{f}(\mathbf{x}) - \ell(\mathbf{x})\| \leq \rho \quad (5.2)$$

where

$$\rho = \rho_{\mathbf{f}}([\mathbf{x}]) = ub \left(\left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| \right) \quad (5.3)$$

Moreover

$$\rho = o(w([\mathbf{x}]))$$

i.e., ρ is small compared to $\varepsilon = w([\mathbf{x}])$, when $w([\mathbf{x}])$ is small.

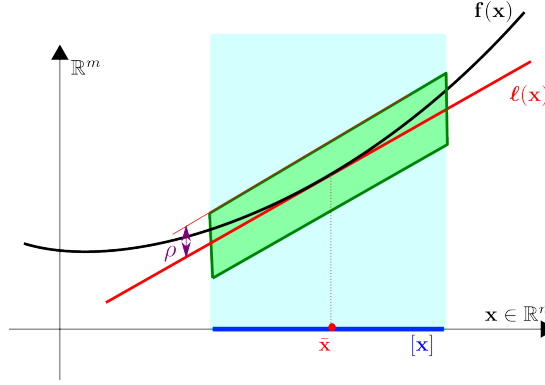


Figure 5.1 – Principle of the parallelepiped enclosure

Proof. Let us bound the error

$$\begin{aligned} \mathbf{e} &= \mathbf{f}(\mathbf{x}) - \ell(\mathbf{x}) \\ &= \mathbf{f}(\mathbf{x}) - \mathbf{f}(\bar{\mathbf{x}}) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \cdot (\mathbf{x} - \bar{\mathbf{x}}) \end{aligned} \quad (5.4)$$

using the methodology given in [60], Section 4.3, based on the centered form [93]. Since $\mathbf{x} \in [\mathbf{x}]$, we get

$$\begin{aligned} \mathbf{e}(\mathbf{x}) &\in [\mathbf{e}_c]([\mathbf{x}]) \\ &= \underbrace{\mathbf{e}(\bar{\mathbf{x}})}_{=0} + \left[\frac{d\mathbf{e}}{d\mathbf{x}} \right]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \\ &= \left[\frac{d(\mathbf{f}(\mathbf{x}) - \ell(\mathbf{x}))}{d\mathbf{x}} \right]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \\ &= \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \end{aligned} \quad (5.5)$$

The smallest ball which encloses the box $[\mathbf{e}_c]([\mathbf{x}])$ has a radius $\rho = \text{ub}(\|[\mathbf{e}_c]([\mathbf{x}])\|)$. Since

$$\begin{aligned} w\left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}})\right) &= O(w([\mathbf{x}])) \\ w([\mathbf{x}] - \bar{\mathbf{x}}) &= O(w([\mathbf{x}])) \end{aligned} \quad (5.6)$$

we have $w([\mathbf{e}_c]) = o(w([\mathbf{x}]))$. Moreover, since $\mathbf{0} \in [\mathbf{e}_c]$, we get $\text{ub}([\mathbf{e}_c]) = o(w([\mathbf{x}]))$, i.e. $\rho = o(w([\mathbf{x}]))$. $\square$

To define a useful corollary of this theorem, we first need to define the Hausdorff distance.

Definition 57. [15, 52] The *Hausdorff distance*, or *Pompeiu-Hausdorff Distance*, expresses how far two compact subsets of a metric space are. Let $(\mathbb{M}, d)$ be a metric space and $\mathbb{A} \subset \mathbb{M}$ and $\mathbb{B} \subset \mathbb{M}$ be two non-empty compact subsets of $\mathbb{M}$. The *Hausdorff distance* between $\mathbb{A}$ and $\mathbb{B}$ is defined by :

$$h(\mathbb{A}, \mathbb{B}) = \max \left\{ \sup_{a \in \mathbb{A}} d(a, \mathbb{B}), \sup_{b \in \mathbb{B}} d(b, \mathbb{A}) \right\} \quad (5.7)$$

where $d(a, \mathbb{B}) = \inf_{b \in \mathbb{B}} d(a, b)$ and $d(b, \mathbb{A}) = \inf_{a \in \mathbb{A}} d(b, a)$.

1. Given a C^1 function $\mathbf{f}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$ and a box $[\mathbf{x}] \in \mathbb{IR}^n$. We have

$$\mathbf{f}([\mathbf{x}]) \subset \boldsymbol{\ell}([\mathbf{x}]) + \rho \mathbb{U} \quad (5.8)$$

where $\mathbb{U}$ is the unit ball of $\mathbb{R}^n$, $\rho = \rho_{\mathbf{f}}([\mathbf{x}])$ as defined in Equation 5.3) and $\boldsymbol{\ell}([\mathbf{x}])$ is defined by Equation 5.1. In addition, the Hausdorff distance $h(\mathbf{f}([\mathbf{x}]), \boldsymbol{\ell}([\mathbf{x}]))$ is $o(w([\mathbf{x}]))$.

Proof. The inclusion is a consequence of Theorem 1. Moreover

$$\begin{aligned} h(\mathbf{f}([\mathbf{x}]), \boldsymbol{\ell}([\mathbf{x}])) &\leq \rho \text{ see(5.2)} \\ &= \text{ub} \left(\left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| \right) \\ &= O \left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| \\ &= O \left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \right) \right\| \cdot O \|([\mathbf{x}] - \bar{\mathbf{x}})\| \\ &= O \|w([\mathbf{x}])\| \cdot O \|w([\mathbf{x}])\| \\ &= o(w([\mathbf{x}])) \end{aligned} \quad (5.9)$$

□

These results will be used for the parallelepiped inflation process

5.2.2 Parallelepiped inflation

The objective of this section is to show how we can inflate the linear approximation $\boldsymbol{\ell}([\mathbf{x}])$ to get a parallelepiped enclosing $\boldsymbol{\ell}([\mathbf{x}]) + \rho \mathbb{U}$. The principle can be found in [95]. The linear approximation $\boldsymbol{\ell}([\mathbf{x}])$ is defined by Corollary 1 as

$$\boldsymbol{\ell}([\mathbf{x}]) = \{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in [-1, 1]^n, \mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}\} \quad (5.10)$$

It is a parallelepiped $\langle \mathbf{y} \rangle$ defined as

$$\langle \mathbf{y} \rangle = \bar{\mathbf{y}} + \mathbf{A} \cdot [-1, 1]^n \quad (5.11)$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\bar{\mathbf{y}} = \mathbf{b}$. This parallelepiped can be *flat* if $n < m$.

The idea of the inflation process is to enclose $\rho\mathbb{U}$ in a parallelepiped with the same shape as $\ell([\mathbf{x}])$, noted $\langle \mathbf{e} \rangle$, and to compute the sum $\langle \mathbf{y} \rangle + \langle \mathbf{e} \rangle$ as defined in Remark 12. This is illustrated Figure 5.2.

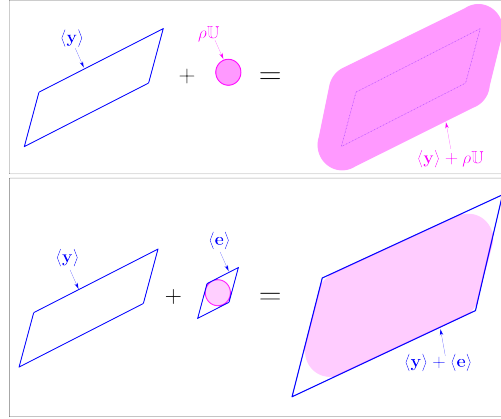


Figure 5.2 – Inflation of $\langle \mathbf{y} \rangle$ by a disk of radius ρ

Centered inflation, square case

Let us first consider the case where $n = m$ and $\bar{\mathbf{y}} = \mathbf{0}$.

1. Define the n dimensional centered parallelepiped

$$\langle \mathbf{y} \rangle = \mathbf{A} \cdot \begin{pmatrix} [-e_1, e_1] \\ \vdots \\ [-e_n, e_n] \end{pmatrix} \quad (5.12)$$

where $\mathbf{A}$ is an invertible square matrix. Consider the n dimensional unit ball $\mathbb{U}$ and an inflation coefficient $\rho > 0$. The parallelepiped

$$\langle \mathbf{z} \rangle = \mathbf{A} \cdot \begin{pmatrix} e_1 + \rho\sqrt{q_1} & 0 & \cdots & 0 \\ 0 & e_2 + \rho\sqrt{q_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & e_n + \rho\sqrt{q_n} \end{pmatrix} \cdot [-1, 1]^n, \quad (5.13)$$

where the q_i is the i th diagonal element of the matrix $\mathbf{Q} = (\mathbf{A}^T \mathbf{A})^{-1}$, satisfies

$$\langle \mathbf{z} \rangle \supset \langle \mathbf{y} \rangle + \rho \mathbb{U} \quad (5.14)$$

Figure 5.3 illustrates this inflation process with $\langle \mathbf{y} \rangle = \begin{pmatrix} 5 & 5 \\ 0 & 2 \end{pmatrix} \cdot [-1, 1]^2$ and $\rho = 1$.

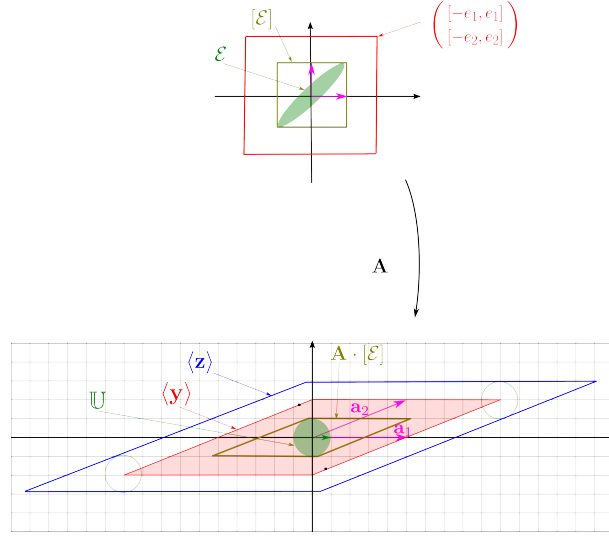


Figure 5.3 – Inflation of a parallelepiped in the case $m = n = 2$

Proof. We want to find a parallelepiped which encloses $\langle \mathbf{y} \rangle + \rho \mathbb{U}$. For this, we need to find a parallelepiped with a shape similar to $\langle \mathbf{y} \rangle$ which encloses the ball $\mathbb{U}$. Since $\mathbf{A}$ is invertible, we have

$$\mathbb{U} = \mathbf{A} \cdot \mathcal{E} \quad (5.15)$$

where $\mathcal{E}$ is the ellipsoid given by

$$\mathcal{E} = \left\{ \mathbf{x} \in \mathbb{R}^m \mid \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} \leq 1 \right\} \quad (5.16)$$

The smallest box with respect to the inclusion which encloses $\mathcal{E}$ is [104]

$$[\mathcal{E}] = \begin{pmatrix} \sqrt{q_1} & 0 & \cdots & 0 \\ 0 & \sqrt{q_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{q_m} \end{pmatrix} \cdot \begin{pmatrix} [-1, 1] \\ \vdots \\ [-1, 1] \end{pmatrix} \quad (5.17)$$

Since

$$\begin{aligned}
 \langle \mathbf{y} \rangle + \rho \mathbb{U} &= \langle \mathbf{y} \rangle + \rho \cdot \mathbf{A} \cdot \mathcal{E} \\
 &\subset \langle \mathbf{y} \rangle + \rho \cdot \mathbf{A} \cdot [\mathcal{E}] \\
 &= \mathbf{A} \cdot \begin{pmatrix} e_1 + \rho\sqrt{q_1} & 0 & \cdots & 0 \\ 0 & e_2 + \rho\sqrt{q_2} & & \vdots \\ \vdots & \ddots & & 0 \\ 0 & \cdots & 0 & e_n + \rho\sqrt{q_n} \end{pmatrix} \cdot \begin{pmatrix} [-1, 1] \\ \vdots \\ [-1, 1] \end{pmatrix} \quad (5.18)
 \end{aligned}$$

we get the enclosure to be proven. $\square$

Example 26. Consider the two dimensional case, where

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix} \quad (5.19)$$

We have

$$\mathbf{A}^T \cdot \mathbf{A} = \begin{pmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \end{pmatrix} \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{pmatrix} = \begin{pmatrix} \|\mathbf{a}_1\|^2 & \mathbf{a}_1^T \mathbf{a}_2 \\ \mathbf{a}_1^T \mathbf{a}_2 & \|\mathbf{a}_2\|^2 \end{pmatrix} \quad (5.20)$$

We get

$$\mathbf{Q} = (\mathbf{A}^T \cdot \mathbf{A})^{-1} = \frac{1}{\|\mathbf{a}_1 \wedge \mathbf{a}_2\|^2} \cdot \begin{pmatrix} \|\mathbf{a}_2\|^2 & \times \\ \times & \|\mathbf{a}_1\|^2 \end{pmatrix} \quad (5.21)$$

From Proposition 1, we get

$$\langle \mathbf{y} \rangle + \rho \mathbb{U} \subset \mathbf{A} \cdot \begin{pmatrix} e_1 + \rho \frac{\|\mathbf{a}_2\|}{\|\mathbf{a}_1 \wedge \mathbf{a}_2\|} & 0 \\ 0 & e_2 + \rho \frac{\|\mathbf{a}_1\|}{\|\mathbf{a}_1 \wedge \mathbf{a}_2\|} \end{pmatrix} \cdot \begin{pmatrix} [-1, 1] \\ [-1, 1] \end{pmatrix} \quad (5.22)$$

Centered flat parallelepiped inflation

Now let us consider the case of a flat parallelepiped, *i.e.* $n < m$, but centered ($\bar{\mathbf{y}} = 0$). In this case the matrix $\mathbf{A}$ is not square. We then first need to complete $\mathbf{A}$ with null-length vectors.

2. Define the m dimensional flat parallelepiped

$$\langle \mathbf{y} \rangle = \mathbf{A} \cdot \begin{pmatrix} [-e_1, e_1] \\ \vdots \\ [-e_n, e_n] \end{pmatrix} \quad (5.23)$$

where $n < m$. The matrix $\mathbf{A}$ is assumed to be full rank. Consider a matrix $\mathbf{K}$ such that the matrix $\tilde{\mathbf{A}} = [\mathbf{A}|\mathbf{K}]$ is square and invertible. Note that the $\mathbf{K}$ used here refers to the kernel or nullspace of $\mathbf{A}$ [122]. Consider the unit ball $\mathbb{U}$ of $\mathbb{R}^m$ and an inflation coefficient $\rho > 0$. The parallelepiped of $\mathbb{R}^m$:

$$\langle \mathbf{z} \rangle = \tilde{\mathbf{A}} \cdot \begin{pmatrix} e_1 + \rho\sqrt{q_1} & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & e_2 + \rho\sqrt{q_2} & & \vdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & \vdots & 0 & 0 \\ 0 & \cdots & 0 & e_n + \rho\sqrt{q_n} & 0 & \vdots & 0 \\ 0 & 0 & & 0 & \rho\sqrt{q_{n+1}} & 0 & \vdots \\ 0 & 0 & 0 & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \rho\sqrt{q_m} \end{pmatrix} \cdot [-1, 1]^m, \quad (5.24)$$

where q_i is the i th diagonal element of the matrix $\mathbf{Q} = (\tilde{\mathbf{A}}^T \tilde{\mathbf{A}})^{-1}$, satisfies

$$\langle \mathbf{z} \rangle \supset \langle \mathbf{y} \rangle + \rho\mathbb{U} \quad (5.25)$$

Proof. We have

$$\begin{aligned} \langle \mathbf{y} \rangle &= \mathbf{A} \cdot \begin{pmatrix} [-e_1, e_1] \\ \vdots \\ [-e_n, e_n] \end{pmatrix} + \mathbf{K} \cdot \begin{pmatrix} [0, 0] \\ \vdots \\ [0, 0] \end{pmatrix} \\ &= \underbrace{[\mathbf{A}|\mathbf{K}]}_{=\tilde{\mathbf{A}}} \cdot \begin{pmatrix} [-e_1, e_1] \\ \vdots \\ [-e_n, e_n] \\ [-e_{n+1}, e_{n+1}] \\ \vdots \\ [-e_m, e_m] \end{pmatrix} \end{aligned}$$

where $e_{n+1} = \cdots = e_m = 0$. By Proposition 1, $\langle \mathbf{z} \rangle$ in Equation 5.24 encloses $\langle \mathbf{y} \rangle + \rho\mathbb{U}$

which completes the proof. $\square$

This inflation process is summarized in Algorithm 5.

Algorithm 5 Parallelepiped inflation

```

1: Input:  $\langle \mathbf{y} \rangle, \rho$  such that  $\langle \mathbf{y} \rangle = \mathbf{A} \cdot \begin{pmatrix} [-e_1, e_1] \\ \vdots \\ [-e_n, e_n] \end{pmatrix}$ 
2: Output:  $\langle \mathbf{z} \rangle$ 
3:
4: function Inflate( $\langle \mathbf{y} \rangle, \rho$ )
5:    $\mathbf{K} \leftarrow \text{Kernel}(\mathbf{A})$ 
6:    $\tilde{\mathbf{A}} \leftarrow [\mathbf{A} \mid \mathbf{K}]$ 
7:    $\mathbf{Q} \leftarrow (\tilde{\mathbf{A}}^T \tilde{\mathbf{A}})^{-1}$ 
8:    $\mathbf{M} \leftarrow \mathbf{0}^{m \times m}$ 
9:   for  $i = 1$  to  $m$  do
10:     $\mathbf{M}(i, i) \leftarrow \rho \sqrt{\mathbf{Q}(i, i)}$ 
11:   end for
12:   for  $i = 1$  to  $n$  do
13:     $\mathbf{M}(i, i) \leftarrow \mathbf{M}(i, i) + e_i$ 
14:   end for
15:    $\langle \mathbf{z} \rangle \leftarrow \tilde{\mathbf{A}} \cdot \mathbf{M} \cdot [-1, 1]^m$ 
16: end function
    
```

Example 27. For this example, $n = 2$ and $m = 3$. Assume that we want to inflate by ρ the parallelepiped

$$\langle \mathbf{y} \rangle = \underbrace{\begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{pmatrix}}_{=\mathbf{A}} \cdot [-1, 1]^2 \quad (5.26)$$

We have

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{k} \end{pmatrix} \quad (5.27)$$

where $\mathbf{k} = \mathbf{a}_1 \wedge \mathbf{a}_2$. We have

$$\tilde{\mathbf{A}}^T \cdot \tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{k}^T \end{pmatrix} \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{k} \end{pmatrix} = \begin{pmatrix} \|\mathbf{a}_1\|^2 & \mathbf{a}_1^T \mathbf{a}_2 & 0 \\ \mathbf{a}_1^T \mathbf{a}_2 & \|\mathbf{a}_2\|^2 & 0 \\ 0 & 0 & \|\mathbf{k}\|^2 \end{pmatrix} \quad (5.28)$$

Since $\det \tilde{\mathbf{A}} = \|\mathbf{n}\|^2$, we get

$$\begin{aligned} \mathbf{Q} = (\tilde{\mathbf{A}}^T \cdot \tilde{\mathbf{A}})^{-1} &= \frac{1}{\|\mathbf{k}\|^4} \begin{pmatrix} \|\mathbf{a}_2\|^2 \|\mathbf{k}\|^2 & \times & \times \\ \times & \|\mathbf{a}_1\|^2 \|\mathbf{k}\|^2 & \times \\ \times & \times & \|\mathbf{a}_1\|^2 \|\mathbf{a}_2\|^2 - (\mathbf{a}_1^T \mathbf{a}_2)^2 \end{pmatrix} \\ &= \begin{pmatrix} \frac{\|\mathbf{a}_2\|^2}{\|\mathbf{k}\|^2} & \times & \times \\ \times & \frac{\|\mathbf{a}_1\|^2}{\|\mathbf{k}\|^2} & \times \\ \times & \times & \frac{\|\mathbf{a}_1\|^2 \|\mathbf{a}_2\|^2 - (\mathbf{a}_1^T \mathbf{a}_2)^2}{\|\mathbf{k}\|^4} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\|\mathbf{a}_2\|^2}{\|\mathbf{k}\|^2} & \times & \times \\ \times & \frac{\|\mathbf{a}_1\|^2}{\|\mathbf{k}\|^2} & \times \\ \times & \times & \frac{1}{\|\mathbf{k}\|^2} \end{pmatrix} \end{aligned} \quad (5.29)$$

Therefore

$$\begin{aligned} \langle \mathbf{z} \rangle &= \tilde{\mathbf{A}} \cdot \begin{pmatrix} 1 + \rho \frac{\|\mathbf{a}_2\|}{\|\mathbf{k}\|} & 0 & 0 \\ 0 & 1 + \rho \frac{\|\mathbf{a}_1\|}{\|\mathbf{k}\|} & 0 \\ 0 & 0 & \rho \frac{1}{\|\mathbf{k}\|} \end{pmatrix} \cdot [-1, 1]^3 \\ &= \left((1 + \rho \frac{\|\mathbf{a}_2\|}{\|\mathbf{k}\|}) \mathbf{a}_1 \quad (1 + \rho \frac{\|\mathbf{a}_1\|}{\|\mathbf{k}\|}) \mathbf{a}_2 \quad \rho \frac{\mathbf{k}}{\|\mathbf{k}\|} \right) \cdot [-1, 1]^3 \end{aligned} \quad (5.30)$$

Figure 5.4 illustrates this case.

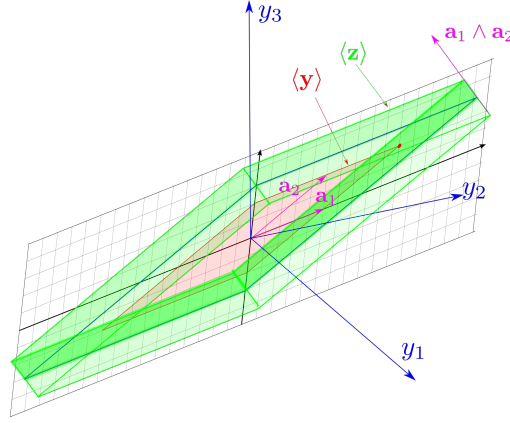
Non-centered case

If we relax the centered condition, we can decompose the parallelepiped $\langle \mathbf{y} \rangle$ as

$$\langle \mathbf{y} \rangle = \langle \mathbf{y}_c \rangle + \bar{\mathbf{y}} \quad (5.31)$$

where $\langle \mathbf{y}_c \rangle$ is a centered parallelepiped and $\bar{\mathbf{y}}$ is the center of $\langle \mathbf{y} \rangle$. To inflate $\langle \mathbf{y} \rangle$ by ρ , we simply inflate $\langle \mathbf{y}_c \rangle$ by ρ as described in Algorithm 5 and then translate the result by $\bar{\mathbf{y}}$.

We know are able to inflate a non-centered and flat parallelepiped. This result is the


 Figure 5.4 – Inflation of the flat parallelepiped $\langle \mathbf{y} \rangle$

basis of our parallelepiped inclusion function where we inflate a linear approximation of the image set. This is described in the following sub-section.

5.2.3 Parallelepiped enclosure of the image of a box

Parallelepiped inclusion function

The main result of this Subsection is a parallelepiped inclusion function for a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ if $n < m$. It is formulated as a Proposition.

3. Given $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $n < m$. A parallelepiped inclusion function is given by :

$$\begin{aligned} \langle \mathbf{f} \rangle : \mathbb{IR}^m &\mapsto \mathbb{PR}^n \\ [\mathbf{x}] &\rightarrow \langle \mathbf{f} \rangle([\mathbf{x}]) = \bar{\mathbf{y}} + \mathbf{A} \cdot [-1, 1]^n \end{aligned} \quad (5.32)$$

with

$$\begin{aligned} \bar{\mathbf{y}} &= \mathbf{f}(\bar{\mathbf{x}}) \\ \mathbf{A} &= \text{Inflate}\left(\frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}), \rho_{\mathbf{f}}([\mathbf{x}])\right) \\ \bar{\mathbf{x}} &= \text{center}([\mathbf{x}]) \end{aligned} \quad (5.33)$$

where the Inflate function is described by Algorithm 5.

Proof. A first order Taylor approximation of $\mathbf{f}(\mathbf{x})$ is

$$\mathbf{f}(\mathbf{x}) \simeq \underbrace{\mathbf{f}(\bar{\mathbf{x}}) + \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}}) \cdot (\mathbf{x} - \bar{\mathbf{x}})}_{\ell(\mathbf{x})} \quad (5.34)$$

Now, from Theorem 1 the error made by this approximation when $\mathbf{x} \in [\mathbf{x}]$ is lower than $\rho_{\mathbf{f}}([\mathbf{x}])$. The inflation has thus to be performed as defined in Algorithm 5. This yields the expression for $\mathbf{A}$. The translation by $\bar{\mathbf{y}}$ is explained in Subsection 5.2.2. $\square$

We can then study the convergence order of this inclusion function. To do so, we will consider the volumetric definition of the convergence order of Definition 34.

