Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics

Colas Le Guernic

Thèse de Doctorat
Université Grenoble I – Joseph Fourier
Octobre 2009

Université Grenoble I – Joseph Fourier

École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics

Calcul d'Atteignabilité des Systèmes Hybrides à Partie Continue Linéaire

THÈSE

pour obtenir le grade de

Docteur de l'Université Joseph Fourier

Spécialité : Mathématiques et Informatique

Préparée au Laboratoire VERIMAG, Équipe TEMPO

Présentée et soutenue publiquement par

Colas LE GUERNIC

le 28 Octobre 2009

Contents

Chapter 1

Introduction

Remark. *Every chapter of this thesis starts by a short abstract in French, except for the introduction and conclusion for which translations can be found on Appendices C and D respectively.*

Remarque. *Chaque chapitre de cette thèse débute par un court résumé en Français, à l'exception de cette introduction et de la conclusion qui sont traduites en Annexes C et D respectivement.*

1.1 Motivations

This thesis consists of several contributions to the problem of computing reachable states of continuous and hybrid systems. For continuous systems, the problem can be formulated as follows:

Consider a continuous dynamical system with input over some state space $\mathcal{X}$ defined via a differential equation of the form

$$\dot{x} = f(x, u) \tag{1.1}$$

where u ranges over some pre-specified set of admissible input signals. Given a set $\mathcal{X}_0 \subset \mathcal{X}$, compute all the states visited by trajectories of the system starting from any $x_0 \in \mathcal{X}_0$.

The significance of this question is that it allows us to verify whether a system behaves correctly in the presence of *all* admissible disturbances, where "correctly" can mean the avoidance of a "bad" subset of the state space or, potentially, more complex temporal properties. Such reachability computation can be used to check the robustness of a control system or an analog electrical circuit against disturbances or to compute the implications of biological models where parameter values and environmental conditions are not precisely known. Historically this line of research originated from the study of hybrid (discrete-continuous) systems and the attempts to apply ideas coming from the verification of programs and digital hardware to systems admitting real-valued state variables.

An intuitive explanation of what is going on in reachability computation can be given in terms of *numerical simulation*, the most commonly-used approach for validating complex systems. Each individual simulation consists in picking *one* initial condition and *one* input stimulus (random, periodic, step, etc.), producing the induced trajectory using numerical integration and observing whether this trajectory behaves correctly. Ideally we would like to repeat this procedure with *all* possible disturbances whose number is huge and typically even non countable. Reachability computation achieves the same effect as *exhaustive* simulation by performing the simulation in a "breadth-first" manner: rather than running each individual simulation to completion and then starting a new one, one computes at each time step all the states reachable by all possible inputs. This set-based simulation is, of course, more costly than the simulation of individual trajectories but provides more confidence in the correctness of the system.

Unlike other methods used for analysis of continuous (and mostly linear) systems, the information obtained from reachability computation covers also the *transient behavior* of the system in question, and not only its stationary behavior. This property makes the approach particularly attractive for the analysis of *hybrid* (discrete-continuous) systems where the applicability of analytic methods is rather limited. Such hybrid models can express, for example, deviation from idealized linear models due to constraints and saturation as well as other switching phenomena.

It is worth noting that reachability computation has a lot in common with the domain known as *interval analysis* which is concerned with providing rigorous results to numerical computations, despite rounding errors. What is common to these two domains is that they provide answers in terms of *sets* that are guaranteed to contain the exact answer and hence some ideas and techniques of set-based computations are shared. However the source of uncertainty in interval analysis are internal errors in the computation due to the use of bounded-precision arithmetics while in our case they are due to the external environment of the modeled system or the fact that the model itself has parameters which are not precisely known.

The major contributions of this thesis are:

1. *Linear Systems*: for systems defined by linear (time-invariant) differential equations we provide a new algorithmic scheme which improves complexity both theoretically and empirically. This scheme increases the dimensionality of systems that can be treated by more than an order of magnitude while avoiding the wrapping effect [Moo66];

2. *Hybrid Systems*: we extend the above scheme to handle piecewise-linear[1] systems defined by hybrid automata with linear differential equations. The

[1]or piecewise affine.

main challenge here is to compute efficiently the *intersection* of the reachable sets in one mode with the switching surfaces (transition guard);

In the rest of this chapter we give more insight into the question of what it means to compute the reachable set.

1.2 Proving Safety

In this study, our main interest is not on the asymptotic behavior of a dynamical system, or its behavior around a reference trajectory, but on the transient behavior of all the trajectories generated by the system:

$$\dot{x} = f(x, u)$$

where u ranges over some pre-specified set of admissible input signals $\widetilde{\mathcal{U}} \subset \mathbb{R}^+ \mapsto \mathcal{U}$ and $x(0)$ lies in $\mathcal{X}_0$.