4. If $\mathbf{f}$ is of class C^1 , the parallelepiped inclusion function is of order $k = 1$.

This is illustrated by Figure 5.5 in the case where $n = 1$ and $m = 2$. We have $w([\mathbf{x}]) = \varepsilon$ and $\text{vol}([\mathbf{f}](\mathbf{x})) = \varepsilon^2$ and $\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])) = \varepsilon^3$

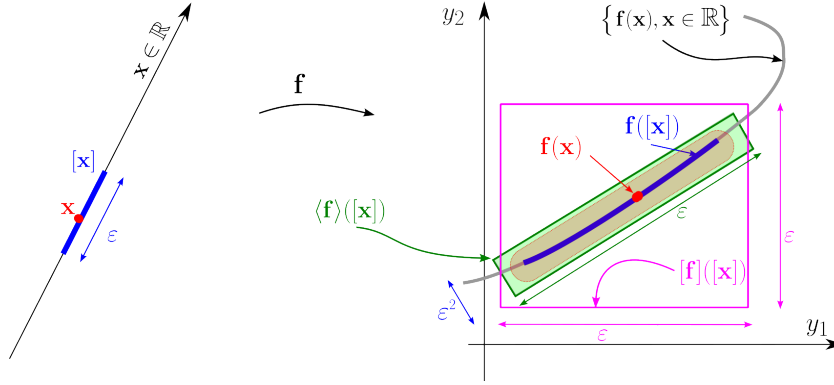


Figure 5.5 – Illustration of the convergence order for $n = 1$ and $m = 2$

Proof. Consider a tiny box $[\mathbf{x}]$ with center $\bar{\mathbf{x}}$. We have

$$w([\mathbf{x}] - \bar{\mathbf{x}}) = \varepsilon \quad (5.35)$$

As a consequence (see [99])

$$\begin{aligned} & w\left(\left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right](\mathbf{x}) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}})\right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}})\right) = \varepsilon^2 \\ \Rightarrow & \left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right](\mathbf{x}) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}})\right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| = \varepsilon^2 \\ \Rightarrow & \rho([\mathbf{x}]) = \text{ub} \left(\left\| \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right](\mathbf{x}) - \frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}})\right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| \right) = \varepsilon^2 \end{aligned} \quad (5.36)$$

This is a well known property of the centered form already shown in [93]. Now, due to our inflation process, we have

$$\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])) = \rho^{m-n}([\mathbf{x}]) \cdot \varepsilon^n \quad (5.37)$$

as illustrated by Figure 5.5 for $n = 1$ and $m = 2$. Since $\rho([\mathbf{x}]) = \varepsilon^2$, we have

$$\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}])) = \varepsilon^{2(m-n)} \cdot \varepsilon^n = \varepsilon^m \varepsilon^{1(m-n)} \quad (5.38)$$

which concludes the proof. $\square$

Parallelepiped enclosure of the image of a large box

With the previous Subsection, we defined a parallelepiped inclusion function which is efficient to enclose the image of a small box $[\mathbf{x}]$ by a function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $n < m$. Although this inclusion function also works for larger box, the conservatism of the enclosure may get to large. The box then has to be splitted into smaller subboxes. For each of the subboxes, we can compute a parallelepiped enclosing their image by $\mathbf{f}$. The union of the resulting parallelepipeds is finally an outer approximation of $\mathbf{f}([\mathbf{x}])$.

An example of implementation is proposed in Algorithm 6. In this algorithm we introduce a size $\varepsilon > 0$ such that all the subboxes have a width inferior to ε . The subdivision of line 5 can be done by a recursive bisection of $[\mathbf{x}_0]$ alongs its largest dimension until all the resulting boxes have a width inferior (or equal) to ε .

Algorithm 6 Parallelepiped Enclosure of the Image of a Box

```

1: Inputs:  $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$  such that  $n < m$ ,  $[\mathbf{x}_0] \in \mathbb{IR}^n$ ,  $\varepsilon > 0$ 
2: Output:  $L_{\langle \mathbf{y} \rangle}$  a list of parallelepipeds such that  $\mathbf{f}([\mathbf{x}]) \subseteq \bigcup_{\langle \mathbf{y} \rangle \in L_{\langle \mathbf{y} \rangle}} \langle \mathbf{y} \rangle$ 
3:
4: function PEIB( $\mathbf{f}$ ,  $[\mathbf{x}]$ ,  $\varepsilon$ )
5:    $L_{\langle \mathbf{y} \rangle} \leftarrow \{\}$ 
6:   Subdivide  $[\mathbf{x}_0]$  into a list of boxes  $L_{[\mathbf{x}]}$  of diameter  $\leq \varepsilon$ 
7:   for  $[\mathbf{x}]$  in  $L_{[\mathbf{x}]}$  do
8:      $\langle \mathbf{y} \rangle \leftarrow \langle \mathbf{f} \rangle([\mathbf{x}])$  ▷ As defined by Proposition 3
9:     Store  $\langle \mathbf{y} \rangle$  in  $L_{\langle \mathbf{y} \rangle}$ 
10:  end for
11: end function

```

By doing so, we have approximately ε^{-n} subboxes covering $[\mathbf{x}_0]$ noted $\{[\mathbf{x}](i), i \in \{1, 2, \dots\}\}$. Since $n < m$, $\mathbf{f}([\mathbf{x}]) \neq \emptyset$. Our parallelepiped approximation $\bigcup_{i \in \{1, 2, \dots\}} \langle \mathbf{f} \rangle([\mathbf{x}](i)) \subset \mathbb{R}^n$ has a strictly positive volume V which satisfies

$$\begin{aligned}
 \log V &= \log(\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}](i))) \cdot \varepsilon^{-n}) \\
 &= \log(\text{vol}(\langle \mathbf{f} \rangle([\mathbf{x}](i)))) - n \log \varepsilon \\
 &= (m + k(m - n)) \log \varepsilon - n \log \varepsilon \quad \text{See (2.53)} \\
 &= (k + 1)(m - n) \log \varepsilon
 \end{aligned} \quad (5.39)$$

Since $k = 1$ for the parallelepiped approximation (see Proposition 4), we get

$$\log V_{\langle \mathbf{f} \rangle} = 2(m - n) \log \varepsilon \quad (5.40)$$

If we consider the natural inclusion function, $k = 0$ and we get

$$\log V_{[\mathbf{f}]} = (m - n) \log \varepsilon \quad (5.41)$$

The parallelepiped approximation provides a better convergence order if

$$\begin{aligned} & \log V_{\langle \mathbf{f} \rangle} < \log V_{[\mathbf{f}]} \\ \Leftrightarrow & (2m - 2n) \log \varepsilon < (m - n) \log \varepsilon \\ \Leftrightarrow & 2m - 2n > m - n \\ \Leftrightarrow & m > n \end{aligned} \quad (5.42)$$

Meaning that our parallelepiped approximation becomes interesting as soon as $n < m$.

Example 28. Assume that we want to enclose the set $\boldsymbol{\psi}_0([\mathbf{x}_0])$ with $[\mathbf{x}_0] = [-1, 1]^2$ and

$$\boldsymbol{\psi}_0(\mathbf{x}) = \begin{pmatrix} \frac{1}{\sqrt{1+x_1^2+x_2^2}} \\ \frac{x_1}{\sqrt{1+x_1^2+x_2^2}} \\ \frac{x_2}{\sqrt{1+x_1^2+x_2^2}} \end{pmatrix} \quad (5.43)$$

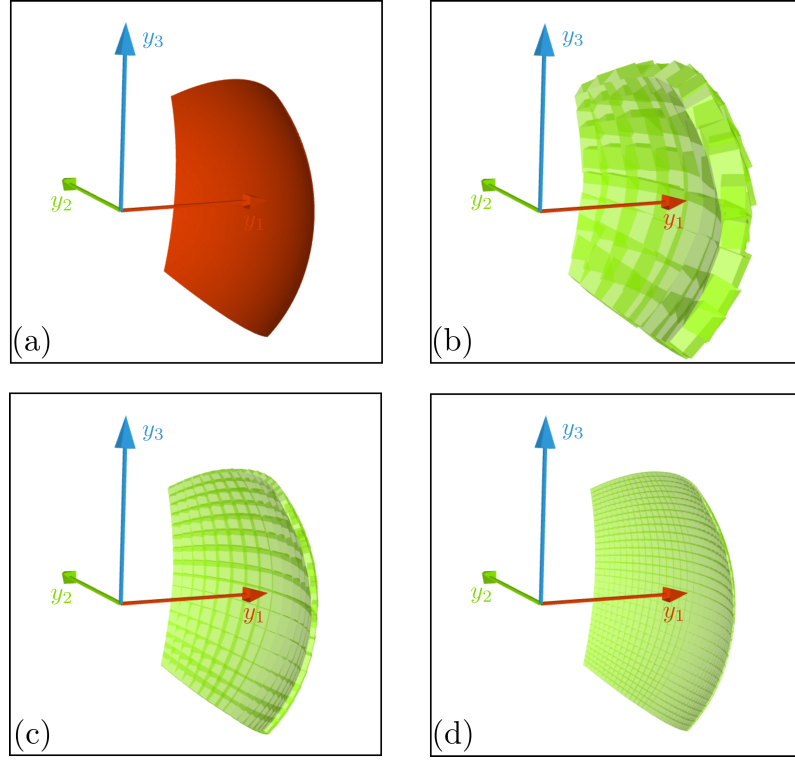
which corresponds to the *inverse gnomonic chart* that will be reconsidered later. The set $\boldsymbol{\psi}_0([\mathbf{x}])$ is painted red in Figure 5.6(a). Since $\|\boldsymbol{\psi}_0(\mathbf{x})\| = 1$, the red surface is a part of the unit sphere of $\mathbb{R}^3$. The Jacobian matrix is

$$\frac{d\boldsymbol{\psi}_0}{d\mathbf{x}} = \frac{1}{\sqrt{1+x_1^2+x_2^2}^3} \begin{pmatrix} -x_1 & -x_2 \\ 1+x_2^2 & -x_1x_2 \\ -x_1x_2 & 1+x_1^2 \end{pmatrix} \quad (5.44)$$

We can thus get the parallelepiped approximation. For $\varepsilon = 0.25$, we obtain Figure 5.6(b) with a pessimism of $\rho = 0.068$. For $\varepsilon = 0.125$, we obtain Figure 5.6(c) with $\rho = 0.015$. For $\varepsilon = 0.0625$, we get Figure 5.6(d) with $\rho = 0.003$.

Extension to set-valued functions

The parallelepiped inclusion function can be extended to set-valued functions, sometimes referred to as thick functions. Let us consider a C^1 function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathcal{P}(\mathbb{R}^m)$ and


 Figure 5.6 – Approximations of $\psi_0([-1, 1]^2)$

a box $[\mathbf{x}] \in \mathbb{I}\mathbb{R}^n$ with $\bar{\mathbf{x}} \in \mathbb{R}^n$ its center. Since $\mathbf{f}(\bar{\mathbf{x}}) \in \mathcal{P}(\mathbb{R}^m)$ is a set and not a point, it can not be used directly as an offset for the linear approximation. The same statement can be done about $\frac{d\mathbf{f}}{d\mathbf{x}}(\bar{\mathbf{x}})$. We then redefine Theorem 1 for this case.

2. Consider a function $\mathbf{f}$ of class C^1 from $\mathbb{R}^n$ to $\mathcal{P}(\mathbb{R}^m)$ and a box $[\mathbf{x}] \in \mathbb{I}\mathbb{R}^n$ with center $\bar{\mathbf{x}} \in \mathbb{R}^n$. Define the linear approximation

$$\ell(\mathbf{x}) = \mathbf{b} + \mathbf{A} \cdot (\mathbf{x} - \bar{\mathbf{x}}) \quad (5.45)$$

We have

$$\forall \mathbf{x} \in [\mathbf{x}], \|\mathbf{f}(\mathbf{x}) - \ell(\mathbf{x})\| \leq \rho \quad (5.46)$$

where

$$\rho = \rho_{\mathbf{f}}([\mathbf{x}]) = ub \left(\left\| [\mathbf{f}](\bar{\mathbf{x}}) - \mathbf{b} + \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \mathbf{A} \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \right\| \right) \quad (5.47)$$

Proof. The proof of this Theorem is similar to the one of Theorem 1. Let us bound the

error

$$\begin{aligned} \mathbf{e}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}) - \boldsymbol{\ell}(\mathbf{x}) \\ &= \mathbf{f}(\mathbf{x}) - \mathbf{b} - \mathbf{A} \cdot (\mathbf{x} - \bar{\mathbf{x}}) \end{aligned} \quad (5.48)$$

Its derivative is

$$\frac{d\mathbf{e}}{d\mathbf{x}}(\mathbf{x}) = \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}) - \mathbf{A} \quad (5.49)$$

using the methodology given in [60], Section 4.3, based on the centered form [93]. Since $\mathbf{x} \in [\mathbf{x}]$, we get

$$\begin{aligned} \mathbf{e}(\mathbf{x}) &\in [\mathbf{e}_c]([\mathbf{x}]) \\ &\in [\mathbf{e}](\bar{\mathbf{x}}) + \left[\frac{d\mathbf{e}}{d\mathbf{x}} \right]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \\ &= [\mathbf{f}](\bar{\mathbf{x}}) - \mathbf{b} + \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \mathbf{A} \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \end{aligned} \quad (5.50)$$

The ball of radius $\rho = \text{ub}(\| [\mathbf{f}](\bar{\mathbf{x}}) - \mathbf{b} + \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}]) - \mathbf{A} \right) \cdot ([\mathbf{x}] - \bar{\mathbf{x}}) \|)$ finally encloses $[\mathbf{e}_c]([\mathbf{x}])$. $\square$

We will further take $\mathbf{b}$ equal or close to $\overline{[\mathbf{f}](\bar{\mathbf{x}})}$ and $\mathbf{A}$ equal or close to $\overline{\left[\frac{d\mathbf{f}}{d\mathbf{x}} \right]([\mathbf{x}])}$.

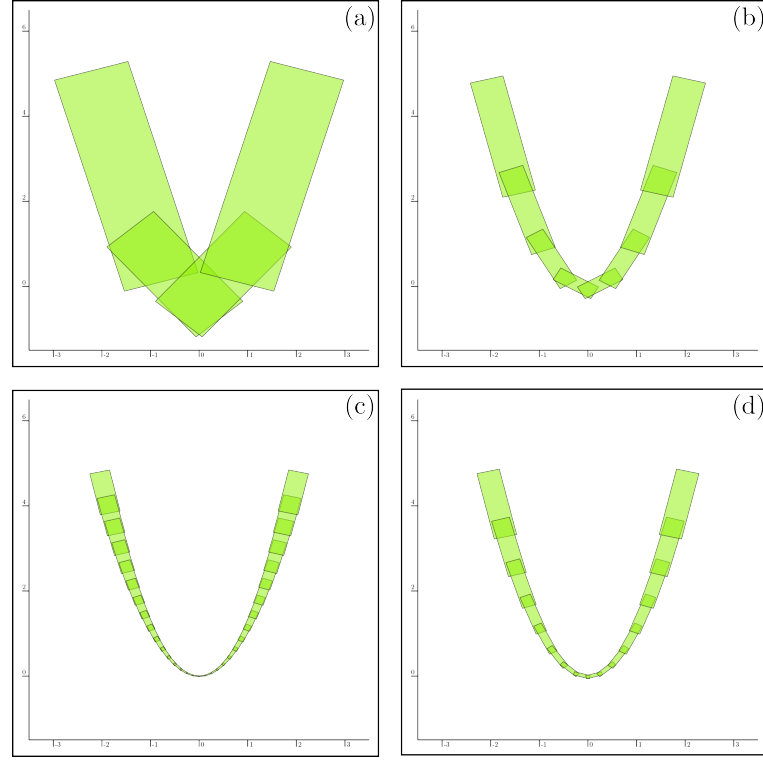
Example 29. Consider the set-value function

$$\begin{aligned} \mathbf{f} : \mathbb{R} &\rightarrow \mathbb{IR}^2 \\ x &\mapsto \begin{pmatrix} x \\ [1.1, 1.2] \cdot x^2 \end{pmatrix} \end{aligned} \quad (5.51)$$

Figure 5.7 shows approximations of $\mathbf{f}([-2, 2])$ obtained using the PEIB Algorithm 6 for $\varepsilon = 1$ (Figure 5.7(a)), $\varepsilon = 0.5$ (Figure 5.7(b)), $\varepsilon = 0.25$ (Figure 5.7(c)) and $\varepsilon = 0.125$ (Figure 5.7(d)).

5.3 Computing the image of a set by a nonlinear $\mathbb{R}^m \rightarrow \mathbb{R}^m$ function

In the previous section, we have shown how to compute an outer approximation of the image of a large box by a nonlinear function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $n < m$. Now, we want to compute an *inner* and an *outer* approximation of the image $\mathbb{Y} = \mathbf{f}(\mathbb{X})$ of a set $\mathbb{X}$ in the case where $n = m$. The set $\mathbb{X}$ can have any shape (*e.g.*, a ball), but it is assumed to


 Figure 5.7 – Approximations of $\mathbf{f}([-2, 2])$

be compact and to have a boundary $\mathbb{M} = \partial\mathbb{X}$ which is a smooth manifold of dimension $m - 1$. This assumption is needed to use our boundary approach. An atlas [69] covering $\mathbb{M}$ can be built. Now, to be able to use the results of the previous section, the atlas for $\mathbb{M}$ should be a box atlas as defined in Definition 45. The construction of the charts can be facilitated using symmetries. Yet to our knowledge, characterizing the class of manifolds for which a box atlas exists is still an open problem.

5.3.1 The gnomonic atlas

Let us introduce the notion of *gnomonic atlas*. They are box atlases, and rely on symmetries to simplify their construction.

The term *gnomonic* comes from the gnomonic projection as defined in [118]. Figure 5.8 shows a possible projection used to map the Earth. All great circles map to straight lines of the chart and large distortions occur far from the center. This projection will be used to cut the unit sphere into similar regions called quarters.

In the example of the sphere, all charts can be obtained from one of them using

$$P_\sigma = \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (5.53)$$

If we apply the symmetry σ to a point

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad (5.54)$$

, we get

$$\sigma(\mathbf{x}) = P_\sigma \cdot \mathbf{x} = \begin{pmatrix} x_3 \\ -x_1 \\ x_2 \end{pmatrix} \quad (5.55)$$

In the CODAC library [109] and further on in this manuscript, we adopt a simpler notation where we only write the second line of the Cauchy's representation. The first line is assumed to be $\begin{pmatrix} 1 & 2 & \dots \end{pmatrix}$. The symmetry from Equation 5.52 will then be simply noted $\sigma = \begin{pmatrix} 3 & -1 & 2 \end{pmatrix}$.

We can take for instance $s_1 = \begin{pmatrix} -2 & 1 & 3 \end{pmatrix}$ the rotation of $\frac{\pi}{2}$ with respect to the y_3 axis and $s_2 = \begin{pmatrix} 3 & 2 & -1 \end{pmatrix}$ the rotation of $\frac{\pi}{2}$ with respect to the y_2 axis. The six symmetry actions that are needed to cover the sphere are

$$\Sigma = \{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\} \quad (5.56)$$

We can thus write that the sphere corresponds to the set

$$\mathbb{S} = \bigcup_{\sigma \in \Sigma} \sigma \circ \psi_0([-1, 1]^2)$$

This is how we define our gnomonic atlas.

Definition 58. Given a manifold $\mathbb{M} \subset \mathbb{R}^m$ of dimension n . A *gnomonic atlas* $\{\psi_0, \Sigma\}$ for a manifold $\mathbb{M}$ is composed of the inverse chart ψ_0 and a set of symmetries Σ of $\mathbb{R}^m$ such that

$$\mathbb{M} = \bigcup_{\sigma \in \Sigma} \sigma \circ \psi_0([-1, 1]^m) \quad (5.57)$$

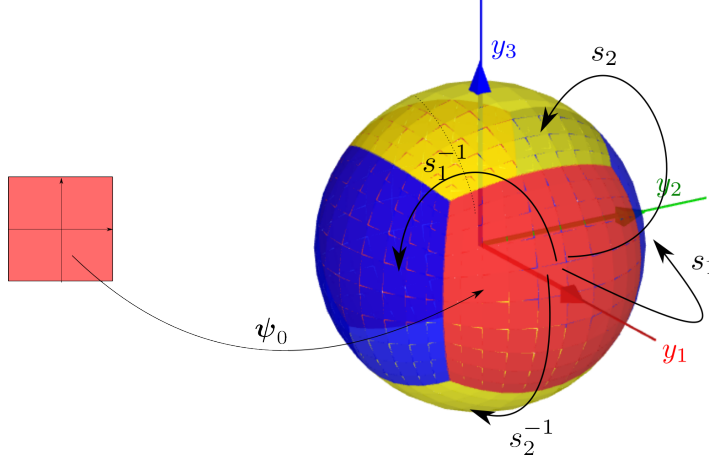


Figure 5.9 – Charts of the sphere

Moreover, the quarters $\sigma \circ \psi_0([-1, 1]^m)$, $\sigma \in \Sigma$ should not have overlapping interiors.

5.3.2 Inner and outer approximation of the image of a set

Consider a C^1 function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ and a compact set $\mathbb{X} \subset \mathbb{R}^m$. Its boundary $\mathbb{M} = \partial\mathbb{X}$ is assumed to be a smooth, compact manifold $\mathbb{M}$ of $\mathbb{R}^n$.

We say $\mathbf{f}$ is locally injective at a point $\mathbf{x}_0 \in \mathbb{X}$ if it is injective on some neighbourhood of $\mathbf{x}_0$; it is locally injective on $\mathbb{X}$ if it is locally injective at each point of $\mathbb{X}$. By the Inverse Function Theorem, if $\mathbf{f}$ is $\mathcal{C}^1$ and the Jacobian $\frac{d\mathbf{f}}{d\mathbf{x}}$ is nonsingular at $\mathbf{x}_0$ then $\mathbf{f}$ has a $\mathcal{C}^1$ inverse in a neighbourhood of $\mathbf{x}_0$ and in particular is locally injective.

In this case, an inner and an outer approximations of $\mathbb{Y} = \mathbf{f}(\mathbb{X})$ can be obtained by computing an enclosure for $\mathbf{f}(\mathbb{M})$.

Example 31. Take $f(x) = x^2$ and $\mathbb{X}_1 = [2, 3]$. We have $\mathbb{M}_1 = \{2, 3\}$, $f(\mathbb{M}_1) = \{4, 9\}$ and $f(\mathbb{X}_1) = [4, 9]$, which illustrates the fact that $f(\mathbb{X}_1)$ can be deduced from $f(\mathbb{M}_1)$. The condition of local injectivity is valid, since $\frac{df}{dx}(x) \neq 0, \forall x \in \mathbb{X}_1$.

If we now take $\mathbb{M}_2 = \{-2, 3\}$. We have $f(\mathbb{M}_2) = \{4, 9\}$ whereas $f(\mathbb{X}_2) = [0, 9]$. There is a fold at $x = 0$ and $f(\mathbb{X}_2)$ cannot be deduced from $f(\mathbb{M}_2)$. The condition of local injectivity is not valid, since $\frac{df}{dx}(0) = 0$.

In this subsection, we explain how this boundary approach can be implemented and illustrate why it is interesting in terms of convergence order.

If $\mathbf{f}$ is locally injective, from [51], we have: $\partial\mathbb{Y} \subset \mathbf{f}(\mathbb{M})$. If it is not case, we may have folds that contribute to $\partial\mathbb{Y}$. Since we assume the local injectivity of $\mathbf{f}$, to get an

inner and an outer approximations of $\mathbb{Y}$, it is enough to compute an outer enclosure for $\mathbf{f}(\mathbb{M})$. We thus get an outer enclosure for $\partial\mathbb{Y}$ from which we can deduce an inner and an outer approximations for $\mathbb{Y}$. Indeed, the $\partial\mathbb{Y}$ separates the space into two zones: $\mathbb{Y}$ and its complementary $\overline{\mathbb{Y}}$. Only the set $\mathbb{Y}$ is bounded, which allows us to know which zone is $\mathbb{Y}$ and which one is $\overline{\mathbb{Y}}$. The classification can be done using the Alexander rules [3]. For a given set $\mathbb{X} \subset \mathbb{R}^m$, the topological boundary $\mathbb{M}$ of $\mathbb{X}$ is defined as in Definition 38.

Enclosing the image of a manifold

The computation of $\mathbf{f}(\mathbb{M})$ can be done by relying on the parallelepiped inclusion function from Section 5.2 and the gnomonic atlas of Subsection 5.3.1. For our application, $\mathbb{M} = \partial\mathbb{X}$ is the boundary of a set.

3. Consider gnomonic atlas $\{\psi_0, \Sigma\}$ of a manifold $\mathbb{M} \subset \mathbb{R}^m$ of dimension n and a function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$. We have

$$\mathbf{f}(\mathbb{M}) \subset \bigcup_i \langle \mathbf{g}_i \rangle([-1, 1]^n) \quad (5.58)$$

where $\langle \mathbf{g}_i \rangle$ is a parallelepiped inclusion function of

$$\mathbf{g}_i = \mathbf{f} \circ \sigma_i \circ \psi_0 \quad (5.59)$$

and σ_i is the i th element of Σ . The construction of $\mathbf{g}_i([-1, 1]^n)$ is illustrated by Figure 5.10.

Proof. Since $\{\psi_0, \Sigma\}$ is a box atlas, we have

$$\mathbb{M} = \bigcup_i \sigma_i \circ \psi_0([-1, 1]^n)$$

Thus

$$\begin{aligned} \mathbf{f}(\mathbb{M}) &= \mathbf{f}(\bigcup_i \sigma_i \circ \psi_0([-1, 1]^n)) \\ &= \bigcup_i \mathbf{f} \circ \sigma_i \circ \psi_0([-1, 1]^n) \\ &= \bigcup_i \mathbf{g}_i([-1, 1]^n) \end{aligned}$$

Now, $\mathbf{g}_i([-1, 1]^n) \subset \langle \mathbf{g}_i \rangle([-1, 1]^n)$ which concludes the proof. $\square$

This Theorem naturally results in an algorithm called PEIBOS for the Parallelepipedic Enclosure of the Image of the BOundary of a Set. It is detailed by Algorithm 7

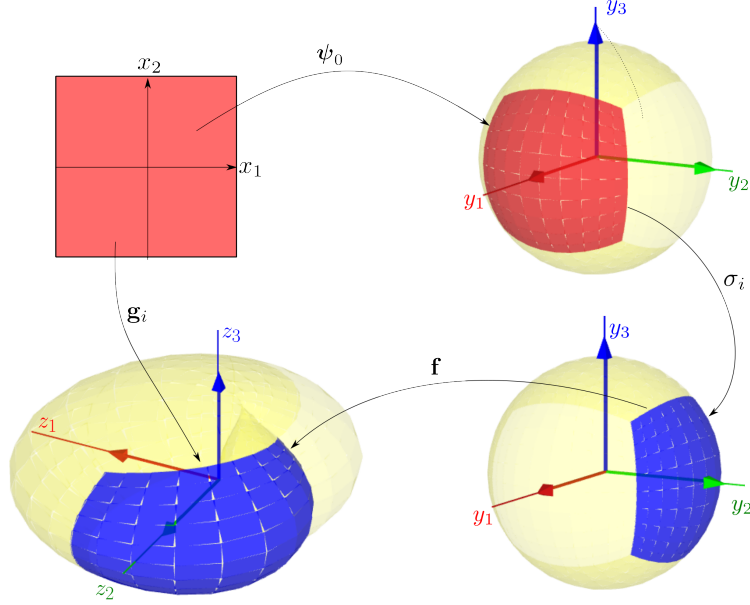


Figure 5.10 – Construction of the function $\mathbf{g}_i$

and outputs a set of parallelepipeds whose union encloses $\mathbf{f}(\mathbb{M})$. The PEIB algorithm mentioned line 7 refers to Algorithm 6.

Algorithm 7 Parallelepipedic Enclosure of the Image of the BOundary of a Set

- 1: **Inputs:** $\mathbf{f} : \mathbb{R}^m \mapsto \mathbb{R}^m$, A gnomonic atlas $\{\psi_0, \Sigma\}$ of $\mathbb{M} \subset \mathbb{R}^m$ with $\psi_0 : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $\varepsilon > 0$
 - 2: **Output:** $L_{\langle \mathbf{y} \rangle}$ a list of parallelepipeds such that $\mathbf{f}(\mathbb{M}) \subseteq \bigcup_{\langle \mathbf{y} \rangle \in L_{\langle \mathbf{y} \rangle}} \langle \mathbf{y} \rangle$
 - 3:
 - 4: **function** PEIBOS($\mathbf{f}, \psi_0, \Sigma, \varepsilon$)
 - 5: $L_{\langle \mathbf{y} \rangle} \leftarrow \{\}$
 - 6: **for** σ in Σ **do**
 - 7: $L_{\langle \mathbf{y} \rangle} \leftarrow L_{\langle \mathbf{y} \rangle} \cup \text{PEIB}(\mathbf{f} \circ \sigma \circ \psi_0, [-1, 1]^n, \varepsilon)$
 - 8: **end for**
 - 9: **end function**
-

5.3.3 Examples

3D case

Consider the mapping

$$\mathbf{f} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \rightarrow \begin{pmatrix} y_1^2 - y_2^2 + y_1 \\ 2y_1y_2 + y_2 \\ y_3 \end{pmatrix} \quad (5.60)$$

This function has been built using complex polynomials to avoid folding. The two first components of this function correspond to the conformal mapping in $\mathbb{C}$ given by $z \mapsto z^2 + z$. Indeed

$$(y_1 + iy_2)^2 + (y_1 + iy_2) = y_1^2 - y_2^2 + y_1 + i(2y_1y_2 + y_2) \quad (5.61)$$

We want to compute the image $\mathbb{Z}$ of the unit ball

$$\mathbb{Y} = \{\mathbf{y} \in \mathbb{R}^3 \text{ such that } \|\mathbf{y}\| \leq 1\} \quad (5.62)$$

We have

$$\frac{d\mathbf{f}}{d\mathbf{y}}(\mathbf{y}) = \begin{pmatrix} 2y_1 + 1 & -2y_2 & 0 \\ 2y_2 & 2y_1 + 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.63)$$

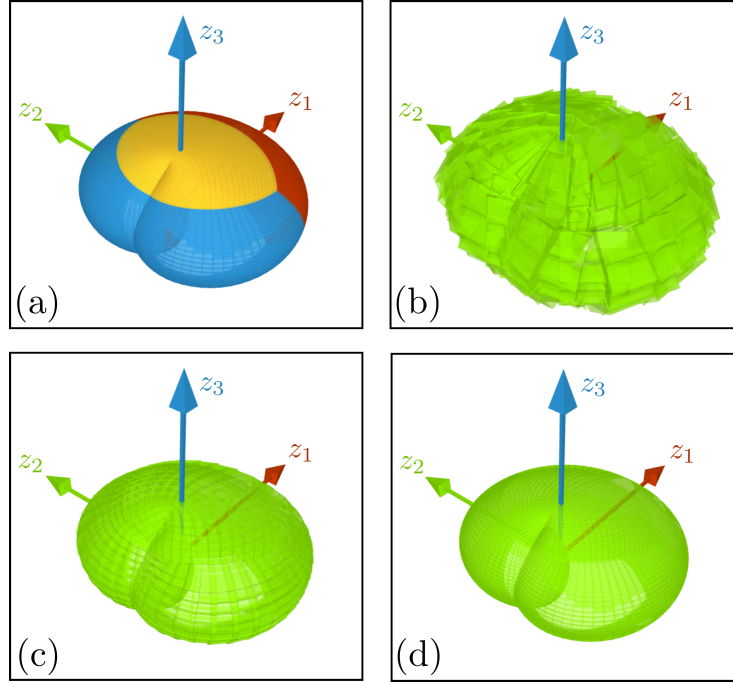
Since $\det\left(\frac{d\mathbf{f}}{d\mathbf{y}}(\mathbf{y})\right) = (2y_1 + 1)^2 + 4y_2^2$, is nonzero except when $y_1 = -\frac{1}{2}$ and $y_2 = 0$, $\mathbf{f}$ has no fold on $\mathbb{Y}$. This means that the inside and the outside of $\mathbb{Z} = \mathbf{f}(\mathbb{Y})$ are separated by $\mathbf{f}(\partial\mathbb{Y})$ which makes our approach valid. Moreover, $\mathbf{f}$ tends to wrap $\mathbb{Y}$ into itself creating a continuum of self intersections. A parallelepiped approximation of $\mathbf{f}(\partial\mathbb{Y})$ has been computed by the PEIBOS algorithm and is depicted Figure 5.11(a). For $\varepsilon = 0.25$, Algorithm 7 generates Figure 5.11(b). For $\varepsilon = 0.125$, we obtain Figure 5.11(c) and for $\varepsilon = 0.0625$, we get Figure 5.11(d).

Let us now use this example to illustrate Formula 2.53 in order to visualize the convergence order of the approximation. For this, we take different precisions for the paving and compute the volume of the parallelepiped enclosure. Since $n = 2$ and $m = 3$, we get for the parallelepiped approximation (see Equation 5.40):

$$\log V_{\langle \mathbf{f} \rangle} = 2(n - m) \log \varepsilon = 2 \log \varepsilon \quad (5.64)$$

and for the interval extension (see Equation 5.41).

$$\log V_{[\mathbf{f}]} = (n - m) \log \varepsilon = \log \varepsilon$$


 Figure 5.11 – Covering of $\mathbf{f}(\mathbb{M})$ using the PEIBOS algorithm

This is consistent with our numerical experiments depicted in Figure 5.12. The horizontal axis corresponds to $\log \varepsilon$ and the vertical axis corresponds to the volume of the approximations. The values used for this graph are summarized Table 5.1. For more detailed value, please refer to Subection 9.3.

The blue box illustrates an asymptotic slope $\frac{\log V_{[\mathbf{f}]}}{\log \varepsilon}$ of 1 and the orange boxes illustrates an asymptotic slope $\frac{\log V_{(\mathbf{f})}}{\log \varepsilon}$ of 2.

ε	Natural inclusion	Parallelepiped inclusion
0.25	31.2	12.4
0.125	15.5	1.2
0.0625	7.5	0.18
2^{-5}	3.9	0.032
2^{-6}	1.9	$6.6e - 3$
2^{-7}	0.97	$1.5e - 3$
2^{-8}	0.48	$3.6e - 4$

Table 5.1 – Volume comparison between the natural and parallelepiped inclusions

For a visual comparison, Figure 5.13 show the box and parallelepiped enclosures for $\varepsilon = 0.125$. The boxes are represented in blue Figure 5.13(a) and 5.13(c), while parallelepipeds

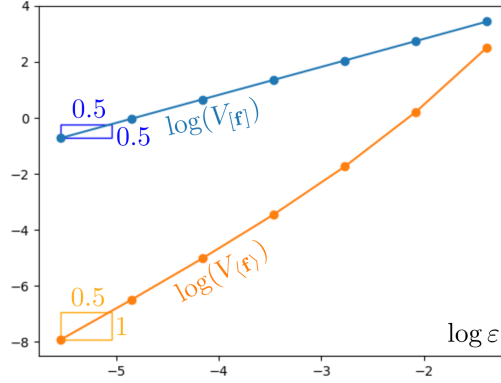


Figure 5.12 – Convergence order in terms of volume with respect to $\log \varepsilon$ using parallelepipeds (orange) and boxes (blue)

are shown in green Figure 5.13(b) and 5.13(d).

2D case

We choose here an example with a two dimensional solution set $\mathbb{Z}$. This makes the visualization of the inner and the outer approximations easier. Consider the mapping

$$\mathbf{f} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \rightarrow \begin{pmatrix} y_1^2 - y_2^2 + y_1 \\ 2y_1y_2 + y_2 \end{pmatrix} \quad (5.65)$$

which is a two dimensional version of the previous example. Since this case is two-dimensional, let us define the inverse chart

$$\psi_0(x) = \begin{pmatrix} \sin \left(x \cdot \frac{\pi}{4} \right) \\ \cos \left(x \cdot \frac{\pi}{4} \right) \end{pmatrix} \quad (5.66)$$

The image of $[-1, 1]$ by this function is a quarter of the unit circle. If we define s the rotation of $\frac{\pi}{2}$, the unit circle can be covered by the gnomonic atlas $\{\psi_0, \{1, s, s^2, s^{-1}\}\}$ as shown Figure 5.14.

We can then apply the PEIBOS algorithm with $\varepsilon = 0.125$ and the method from Subsection 2.3.3 to get the result from Figure 5.15a. We can notice that yellow parallelepipeds appear inside of the image set. This phenomenon of *fake boundaries* is characteristic of boundary based approaches and will be tackled later in Chapter 8. Figure 5.15b shows the result that we can expect after the removal of these fake boundaries.

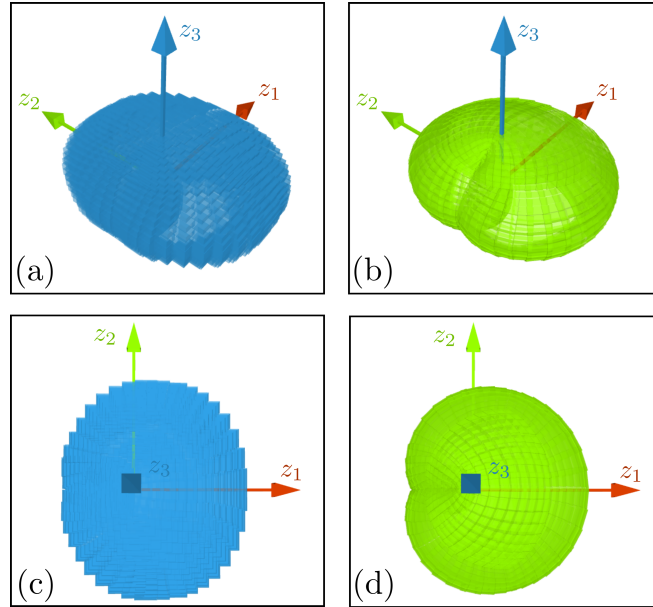


Figure 5.13 – Box (blue) and Parallelepiped (green) enclosures for $\varepsilon = 0.125$

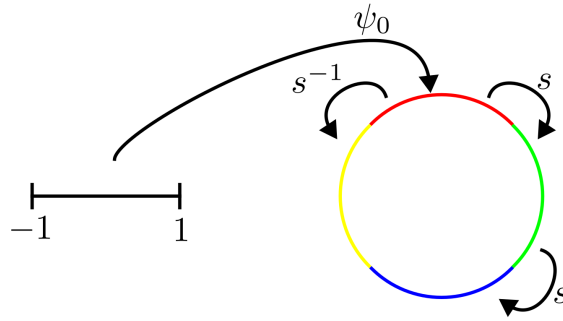


Figure 5.14 – Gnomonic atlas of the unit circle

5.4 Extension to ODE

In the context of this manuscript, we want to use this method to enclose the image of a set through an ODE. This will be further used for the computation of the reachable set.

5.4.1 Method

Consider the dynamical system

$$\dot{\mathbf{x}} = \gamma(\mathbf{x}) \quad (5.67)$$

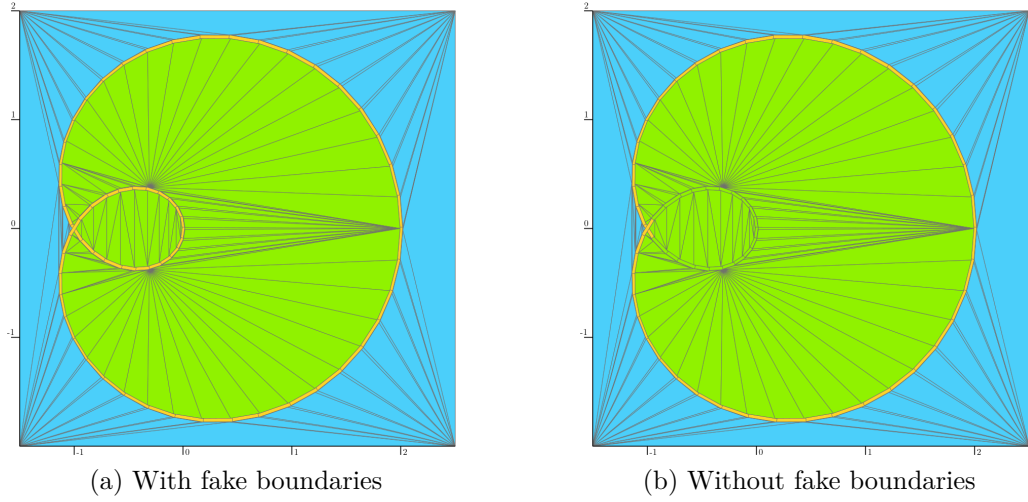


Figure 5.15 – Inner (green) and outer (green+yellow) approximation of the image set

defined by its state $\mathbf{x} \in \mathbb{R}^m$. The flow function ϕ_t as defined in Subsection 4.2.3 is the studied function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ and we want to compute the image of an initial set $\mathbb{X}_0$ by this function. We then need a way to compute the Jacobian $\frac{\partial \phi_t}{\partial \mathbf{x}}$ at the same time.

Define $\mathbf{J}(\mathbf{x}_0, t) = \frac{\partial \phi_t(\mathbf{x}_0)}{\partial \mathbf{x}}$ this Jacobian, it satisfies the *variational equation* [47]

$$\dot{\mathbf{J}} = \frac{\partial \gamma(\mathbf{x})}{\partial t} \cdot \mathbf{J} \quad (5.68)$$

with $\forall \mathbf{x}_0 \in \mathbb{R}^m$, $\mathbf{J}(\mathbf{x}_0, 0) = \mathbf{I}$. Integrating this ODE gives the desired Jacobian. The resulting IVP is

$$\left\{ \begin{array}{l} \begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{J}} \end{pmatrix} = \begin{pmatrix} \gamma(\mathbf{x}) \\ \frac{\partial \gamma(\mathbf{x})}{\partial t} \cdot \mathbf{J} \end{pmatrix} \\ \begin{pmatrix} \mathbf{x} \\ \mathbf{J} \end{pmatrix}(0) = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{I} \end{pmatrix} \end{array} \right. \quad (5.69)$$

And solving it allows us to use our parallelepipedic enclosure.

5.4.2 Examples

For the following examples, we will solve the IVP with the CAPD library. Note that the library already allows us to integrate the variational equation alongside the base ODE, so we will not detail it. However since the integrator is guaranteed, it outputs guaranteed bound as boxes, meaning to we will use the formulation of Subsection 5.2.3

to perform our parallelepipedic enclosure.

Pendulum

Consider the pendulum system with friction defined by the ODE

$$\begin{pmatrix} \dot{x}_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{g}{L} \sin(x_1) - 0.5x_2 \end{pmatrix} \quad (5.70)$$

For this example, we consider an initial set $\mathbb{X}_0$ as a disk of radius 0.1 centered around $\begin{pmatrix} -\frac{\pi}{4} \\ 0 \end{pmatrix}$ and integrate it over 10 seconds. The result is displayed Figure 5.16 where we show the set $\phi_t(\partial\mathbb{X}_0)$ every 0.2 second with a color map. The initial set $\mathbb{X}_0$ is in pink on the left of the figure, and the last time step is in red in the middle. Note that this result can be seen as a Reachability Analysis of the pendulum in an undisturbed case.

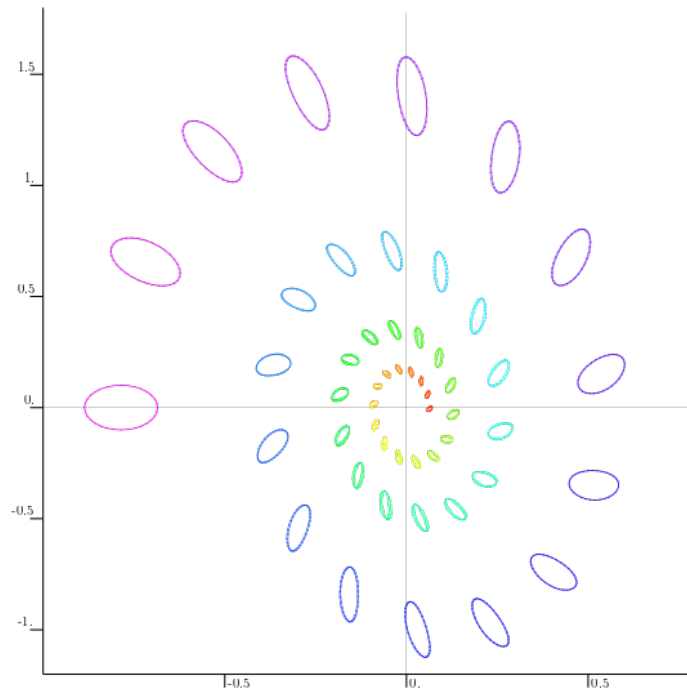


Figure 5.16 – Integration of the pendulum for 10 seconds

Lorenz system

Now consider the Lorenz system defined by

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 10(x_2 - x_1) \\ x_1(28 - x_2) - x_2 \\ x_1x_2 - \frac{8}{3}x_3 \end{pmatrix} \quad (5.71)$$

An integration of the unit ball $\mathbb{X}_0$ for $t \in \{0, 0.05, 0.1\}$ generates Figure 5.17. Quickly, we observe a strong distortion of the ball which is consistent with the chaotic behavior of the Lorenz system.

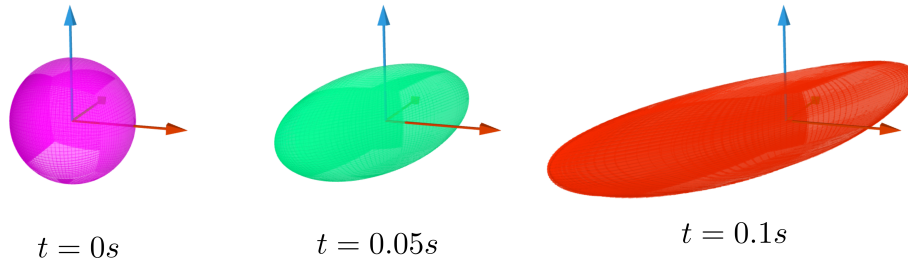


Figure 5.17 – Integration of the Lorenz system for 0.1 seconds

5.5 Conclusion

As suggested earlier in Section 1.3 the problematic of reachability analysis can be seen as the computation of the image of a set by a function.

Hence in this chapter we have proposed an original approach to compute an inner and an outer approximations for a set $\mathbb{Y}$ defined as the image of a set $\mathbb{X}$ by nonlinear function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$. This problem is fundamental as soon as we want to propagate uncertainties in a set-membership framework. Depending on the use case, both the inner and outer approximations of the reachable set have an interest. The method presented here is then interesting in this context.

Our approach assumes that $\mathbf{f}$ is locally injective and that the boundary of $\mathbb{X}$ can be covered by a gnomonic atlas. The principle is to cover $\partial\mathbf{f}(\mathbb{X})$, the boundary of $\mathbf{f}(\mathbb{X})$, by small parallelepipeds. Each parallelepiped is computed using a linearisation followed by an inflation. The inflation rate is evaluated using the centered form provided by interval arithmetic to guarantee the enclosure. This principle is summarized in two algorithms, respectively called PEIB and PEIBOS.

This method has a nice property of convergence, and we showed that it can even be

applied to propagate a set through an ODE. This result will be useful in the next chapter to perform a reachability analysis.

INNER AND OUTER APPROXIMATION OF THE REACHABLE SET

6.1 Introduction

As presented in Chapter 4 this manuscript focuses on forward and finite time reachability analysis. The result of a reachability analysis is an approximation of the reachable set, noted $\mathcal{R}_t$ further on. This approximation can either be an inner approximation [44], i.e. a subset $\mathcal{R}_t^- \subset \mathcal{R}_t$, or an outer approximation [39, 85], i.e. a superset $\mathcal{R}_t^+ \supset \mathcal{R}_t$. Depending on the application, both approximations can be useful.

The inner approximation is the set of states that the system will for sure be able to reach under an admissible control input. A typical use case is the computation of the region of attraction of a dynamical system [66, 125].

The outer approximation is a pessimistic and guaranteed estimation of the possible states of the system, i.e. the system can not be outside of this set. It is used in more critical applications such as system verification [5] and obstacle avoidance [131].

Both approximations are often represented as convex sets such as boxes [59, 92], ellipsoids [71] or zonotopes [8, 25]. The choice of a convex representation makes the analysis easier, but adds conservatism to the result. This is especially true for non-convex sets.

A solution, and the one explored in this chapter, is to first compute an outer-approximation of the reachable set boundary [75]. From this boundary approximation, one can identify regions that are inside or outside the reachable set. The Alexander rules [3] can be used to perform this classification. An inner approximation of the reachable set is obtained by computing the union of the “inside” areas. If we add the outer approximation of the boundary to this, we get an outer approximation of the reachable set. In addition, boundary based methods tend to be computationally lighter than classical methods [77].

Our method is based on the work from [85] that shows it is possible to recover the

convex hull of the reachable set by integrating an augmented ODE as described in Subsection 4.3.1. As mentioned earlier, this convexification adds an undesired pessimism to the resulting approximation.

In this chapter, we then extend the work from [85] in order to provide the reachable set boundary instead of its convex hull. In addition, our study relies on an initial set instead of a single state. This ability to handle an initial set allows us to handle uncertainties on the initial state of the system.

The main contribution of this paper is a method to compute an inner and outer-approximation of the reachable set. We provide an *augmented ODE* defined only on the boundary of the initial set. In this IVP, only one initial disturbance needs to be considered for each initial state. Under the stated assumptions, integrating it gives the reachable set boundary. In particular, we prove that we recover the exact boundary of the reachable set if it is smooth, and an outer-approximation of it otherwise.

6.2 Context

In this chapter, we will consider a differential inclusion of the form

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{w}(t) \\ \mathbf{x}(0) \in \mathbb{X}_0 \\ \forall t \in [0, T], \mathbf{w}(t) \in \mathbb{W} \end{cases} \quad (6.1)$$

And the following assumptions will be made: $\mathbb{X}_0$ is compact and its boundary $\partial\mathbb{X}_0$ is a smooth manifold $\mathbb{W}$ is compact and its boundary $\partial\mathbb{W}$ is an ovaloid $\mathbf{f}$ is at least C^1 , globally Lipschitz and its Jacobian $\frac{d\mathbf{f}}{d\mathbf{x}}$ is Lipschitz The disturbances $\mathbf{w} \in L^\infty([0, T], \mathbb{W})$ are integrable Assumption 6.2 is a standard smoothness assumption [88], and along with Assumption 6.2 they ensure that the flow function is always well defined. The compactness of $\mathbb{X}_0$ and $\mathbb{W}$ ensure that the reachable set is also compact if Assumption 6.2 is respected.

The smoothness assumption from Assumption 6.2 is further needed to define the normal to $\partial\mathbb{X}_0$ as defined in Section 3.3. Assumption 6.2 is more restrictive and requires $\partial\mathbb{W}$ to be an ovaloid so that its Gauss map (see Definition 48) is a bijection.

6.3 Characterization of the smooth reachable set

6.3.1 Preliminary result

For this subsection, we will consider the case where the boundary $\partial\mathcal{R}_t$ is smooth at any time $t > 0$. In this case there exists a bijection between the boundaries $\partial\mathbb{X}_0$ and $\partial\mathcal{R}_t$, i.e. the boundary of the reachable set is generated by the whole boundary of the initial set and the boundary only.

Before introducing the main contribution of this section, some intermediate results are needed.

5. Let us consider an ODE

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, t) \\ \mathbf{x}(0) \in \mathbb{X}_0 \end{cases} \quad (6.2)$$

where $\mathbf{f}$ is at least C^1 and the set $\mathbb{X}_0$ has a smooth boundary. Let us denote ϕ the flow function associated to this ODE, and $\mathbb{X}_t = \phi(t, \mathbb{X}_0)$.

Let $\mathbf{x}(0) \in \partial\mathbb{X}_0$ be an initial state and $\mathbf{x}(t) = \phi(t, \mathbf{x}(0))$ be the solution of the ODE for this initial state. Let $\mathbf{p}$ be a state such that

$$\mathbf{p}(0) \in N_{\mathbf{x}(0)}\partial\mathbb{X}_0 \quad (6.3)$$

and

$$\forall t \in [0, T], \dot{\mathbf{p}}(t) = -\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}(t))^\top \cdot \mathbf{p} \quad (6.4)$$

Then as depicted Figure 6.1

$$\forall t \in [0, T], \mathbf{p}(t) \in N_{\mathbf{x}(t)}\partial\mathbb{X}_t \quad (6.5)$$

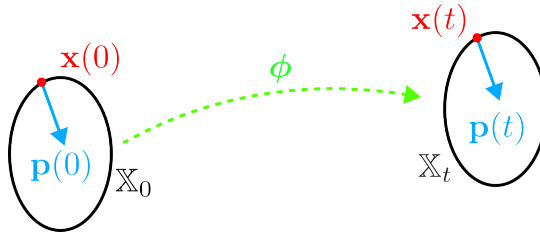


Figure 6.1 – Conservation of the orthogonality

Proof. Let $\delta \mathbf{x}(t)$ be a variation of the trajectory induced by an initial variation on the initial condition $\mathbf{x}_0$. Then $\delta \mathbf{x}(t)$ is tangent to $\mathbb{X}_t$ and satisfies the variational equation associated with the flow [86]:

$$\dot{\delta \mathbf{x}}(t) = \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}(t)) \cdot \delta \mathbf{x}(t) \quad (6.6)$$

In addition, $\mathbf{p}$ follows the adjoint dynamic from Equation 6.4.