The set of all possible trajectories is then:

$$\Xi(\mathcal{X}_0) = \left\{ \xi : \xi(0) \in \mathcal{X}_0 \text{ and } \exists u \in \widetilde{\mathcal{U}},\, \forall t \geq 0,\, \dot{\xi}(t) = f(\xi(t), u(t)) \right\}$$

Given a trajectory one can check if it verifies some properties we are interested in. One property of interest is the safety property. A trajectory ξ is said to be safe if it does not intersect a *bad* set $\mathcal{F}$:

$$\forall t \geq 0,\, \xi(t) \notin \mathcal{F}$$

The set $\mathcal{F}$ is a region of the state space we want to avoid. As an example, we do not want an airplane to fly below its stalling speed, or the temperature of the core of a nuclear reactor to heat beyond the critical fuel surface temperature.

Here we are not interested in one trajectory, but we want the whole system to be safe:

$$\forall \xi \in \Xi(\mathcal{X}_0),\, \forall t \geq 0,\, \xi(t) \notin \mathcal{F}$$

If $\mathcal{F}$ is avoided, then the system is said to be *safe*.

In this section we briefly describe three methods used to tackle this problem, the last of which, reachability analysis, is the subject of this thesis.

1.2.1 Simulations

If the set $\mathcal{X}_0$ of initial conditions and the set of admissible input signals are finite, it is possible to analyse the system exhaustively. Unfortunately $\Xi(\mathcal{X}_0)$ is often infinite and rarely countable, it is impossible to study all the trajectories individually. Moreover simulations are usually done on a finite time horizon, thus we often only study the bounded time safety of the system:

$$\forall \xi \in \Xi(\mathcal{X}_0),\, \forall t \in [0; T],\, \xi(t) \notin \mathcal{F}$$

where T is our time horizon.

One can apply statistical methods to solve this problem [CDL08]. Checking a sufficient (finite) number of randomly chosen trajectories might give sufficient confidence in the system.

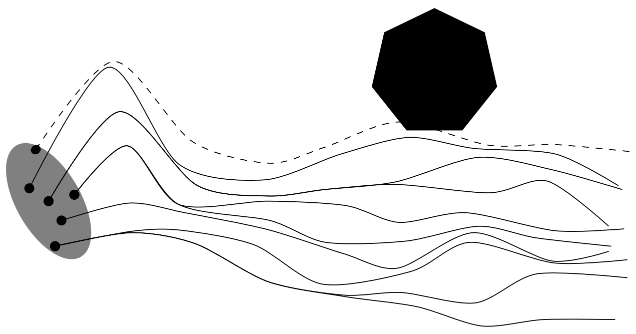


Figure 1.1: A few trajectories leaving the initial set (in grey). Only one trajectory (dashed) intersect the *bad* set (in black).

Another solution is to try to generate a finite set of trajectories that will exhibit all the behaviors of the system [KKMS03, BF04, BCLM06, GP06, DM07]. Most of these techniques are based on Rapidly-exploring Random Trees (RRT) originally developed in the field of motion planning, sensitivity analysis, or approximate bisimulation. The idea is to cover the set of reachable states by a small neighborhood of a finite set of trajectories. In other words we want this finite set of trajectories to be such that any state reachable by the original system is at a small distance from one of the chosen trajectories. Under some condition it is possible to bound this distance.

To recap, if an unsafe trajectory is found, the system is known to be unsafe. Otherwise the analysis is inconclusive. But one may try to use statistical analysis or generate the trajectories in a way that will give some confidence in the safety of the system. Under certain restricted conditions it is even possible to guarantee that the system is actually safe.

1.2.2 Barrier Certificates

Another way to tackle the safety problem, is to search for a surface, separating the set of initial points $\mathcal{X}_0$ and the bad set $\mathcal{F}$, which can not be crossed by any trajectory. More formally we have the following definition:

Definition 1.1 ([PJ04]). *A differentiable function $B : \mathcal{X} \to \mathbb{R}$ is a barrier certificate iff it satisfies the following conditions:*

$$\begin{aligned} B(x) &> 0 && \forall x \in \mathcal{F} \\ B(x) &\le 0 && \forall x \in \mathcal{X}_0 \\ \frac{\partial B}{\partial x} \cdot f(x,u) &\le 0 && \forall (x,u) \in \mathcal{X} \times \mathcal{U} \textit{ such that } B(x) = 0 \end{aligned}$$

An illustration is given on Figure 1.2.

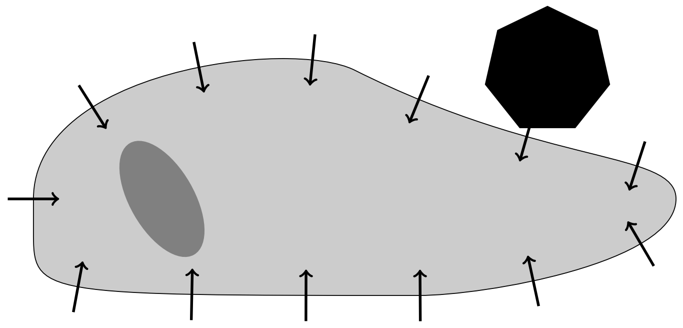


Figure 1.2: A barrier certificate proving safety of the system.

This notion of barrier certificate is closely linked to the one of Lyapunov function originally developed for stability analysis at the end of the 19th century by Aleksandr Lyapunov and is still extensively investigated. Intuitively a Lyapunov function V is positive everywhere except on one point x_0 where $V(x_0) = 0$, and its time derivative, which is equal to $\frac{\partial V}{\partial x} \cdot f(x, u)$ if it is differentiable, is nowhere positive. If such a function exists then x_0 is a stable equilibrium.

Here we are not interested in stability but in safety. If there is a $k \in \mathbb{R}$ such that $V(x) > k$ for all x in $\mathcal{F}$, and $V(x) \leq k$ for all x in $\mathcal{X}_0$ then $x \mapsto V(x) - k$ is a barrier certificate for our problem. Thus finding a barrier certificate can be done by finding a Lyapunov function.

But even for stable linear system, for which Lyapunov functions can be systematically found, once we have a Lyapunov function there is no guarantee that such a k exists. To summarize, barrier certificates might be hard to find, but they can guarantee the safety of a system over an infinite time horizon.

1.2.3 Reachability

The approach we take in this thesis is to compute the reachable set[2]. The reachable set is the set of all points reachable by the system:

$$\mathcal{R}(\mathcal{X}_0) = \{x : \exists \xi \in \Xi(\mathcal{X}_0),\, \exists t \in \mathbb{R}^+,\, x = \xi(t)\}$$

Unfortunately, this set is often impossible to compute. Instead we compute approximations of this set. In order to be able to verify properties like safety, this approximation must be conservative, that is, an over-approximation: $\mathcal{R} \subseteq \overline{\mathcal{R}}$.

Then, if $\overline{\mathcal{R}} \cap \mathcal{F} = \emptyset$, we can guarantee that $\mathcal{R} \cap \mathcal{F} = \emptyset$, and that the system is safe. If $\overline{\mathcal{R}} \cap \mathcal{F}$ is not empty the analysis is inconclusive and one might try to improve

[2]Another object of interest to reachability analysis is the reachability tube, or flow pipe:

$$\mathcal{R}_{\text{tube}}(\mathcal{X}_0) = \{(x, t) : \exists \xi \in \Xi(\mathcal{X}_0),\, x = \xi(t)\}$$

but we will mainly focus here on the reachable set.

the over-approximation. But if the system is unsafe, any over-approximation will intersect the bad set $\mathcal{F}$ and we will not be able to conclude. Thus it is sometimes also interesting to compute an under-approximation $\underline{\mathcal{R}} \subseteq \mathcal{R}$. If $\underline{\mathcal{R}}$ intersects $\mathcal{F}$ then the system is unsafe.

There are several ways to compute approximations of the reachable sets, the most obvious and useless way is to remark that $\emptyset \subseteq \mathcal{R} \subseteq \mathcal{X}$. Thus the fastest algorithm simply returns $\emptyset$ and $\mathcal{X}$ as under-and over-approximations of the reachable set. This is why we will need a notion of quality of approximation.

More interesting ways of approximating the reachable sets were discussed in the two previous sections. Indeed, a finite set of trajectories gives an under-approximation of $\mathcal{R}$. Under some conditions, it is even possible to guarantee that if we keep generating trajectories for an infinite time, the set of visited points will be dense in $\mathcal{R}$. Moreover, if all the points of $\mathcal{R}$ are known to be in an ϵ-neighborhood of our finite set of trajectories, then we also have an over-approximation of $\mathcal{R}$.

A barrier certificate V also defines directly an over-approximation of the reachable set: $\mathcal{R} \subseteq \{x : V(x) \leq 0\}$. The frontier of the reachable set defines a barrier certificate, the smallest with respect to inclusion.

The approach we will take is related to simulation. But instead of having points evolving in the state space we will have sets. Like for simulation it is rather difficult to consider an infinite time horizon. That is why we will focus on bounded time reachability. The set we want to over-, or under-, approximate is:

$$\mathcal{R}_{[0;T]}(\mathcal{X}_0) = \{x : \exists \xi \in \Xi(\mathcal{X}_0),\, \exists t \in [0;T],\, x = \xi(t)\}$$

where T is our time horizon.