The duality between the variational equation of the flow and the adjoint dynamics [86] gives us that the inner product between $\mathbf{p}(t)$ and $\delta \mathbf{x}(t)$ is constant, i.e.

$$\frac{d(\mathbf{p}(t)^T \delta \mathbf{x}(t))}{dt} = 0 \quad (6.7)$$

Since $\mathbf{p}(0) \in N_{\mathbf{x}_0} \partial \mathbb{X}_0$, we have

$$\mathbf{p}(0)^T \delta \mathbf{x}(0) = 0 \quad (6.8)$$

Then,

$$\forall t \in [0, T], \mathbf{p}(t)^T \delta \mathbf{x}(t) = 0 \quad (6.9)$$

Therefore $\forall t \in [0, T], \mathbf{p}(t) \in N_{\mathbf{x}(t)} \partial \mathbb{X}_t$. $\square$

The following proposition is inspired by the work from [85]. Since in this article we want to recover the boundary of the reachable set instead of its convex hull, we redemonstrate the result.

6. *Let us consider a differential inclusion as defined in Equation 6.1 such that the Assumptions 6.2 to 6.2 are verified. The disturbances responsible for the boundary points of the reachable set at any time $t \in [0, T]$ can be expressed as*

$$\mathbf{w}(\mathbf{x}, t) = (\mathbf{n}^{\partial \mathbb{W}})^{-1}(\mathbf{q}(\mathbf{x}, t)) \quad (6.10)$$

Where $\mathbf{q}(\mathbf{x}, t)$ is a unit length, outward pointing vector, orthogonal to the reachable set $\mathcal{R}_t$ at point $\mathbf{x}(t)$, i.e. $\mathbf{q}(\mathbf{x}, t) = \mathbf{n}^{\partial \mathcal{R}_t}(\mathbf{x}(t))$.

Proof. In this subsection the boundary of the reachable set $\mathcal{R}_t$ is smooth. At any time $t \in [0, T]$, the reachable set $\mathcal{R}_t$ can then be defined as the zero sublevel of a function v

$$\mathcal{R}_t = \{\mathbf{x} \in \mathbb{R}^n \mid v(\mathbf{x}, t) \leq 0\} \quad (6.11)$$

where $v : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}$ is the solution of the Hamilton-Jacobi PDE [89]

$$\dot{v}(\mathbf{x}, t) + H\left(\mathbf{x}, \frac{dv}{d\mathbf{x}}(\mathbf{x}, t)\right) = 0 \quad (6.12)$$

where H is the Hamiltonian defined by

$$\begin{aligned} H(\mathbf{x}, \mathbf{p}) &= \max_{\mathbf{w} \in \mathbb{W}} \mathbf{p}^\top \cdot (\mathbf{f}(\mathbf{x}) + \mathbf{w}) \\ &= \mathbf{p}^\top \cdot \mathbf{f}(\mathbf{x}) + \max_{\mathbf{w} \in \mathbb{W}} \mathbf{p}^\top \cdot \mathbf{w} \end{aligned} \quad (6.13)$$

This result is a specific case of Lemma 7 from [89] in the case where there is only a disturbance player.

Then the optimal disturbance is

$$\mathbf{w}(\mathbf{x}, t) = \arg \max_{\mathbf{w} \in \mathbb{W}} \left(\left(\frac{dv}{d\mathbf{x}}(\mathbf{x}, t) \right)^\top \cdot \mathbf{w} \right) \quad (6.14)$$

We want to characterize the boundary of the reachable set

$$\partial \mathcal{R}_t = \{\mathbf{x} \in \mathbb{R}^n \mid v(\mathbf{x}, t) = 0\} \quad (6.15)$$

Let us introduce

$$\mathbf{q}(\mathbf{x}, t) = \frac{\frac{dv}{d\mathbf{x}}(\mathbf{x}, t)}{\left\| \frac{dv}{d\mathbf{x}}(\mathbf{x}, t) \right\|} \quad (6.16)$$

Since in this subsection the boundary $\partial \mathcal{R}_t$ is smooth, $\mathbf{q}$ is well defined (*i.e.* $\left\| \frac{dv}{d\mathbf{x}}(\mathbf{x}, t) \right\|$ is never null).

Since v is the value function describing the reachable set, we have at every time $t \in [0, T]$

$$\forall \mathbf{x} \in \partial \mathcal{R}_t, \nabla v(\mathbf{x}, t) \in N_{\mathbf{x}} \partial \mathcal{R}_t \quad (6.17)$$

This relation is directly deduced from the definition of the gradient and is depicted Figure 6.2. Equation 6.14 can then be rewritten

$$\mathbf{w}(\mathbf{x}, t) = \arg \max_{\mathbf{w} \in \mathbb{W}} \left(\mathbf{q}(\mathbf{x}, t)^\top \cdot \mathbf{w} \right) \quad (6.18)$$

Since $\mathbb{W}$ is compact and its boundary $\partial \mathbb{W}$ is an ovaloid, the Gauss map of $\partial \mathbb{W}$ is a bijection and is invertible.

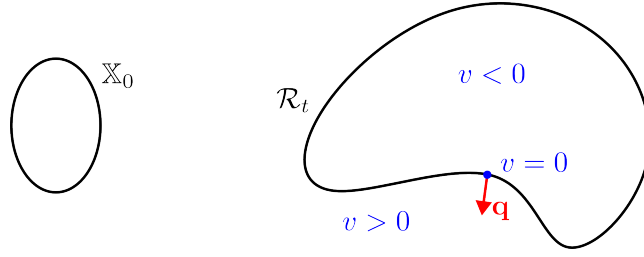


Figure 6.2 – Reachable set as a zero sublevel

Finally, since $\mathbf{q}$ is a unit length vector the maximizer is given by the inverse Gauss map as depicted Figure 6.3:

$$\mathbf{w}(\mathbf{x}, t) = \left(\mathbf{n}^{\partial \mathbb{W}} \right)^{-1} (\mathbf{q}(\mathbf{x}, t)) \quad (6.19)$$

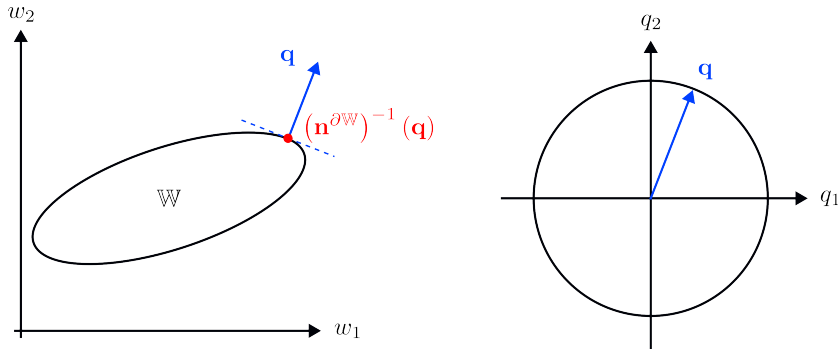


Figure 6.3 – Optimal disturbance

And $\mathbf{q}$ is a unit normal, outward pointing vector to the reachable set. $\square$

6.3.2 Main contribution

From these two intermediate results, we can introduce the main contribution of this article in the form of a proposition.

7. *Let us consider a differential inclusion as defined in Equation 6.1 such that the Assumptions 6.2 to 6.2 are verified. Solving over a time t the augmented IVP*

$$\left\{ \begin{array}{l} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \left(\mathbf{n}^{\partial \mathbb{W}} \right)^{-1} (\mathbf{q}) \\ \dot{\mathbf{q}} = -\text{Proj}_{\mathbf{q}} \left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x})^\top \mathbf{q} \right) \\ \left(\mathbf{x}(0), \left(\mathbf{n}^{\partial \mathbb{W}} \right)^{-1} (\mathbf{q}(0)) \right) \in \partial \mathbb{X}_0 \times \partial \mathbb{W} \\ \mathbf{q}(0) = \mathbf{n}^{\partial \mathbb{X}_0}(\mathbf{x}(0)) \end{array} \right. \quad (6.20)$$

where the *Proj* operator is the one from Equation 3.8, for every $\mathbf{x}(0) \in \partial\mathbb{X}_0$ gives a manifold $\mathbb{M}_t$. The projection of this manifold on the n first dimensions $\Pi_n(\mathbb{M}_t)$ is the boundary of the reachable set $\partial\mathcal{R}_t$. This proposition is displayed on Figure 6.4.

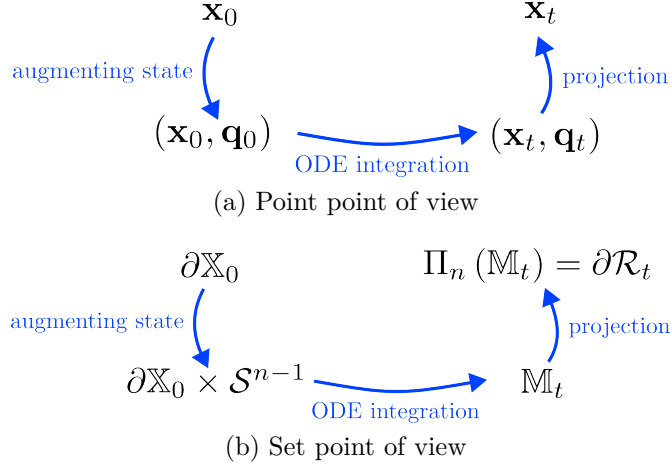


Figure 6.4 – Proposed scheme for reachable set computation

Let us consider an initial state $\mathbf{x}_0 \in \partial\mathbb{X}_0$ and an adjoint state $\mathbf{p} : [0, T] \in \mathbb{R}^m$ such that

$$\begin{cases} -\mathbf{p}(0) \in N_{\mathbf{x}_0} \partial\mathbb{X}_0 \\ \dot{\mathbf{p}}(t) = -\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}(t))^\top \cdot \mathbf{p} \end{cases} \quad (6.21)$$

Under a given disturbance $\mathbf{w} \in L^\infty([0, T], \mathbb{W})$, the differential inclusion from Equation 6.1 becomes an ODE. From Proposition 5, we know that

$$\forall t \in [0, T], \mathbf{p}(t) \in N_{\mathbf{x}(t)} \partial\mathbb{X}_t \quad (6.22)$$

Let us define

$$\begin{aligned} \mathbf{q} : \mathbb{R} &\rightarrow \mathbb{R}^n \\ t &\mapsto -\frac{\mathbf{p}(t)}{\|\mathbf{p}(t)\|} \end{aligned} \quad (6.23)$$

At all time $t \in [0, T]$, $\mathbf{q}(t)$ is a unit vector and

$$\forall t \in [0, T], \mathbf{q}(t) \in N_{\mathbf{x}(t)} \partial\mathbb{X}_t \quad (6.24)$$

If $\mathbf{p}$ is chosen to point inward, then $\mathbf{q}$ points outward and

$$\forall t \in [0, T], \mathbf{q}(t) = \mathbf{n}^{\partial \mathbb{X}_t}(\mathbf{x}(t)) \quad (6.25)$$

This result is depicted Figure 6.5.

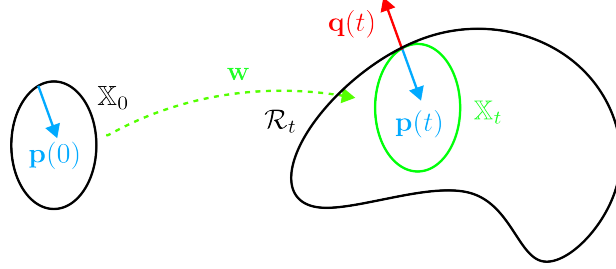


Figure 6.5 – Gauss map of the reachable set

By definition of the normal space from, we also have

$$\forall t \in [0, T], \mathbf{q}(t) = \mathbf{n}^{\partial \mathcal{R}_t}(\mathbf{x}(t)) \quad (6.26)$$

From Proposition 6 we know that all the points on the boundary of the reachable set are obtained by disturbances of the form

$$\mathbf{w}(\mathbf{x}, t) = \left(\mathbf{n}^{\partial \mathcal{W}} \right)^{-1}(\mathbf{q}(t)) \quad (6.27)$$

Finally from [85] we know that

$$\dot{\mathbf{q}} = -\text{Proj}_{\mathbf{q}}\left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x})^\top \mathbf{q}\right) \quad (6.28)$$

and solving the IVP 6.20 over a time t and projecting the result gives the boundary of the reachable set at time t , $\partial \mathcal{R}_t$.

6.3.3 Examples

For the following examples, we will use the CAPD library along with the work from [42] to get a parallelepipedic outer approximation of $\mathbb{M}_t$ as suggested in Section 5.4. The PEIBOS implementation from the CODAC library [109] is used to generate the parallelepipeds¹.

1. <https://codac.io/manual/functions/peibos/peibos.html>

Spiral system

For this application we will consider the spiral system from [113]. This kind of system is modeled as a neural ODE [21].

We note $\mathbf{x} \in \mathbb{R}^2$ the state of the system. It follows the differential inclusion

$$\begin{cases} \dot{\mathbf{x}}(t) = W_2 \tanh(W_1 \mathbf{x} + b_1) + b_2 + \mathbf{w}(t) \\ \mathbf{x}(0) \in \mathbb{X}_0 \\ \forall t \in [0, T], \mathbf{w}(t) \in \mathbb{W} \end{cases} \quad (6.29)$$

Where $W_1 \in \mathbb{R}^{10 \times 2}$ and $W_2 \in \mathbb{R}^{2 \times 10}$ are weight matrices and $b_1 \in \mathbb{R}^{10}$ and $b_2 \in \mathbb{R}^2$ are bias vectors. The values of these matrices and vector are available in [113].

We choose $\mathbb{X}_0$ as a disk of radius 0.3, centered around $\begin{pmatrix} 2 & 0 \end{pmatrix}^\top$, and $\mathbb{W}$ as a disk of radius 0.1, centered around $\begin{pmatrix} 0 & 0 \end{pmatrix}^\top$. Assumptions 6.2 and 6.2 are then satisfied. We integrate the ODE over 1s and display the result every 0.25s.

To do so we use the PEIBOS algorithm. The inverse chart in this case is

$$\psi_0(x) = \begin{pmatrix} 0.3 \cdot \sin\left(x \cdot \frac{\pi}{4}\right) \\ 0.3 \cdot \cos\left(x \cdot \frac{\pi}{4}\right) \\ \sin\left(x \cdot \frac{\pi}{4}\right) \\ \cos\left(x \cdot \frac{\pi}{4}\right) \end{pmatrix} \quad (6.30)$$

We define the symmetry $s = \begin{pmatrix} -2 & 1 & -4 & 3 \end{pmatrix}$. A gnomonic atlas of $\partial\mathbb{X}_0 \times S^1$ is then $\{\psi_0, \{1, s, s^2, s^{-1}\}\}$

For each time step we display as shown on Figure 6.6a:

- In pink the integration of $\partial\mathbb{X}_0$ with no disturbance
- In grey 50 integrations of $\partial\mathbb{X}_0$ with a random and constant disturbance
- In red the boundary of the reachable set, computed with our method

The total 1s of integration is visible on the left of Figure 6.6a, while Figure 6.6b show a zoomed view of the last time step.

At any time, the undisturbed and disturbed trajectories give a set which is included in the reachable set computed with our method. We can then say that we are indeed able to characterize the reachable set of the spiral at any time.

Note that the process used in Subsection 5.3.3 could also be used here since the red zonotope constitute an over approximation of the boundary. The interior of the red area is an inner approximation of the reachable set, while its union with the red over boundary

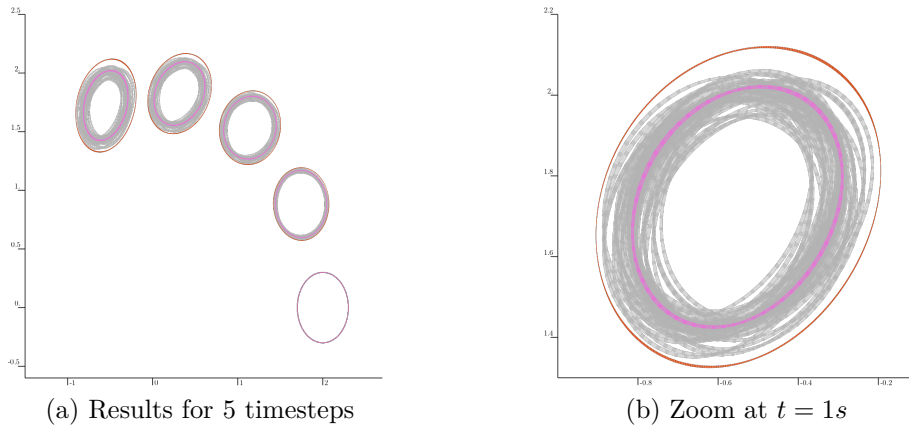


Figure 6.6 – 1s of integration of the spiral system

is an outer approximation of it.

A real Dubins car : Hlios

For this application we will study an USV (Unmanned Surface Vehicle) named Hlios and shown Figure 6.7. This robot is a boat equipped with four propellers : two on the right and two on the left (see Figure 6.7b).

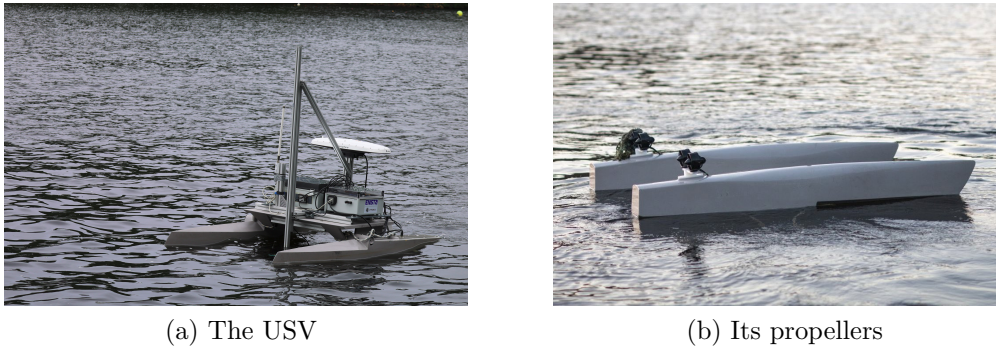


Figure 6.7 – Hélios USV

The robot is controlled in differential, *i.e.* at constant forward speed with a difference between the left and right motors to make it turn. It can then be modelled as a Dubins car. For this application we asked the robot to follow a desired heading θ_d . Its behavior can then be described by the ODE

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -v \cdot \cos(\theta) \\ v \cdot \sin(\theta) \\ K \cdot (\theta_d - \theta) \end{pmatrix} \quad (6.31)$$

where v is the forward speed and K is a regulation constant. For this application, we take $v = 1.48$ and $K = 0.2$. The initial set is a sphere of radius 0.15. Figure 6.8 show in three dimensions the states measured by the robot (in green) and the states obtained by integrating the undisturbed ODE from Equation 6.31. Both cover 1.83s of integration to match the measurements, the pink sets are drawn at $t \in \{0, 0.5, 1.0, 1.5, 1.83\}$.

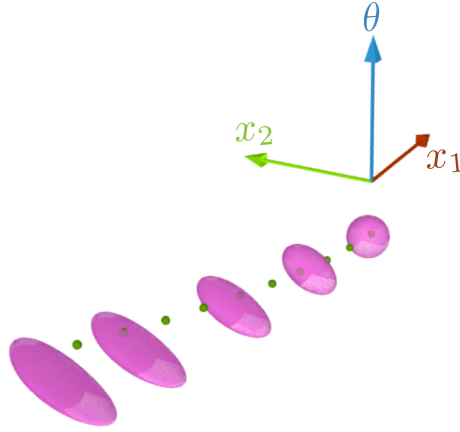


Figure 6.8 – Undisturbed (pink) and reference (green) trajectories

In addition, Figure 6.9 shows these trajectories projected in the three orthogonal planes for a better understanding. Note the the projection on the (x_1, x_2) plane Figure 6.9a is the trace of the robot, *i.e.* its position through time. We can observe that it turns right.

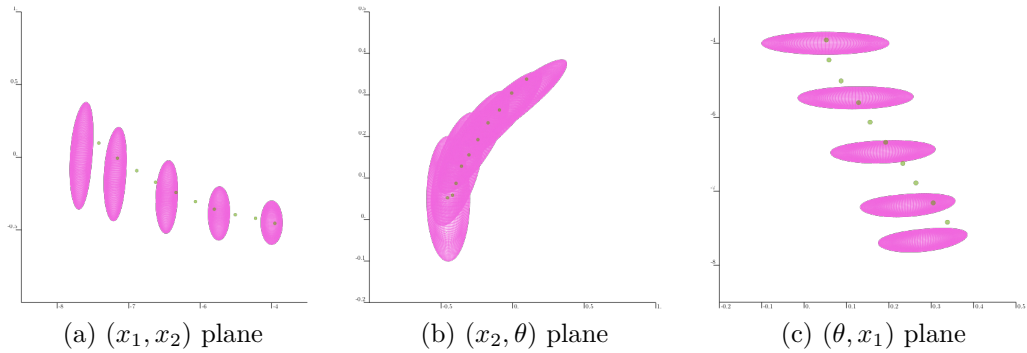


Figure 6.9 – Projections of the trajectories

We model the noise on the system (wind, current...) as a sphere $\mathbb{W}$ of radius 0.06 and use our method to compute the reachable set at time $t = 1.83s$. For a better understanding of the inclusion of the volumes, Figure 6.10 shows in green the true state of the robot (noted $\mathbf{x}_t$), in pink the undisturbed set, and in red the reachable set.

Note that not all the parallelepipeds are projected since we performed a “cut” of the volumes. For example in Figure 6.10a we projected only the parallelepipeds intersecting the (x_1, x_2) plan passing through $\mathbf{x}_t$.

If no pink zonotope is visible Figure 6.10b, it is because the undisturbed trajectory is too far from the true state and no parallelepiped intersect the (x_2, θ) passing through $\mathbf{x}_t$. In the three planes, we can see that the undisturbed set does not contain the true state of the system. However, the reachable set computed with our method does.

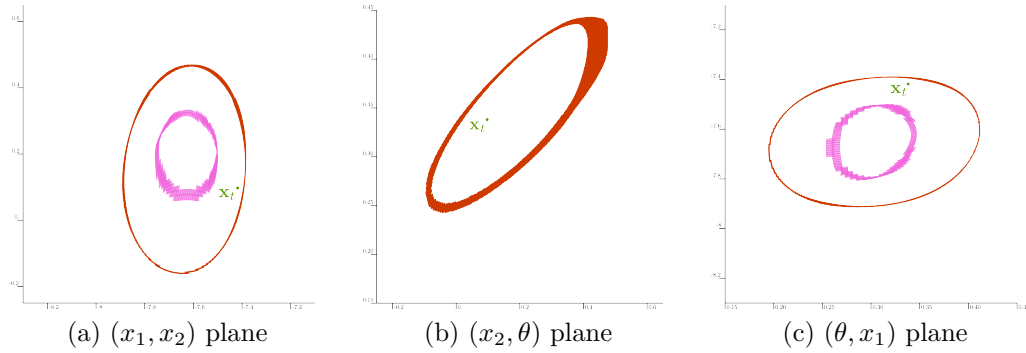


Figure 6.10 – Projections of the reachable set

We showed here that we are able to handle the disturbances of a real system. However, one can notice that some parallelepipeds are notably larger than other (for example Figure 6.10b). This phenomenon could have been limited by a thinner slicing of the charts, but the integration of the sixth-dimensional ODE of this case is expansive in terms of computation time. We will explore a solution to reduce the computation time of the PEIBOS algorithm Chapter 7.

6.4 Extension to non smooth reachable sets

Under the stated assumptions, it may occur that the boundary of the reachable set is not smooth. Our method still applies, and we are able to recover an outer approximation of this boundary.

6.4.1 Contribution

Let us consider the case depicted Figure 6.11 where the boundary $\partial\mathcal{R}_t$ is not smooth and may even be disconnected.

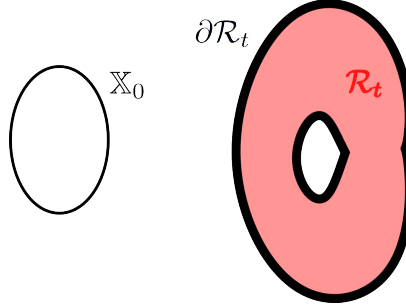


Figure 6.11 – Discontinuous boundary

In this case, the results from the previous subsection is still applicable. Instead of recovering the boundary of the reachable set, the integration of the augmented ODE will provide an outer approximation of it.

8. Let us consider a differential inclusion as defined in Equation 6.1 such that the Assumptions 6.2 to 6.2 are verified. Let us denote by $\mathbb{M}_t$ the manifold obtained by solving the IVP 6.20 and $\mathcal{R}_t$ the reachable set at time t . We then have

$$\partial\mathcal{R}_t \subseteq \Pi_m(\mathbb{M}_t) \quad (6.32)$$

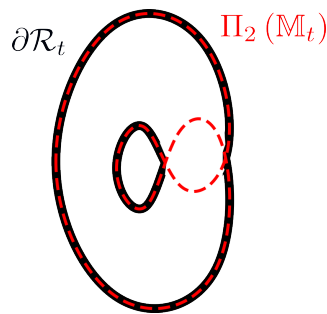


Figure 6.12 – Outer approximation of the reachable set boundary

An illustration of this property is depicted Figure 6.12.

Proof. Let us reason by the absurd.