In the following we will denote by $\mathcal{R}_{[t_0;t_1]}(\mathcal{Y})$ the set of points reachable from $\mathcal{Y}$ between times t_0 and t_1 by system (1.1), or, more formally:

$$\mathcal{R}_{[t_0;t_1]}(\mathcal{Y}) = \{x : \exists \xi \in \Xi(\mathcal{Y}),\ \text{and } \exists t \in [t_0;t_1],\, x = \xi(t)\}$$

We also define $\mathcal{R}_t(\mathcal{Y})$ as $\mathcal{R}_{[t;t]}(\mathcal{Y})$. When $\mathcal{Y}$ is omitted it is assumed to be $\mathcal{X}_0$.

Let us remark that for bounded horizon in discrete time there exists an alternative approach for proving safety which consists of formulating the existence of a property-violating (or safety-violating) trajectory as an existence of a satisfying assignment to a formula which is obtained by unfolding the recurrence relation defining the system k times with a new set of variables for each time step. This approach underlies, in fact, model-predictive control, and its adaptation to verification of hybrid system has been investigated in [BM99].

1.3 Outline of the Thesis

The thesis is separated in two parts. The first part is concerned with reachability analysis of continuous linear systems. The second part extend this work to hybrid systems reachability. Before that, Chapter 2 investigates some of the set representations that have been used in the context of reachability analysis.

Part I The first part of this thesis is separated into three chapters. After exposing approaches related to ours in Chapter 3, we present the main contribution of this thesis: a new algorithmic scheme for reachability analysis of linear time-invariant systems in Chapter 4. The only drawback of its exact implementation is that its output is hard to manipulate. It can also be implemented into an approximation algorithm that is not subject to the *wrapping effect*, an uncontrolled growth of the approximation which is the subject of numerous research papers since the early 1960s. Finally, Chapter 5 discusses a variant of this algorithm specialized to support functions, a functional representation of convex sets.

Part II The second part of this thesis starts by a short introduction to hybrid systems reachability analysis in Chapter 6. Then, we focus on two aspects of hybrid systems: firstly, we investigate the implications of having an invariant for the continuous reachability analysis in Chapter 7, and finally, in Chapter 8, we explain how to intersect the outputs of the algorithms described in Part I with hyperplanar guards.

The concluding chapter summarizes the contributions of this thesis and suggests some future research directions.

Chapter 2

Representing Sets

Résumé : *Avant de calculer l'ensemble atteignable nous devons clarifier la notion de calcul d'un ensemble. Si il s'agit de trouver une représentation pour cet ensemble alors $\mathcal{R}_{[0;T]}$ peut être représenté par $\mathcal{X}_0$, $\mathcal{U}$, f et T. Pour pouvoir faire des calculs, et résoudre des problèmes, nous devons choisir une* bonne *représentation. Par exemple, pour vérifier la sûreté d'un système nous devons pouvoir déterminer si l'intersection avec $\mathcal{F}$ est vide. Dans cette thèse nous décrivons des algorithmes qui permettent d'exprimer l'ensemble atteignable, initialement représenté par $\mathcal{X}_0$, $\mathcal{U}$, f et T, sous une forme exploitable par d'autres outils pour déterminer certaines propriétés du système.*
Dans ce chapitre, nous nous intéressons aux notions de sur- et sous-approximation ainsi qu'à la distance de Hausdorff ; cette distance nous permet d'évaluer la qualité *d'une approximation. Nous présentons ensuite plusieurs classes d'ensembles et leurs représentations : produits d'intervalles, ellipsoïdes, polytopes, zonotopes, et ensembles convexes représentés par leur fonction support. Nous montrons enfin comment effectuer certaines opérations sur ces ensembles : transformation affine, somme de Minkowski, union convexe, et intersection avec un hyperplan.*

The exact reachable set is very hard to compute, for this reason we only compute an over-approximation, and only for a bounded time horizon.

Before that, we must clarify what it means to compute a set. Indeed, if it means finding a computer representation of this set, then, $\mathcal{R}_{[0;T]}$ can already be represented by $\mathcal{X}_0$, $\mathcal{U}$, f and T. In order to be able to do some computations one must first choose a class of sets with a good representation.

A good representation should allow us to solve problems. If we want to check safety, we need to be able to check intersection with $\mathcal{F}$. However, safety is not the only problem one may solve with an approximation of the reachable set. The aim of this thesis is to design algorithms that will transform the reachable set, represented initially by $\mathcal{X}_0$, $\mathcal{U}$, f and T into a form that other tools can use to solve efficiently problems of interest.

In this chapter, we will first define the notions of over-and under-approximation, as well as the notion of Hausdorff distance, which is a way to evaluate the *quality* of an approximation. Then, we will present some useful classes of sets together with their representation. Finally we will show how some operations on these sets can be performed. For improved readability, all sets are considered to be closed.

2.1 Distances and Approximations

We want to approximate a set $\mathcal{R}$ by a set $\widetilde{\mathcal{R}}$. One of the properties we are interested in, is the safety property: we want to know if $\mathcal{R}$ intersects a *bad* set $\mathcal{F}$. Ideally, we would like to have the following property:

$$\mathcal{R} \cap \mathcal{F} = \emptyset \Longleftrightarrow \widetilde{\mathcal{R}} \cap \mathcal{F} = \emptyset$$

This is generally very hard to achieve. Moreover if the approximation is computed independently from $\mathcal{F}$, then this would imply that $\mathcal{R} = \widetilde{\mathcal{R}}$. That is why we will only consider one direction of the equivalence relation.

$$\mathcal{R} \cap \mathcal{F} = \emptyset \Longrightarrow \widetilde{\mathcal{R}} \cap \mathcal{F} = \emptyset$$

or

$$\mathcal{R} \cap \mathcal{F} = \emptyset \Longleftarrow \widetilde{\mathcal{R}} \cap \mathcal{F} = \emptyset$$

One sufficient condition to ensure these properties is that $\widetilde{\mathcal{R}} \subseteq \mathcal{R}$ or $\mathcal{R} \subseteq \widetilde{\mathcal{R}}$, respectively. In the first case $\widetilde{\mathcal{R}}$ is called an *under*-approximation of $\mathcal{R}$, and will be denoted $\underline{\mathcal{R}}$. In the second case $\widetilde{\mathcal{R}}$ is called an *over*-approximation of $\mathcal{R}$, and will be denoted $\overline{\mathcal{R}}$.

Our main focus in this thesis is on computing over-approximations, but we will also consider under-approximations.

For any $\mathcal{X} \subseteq \mathbb{R}^d$, any subset of $\mathcal{X}$ and any set containing $\mathcal{X}$ are valid under- and over-approximations of $\mathcal{X}$, respectively. The notion of Hausdorff distance will help us evaluate the quality of an approximation. This distance on sets is defined with respect to a distance on points. For the latter we will take the distance induced by an unspecified norm.

Definition 2.1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets. The Hausdorff distance between $\mathcal{X}$ and $\mathcal{Y}$, denoted $d_H(\mathcal{X}, \mathcal{Y})$ is:*

$$d_H(\mathcal{X}, \mathcal{Y}) = \max\left(\sup_{x\in\mathcal{X}} \inf_{y\in\mathcal{Y}} \|x-y\|, \sup_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} \|x-y\|\right)$$

The Hausdorff distance is a distance on sets in the metric sense, it thus has some nice properties:

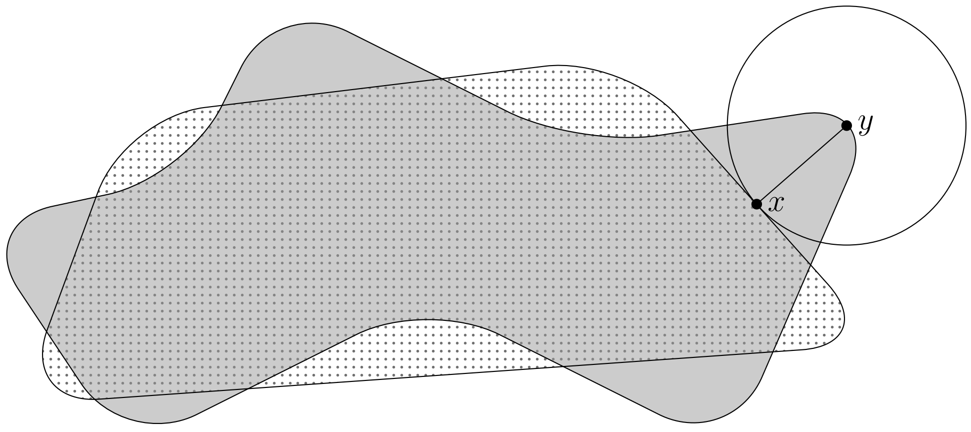


Figure 2.1: The Hausdorff distance (induced by the Euclidean norm) between these two sets is realized on x and y.

- *identity of indiscernibles*: For any two closed sets $\mathcal{X}$ and $\mathcal{Y}$,

$$d_H(\mathcal{X}, \mathcal{Y}) = 0 \Longleftrightarrow \mathcal{X} = \mathcal{Y}$$

- *symmetry*: for any two sets $\mathcal{X}$ and $\mathcal{Y}$,

$$d_H(\mathcal{X}, \mathcal{Y}) = d_H(\mathcal{Y}, \mathcal{X})$$

- *triangle inequality*: for any three sets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$,

$$d_H(\mathcal{X}, \mathcal{Z}) \leq d_H(\mathcal{X}, \mathcal{Y}) + d_H(\mathcal{Y}, \mathcal{Z})$$

One property of particular interest uses the notion of Minkowski sums.