Let us consider a point $\hat{\mathbf{x}}(T)$ such that

$$\hat{\mathbf{x}}(T) \in \partial \mathcal{R}_T(\mathbb{X}_0) \quad (6.33)$$

$$\hat{\mathbf{x}}(T) \notin \Pi_m(\mathbb{M}_t) \quad (6.34)$$

For topological reasons, we know that this point originated from the boundary of $\mathbb{X}_0$, *i.e.*

$$\hat{\mathbf{x}}(0) = \mathbf{x}_0 \in \partial \mathbb{X}_0 \quad (6.35)$$

Let us denote $(\tilde{\mathbf{x}}, \tilde{\mathbf{q}})$ the unique solution of the IVP 6.20 from the initial conditions

$$\begin{aligned} \tilde{\mathbf{x}}(0) &= \mathbf{x}_0 \\ \tilde{\mathbf{q}}(0) &= \mathbf{n}^{\partial \mathbb{X}_0}(\mathbf{x}(0)) \end{aligned} \quad (6.36)$$

We then have

$$\tilde{\mathbf{x}}(0) = \hat{\mathbf{x}}(0) \quad (6.37)$$

From Equation 6.34 we have

$$\tilde{\mathbf{x}}(T) \neq \hat{\mathbf{x}}(T) \quad (6.38)$$

We can deduce that there exist a time $t_1 \in [0, T]$ such that

$$\tilde{\mathbf{x}}(t_1) = \hat{\mathbf{x}}(t_1) = \mathbf{x}_{t_1} \quad (6.39)$$

$$\forall \epsilon > 0, \tilde{\mathbf{x}}(t_1 + \epsilon) \neq \hat{\mathbf{x}}(t_1 + \epsilon) \quad (6.40)$$

During this time interval $[0, t_1]$, the disturbance $\tilde{\mathbf{w}}$ is applied to the state $\tilde{\mathbf{x}}$ so that $\tilde{\mathbf{x}}(t_1) = \mathbf{x}_{t_1}$. Given the initial state $\mathbf{x}_0$, this disturbance is a fixed trajectory $\tilde{\mathbf{w}} \in L^\infty([0, t_1], \mathbb{W})$.

Now let us consider a ball $\mathbf{B} \subset \mathbb{X}_0$ such that

$$\mathbf{B} \cap \partial \mathbb{X}_0 = \mathbf{x}_0 \quad (6.41)$$

Such ball always exists since $\partial \mathbb{X}_0$ is smooth.

For the time interval $[0, t_1]$, the same disturbance $\tilde{\mathbf{w}}$ is applied to the ball $\mathbf{B}$. The

resulting set is denoted $\mathbf{B}_1$. As this input is purely time-dependant, $\partial\mathbf{B}_1$ remains smooth. In addition,

$$\begin{cases} \mathbf{B}_1 \subset \mathcal{R}_{t_1}(\mathbb{X}_0) \\ \mathbf{B}_1 \cap \partial\mathcal{R}_{t_1}(\mathbb{X}_0) = \mathbf{x}_{t_1} \in \partial\mathbf{B}_1 \end{cases} \quad (6.42)$$

This is summed up in Figure 6.13.

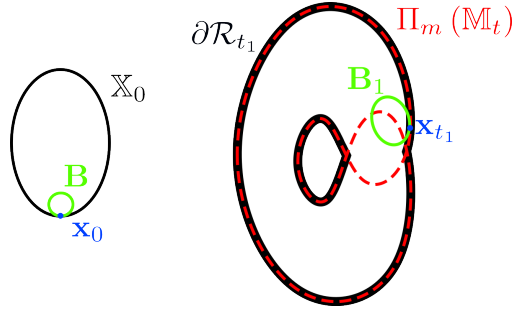


Figure 6.13 – Inclusion of the boundaries at time t_1

Since $\frac{d\mathbf{f}}{d\mathbf{x}}$ is Lipschitz, there exists a constant $K > 0$ such that

$$\forall (x_1, x_2) \in \mathbb{R}^n \times \mathbb{R}^n, x_1 \neq x_2, \left\| \frac{d\mathbf{f}}{d\mathbf{x}}(x_1) - \frac{d\mathbf{f}}{d\mathbf{x}}(x_2) \right\| \leq K \cdot \|x_1 - x_2\| \quad (6.43)$$

Then from Equation 6.21

$$\|\dot{\mathbf{p}}\| \leq K \cdot \|x_1 - x_2\| \cdot \|\mathbf{p}\| \quad (6.44)$$

As done in the previous subsection, we can define $\mathbf{B}_1$ by a value function v . From Equations 6.16 and 6.23, we know that when $\mathbf{q}$ is defined there exists $k_t \in \mathbb{R}$ such that

$$\frac{dv}{d\mathbf{x}}(t) = k_t \cdot \mathbf{p}(t) \quad (6.45)$$

We then have

$$\left\| \frac{dv}{d\mathbf{x}} \right\| \leq K \cdot \|x_1 - x_2\| \cdot \left\| \frac{dv}{d\mathbf{x}} \right\| \quad (6.46)$$

This means that the temporal evolution of the gradient of the value function is limited.

For the ball $\mathbf{B}_1$, $\mathbf{q}$ is well defined. Since $\partial\mathbf{B}_1$ is smooth, this gradient never cancels itself.

From Equation 6.46, we can say that there exist a duration $\epsilon > 0$ such that $\partial\mathcal{R}_\epsilon(\mathbf{B}_1)$ is still smooth, *i.e.* the gradient still never cancels itself.

In this case, the results from the previous subsection gives us

$$\partial\mathcal{R}_\epsilon(\mathbf{B}_1) = \Pi_m(\mathbb{M}_\epsilon(\partial\mathbf{B}_1)) \quad (6.47)$$

By the definitions of the tangent and normal spaces,

$$N_{\mathbf{x}_{t_1}}\partial\mathbf{B}_1 \subset N_{\mathbf{x}_{t_1}}\partial\mathcal{R}_{t_1}(\mathbb{X}_0) \quad (6.48)$$

With Equation 6.42 we then get

$$\tilde{\mathbf{x}}(t_1 + \epsilon) \in \Pi_m(\mathbb{M}_\epsilon(\partial\mathbf{B}_1)) \quad (6.49)$$

In addition from 6.42 we have

$$\hat{\mathbf{x}}(t_1 + \epsilon) \in \partial\mathcal{R}_\epsilon(\mathbf{B}_1) \quad (6.50)$$

This means that both $\tilde{\mathbf{x}}(t_1 + \epsilon)$ and $\hat{\mathbf{x}}(t_1 + \epsilon)$ belong to $\Pi_m(\mathbb{M}_\epsilon(\partial\mathbf{B}_1))$. Under the stated assumptions, the (flow) function mapping each point of $\partial\mathbf{B}_1$ to $\Pi_m(\mathbb{M}_\epsilon(\partial\mathbf{B}_1))$ is a bijection. Since both $\tilde{\mathbf{x}}(t_1 + \epsilon)$ and $\hat{\mathbf{x}}(t_1 + \epsilon)$ originate from the same point $\mathbf{x}_{t_1} \in \partial\mathbf{B}_1$, we have

$$\tilde{\mathbf{x}}(t_1 + \epsilon) = \hat{\mathbf{x}}(t_1 + \epsilon) \quad (6.51)$$

Which contradicts 6.40 and concludes the proof. $\square$

6.4.2 Example

For this application we will note $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ the state of the system. Let us consider the differential inclusion

$$\left\{ \begin{array}{l} \dot{\mathbf{x}}(t) = \begin{pmatrix} x_2(t)^2 + 0.6 \\ 0 \end{pmatrix} + \mathbf{w}(t) \\ \mathbf{x}(0) \in \mathbb{X}_0 \\ \forall t \in [0, T], \mathbf{w}(t) \in \mathbb{W} \end{array} \right. \quad (6.52)$$

We choose $\mathbb{X}_0$ as a disk of radius 0.4 and $\mathbb{W}$ as a disk of radius 0.1, both centered

around $\begin{pmatrix} 0 & 0 \end{pmatrix}^\top$. Assumptions 6.2 to 6.2 are then satisfied. We integrate the ODE over $8s$ and display the result every $4s$.

For each time step we display as shown earlier on Figure 6.6b:

- In pink the integration of $\partial\mathbb{X}_0$ with no disturbance
- In grey 50 integrations of $\partial\mathbb{X}_0$ with a random and constant disturbance
- In red the boundary of the reachable set, computed with our method

The total $8s$ of integration are visible on Figure 6.14.

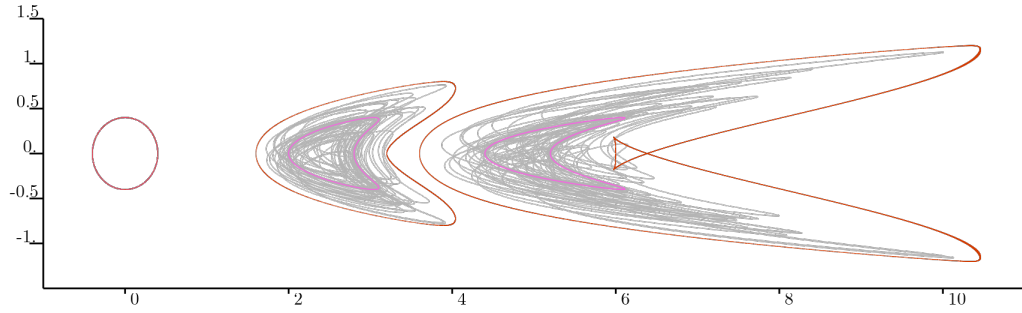


Figure 6.14 – Reachability of the system over $8s$

In this simple example we can clearly see that the projected manifold $\Pi_2(\mathbb{M}_8)$, in red, intersects itself. This is due to the fact that the boundary of the reachable set is not smooth. Note that the manifold itself has no intersection, the phenomenon visible here is an overlap due to the projection.

Since some parts of $\Pi_2(\mathbb{M}_8)$ end up inside of the reachable set, they bring no information about its boundary. They can be removed to obtain the result from Figure 6.15. A method to remove them will be presented later on in Chapter 8.

The undisturbed and disturbed trajectories give a set which is included in the reachable set computed with our method. We can then say that our outer approximation is still valid, even in the non smooth case.

6.5 Conclusion

We displayed a method to compute the inner and outer approximation of a dynamical system reachable set. This method tackles the classical problem of finite-time reachability analysis by the integration of an augmented ODE. This ODE is defined on the boundary of the initial set, and associates a single initial disturbance to each initial state. Its integration over a time t gives a manifold.

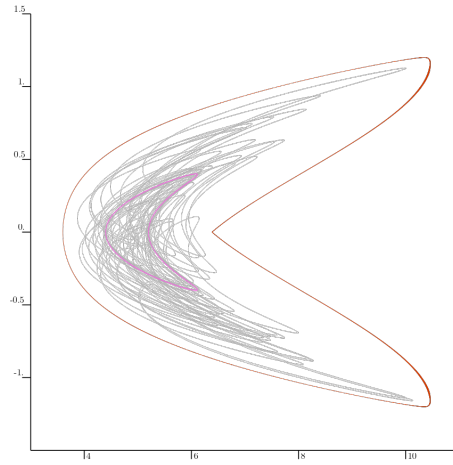


Figure 6.15 – Outer Approximation of the boundary at $t = 8s$

After projection of the resulting manifold, we are able to characterize the exact boundary of the reachable set at time t if it is smooth. Otherwise, we are able to provide an approximation of it. From this outer approximation of the boundary, it is then possible to identify an inner and outer approximation of the reachable set. This result has been displayed on examples in two and three dimensions.

Two problematic arose from these examples. As the method doubles the dimension of the studied system, the integration of the ODE gets expensive in terms of computation time. In addition in the non-smooth case, we observe that fake boundaries appear in the projection of the manifold. These issues will be tackled respectively in Chapter 7 and 8.

CONSTRAINED ADAPTIVE PARALLELEPIPEDIC APPROXIMATION OF THE IMAGE OF A SET BY A NONLINEAR FUNCTION

7.1 Introduction

Consider an initial set $\mathbb{X} \subset \mathbb{R}^n$ and a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. We will denote by $\mathbb{Y} = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathbb{X}\}$ the image set. Let us consider the case depicted in Figure 7.1 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 or the computation of a reachable set as done previously in Chapter 6.

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 chapter 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 7.1 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 7.1a the pessimism of the parallelepipedic approximation (in green) is small enough to not cross the unsafe area. On Figure 7.1b 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. When the function is not explicit but is the flow function of an ODE (see Section 5.4), such method requires numerous integrations of the same ODE, which is a very expensive operation.

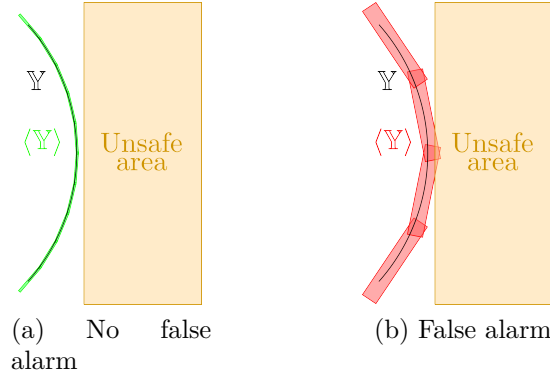


Figure 7.1 – Outer approximation of $\mathbb{Y}$ next to an unsafe area

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 chapter is then to use a low resolution everywhere, and to refine the approximation where needed. This is depicted on Figure 7.2 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.

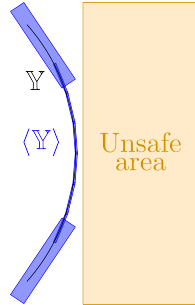


Figure 7.2 – Adaptive parallelepipedic enclosure

7.2 Adaptive parallelepipedic enclosure

This section presents the main contribution of this chapter 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.

7.2.1 Enclosure of the image of a box

Consider a C^1 function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, 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 $\mathcal{L} : \mathbb{R}^m \rightarrow \{0, 1\}$, i.e. we want to ensure that:

$$\forall \mathbf{x} \in [\mathbf{x}], \mathcal{L}(\mathbf{f}(\mathbf{x})) = 1 \quad (7.1)$$

Figure 7.3 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 Definition 24) 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 $\mathcal{L}$.

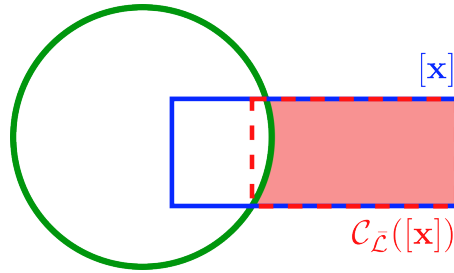


Figure 7.3 – Contractor on the complementary

A classical method consists in directly computing the box $[\mathbf{y}] = [\mathbf{f}]([\mathbf{x}])$ where $[\mathbf{f}]$ is an interval inclusion function of $\mathbf{f}$ [59]. 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 [78] and branch-and-prune [49, 61] algorithms.

The method presented here relies on the same principle. Algorithm 8 presents a step of what will be the *Ad-PEIBOS* algorithm. Its name *Ad-PEIB* stands for **A**daptive **P**arallelepipedic **E**nlcosure of the **I**mage of a **B**ox. 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 7.1 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 subdivided 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 return a parallelepiped enclosing $\mathbf{f}([\mathbf{x}])$ as defined in Definition 33.

Algorithm 8 Adaptive Parallelepiped Enclosure of the Image of a Box (Ad-PEIB)

```

1: Inputs:  $\langle \mathbf{f} \rangle : \mathbb{R}^n \rightarrow \mathbb{P}^m$ ,  $[\mathbf{x}_0] \in \mathbb{R}^n$ ,  $\mathcal{C}_{\bar{\mathcal{L}}}$ ,  $\varepsilon > 0$ ,  $\varepsilon_{lim} > 0$ ,  $L_P$ 
2: Output:  $L_{\langle \mathbf{y} \rangle}$  completed in place
3:
4: function AD-PEIB( $\langle \mathbf{f} \rangle$ ,  $[\mathbf{x}_0]$ ,  $\mathcal{C}_{\bar{\mathcal{L}}}$ ,  $\varepsilon$ ,  $\varepsilon_{lim}$ ,  $L_{\langle \mathbf{y} \rangle}$ )
5:   if  $\varepsilon < \varepsilon_{lim}$  then
6:     Raise an alarm                                      $\triangleright$  Constraint could not be verified
7:   else
8:      $L_{\mathbf{x}} \leftarrow \text{subdivide}([\mathbf{x}_0], \varepsilon)$               $\triangleright$  Boxes of diameter  $\leq \varepsilon$ 
9:     for all  $[\mathbf{x}] \in L_{\mathbf{x}}$  do
10:       $\langle \mathbf{y} \rangle \leftarrow \langle \mathbf{f} \rangle([\mathbf{x}])$ 
11:      if  $\mathcal{C}_{\bar{\mathcal{L}}}(\langle \mathbf{y} \rangle) = \emptyset$  then
12:         $L_{\langle \mathbf{y} \rangle} \leftarrow L_{\langle \mathbf{y} \rangle} \cup \{\langle \mathbf{y} \rangle\}$ 
13:      else
14:        AD-PEIB( $\langle \mathbf{f} \rangle$ ,  $[\mathbf{x}]$ ,  $\mathcal{C}_{\bar{\mathcal{L}}}$ ,  $\varepsilon/2$ ,  $\varepsilon_{lim}$ ,  $L_{\langle \mathbf{y} \rangle}$ )
15:      end if
16:    end for
17:  end if
18: end function

```

At the end of Algorithm 8 if no alarm was raised we obtain a list of parallelepipeds $L_{\langle \mathbf{y} \rangle}$ such that

$$\left\{ \begin{array}{l} \forall \langle \mathbf{y} \rangle \in L_{\langle \mathbf{y} \rangle}, \mathcal{L}(\langle \mathbf{y} \rangle) = 1 \\ \mathbf{f}([\mathbf{x}_0]) \subseteq \bigcup_{\langle \mathbf{y} \rangle \in L_{\langle \mathbf{y} \rangle}} \langle \mathbf{y} \rangle \end{array} \right. \quad (7.2)$$

7.2.2 Enclosure of the image of a set

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. 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 $\partial\mathbb{X}$. As suggested in Subection 5.3.2, gnomonic atlases can be used to make such computations easier.

Algorithm 9 combines the notion of gnomonic atlas and the adaptive parallelepiped enclosure of Algorithm 8. 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 smooth manifold, this algorithm computes the image of $\partial\mathbb{X}$ by a function $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ while asserting that a constraint $\mathcal{L} : \mathbb{R}^m \rightarrow \{0, 1\}$ is satisfied. To do so, the algorithm relies on a gnomonic atlas $\{\psi_0, \Sigma\}$ which satisfies as defined in Subsection 5.3.1 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 8 can then be used on each of these functions. Finally we know that we will compute an outer approximation of $\partial(\mathbf{f}(\mathbb{X}))$.

Algorithm 9 Ad-PEIBOS

```

1: Inputs:  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ ,  $\{\psi_0, \Sigma\}$ ,  $\mathcal{C}_{\bar{\mathcal{L}}}$ ,  $\varepsilon > 0$ ,  $\varepsilon_{\text{lim}} > 0$ 
2: Output:  $L_{\langle \mathbf{y} \rangle}$ , a list of parallelepipeds enclosing the image set
3:
4: function Ad-PEIBOS( $\mathbf{f}, \{\psi_0, \Sigma\}, \mathcal{C}_{\bar{\mathcal{L}}}, \varepsilon, \varepsilon_{\text{lim}}$ )
5:    $L_{\langle \mathbf{y} \rangle} \leftarrow \emptyset$ 
6:   for all  $\sigma \in \Sigma$  do
7:     Ad-PEIB( $\mathbf{f} \circ \sigma \circ \psi_0, [-1, 1]^n, \mathcal{C}_{\bar{\mathcal{L}}}, \varepsilon, \varepsilon_{\text{lim}}, L_{\langle \mathbf{y} \rangle}$ )
8:   end for
9:   return  $L_{\langle \mathbf{y} \rangle}$ 
10: end function

```

7.3 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.

7.3.1 Ad-PEIB

Let us consider an initial box $[\mathbf{x}] \in \mathbb{IR}^n$ with $w([\mathbf{x}])$ its width as defined Definition 7.

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.* $\forall i \in \llbracket 1, n \rrbracket, w([x_i]) = w([\mathbf{x}])$,
- The box is initially not bisected, *i.e.* at the first iteration $\varepsilon = w([\mathbf{x}])$,
- 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 k the smallest integer such that

$$\frac{w([\mathbf{x}])}{2^k} \leq \varepsilon_{\text{lim}} \quad (7.3)$$

Then the value of k is given by

$$k = \left\lceil \log_2 \left(\frac{w([\mathbf{x}])}{\varepsilon_{\text{lim}}} \right) \right\rceil \quad (7.4)$$

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 k .

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 $O(n^i)$.

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

7.3.2 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 $O(n^k)$. The Ad-PEIBOS algorithm then has a time complexity in $O(p \cdot n^k)$.

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

$$\begin{aligned} \log_2 \left(\frac{2}{\epsilon_{\text{lim}}} \right) &= \log_2(2) - \log_2(\epsilon_{\text{lim}}) \\ &= 1 - \log_2(\epsilon_{\text{lim}}) \end{aligned} \tag{7.5}$$

and the expression of k becomes :

$$k = 1 - \lceil \log_2(\epsilon_{\text{lim}}) \rceil \tag{7.6}$$

7.3.3 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 ϵ_{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 could be evaluated on a case-by-case basis, but an adaptive approach is often beneficial.

7.4 Example

Let us now 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 Section 5.2. It defines a parallelepiped inclusion function for any C^1 function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ if $n < m$ which is the case here.

7.4.1 3d robotic arm

Let us consider the 3-dimensional robotic arm described Figure 7.4.

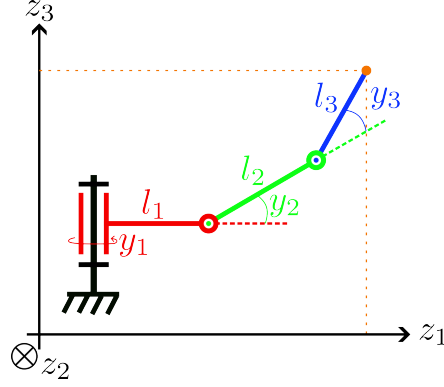


Figure 7.4 – 3D robotic arm

We will study the position of the effector (in orange Figure 7.4). It is defined by

$$\mathbf{z} = \mathbf{f} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \cos(y_1)(l_1 + l_2 \cos(y_2) + l_3 \cos(y_2 + y_3)) \\ \sin(y_1)(l_1 + l_2 \cos(y_2) + l_3 \cos(y_2 + y_3)) \\ l_2 \sin(y_2) + l_3 \sin(y_2 + y_3) \end{pmatrix} \quad (7.7)$$

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

We then define

$$\text{— } \psi_0(\mathbf{x}) = \begin{pmatrix} \frac{\pi}{2} \\ x_1 \cdot \frac{\pi}{2} \\ x_2 \cdot \frac{\pi}{2} \end{pmatrix}$$

— $s_1 = \begin{pmatrix} -2 & 1 & 3 \end{pmatrix}$ the rotation of $\frac{\pi}{2}$ with respect to the y_3 axis

— $s_2 = \begin{pmatrix} 3 & 2 & -1 \end{pmatrix}$ the rotation of $\frac{\pi}{2}$ with respect to the y_2 axis

For this application we will consider that our robotic arm needs to operate in a cylinder as depicted in Figure 7.5, without touching it. We then get two constraints :

$$\begin{cases} \mathcal{L}_1(\mathbf{z}) = \left(\sqrt{z_2^2 + z_3^2} < 1.2 \right) & \text{(radius of the cylinder)} \\ \mathcal{L}_2(\mathbf{z}) = (z_1 < 1.055) & \text{(depth of the cylinder)} \end{cases} \quad (7.8)$$

The resulting constraint is then $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$.

Note that in this application the criterium from Chapter 5 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

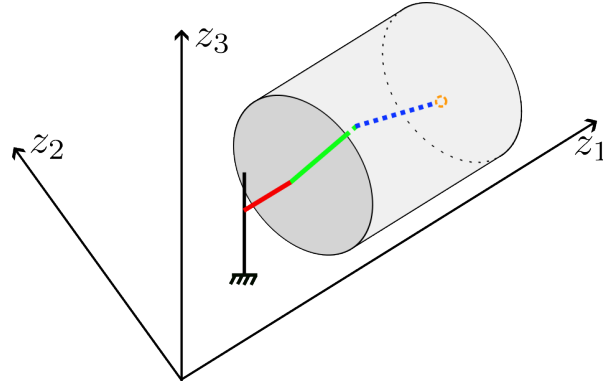


Figure 7.5 – Robotic arm in a cylinder

are aligned, i.e. when $y_3 = 0$. This result can be found by searching for solutions to the equation

$$\det \left(\frac{d\mathbf{f}}{d\mathbf{y}}(\mathbf{y}) \right) = 0 \quad (7.9)$$

where $\det$ is the determinant operator. This singularity is depicted on Figure 7.6. Figure 7.6b 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 7.6a where we can easily visualise how this singularity is generated by the alignment of the two last segments of the arm.

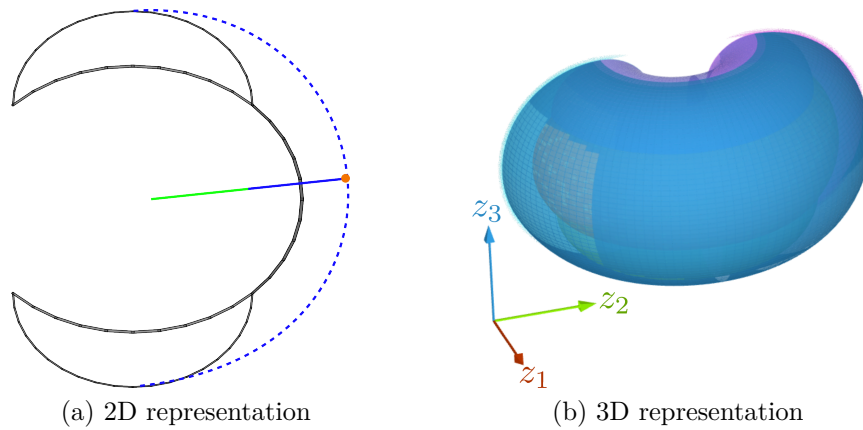


Figure 7.6 – Singularity in blue in the robotic arm workspace

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

$$\psi(\mathbf{x}) = \begin{pmatrix} x_1 \cdot \frac{\pi}{2} \\ x_2 \cdot \frac{\pi}{2} \\ 0 \end{pmatrix} \quad (7.10)$$

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 $\{\psi_0, \{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\}\}$. The result is then completed by using the Ad-PEIB algorithm on the function $\mathbf{f} \circ \psi$.