Definition 2.2 (Minkowski sum). *The Minkowski sum of two sets $\mathcal{X}$ and $\mathcal{Y}$ is the set of sums of elements from $\mathcal{X}$ and $\mathcal{Y}$:*

$$\mathcal{X} \oplus \mathcal{Y} = \{x + y : x \in \mathcal{X} \textit{ and } y \in \mathcal{Y}\}$$

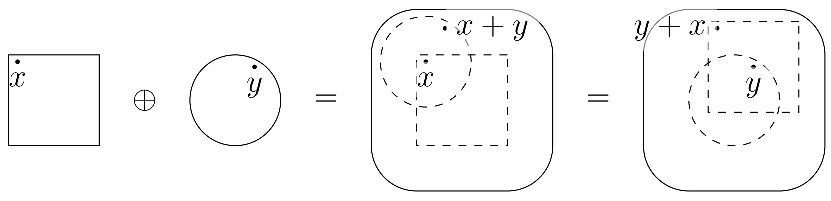


Figure 2.2: Minkowski sum of a square and a disk.

This concept is illustrated on Figure 2.2 with the example of two points x and y from $\mathcal{X}$ and $\mathcal{Y}$ respectively, and their sum $x + y$ inside of $\mathcal{X} \oplus \mathcal{Y}$. Naturally, the commutativity of $+$ implies the commutativity of $\oplus$.

Proposition 2.1. *For any two closed sets $\mathcal{X}$ and $\mathcal{Y}$, if $\mathcal{B}$ is the ball of radius $d_H(\mathcal{X}, \mathcal{Y})$, then:*

$$\mathcal{X} \subseteq \mathcal{Y} \oplus \mathcal{B} \text{ and } \mathcal{Y} \subseteq \mathcal{X} \oplus \mathcal{B}$$

Furthermore, $\mathcal{B}$ is the smallest such ball.

2.2 Set Representation

In this section we present some useful classes of subsets of $\mathbb{R}^d$, and their representations, that have been used in the context of reachability analysis. They all have also been successfully used in numerous other fields.

We will illustrate how an arbitrary set $\mathcal{S}$ (see Figure 2.3) can be approximated by a member of each class.

2.2.1 Boxes

One of the most simple class of sets is the class of boxes[1].

Definition 2.3 (Box)**.** *A set $\mathcal{B}$ is a* box *iff it can be expressed as a product of intervals.*

$$\mathcal{B} = [\underline{x_1}; \overline{x_1}] \times \ldots \times [\underline{x_d}; \overline{x_d}]$$

$\mathcal{B}$ is the set of points x whose i^{th} coordinate, x_i, lies between $\underline{x_i}$ and $\overline{x_i}$.

One of their strengths is that they admit a compact representation: boxes in $\mathbb{R}^d$ only require $2d$ parameters. Moreover, most operations can be done very efficiently. They are successfully used in numerous fields, notably in the context of interval analysis [MKC09] and interval arithmetic. They allow us to compute mathematically rigorous bounds by extending real arithmetic and functions over reals in general to intervals. Unfortunately they yield a coarse over-approximation in our context as we will see in the next chapter.

For a set $\mathcal{S}$ we define its *interval hull*, denoted $\square(\mathcal{S})$.

Definition 2.4 (Interval Hull)**.** *The* interval hull *of a set $\mathcal{S}$, denoted $\square(\mathcal{S})$, is its smallest enclosing box.*

$$\square(\mathcal{S}) = [\underline{x_1}; \overline{x_1}] \times \ldots \times [\underline{x_d}; \overline{x_d}]$$

where for all i, $\underline{x_i} = \inf\{x_i : x \in \mathcal{S}\}$ and $\overline{x_i} = \sup\{x_i : x \in \mathcal{S}\}$.

An illustration of the interval hull is given on Figure 2.3.

We also define the symmetric interval hull of $\mathcal{S}$, denoted $\boxdot(\mathcal{S})$.

Definition 2.5 (Symmetric Interval Hull)**.** *The* symmetric interval hull *of a set $\mathcal{S}$, denoted $\boxdot(\mathcal{S})$, is its smallest enclosing interval product symmetric with respect to the origin:*

$$\boxdot(\mathcal{S}) = [-\overline{|x_1|}; \overline{|x_1|}] \times \ldots \times [-\overline{|x_d|}; \overline{|x_d|}]$$

where for all i, $\overline{|x_i|} = \sup\{|x_i| : x \in \mathcal{S}\}$.

[1]Also known as interval products, (axis aligned) hyper-rectangles, ...