Figure 7.7 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($\mathbf{f}$, $\{\psi_0, \{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\}\}$, $\mathcal{C}_{\tilde{\mathcal{L}}}$, 1.0, 0.01) and Ad-PEIB($\mathbf{f} \circ \psi$, $[-1, 1]^2, \mathcal{C}_{\tilde{\mathcal{L}}}$, 1.0, 0.01, $\{\}$)

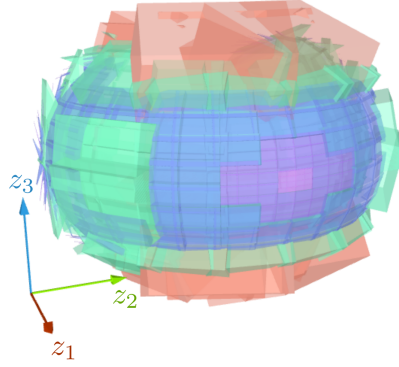


Figure 7.7 – Adaptive outer approximation of the arm workspace

On Figures 7.8 and 7.9, 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 7.8 that we are able to satisfy both constraints, while refining only the necessary areas. The smallest resolution used to get this result is $\varepsilon_{min} = 0.03125$.

To compare with the standard implementation, Figure 7.9 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 0.3s to 7s. This means that our method allowed us to divide the computation time by a factor 20.

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 7.10 shows that we are able to detect the unsafe arm positions, in red Figure 7.10b. Note that since the problem is symmetric, red parallelepiepds also

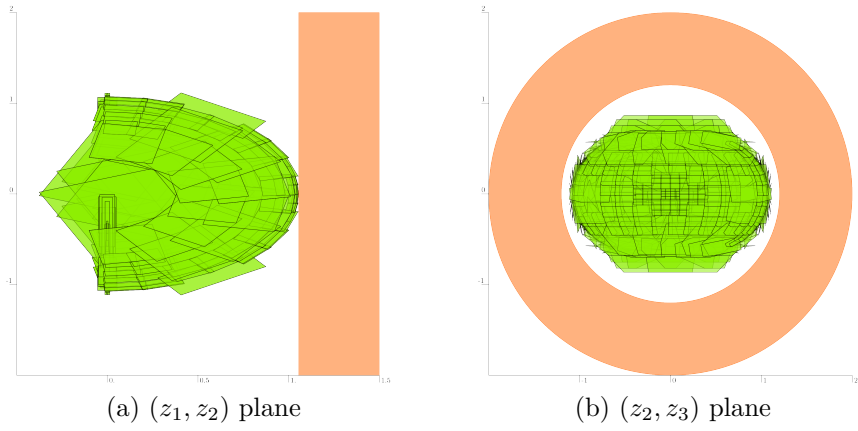


Figure 7.8 – Constraints validated thanks to an adaptive slicing

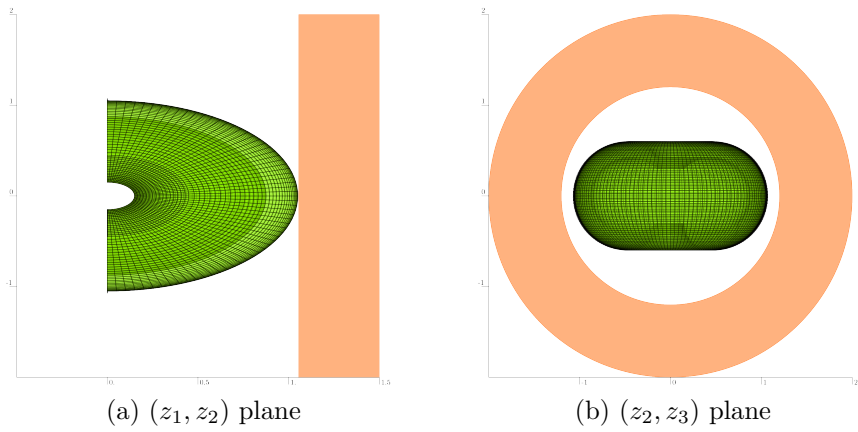


Figure 7.9 – Fine workspace

appear on the other side of the volume (not visible here). In Figure 7.10a, we can see in red the configurations that led to the unsafe positions of the robotic arm.

This property of the adaptive parallelepiped inclusion will be the subject of Section 7.5.

7.4.2 Area reaching

As done before, the method of this Chapter can be used to propagate sets through an ODE. For this application, we will consider the dynamic of the Dubins car :

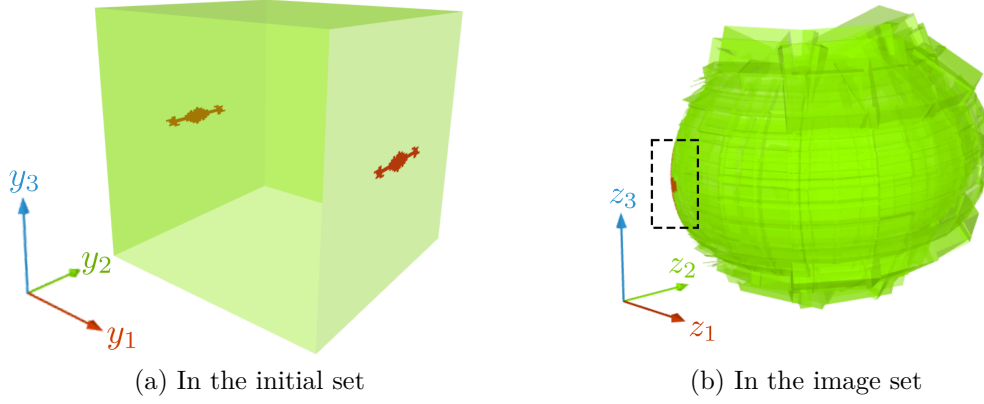


Figure 7.10 – Detection of the unsafe configurations

$$\dot{\mathbf{x}} = \beta(\mathbf{x}, u) = \begin{pmatrix} \cos(x_3) \\ \sin(x_3) \\ u \end{pmatrix} \quad (7.11)$$

With $u(t) = \cos(2\pi t)$. We can then rewrite Equation 7.11

$$\dot{\mathbf{x}} = \gamma(t, \mathbf{x}) = \begin{pmatrix} \cos(x_3) \\ \sin(x_3) \\ \cos(2\pi t) \end{pmatrix} \quad (7.12)$$

We can then define a new system state as :

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ t \end{pmatrix} \quad (7.13)$$

This new system follows an equation of the form :

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) = \begin{pmatrix} \gamma(t, \mathbf{x}) \\ 1 \end{pmatrix} \quad (7.14)$$

We will denote ϕ the flow associated to this ODE. We can use this formulation to compute the set reachable by the system at any time $t \in \mathbb{R}^+$ noted $\mathbb{X}_t$. Note that in this case the input is a simple trajectory which simplifies the computation of the reachable set.

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

then define

$$\begin{aligned} \text{--- } \psi_0(\mathbf{b}) &= 0.02 \cdot \begin{pmatrix} \frac{1}{\sqrt{1+b_1^2+b_2^2}} \\ \frac{b_1}{\sqrt{1+b_1^2+b_2^2}} \\ \frac{b_2}{\sqrt{1+b_1^2+b_2^2}} \\ 0 \end{pmatrix} \\ \text{--- } s_1 &= \begin{pmatrix} -2 & 1 & 3 & 4 \end{pmatrix} \text{ the rotation of } \frac{\pi}{2} \text{ with respect to the } x_3 \text{ axis} \\ \text{--- } s_2 &= \begin{pmatrix} 3 & 2 & -1 & 4 \end{pmatrix} \text{ the rotation of } \frac{\pi}{2} \text{ with respect to the } x_2 \text{ axis} \end{aligned}$$

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 7.11. We then define the constraint

$$\mathcal{L}(\mathbf{y}) = \left(\sqrt{(a_1 - y_1)^2 + (a_2 - y_2)^2} < r \right) \quad (7.15)$$

For numerical applications we will consider the constraint of a circle of center $\mathbf{a} = (1, 0)$ and radius $r = 0.03$.

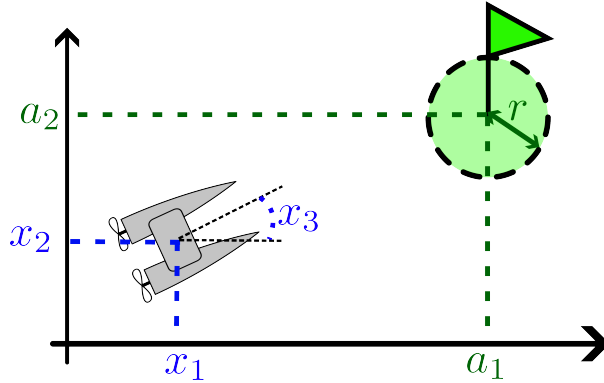


Figure 7.11 – Area reaching

Figure 7.12 shows the result of a naive parallelepipedic approximation of the state of the Dubins car at 8 time points in $[0, 1]$ for $\varepsilon = 0.25$ in the (x_1, x_2, x_3) space.

To get the set reached by the car after $1s$, noted $\mathcal{R}_1$, we then perform the algorithm $\text{Ad-PEIBOS}(\phi, \{\psi_0, \{1, s_1, s_1^2, s_1^{-1}, s_2, s_2^{-1}\}\}, \mathcal{C}_{\bar{\mathcal{L}}}, 0.5, 0.01)$. Figure 7.13 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.

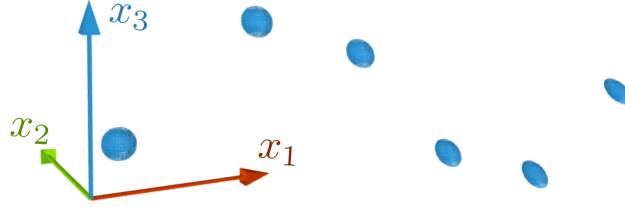


Figure 7.12 – Evolution of the Dubins car

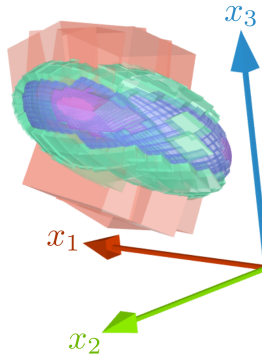


Figure 7.13 – Adaptive outer approximation of $\mathcal{R}_1$

Figure 7.14 shows the projection of $\mathcal{R}_1$ 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 18s and the smallest resolution needed was $\varepsilon_{min} = 0.03125$. The result on the right was obtained in 130s by using the resolution ε_{min} everywhere. In this example the use of the adaptive slicing divided the computation time by a factor 7.

7.5 Inner and outer approximation of the unsafe areas

In the case where the constraints can not be validated, the result of the Ad-PEIBOS algorithm can be used to get both an inner and outer approximation of the unsafe areas.

Consider that we have an initial set $\mathbb{X}$ and we want to compute its image by the

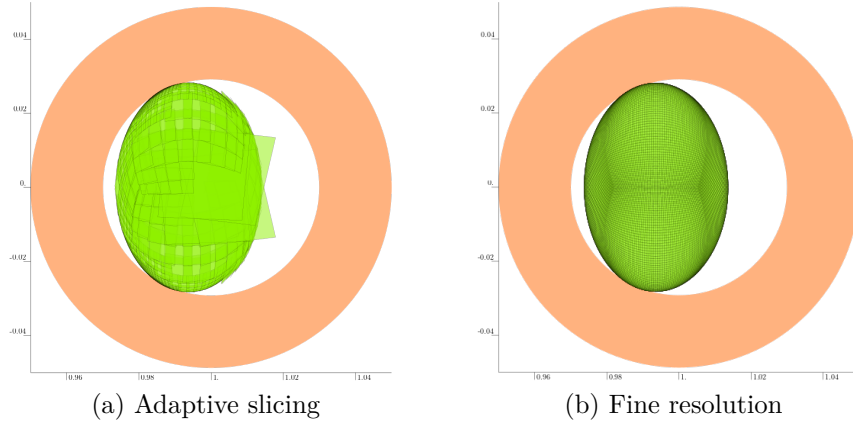


Figure 7.14 – Projected parallelepipedic approximation of $\mathcal{R}_1$ with (left) and without (right) adaptive slicing

function $\mathbf{f}$. As suggested in Subsection 2.2.6, constraints can be seen as a set $\mathbb{Y}$. The validation problem presented here comes down to asserting that

$$\mathbf{f}(\mathbb{X}) \cap \overline{\mathbb{Y}} = \emptyset \quad (7.16)$$

If the constraint can not be validated, *i.e.* $\mathbf{f}(\mathbb{X}) \cap \overline{\mathbb{Y}} \neq \emptyset$, it is possible to get both an inner and an outer approximation of this set, further noted $\mathbb{S}$.

For illustration purpose, let us consider the example from Subsection 5.3.3 where $\mathbb{X}$ is the unit disk and $\mathbf{f}$ is defined by Equation 5.65. For the constraint, we will consider $\mathbb{Y}$ the disk of radius $r = 1.75$ centered around $\begin{pmatrix} 0.5 & 0 \end{pmatrix}$. Figure 7.15 shows both $\mathbf{f}(\partial\mathbb{X})$ computed with PEIBOS (in black) and $\partial\mathbb{Y}$ (in blue). We can see that parts of $\mathbf{f}(\mathbb{X})$ are outside of $\mathbb{Y}$.

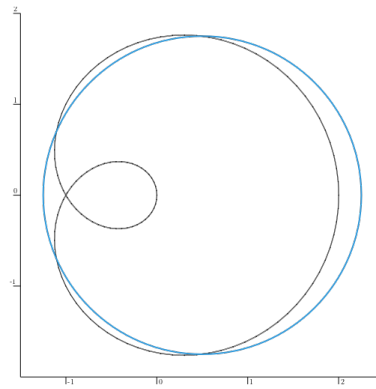


Figure 7.15 – The constraint is not validated, $\mathbb{S} \neq \emptyset$

A standard way to this in Interval Analysis is to look for an inner and an outer approximation of $\mathbf{f}^{-1}(\mathbb{S})$ in this initial space. To do so, let us introduce two functions $\mathbf{c}$ and $\mathbf{g}$ which respectively handle the sets $\mathbb{X}$ and $\mathbb{Y}$. In our case we have

$$\mathbf{c}(\mathbf{x}) = \sqrt{x_1^2 + x_2^2} - 1 \quad (7.17)$$

$$\mathbf{g}(\mathbf{y}) = \sqrt{(y_1 - 0.5)^2 + y_2^2} - 1.75 \quad (7.18)$$

Meaning that a point $\mathbf{x} \in \mathbb{R}^2$ belongs to $\mathbb{X}$ if $\mathbf{c}(\mathbf{x}) < 0$, and a point $\mathbf{y} \in \mathbb{R}^2$ belongs to $\mathbb{Y}$ if $\mathbf{g}(\mathbf{y}) < 0$. Looking for points in $\mathbf{f}^{-1}(\mathbb{S})$ then comes down to look for point $\mathbf{x} \in \mathbb{R}^2$ such that

$$\begin{cases} \mathbf{c}(\mathbf{x}) \leq 0 \\ \mathbf{g}(\mathbf{f}(\mathbf{x})) > 0 \end{cases} \quad (7.19)$$

In other words, we look for points in $\mathbb{X}$ whose image by $\mathbf{f}$ are outside of $\mathbb{Y}$. This can be done by a paving algorithm as presented in Subsection 2.2.5 to get the result from Figure 7.16 where we get both an inner and an outer approximation of $\mathbf{f}^{-1}(\mathbb{S})$. Since we were able to find an inner approximation, we are certain that $\mathbf{f}^{-1}(\mathbb{S})$ is not empty and so is $\mathbb{S}$, as expected.

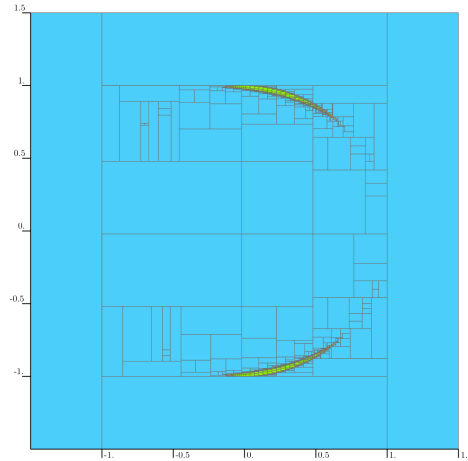


Figure 7.16 – Inner (green) and outer (green + yellow) approximations of $\mathbf{f}^{-1}(\mathbb{S})$

From there we can observe that the yellow boxes represent an outer approximation of the boundary $\partial\mathbf{f}^{-1}(\mathbb{S})$. Computing their image by $\mathbf{f}$ then provides an outer approximation of $\partial\mathbb{S}$ as visible Figure 7.17

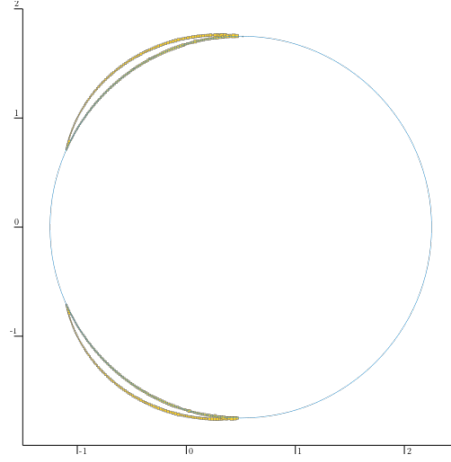


Figure 7.17 – Outer approximation of $\partial\mathbb{S}$

As done previously, this result can be interpreted by the CGAL library as a polygon with holes to get both an inner and outer approximation of $\mathbb{S}$ itself. The result of this step is shown Figure 7.18

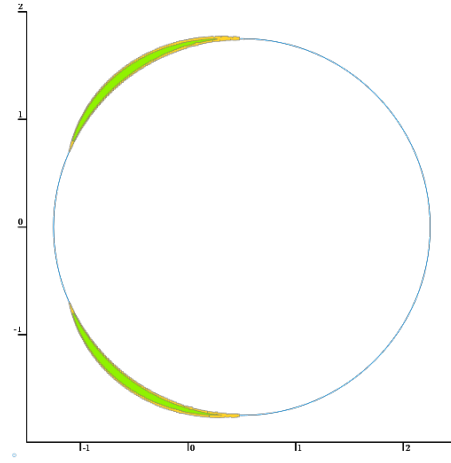


Figure 7.18 – Inner (green) and outer (green + yellow) approximation of $\mathbb{S}$ with the inversion method

Now if we want to compare this method with ours, we first need to use the Ad-PEIBOS algorithm. The result of the algorithm is shown Figure 7.19 in green, for the valid parallelepipeds, and in red for the invalid ones. For graphical purposes the boundary of $\mathbb{Y}$ is once again drawn in blue. As expected, an alarm is raised.

Since the algorithm provides an outer approximation of the boundary $\partial\mathbf{f}(\mathbb{X})$, we can easily get both an outer and an inner approximation of $\mathbf{f}(\mathbb{X})$ as shown Figure 7.20a. With

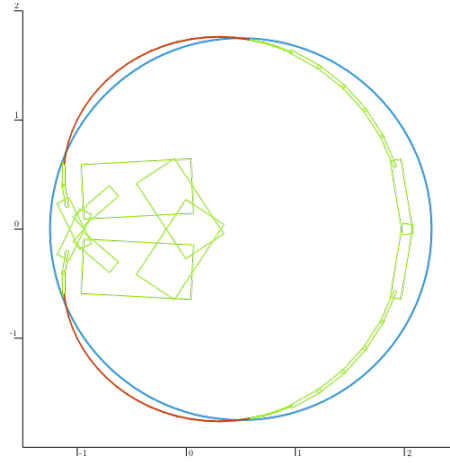


Figure 7.19 – Output of the Ad-PEIBOS algorithm

an quick use of the classical PEIBOS algorithm, we can also get the result of Figure 7.20b which is an inner and an outer approximation of $\mathbb{Y}$.

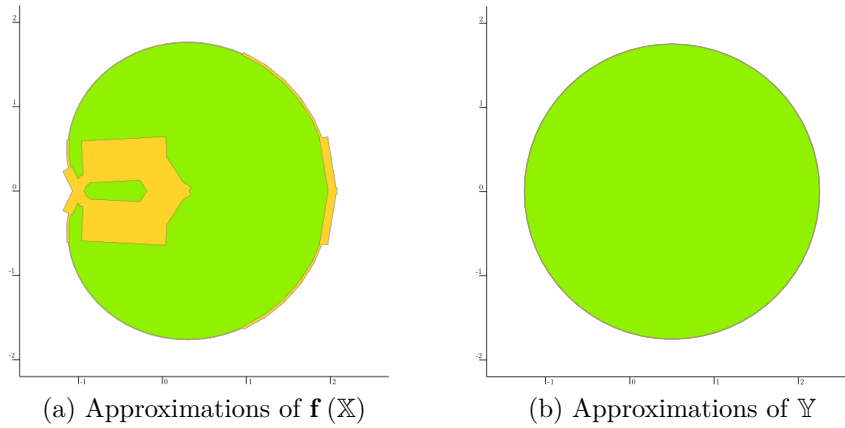


Figure 7.20 – Inner (green) and outer (green + yellow) approximation of both sets

With both results, we can easily get an inner and an outer approximation of $\mathbb{S}$. As illustrated by the Figures 7.21 and 7.22 , we have :

$$\begin{cases} \mathbb{S}^- = \mathbf{f}(\mathbb{X})^- / \mathbb{Y}^+ \\ \mathbb{S}^+ = \mathbf{f}(\mathbb{X})^+ / \mathbb{Y}^- \end{cases} \quad (7.20)$$

We finally get the result from Figure 7.23 which is similar to the one from Figure 7.18.

To be able to compare both result, Table 7.1 summarizes the time taken by both methods and the areas of the resulting inner and outer approximations. In this example

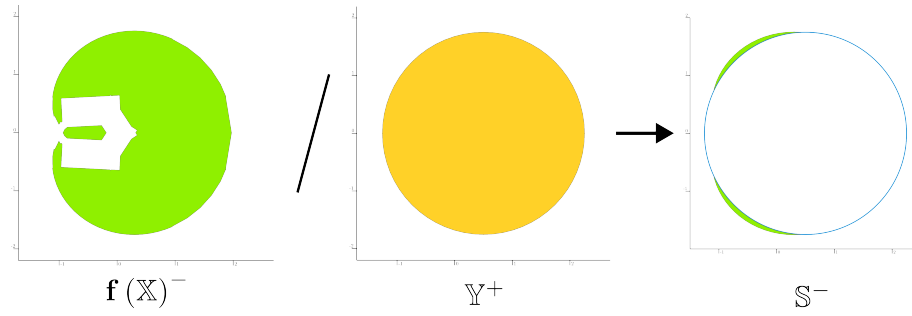


Figure 7.21 – Construction of the inner approximation

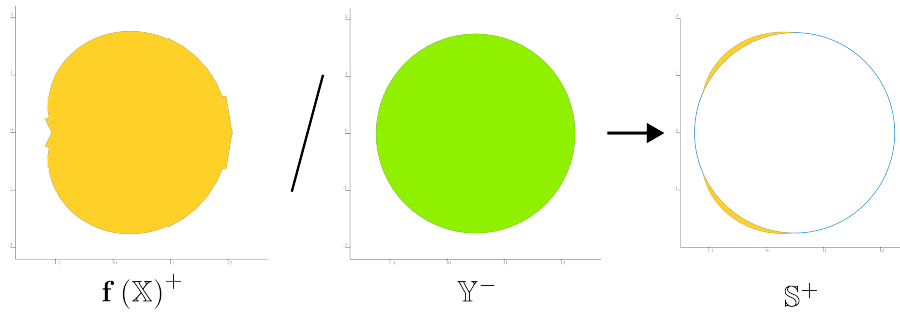


Figure 7.22 – Construction of the outer approximation

we can see that our method is faster, and provides both a better, *i.e.* larger, inner approximation and a better, *i.e.* smaller, outer approximation.

	inversion based	Ad-PEIBOS based
time (s)	0.3	0.13
Area of $\mathbb{S}^-$	0.176	0.203
Area of $\mathbb{S}^+$	0.295	0.264

Table 7.1 – Performances comparison : time and areas

This result encourages us to think that this method could generally be more efficient to compute both approximation of the set $\mathbb{S}$ if it exists. It has to be noted that the result shown here relies on a single example. The same process could be used in other configurations, maybe in higher dimensions, to ensure the interest of this approach.

7.6 Conclusion

In this chapter we proposed an algorithm Ad-PEIB to compute a constrained adaptive outer approximation for a set $\mathbb{Y}$ defined as the image of a $\text{box}[\mathbf{x}]$ by a nonlinear function

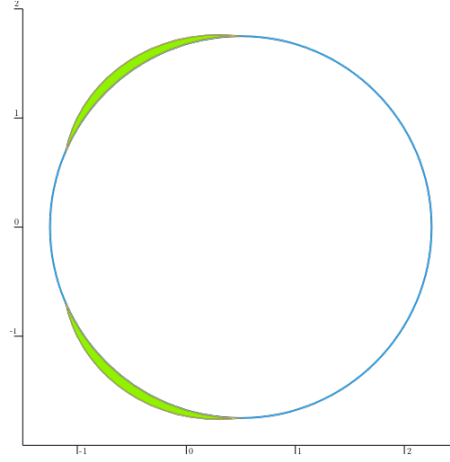


Figure 7.23 – Inner (green) and outer (green + yellow) approximation of $\mathbb{S}$ with Ad-PEIBOS

$\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. 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 part of the initial set is responsible and refine it. This could also be a step towards a parallelepiped-based set inversion algorithm.

The displayed extension of this work to the propagation of an initial set through an ODE makes a direct link with Chapter 6. The Ad-PEIBOS algorithm could be used for the reachability analysis to reduce the computation time of our method. However, the problematic of fake boundaries in the outer approximation of the boundary remains (for example in the robotic arm workspace). The next chapter provides a solution to remove them.

REMOVAL OF FAKE BOUNDARIES IN BOUNDARY BASED APPROACHES

8.1 Introduction

The methods to calculate the image of a set by a function can be divided into two subclasses. The set-based methods compute directly an outer, and sometimes an inner, approximation of the image of the whole set. They can rely on Interval Analysis for the operations on sets. The boundary-based methods compute instead the image of the boundary of the set, and from this image an inner and outer approximation of the image set can be deduced. These methods are lighter to calculate, but they are subject to the appearance of fake boundaries in the image set.

These fake boundaries are the result of the difference between the boundary of the image set, and the image of the boundary. These fake boundaries are parts of the image of the boundary that are in facts inside of the image set and should be classified as such. As they add an unwanted pessimism to the estimation of the inner approximation of the image set, these fake boundaries have to be removed.

This problematic has often been addressed as different applications favor different solutions. The papers [128, 18] both expose the fact that the result obtained when using interval methods depends on the way the problem itself is formulated. For example the union of adjacent but non-overlapping sets is subject to the appearance of fake boundaries, whereas a reformulation of the problem can avoid this issue. In [128] a first solution is presented by using a DNF (Disjunctive Normal Form) / CNF (Conjunctive Normal Form) to prevent the fake boundaries from appearing. The paper [18] is a direct continuation of this work and proposes to rely on Karnaugh maps [63] to highlight how an outer approximation of the boundary can be constructed part by part. From this outer approximation of the boundary without the fake boundaries, an inner and outer approximation of the whole set can be obtained.

The problematic of fake boundaries can also be seen from a topological point of view. In [127], the notion of winding number is used to detect the fake boundaries. More precisely, it extends the works from [121] and [36] in order to compute the winding number, i.e. the topological degree, of a whole area. In the context of a coverage measure, the winding number represents how many times a given area was seen. The boundaries then have an interval of winding number, for example $[0, 1]$ if the boundary is between an area seen one time and an unseen area. Fake boundaries can finally be spotted as their winding number does not contain 0.

The method presented in this chapter is a continuation of these works as the problem considered here can easily be seen as the computation of a covered or visible area without fake boundaries. It builds up on the work presented in [40] and extends it to higher dimension. In addition, the notion of Box Chains from [40] will be extended to parallelepipeds.

8.2 Problem presentation

Consider a set $\mathbb{X} \subset \mathbb{R}^m$ whose boundary $\partial\mathbb{X}$ is a smooth manifold. for graphical purposes, we will use $\mathbb{X} = D^2$ the unit disk and $\partial\mathbb{X} = S^1$ the unit circle.

As computing the image of the whole set $\mathbb{X}$ can give a result too pessimistic to be usable, it is often preferred to compute the image of its boundary $\partial\mathbb{X}$ instead. For this section we will consider functions of the form $\mathbf{f} : \mathbb{X} \rightarrow \mathbb{R}^m$ that satisfies :

- $\mathbf{f}$ is C^1 , we will note $\frac{d\mathbf{f}}{dx}$ its Jacobian function.
- $\mathbf{f}$ has no singularity on $\mathbb{X}$, i.e. the determinant of its Jacobian is never zero

Thanks to these assumptions, cases where the image of the set contains a cusp or a fold, as depicted Figure 8.1, will not be considered.

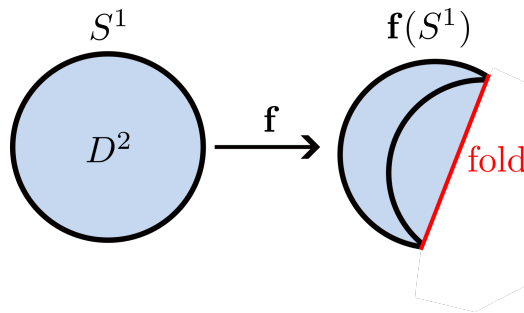


Figure 8.1 – Fold in the image of the set

As depicted Figure 8.2a for the case of the unit disk, we have

$$\partial \mathbf{f}(\mathbb{X}) \subseteq \mathbf{f}(\partial \mathbb{X}) \quad (8.1)$$

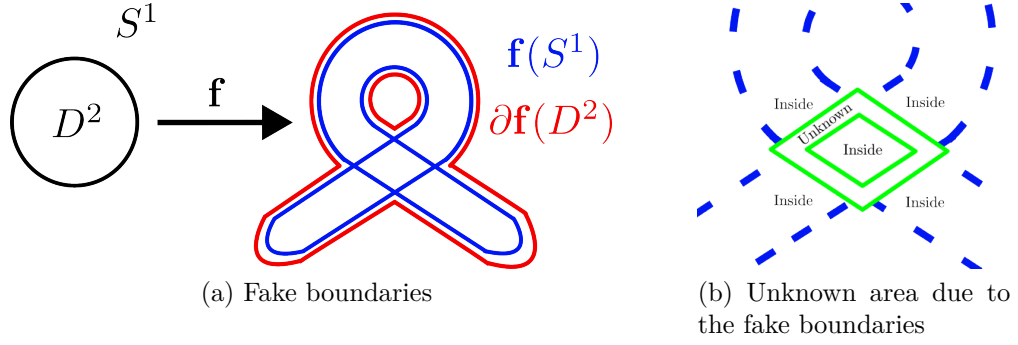


Figure 8.2 – Pessimism induced by the fake boundaries

Proof. Consider a function $\mathbf{f} : \mathbb{X} \rightarrow \mathbb{R}^m$. Under the assumptions cited earlier, we can say that $\mathbf{f}$ is locally bijective at every point of $\mathbb{X}$ and is an open mapping. This mean that the image of an open set by $\mathbf{f}$ is an open set:

$$\forall \mathbf{x} \in \mathbb{X}, \mathbf{x} \notin \partial \mathbb{X} \Rightarrow \mathbf{f}(\mathbf{x}) \notin \partial \mathbf{f}(\mathbb{X}) \quad (8.2)$$

Then,

$$\forall \mathbf{y} \in \mathbb{R}^m, \mathbf{y} \in \partial \mathbf{f}(\mathbb{X}) \implies \exists \mathbf{x} \in \partial \mathbb{X}, \mathbf{y} = \mathbf{f}(\mathbf{x}) \quad (8.3)$$

Finally, all the points of $\partial \mathbf{f}(\mathbb{X})$ are points of $\mathbf{f}(\partial \mathbb{X})$. $\square$

Calculating the image of $\partial \mathbb{X}$ instead of the image of $\mathbb{X}$ makes fake boundaries appear. These fake boundaries are defined by:

$$FB_{\mathbf{f}}(\mathbb{X}) = \mathbf{f}(\partial \mathbb{X}) \setminus \partial \mathbf{f}(\mathbb{X}) \quad (8.4)$$

When computing the image of a set, we often want to compute both an inner and an outer approximation of the image set. Given $\partial \mathbf{f}(\mathbb{X})$ it is possible to compute the inner and the outer approximation of the image set. However using $\mathbf{f}(\partial \mathbb{X})$ will give a more pessimistic result as the neighborhood of $FB_{\mathbf{f}}(\mathbb{X})$ will be considered as part of the

boundary as depicted on Figure 8.2b. The objective of this chapter is to present how we can automatically detect and remove these fake boundaries.

8.3 Box chains to detect self intersections

8.3.1 Hypothesis

The studied function will be supposed to satisfy the properties exposed in Section 8.2.

As stated in the introduction of the chapter, the contribution presented here is a generalization of the work presented in [40]. This paper was only addressing the case of the unit disk and its boundary. Here we will consider any compact set $\mathbb{X} \subset \mathbb{R}^m$ whose boundary is a smooth n -manifold. For this chapter, we will suppose that a box atlas can be constructed to cover $\partial\mathbb{X}$.

Denote $\{(U_i, \varphi_i)\}_{i \in I}$ a box atlas of $\partial\mathbb{X}$, we will further denote $\psi_i = \varphi_i^{-1}$ the inverse charts (possible since the φ_i are homeomorphisms). The charts $V_i = \varphi_i(U_i)$ are boxes of $\mathbb{R}^n$, and we can introduce

$$\begin{aligned} \mathbf{g}_i : V_i &\rightarrow \mathbb{R}^m \\ \mathbf{x} &\mapsto \mathbf{f}(\psi_i(\mathbf{x})) \end{aligned} \tag{8.5}$$

instead of studying $\mathbf{f}$ on $\partial\mathbb{X}$, we will then study each of the $\mathbf{g}_i$ functions on their domain of definition $V_i \in \mathbb{R}^n$.

8.3.2 Definitions

To define the notion of Box Chain we first need to introduce the relation of neighborhood.

Definition 59. Let $[\mathbf{x}_i] \in \mathbb{R}^n$ and $[\mathbf{x}_j] \in \mathbb{R}^n$ be two boxes. We define the neighborhood relation noted R_n between $[\mathbf{x}_i]$ and $[\mathbf{x}_j]$ as :

$$[\mathbf{x}_i] R_n [\mathbf{x}_j] \Leftrightarrow [\mathbf{x}_i] \cap [\mathbf{x}_j] \neq \emptyset \tag{8.6}$$

This relation can be represented for real-value intervals in the t-plane [110] or by its logical matrix as shown Figure 8.3.

The t-plane represents real-value intervals $[0, x_1], [x_1, x_2], \dots$ on the abscissa and on the ordinate. The grid is then colored where the corresponding intervals verify the relation,

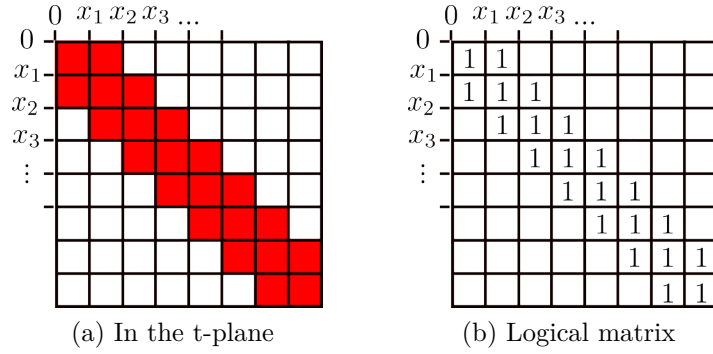


Figure 8.3 – Neighborhood Relation

and is left blank otherwise. The resulting grid can be interpreted as a logical matrix, say M , where the blank boxes are null, and the colored boxes are ones. This graphical representation of the relation highlights its properties:

- **Reflexivity** As the main diagonal of the grid has no blank box, i.e. the identity matrix is included in M , it means that the relation is reflexive
- **Symmetry** As M is symmetric, the relation itself is symmetric

However we can see that $(x_i \cap x_j \neq \emptyset) \wedge (x_j \cap x_k \neq \emptyset) \not\Rightarrow x_i \cap x_k \neq \emptyset$, meaning that this relation is not transitive.

From this definition, we can define the Box Chain relation.

Definition 60. Let $[\mathbf{x}_i] \in \mathbb{IR}^n$ and $[\mathbf{x}_k] \in \mathbb{IR}^n$ be two boxes and $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ a C^1 function. We define the Box Chain relation for $\mathbf{g}$, noted $R_{BC,\mathbf{g}}$ between $[\mathbf{x}_i]$ and $[\mathbf{x}_k]$, as :

$$[\mathbf{x}_i] R_{BC,\mathbf{g}} [\mathbf{x}_k] \iff \exists m \in \mathbb{N}, [\mathbf{x}_{j_1}], \dots, [\mathbf{x}_{j_m}] \in \mathbb{IR}^m, \begin{cases} [\mathbf{x}_i] R_n [\mathbf{x}_{j_1}] \wedge \dots \wedge [\mathbf{x}_{j_m}] R_n [\mathbf{x}_k] \\ \mathbf{g} \text{ is injective on } [\mathbf{x}_i] \cup [\mathbf{x}_{j_1}] \cup \dots \cup [\mathbf{x}_{j_m}] \cup [\mathbf{x}_k] \end{cases} \quad (8.7)$$

Equivalently, $[\mathbf{x}_i]$ and $[\mathbf{x}_k]$ are in Box Chain relation for $\mathbf{g}$ if a trajectory from neighbor to neighbor exists between $[\mathbf{x}_i]$ and $[\mathbf{x}_k]$ on which $\mathbf{g}$ is injective.

8.3.3 Box Chain decomposition

Box Chain decomposition algorithm

As depicted in Figure 8.2a, fake boundaries appear when the considered boundary crosses itself. This means that the studied function is not globally injective as two distinct inputs give the same output.

Since we can not exactly compute the image of a box by a function, we need to consider an inclusion function as explained in Subsection 2.2.3. We will then get a set of boxes which enclose $\mathbf{g}_i(V_i)$. Finding the self intersections in the contour can then be hard as each box of the contour crosses at least two other boxes as shown in the two dimensional case Figure 8.4. This is the result of the continuity of the function $\mathbf{g}_i$, and this phenomenon gets worse in higher dimension.

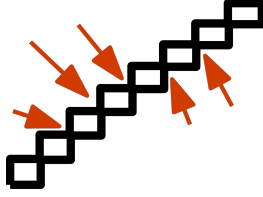


Figure 8.4 – Intersections between the boxes

To make this detection easier, Box Chains can be used to decompose the domain of any function into subsets on which the function is locally injective. Looking for the self intersections in the boundary will then come down to looking for the intersections between different Box Chains. Algorithm 10 is suggested to perform this decomposition. It takes three inputs :

- The studied function $\mathbf{g}_i$
- An injectivity criterion h
- A cover of V_i , C

The injectivity criterion is defined over the powerset of V_i , $\mathcal{P}(V_i)$. For a given subset of V_i it outputs 1 if $\mathbf{g}$ is injective over this subset, 0 otherwise. This criterion must be sufficient, but does not need to be necessary. Example of injectivity criterium will be given in Subection 8.3.3.

The algorithm takes the element of the cover C one by one and try to group them into Box Chains. As soon as an element can not be added to the Box Chain without losing the injectivity of $\mathbf{g}$ on it, a new Box Chain is created. Finally the algorithm sorts the boxes of the cover C into a list of Box Chains.

To get the best result possible, i.e. as few Box Chains as possible, the cover C should be sorted in such a way that each element is in neighborhood relationship with the following one. In addition, the popped element should be the first or the last of the list. In terms of complexity the tests $h(\mathcal{L}_B \cup \{\mathbf{x}\})$ and $\mathcal{L}_B R_n \mathbf{x}$ require $O(k)$ operations, k being the number of elements in the list $\mathcal{L}_B$. These tests then have a complexity in $O(k)$.

Algorithm 10 Box Chain decomposition

```
1: Inputs:  $\mathbf{g}_i : V_i \rightarrow \mathbb{R}^m$ ,  $h : \mathcal{P}(Vi) \rightarrow \{0, 1\}$ ,  $C$ 
2: Output:  $\mathcal{L}_{BC}$ , a list of Box Chains
3:
4: function BC-DECOMPOSITION( $\mathbf{g}_i, h, C$ )
5:    $\mathcal{L}_W \leftarrow C$ 
6:    $\mathcal{L}_{BC} \leftarrow \emptyset$ 
7:   Pop  $\mathbf{x}$  from  $\mathcal{L}_W$ 
8:   while  $\mathcal{L}_W \neq \emptyset$  do
9:      $\mathcal{L}_B \leftarrow \{\mathbf{x}\}$ 
10:    while  $\mathcal{L}_W \neq \emptyset$  do
11:      Pop  $\mathbf{x}$  from  $\mathcal{L}_W$ 
12:      if  $h(\mathcal{L}_B \cup \{\mathbf{x}\})$  and  $\mathcal{L}_B R_n \mathbf{x}$  then ▷ Injectivity criterion
13:         $\mathcal{L}_B \leftarrow \mathcal{L}_B \cup \{\mathbf{x}\}$ 
14:      else
15:        break
16:      end if
17:    end while
18:     $\mathcal{L}_{BC} \leftarrow \mathcal{L}_{BC} \cup \{\mathcal{L}_B\}$ 
19:  end while
20:  return  $\mathcal{L}_{BC}$ 
21: end function
```

The worst case happens when all the elements can be added to the same Box Chain except the last one. The tests mentioned earlier are then runned with $k \in \llbracket 1, p-1 \rrbracket$ elements where p is the number of elements in the cover.. Finally, the complexity of the algorithm in the worst case is $O(p^2)$.

Injectivity criterium

In [40], an injectivity criterion in the case where $\mathbb{X}$ is the unit disk was proposed. it relies on the tangent to the contour. The Jacobian of a function represents this tangent when defined, see Figure 8.5. Note that as the domain of the studied function $\mathbf{g}_i$ is a subset of $\mathbb{R}$, its Jacobian is a vector.

4. Let $\mathbb{X}$ be a subset of $\mathbb{R}$ and $\mathbf{g} : \mathbb{R} \rightarrow \mathbb{R}^2$ be a C^1 function with $\frac{d\mathbf{g}}{dx} : \mathbb{R} \rightarrow \mathbb{R}^2$ its Jacobian. If $0_{\mathbb{R}^2} \notin \left[\bigcup_{t \in \mathbb{T}} \frac{d\mathbf{g}}{dx}(x) \right]$, then $\mathbf{g}$ is injective over $\mathbb{X}$.

Proof. Let us demonstrate the contrapositive.

If $\mathbf{g}$ is not injective, $\exists (x_1, x_2) \in \mathbb{X}^2$, $x_1 \neq x_2$ so that $\mathbf{g}(x_1) = \mathbf{g}(x_2)$ as shown Figure 8.5. Let us denote $g_1 : \mathbb{R} \rightarrow \mathbb{R}$ and $g_2 : \mathbb{R} \rightarrow \mathbb{R}$ the functions that give respectively the first and second component of $\mathbf{g}$ (not to be mistaken with the $\mathbf{g}_i$ functions from the box atlas). x_1 and x_2 then verify

$$\forall i \in \{1, 2\}, g_i(x_1) = g_i(x_2) \quad (8.8)$$

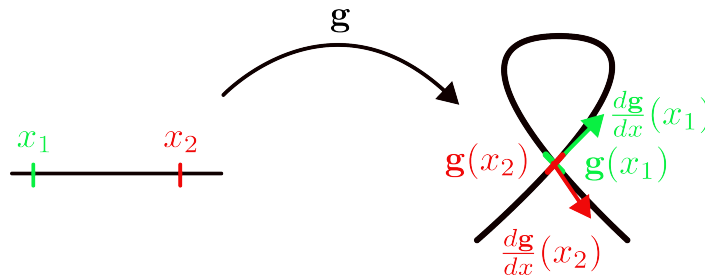


Figure 8.5 – Loop in $\mathbb{R}^2$, $\mathbf{g}$ is not injective

The mean value theorem can then be applied to each of the g_i functions, giving

$$\exists \tau_i \in [x_1, x_2], \frac{dg_i}{dx}(\tau_i) = 0 \quad (8.9)$$

Meaning that

$$\forall i \in \{1, 2\}, 0 \in \frac{dg_i}{dx}([x_1, x_2]) \quad (8.10)$$

Then,

$$0_{\mathbb{R}^2} \in \frac{dg_1}{dx}([x_1, x_2]) \times \frac{dg_2}{dx}([x_1, x_2]) \quad (8.11)$$

Finally per definition of the cartesian product and as shown Figure 8.6,

$$\frac{dg_1}{dx}([x_1, x_2]) \times \frac{dg_2}{dx}([x_1, x_2]) = \left[\bigcup_{x \in [x_1, x_2]} \mathbf{J}_{\mathbf{g}}(x) \right] \quad (8.12)$$

As $(x_1, x_2) \in \mathbb{T}^2$, $[x_1, x_2] \subset \mathbb{X}$. Then $0_{\mathbb{R}^2} \in \left[\bigcup_{x \in \mathbb{T}} \frac{d\mathbf{g}}{dx}(x) \right]$. Finally, we get that if $\mathbf{g}$ is not injective over $\mathbb{T}$, $0_{\mathbb{R}^2} \in \left[\bigcup_{x \in \mathbb{X}} \frac{d\mathbf{g}}{dx}(x) \right]$ which concludes the proof.

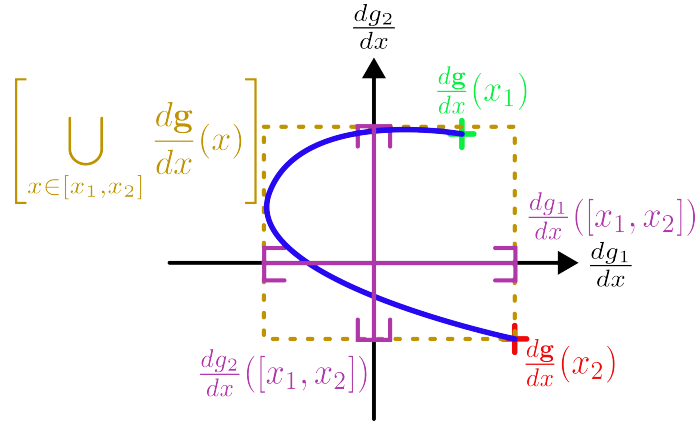


Figure 8.6 – The brown box contains the origin, we can't conclude on the injectivity

□

Since this theorem only works for the case of a function from $\mathbb{R}$ to $\mathbb{R}^2$, we propose an additional injectivity criterion. By construction of the atlas, the studied functions $\mathbf{g}_i$ are defined as $\mathbf{f} \circ \psi_i$ and the ψ_i functions are homeomorphisms. For an input box $[\mathbf{x}] \in V_i$, $\mathbf{g}_i$ is then injective over $[\mathbf{x}]$ if $\mathbf{f}$ is injective over $\psi_i([\mathbf{x}])$.

5. Let $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ be a C^1 function, and $\frac{d\mathbf{f}}{d\mathbf{x}} : \mathbb{R}^m \rightarrow \mathbb{R}^{m \times m}$ its Jacobian. If for a box $[\mathbf{x}] \subset \mathbb{R}^m$, $0 \notin \det \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}([\mathbf{x}]) \right] \right)$, then $\mathbf{f}$ is injective over $[\mathbf{x}]$.

Proof. Let there be $\mathbf{a}, \mathbf{b} \in [\mathbf{x}]$. For the sake of simplicity we will note $M = [\mathbf{J}_f([\mathbf{x}])]$ and for any matrix C we will note $c_{i,j}$ its element at the index (i, j) . Let us consider that $0 \notin \det(M)$.

If we consider a path $\mathbf{s} : [0, 1] \rightarrow [\mathbf{x}]$ such that $\mathbf{s}(0) = \mathbf{a}$ and $\mathbf{s}(1) = \mathbf{b}$ we can write:

$$\mathbf{f}(\mathbf{b}) - \mathbf{f}(\mathbf{a}) = \int_0^1 \mathbf{J}_f(\mathbf{s}(t)) \cdot \mathbf{s}'(t) dt \quad (8.13)$$

If we consider the path defined by $\forall t \in [0, 1], \mathbf{s}(t) = \mathbf{a} + t \cdot (\mathbf{b} - \mathbf{a})$ as depicted Figure 8.7, Equation 8.13 becomes

$$\mathbf{f}(\mathbf{b}) - \mathbf{f}(\mathbf{a}) = \left(\int_0^1 \mathbf{J}_f(\mathbf{s}(t)) dt \right) \cdot (\mathbf{b} - \mathbf{a}) \quad (8.14)$$

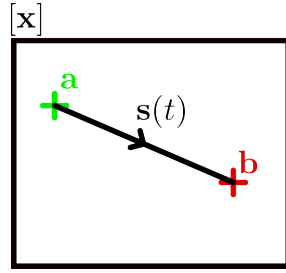


Figure 8.7 – Path $\mathbf{s}$ from $\mathbf{a}$ to $\mathbf{b}$

Moreover as $[\mathbf{x}]$ is convex and $\mathbf{s}([0, 1])$ represents the segment $[\mathbf{a}, \mathbf{b}]$, we get

$$\forall t \in [0, 1], \mathbf{s}(t) \in [\mathbf{x}] \quad (8.15)$$

Meaning that

$$\forall t \in [0, 1], \mathbf{J}_f(\mathbf{s}(t)) \in M \quad (8.16)$$

If we note $\mathbf{h} : t \mapsto \mathbf{J}_f(\mathbf{s}(t))$, we can write

$$\forall t \in [0, 1], \mathbf{h}(t) \in M \quad (8.17)$$

And

$$\int_0^1 \mathbf{J}_f(\mathbf{s}(t)) dt = \begin{pmatrix} \int_0^1 \mathbf{h}_{0,0}(t) dt & \cdots & \int_0^1 \mathbf{h}_{0,n-1}(t) dt \\ \vdots & \ddots & \vdots \\ \int_0^1 \mathbf{h}_{n-1,0}(t) dt & \cdots & \int_0^1 \mathbf{h}_{n-1,n-1}(t) dt \end{pmatrix} \quad (8.18)$$

For all $(i, j) \in [0, n - 1]^2$, $h_{i,j}$ is a real-value function that is bounded and continuous. The $h_{i,j}$ functions are then Riemann-integrable and the $\int_0^1 \mathbf{h}_{i,j}(t)dt$ are Riemann integrals. We can then write the result shown Figure 8.8 [111]:

$$1 * \inf_{[0,1]} h_{i,j} \leq \int_0^1 \mathbf{h}_{i,j}(t)dt \leq 1 * \sup_{[0,1]} h_{i,j} \quad (8.19)$$

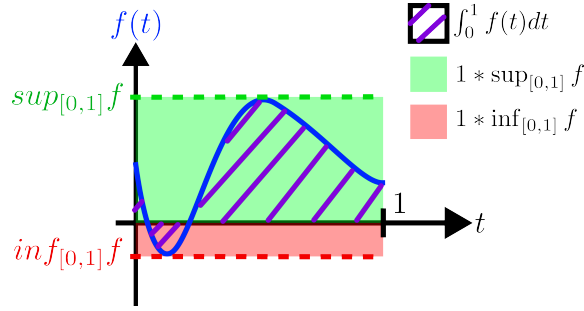


Figure 8.8 – Riemann integrals

From Equation 8.17, we get that $\inf m_{i,j} \leq \inf_{[0,1]} h_{i,j}$ and $\sup_{[0,1]} h_{i,j} \leq \sup m_{i,j}$. From this and Equation 8.19, we get

$$\inf m_{i,j} \leq \int_0^1 \mathbf{h}_{i,j}(t)dt \leq \sup m_{i,j} \quad (8.20)$$

Finally for all $(i, j) \in [0, n - 1]^2$

$$\int_0^1 \mathbf{h}_{i,j}(t)dt \in m_{i,j} \quad (8.21)$$

Meaning that $\int_0^1 \mathbf{J}_{\mathbf{f}}(\mathbf{s}(t))dt \in M$. As $0 \notin \det(M)$, $\det\left(\int_0^1 \mathbf{J}_{\mathbf{f}}(\mathbf{s}(t))dt\right) \neq 0$ meaning that $\int_0^1 \mathbf{J}_{\mathbf{f}}(\mathbf{s}(t))dt$ is invertible.