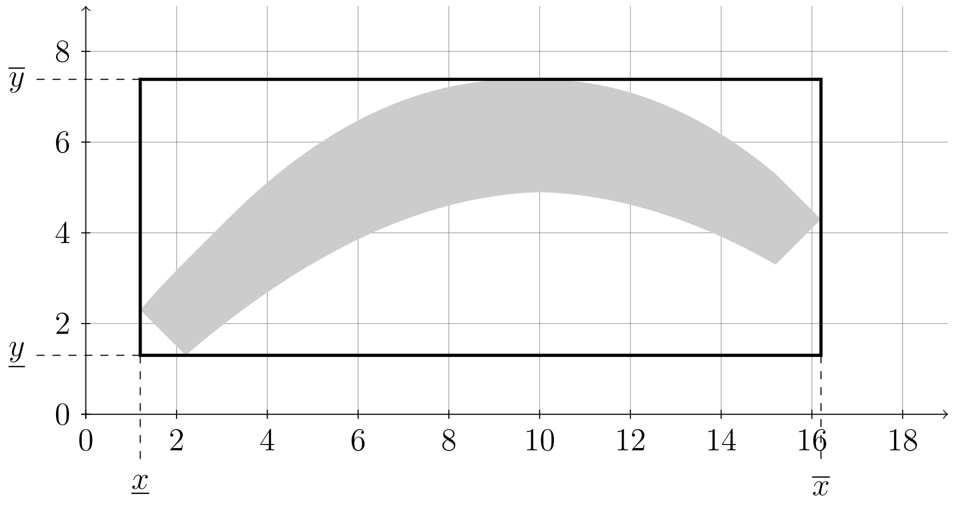


Figure 2.3: The set $\mathcal{S}$ and its interval hull $\square(\mathcal{S})$.

2.2.2 Ellipsoids

An ellipsoid is the image of an Euclidean ball by an invertible linear transformation, A. They are usually represented by their center c and a shape matrix $Q = AA^\top$.

Definition 2.6 (Ellipsoid)**.** *Given a point c and a positive definite*[2] *matrix Q we can define the* ellipsoid $\mathcal{E}(c, Q)$ *of center c and shape matrix Q as:*

$$\mathcal{E}(c, Q) = \{x : (x - c) \cdot Q^{-1}(x - c) \leq 1\}$$

When Q is the identity matrix I, $\mathcal{E}(c, I)$ is the unit Euclidean ball of center c.

Like interval products they have a constant representation size, $d(d+1)$ parameters in dimension d, but they are much more efficient in approximating *smooth* sets. They are used in various ellipsoidal calculus tools [RST02, KV06].

One notable application of ellipsoids is the ellipsoid method [YN76] in optimization. It has been used in [Kha79] to prove that linear programs (optimization of a linear function over a convex polytope, for an interesting review on the subject see [Tod02]) can be solved in polynomial time.

2.2.3 Polytopes

Polytopes are the generalization of polygons and polyhedra[3] to higher dimension. They play an important role in numerous fields and are the subject of numerous textbooks (see e.g. [Zie95]).

[2]A matrix Q is positive definite iff it is equal to its transpose $Q^\top$, and $x \cdot Qx > 0$ for all nonzero x.

[3]Here a polyhedron is a 3-dimensional polytope, in other context it can mean unbounded polytope

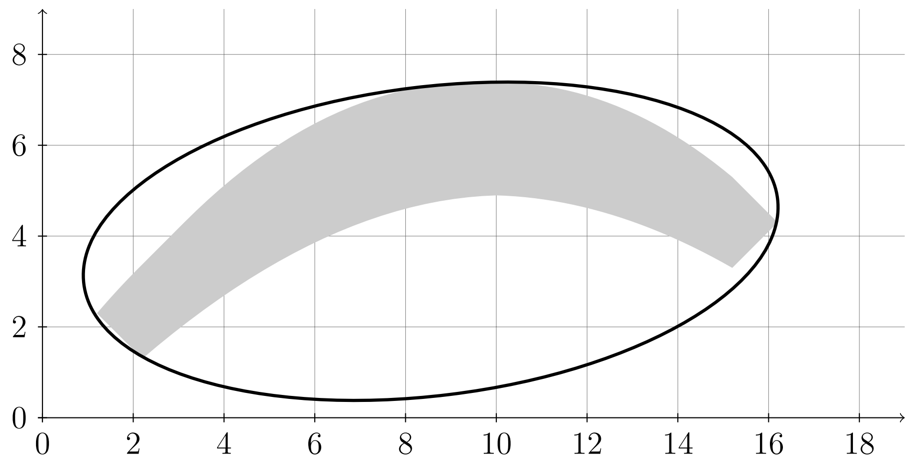


Figure 2.4: A set $\mathcal{S}$ and an enclosing ellipsoid.