Finally Equation 8.14 can be seen as a linear bijective transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$. We can conclude that $\mathbf{f}(\mathbf{b}) - \mathbf{f}(\mathbf{a}) = \mathbf{0}$ only if $\mathbf{b} = \mathbf{a}$, which means that $\mathbf{f}$ is injective and concludes the proof. $\square$

Example

For this example we will consider the function from Equation 5.65 and compute the image of the unit circle by this function. To do so we define

$$\psi_1(x) = \begin{pmatrix} \cos\left(x \cdot \frac{\pi}{2}\right) \\ \sin\left(x \cdot \frac{\pi}{2}\right) \end{pmatrix} \quad (8.22)$$

$$\psi_2(x) = \begin{pmatrix} \cos\left(x \cdot \frac{\pi}{2} + \pi\right) \\ \sin\left(x \cdot \frac{\pi}{2} + \pi\right) \end{pmatrix} \quad (8.23)$$

The image of $[-1, 1]$ by ψ_1 gives half of the unit circle, and its image by ψ_2 gives the other half. In other words, $\left\{(\psi_i^{-1}([-1, 1]), \psi_i^{-1})\right\}_{i \in \{1, 2\}}$ is a box atlas of the unit circle S^1 . We can then construct two functions $\mathbf{g}_i$ and run the Box Chain decomposition Algorithm 10. Figure 8.9 show the result of the algorithm for both functions with a cover of 32 elements. The total 8 box chains are displayed in the initial set Figure 8.9a and in the image set Figure 8.9b, with a different color for each.

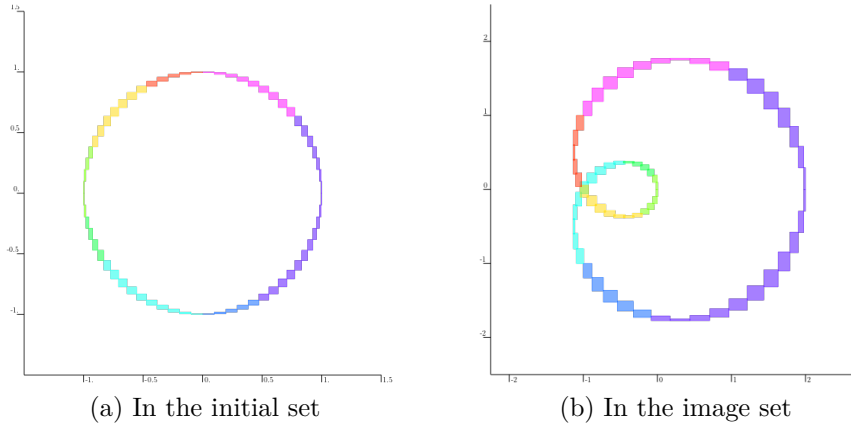


Figure 8.9 – Box Chain decomposition

8.3.4 Removing Fake boundaries

Finding self intersections

As mentionned before and visible on Figure 8.9, the self intersection in the boundary occurs between two different box chains. However due to the continuity of the function $\mathbf{f}$, two box chains that are neighbors (*i.e.* there is a pair of boxes, one in each chain, such that they are neighbors) necessarily intersect each other. Figure 8.10 shows these two cases of Box Chains intersections. The cases in green correspond to a real intersection of the boundary, while the red one is an intersection due to the continuity of $\mathbf{f}$.

Figure 8.10 – Intersections between Box Chains

Say that the two boxes which intersect each other are the image of a box $[\mathbf{x}_1]$ by the function ψ_i and a box $[\mathbf{x}_2]$ by the function ψ_j . We use different indices since we can have $i = j$. An easy way to distinguish both cases is to study the injectivity of $\mathbf{f}$ on $\psi_i([\mathbf{x}_1]) \cup \psi_j([\mathbf{x}_2])$. Figure 8.11 shows the detected self-intersections (in red) in the example from Subsection 8.3.3. Since at least one self intersection has been detected, there may be a fake boundary to remove.

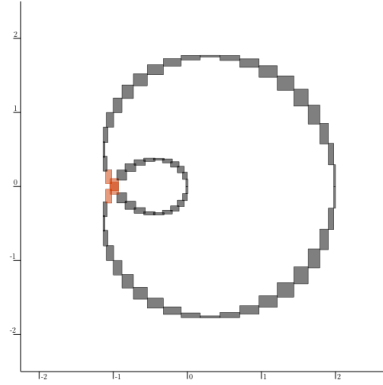


Figure 8.11 – Detected self-intersections in red in the boundary

New Box Chains construction

We can notice that new Box Chains are delimited by the red boxes Figure 8.11. From the result of the last subsection, we are able to get a list of boxes which do not contribute to a self-intersection of the boundary (in grey Figure 8.11). We can group the elements of this list into Box Chains by checking if they intersect with each other to get the result from Figure 8.12. For the sake of clarity, the intersected red boxes from Figure 8.11 have been reported to this Figure but they do not belong to a Box Chain.

Since we know that no self-intersection occurs in these Box Chains, we just need to classify one box of a Box Chain as fake to propagate it to the whole chain and remove it. To do so, we first need to distinguish the inside and the outside of our image set.

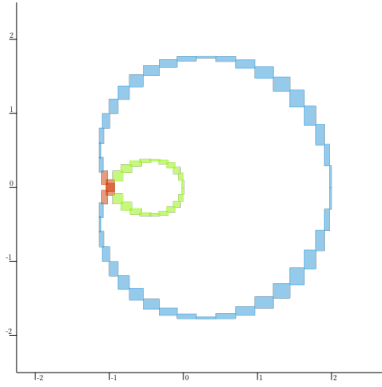


Figure 8.12 – Two new Box Chains (blue and green)

Determination of the inner areas

To determine the inner approximation of the image set an initial paving of the set $S = \bigcup_{i \in I} \left(\bigcup_{[\mathbf{x}] \in C_i} [\mathbf{g}_i](\mathbf{x}) \right)$ is needed. Figure 8.13a shows the result of this initial paving on the example from Subsection 8.3.3 with $\epsilon = 0.1$. By doing so, the union of the yellow boxes is an outer approximation of S , and each yellow box has a width smaller than 0.1.

The blue boxes then represent the complementary of the yellow boxes. Their union is an inner approximation of the complementary of S . They can be divided into connected subsets as shown on Figure 8.13b. For this work we will rely on the connected subset decomposition implemented in the Codac library [109].

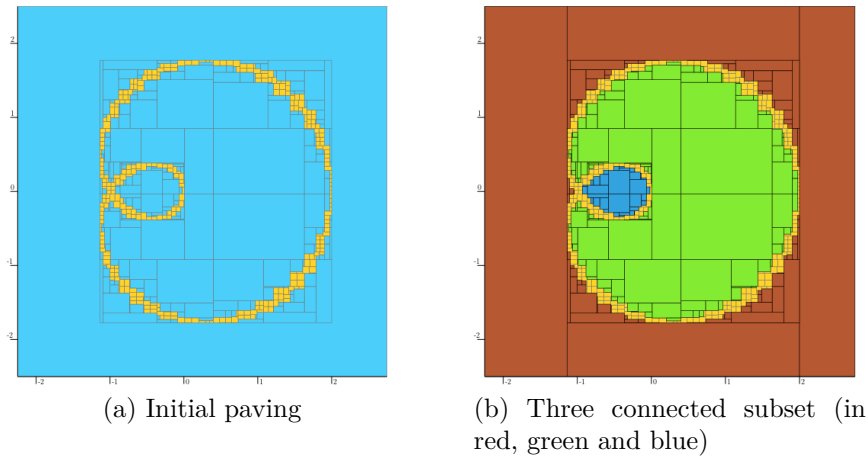


Figure 8.13 – Connected subsets in the paving

Then instead of qualifying each box individually as inside or outside the image set,

we can qualify the whole connected subset. This means that once a box is proven to be inside, the corresponding connected subset can directly be classified as inside.

A solution to distinguish the boxes that are inside and outside is to look at the normal to the boundary. As depicted the normal of a real boundary points towards the exterior. This means that the opposite corner is inside.

Remark 19. Note that in the case of a fake boundary the normal vector points toward the interior of the set, but the opposite corner is still inside. This is due to the fact that a fake boundary is inside the set it should bound.

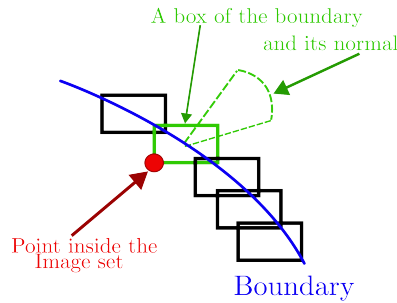


Figure 8.14 – Determination of the inner areas

Applying this to every box of the boundary, except the ones that were previously proven to belong to the self intersections, gives a set of points that are inside. The connected subsets containing at least one of these points can be marked as inside, and the other can be marked as outside. Figure 8.15 shows an example of output for this step.

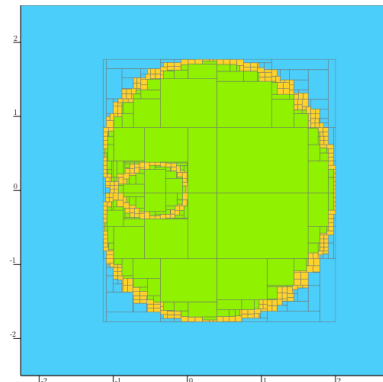


Figure 8.15 – Inner (green) and outer (green + yellow) approximation of the image set with fake boundaries

Suppressing the fake boundaries

As mentioned earlier, the normal to the boundary is supposed to point outwards. Figure 8.16 illustrates the fact that in the case of fake boundaries this normal points towards an area that was classified as inside in the last step. This can be seen as the fact that the normal of a fake boundary aims towards a fake exterior.

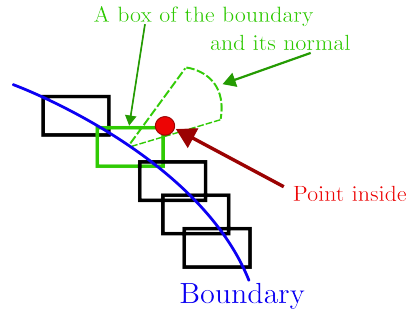
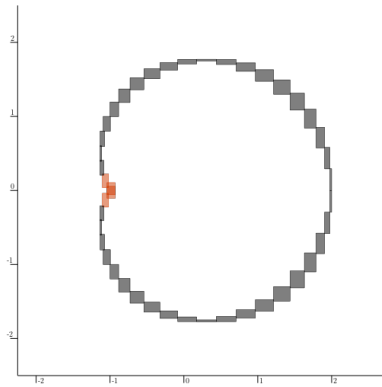
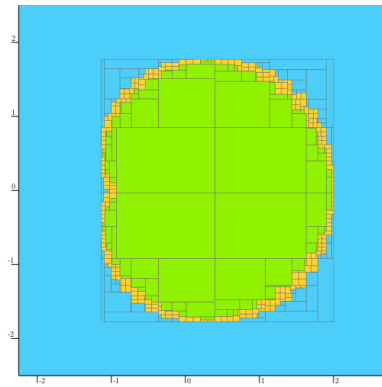


Figure 8.16 – Case of a fake boundary

As mentionned in Subsection 8.3.4, once a box is proven to belong to the fake boundary (*i.e.* with a normal pointing towards an inside area), the whole Box Chain can be deleted as displayed Figure 8.17a. Suppressing these boxes allows us to obtain a less pessimistic approximation of $\partial f(\mathbb{X})$ as visible on Figure 8.17b.



(a) Fake boundaries removed



(b) Inner (green) and outer (green + yellow) approximation of the image set without fake boundaries

Figure 8.17 – Better inner approximation thanks to the removal of fake boundaries

8.4 To go further

In [40], the study was limited to the computation of the image of the unit disk. However, the use of boxes to cover the set $\mathbf{f}(\partial\mathbb{X})$ gets more pessimistic as the dimension of the problem increases. As the Box Chain decomposition algorithm does not depend on the inclusion function chosen for $\mathbf{f}$, we can use other sets to enclose it (polytopes, zonotopes...). In particular, since we use a box atlas to reduce the dimension of the problem, the parallelepiped inclusion function from Chapter 5 can be used.

Let us consider the case of the computation of the image of the unit sphere by the function from Equation 5.60. As explained in Subsection 5.3.3, this function has in fact two singular points on the unit sphere. However, since these singularities are only ponctual, Equation 8.1 still holds. Figure 8.18 shows how the Algorithm 10 gives 159 Parallelepiped chains in this case (for 2400 parallelepipeds computed). Note that the singular points can never satisfy the injectivity criterion and tend to force the creation of new chains near them.

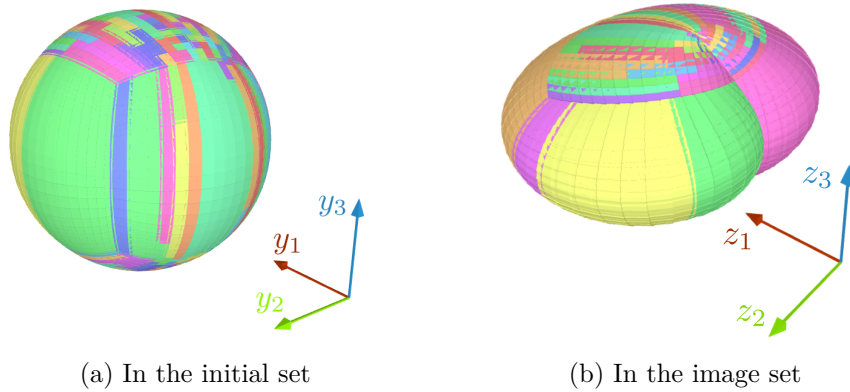


Figure 8.18 – Parallelepiped Chains in three dimensions

Once again, once this decomposition is done it can be used to simplify the research of self intersection in the boundary. Figure 8.19 shows the found intersctions in both the initial and image sets.

After this step, the same logic as before could be done to create new box chains and then remove the fake boundaries. A 3D equivalent of the triangulations could also be used to avoid the conservatism of the paving step.

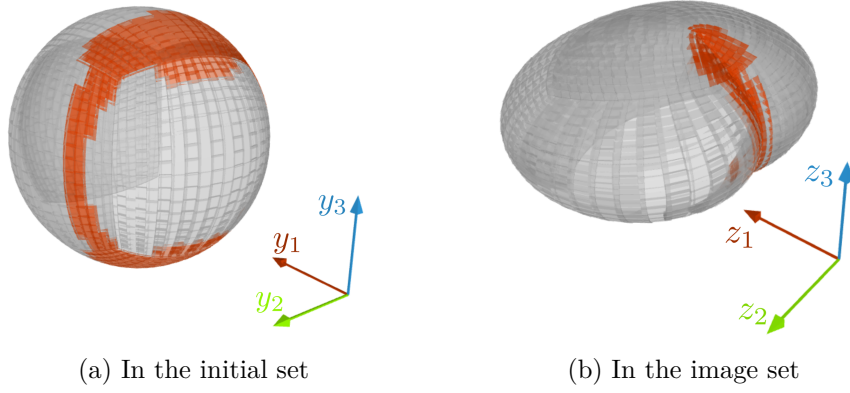


Figure 8.19 – Self-intersection in the boundary

8.5 Conclusion

This chapter presented a solution to the problematic of fake boundaries in the computation of the image of a set by a function. This problem was seen multiple times in the manuscript and is symptomatic of boundary based approaches. A method to remove them in the case of a function from a smooth manifold to was presented. It relies on a box atlas of the manifold to simplify the analysis. The method was finally applied on an example where it was indeed able to remove the fake boundaries.

The notion of Box Chains was introduced with the definition of the neighborhood and the Box Chain relations. This Box Chain relation relies on an injectivity criterion and two examples were proposed. These Box Chains can be used to make the detection of self-intersections in the boundary easier. In addition, a Box Chain decomposition algorithm in $O(p^2)$ was presented.

Finally a boundary simplification algorithm was displayed. It relies on Box Chains to facilitate the detection of self intersections in the contour. Once the fake boundaries have been removed, a better inner approximation of the image set can be obtained.

To build on the previous result, we displayed how the notion of Box Chains can be extended to other set representations with the use of parallelepipeds. This extension gets even more useful in higher dimension to avoid the high wrapping effect of boxes. When trying to compute the inner approximation of the image set, the reduction of the pessimism on the boundary is an important factor to be able to spot small inner volumes. Such method could then be used after the computation of the boundary of the reachable set from Chapter 6 to get a better inner approximation of the reachable set itself.

IMPLEMENTATION

9.1 Introduction

Throughout the thesis, a special focus was to implement the presented methods and to make them available for everyone to use. The objective of this chapter is to discuss these implementations and to give the reader advices on how to reproduce the presented results.

It has to be noted that every result of this manuscript is guaranteed under the stated hypothesis. It is guaranteed in both the theory and its implementation. To understand this second point, let us discuss the topic of numerical uncertainties.

9.2 Numerical guarantee

While using a computer to make a calculation, one has to choose how the numbers are represented : int, float32, float64, double... This choice means that a finite number of bits are used to represent a number. For example float32 uses 32 bits to represent a real number as : 1 bit for the sign, 8 for the exponent, and 23 for the significand. See for example [96] for a discussion on the subject.

Since we use a finite number of bits to represent numbers, it means that all real values can not be represented exactly. For example the real 0.1 can not be exactly stored in a computer due to the way real numbers are stored. In addition, some real numbers like π or $\sqrt{2}$ can not be written as finite decimals, meaning that it would require a computer with an infinite storage to store them.

The notion of ulp (Unit of Least Precision) defines the spacing between two consecutive floating-point numbers. If a number falls “in the ulp” (*i.e.* can not be represented), a choice has to be made on which value will be stored. The main strategies are TO_NEAREST (default strategy in most cases), ROUND_UP and ROUND_DOWN. These three strategies are displayed Figure 9.1.

Figure 9.1 – Rounding strategies

If we want to preserve the guarantee of our interval computations, we can not make an approximation because of the rounding strategy. Real numbers that can not be exactly represented will then be enclosed in a small interval. While handling interval arithmetics, the resulting interval may not land exactly on representable numbers. The upper bound will then be rounded upward, and the lower bound will be rounded downward. By doing so, no value is lost. Figure 9.2 shows an example of how a real number (in red) and an interval (in green) are represented on a computer in a guaranteed way.

Figure 9.2 – Guaranteed computer representation of real numbers and intervals

This process implies that from an implementation point of view, all functions are thick and return an interval for any entry. This has to be taken into account while implementing the methods presented in this manuscript.

9.3 Implementations

As mentionned in Section 2.4, the library used for the interval computations in this manuscript was the CODAC library. It relies on the GAOL library for the low level guaranteed interval arithmetics, and builds upon it to provide inclusion functions, paving algorithm... As explained in the previous subsection, this computationnal guarantee implies that all functions are thick.

To implement the parallelepiped inclusion function from Chapter 5, one should then rely on the version from Subsection 5.2.3 to handle this “thickness”. This has been done in the CODAC library since the inclusion function and the PEIBOS algorithm have been implemented in it. Note that the sources of the library are available on this public repository : github.com/codac-team/codac, and the documentation is available here : codac.io. The Adaptive versions of the PEIB and PEIBOS algorithms presented in Chapter 7 should also join the library soon and will be available at the same adresses.

In addition, all the codes used throughout this manuscript to generate figures are available at this public github repository : github.com/godardma/thesis_example. Since the CODAC library and the implementation of the work presented here may evolve, note that the README of the repository specifies the required version to run the examples. Since

different codes have different dependencies, they have been sorted per library requirement. Table 9.1 summarizes all the available codes, their language, their dependencies, and the figures that they produced.

Code			Figure n°	Requires		
ID	Name	Language		Codac 2	CGAL	CAPD
00	wrapping	Python	2.3	✓	×	×
01	sivia	Python	2.7	✓	×	×
02	himmelblau	Python	2.8	✓	×	×
03	contractor_programming	Python	2.9	✓	×	×
04	paving	Python	2.10	✓	×	×
05	zonotope	Python	2.11, 2.12	✓	×	×
06	parallelepiped_contractor	Python	2.14	✓	×	×
07	paving_parallelepiped_circle	Python	2.15, 2.16	✓	×	×
08	triangulate_parallelepiped_circle	C++	2.17, 2.18	✓	✓	×
09	parallelepiped_enclosure	Python	5.6	✓	×	×
10	parallelepiped_enclosure_set_valued	Python	5.7	✓	×	×
11	peibos	Python	5.11	✓	×	×
12	peibos_convergence_order	Python	5.12	×	×	×
13	peibos_triangulate	C++	5.15	✓	✓	×
14	pendulum	C++	5.16	✓	×	✓
15	lorenz	C++	5.17	✓	×	✓
16	spiral	C++	6.6	✓	×	✓
17	dubins	C++	6.8, 6.9, 6.10	✓	×	✓
18	cshape	C++	6.14, 6.15	✓	×	✓
19	rob_arm_sing	C++	7.6b	✓	×	×
20	rob_arm	C++	7.8,7.9	✓	×	×
21	rob_arm_unvalidated	C++	7.10	✓	×	×
22	area_reach	C++	7.13,7.14	✓	×	✓
23	validation	C++	7.15, 7.16, 7.17, 7.18, 7.19, 7.20a, 7.20b, 7.21, 7.22, 7.23	✓	✓	×
24	box_chains	C++	8.9, 8.11, 8.12, 8.13,8.15, 8.17	✓	×	×
25	parallelepiped_chains	C++	8.18, 8.19	✓	×	×

Table 9.1 – Code to Figure reference

CONCLUSION

10.1 Scope of the contributions

This thesis tackled the problematic of the a priori validation of a robotic mission by relying on set-based methods. The choice of the use of parallelepipeds and a boundary approach has been made to achieve a compromise between the quality of the result and the time required to get it. In addition this choice allowed us to get both an inner and an outer approximation of the reachable set of the robot, allowing direct applications for various fields such as obstacle avoidance and control validation.

10.1.1 Image of a set by a nonlinear function

The first and main contribution of the manuscript is a method to compute the inner and outer approximation of a set by a function. We defined a new parallelepiped inclusion function and showed that it has nice properties in terms of convergence order. We extended it to the computation of the image of the boundary of a set by relying on a box atlas. More precisely, we defined the notion of gnomonic atlas to simplify the notations in the cases where symmetries exist in the initial set. Since an analytical definition of the function is not required, it can also be used to propagate an initial set through an ODE.

10.1.2 Computation of the reachable set of a dynamical system

We then extended the work from [85] to compute the reachable set of a dynamical system starting from an initial set. This method provides the exact boundary of the reachable set if it is smooth, and an outer approximation of it otherwise. Since it relies on the integration of an augmented ODE, the method presented earlier can be applied to get both an inner and an outer approximation of the desired reachable set.

10.1.3 Adaptive enclosure of the image set

As the dimension of the problem increases, it gets more complex to handle and the computation time increases. This is especially true in our case since our method to compute the reachable set doubles the dimension of the ODE to integrate. We then proposed a method to compute an adaptive enclosure of the desired image set to save computation time when possible. This method is adaptive with respect to a constraint or set of constraints. A first large enclosure of the image set is done, and this enclosure is refined only where the constraint(s) can not be validated. This showed to be an effective way to reduce the computation time of our algorithms.

10.1.4 Fake boundaries removal

A drawback to the use of a boundary method is the possible apparition of fake boundaries in the result. This phenomenon has been displayed multiple times in the different examples of the manuscript. These fake boundaries are part of the image of the boundary which should be classified as inside of it. Since our method provides both an inner and an outer approximation of the image set, this make the inner approximation more pessimistic. A method to detect and automatically remove them has then been proposed. It relies on a box atlas and a subdivision of the charts into subsets called Box Chains. On a given Box Chain, we know that the function is injective, which simplifies the research of self intersections in the boundary (symptomatic of a fake boundary). In higher dimension, we also showed how Parallelepiped chains can be used to do the same while have a more precise enclosure.

10.2 Summary of the contributions

10.2.1 Papers

Chapters 5, 6, 7 and 8 are subject to publications. The contributions from Chapters 5 and 7 are already published and available. The contribution which led to Chapter 8 is also published but significant changes have been done to apply it to higher dimension. The publication associated with Chapter 6 is still under review. The list of published articles is available below.

-
- Maël Godard, Luc Jaulin, and Damien Massé, "Boundary approach to characterize the inner and outer approximation of the image of a disk", *Acta Cybernetica* (2025).
 - Maël Godard, Luc Jaulin, and Damien Massé, "Inner and outer approximation of the image of a set by a nonlinear function", *International Journal of Approximate Reasoning* (2025), pp. 109574.
 - Maël Godard, Luc Jaulin, and Damien Massé, "Constrained adaptive parallelepipedic approximation of the image of a set by a nonlinear function", *International Journal of Approximate Reasoning* (2026), pp. 109749.

10.2.2 Implementations

The parallelepiped inclusion function from Chapter 5 and the associated algorithms have been implemented in the CODAC library. In addition, their adaptive versions from Chapter 7 should also join the library. All the codes used to generate figures in this manuscript have been made available for people to try them and compare their methods to the one presented here.

10.3 Limitations and prospects

While the main contribution of this PhD thesis addresses a classical problem in set-based methods, we displayed how it can be used effectively in a context of robotic. Several problematic have been raised and tackled to improve the method both in terms of quality and computation time. However, some work is still remaining to put them all together. This would allow us to get the reachable set of a robot faster, and to produce more experiments to display the effectiveness of the method.

The implementation of our algorithm itself could also be improved. Once the adaptive scheme is in place, one could use a worse and faster integration scheme to get a rapide enclosure of the image set. The refinement could then be done using the same integrator with a smaller input or a better one in the last iterations.

In addition, one could rely on interval analysis to relax some of the hypothesis from Chapter 6. This could increase the possible use cases, with maybe a non-smooth initial set boundary.

Finally, a comparative study could be performed to weight the pros and the cons of our method compared to existing tools. This would however require to run the different

algorithms in the same conditions and on the same problem. A challenge would then be to find a case suited for all tools while being complex enough to discriminate them.