Definition 2.7 (Polytope)**.** *A polytope $\mathcal{P}$ is the bounded intersection of a finite set $\mathcal{H}$ of half-spaces:*

$$\mathcal{P} = \bigcap_{h \in \mathcal{H}} h$$

An half-space is a set defined by a non-null normal vector n and a value γ:

$$\{x : x \cdot n \leq \gamma\}$$

Equivalently a bounded polytope $\mathcal{P}$ is the convex hull of a finite set $\mathcal{V}$:

$$\mathcal{P} = \{\sum_{v \in \mathcal{V}} \alpha_v v : \forall v \in \mathcal{V},\ \alpha_v \geq 0 \textit{ and } \sum_{v \in \mathcal{V}} \alpha_v = 1\}$$

The elements of $\mathcal{V}$ are called the vertices of $\mathcal{P}$.

These two definitions lead to two complementary representations:

- a $\mathcal{V}$-polytope is a polytope represented by its vertices.
- a $\mathcal{H}$-polytope is a polytope represented by a set of half-spaces, defining its facets, each half-space being represented by its normal vector n and a value γ: $\{x : x \cdot n \leq \gamma\}$. Normal vectors and corresponding values are often merged in a matrix A and a vector b, then $\mathcal{P} = \{x : Ax \leq b\}$, where $\leq$ must be interpreted component-wise.

It is common to use both representations, since some operations are easier on the first, for example convex hull of union of polytopes, while others are easier on the second, for example intersection.

Contrary to ellipsoids and interval products, polytopes do not necessarily have a center of symmetry, which is certainly an advantage when trying to approximate

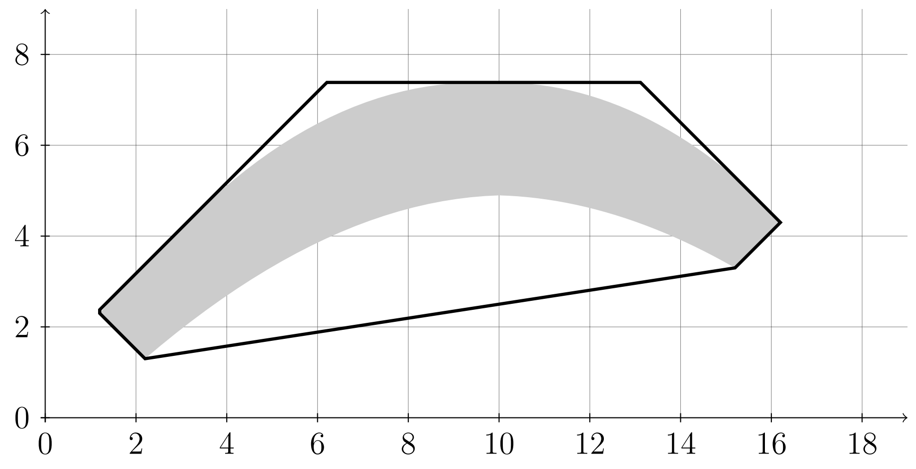


Figure 2.5: A set $\mathcal{S}$ and an enclosing polytope.

a set like $\mathcal{S}$ in Figure 2.5. However one may argue that the basic objects underlying convex polytopes are half-spaces, and that intersections of ellipsoids are much more expressive than intersections of half-spaces[4].

Another difference is that, polytopes do not have a constant representation size, since we can intersect as many half-spaces as we want. If the size of the manipulated polytopes grows during computation, one may need some reduction algorithm in order to reduce the number of vertices, or facets, while introducing as small approximation error as possible [Fre08].

Another solution is to consider a sub-class of the class of polytope by restricting the orientation of the facets to a fixed, possibly finite, set $\mathcal{L}$. We will denote such a class as the class of $\mathcal{L}$-polytopes. These sub-classes are sometimes called template polyhedra[5]. Some of them have custom representations and algorithms, a simple example is the class of interval products, but we can also mention octagons [Min01] and octahedrons [CC04], both developed in the context of abstract interpretation [CC77].

2.2.4 Zonotopes

An interesting sub-class of polytopes is the class of zonotopes[6] which underlies affine arithmetic, an extension of interval arithmetic.

Definition 2.8 (Zonotope)**.** *A zonotope $\mathcal{Z}$ in $\mathbb{R}^d$ is the image of a unit cube $\mathcal{B}_\infty$ in $\mathbb{R}^n$ by an affine transformation.*

$$\mathcal{Z} = \{Ax + c : x \in \mathcal{B}_\infty\}$$

[4]Indeed for any polytope with k facets there is a sequence of intersections of k ellipsoids that converge towards it, which means that the set of all polytopes with k facets is a subset of the boundary of the set of intersections of k ellipsoids.

[5]in this context polyhedron is synonymous to unbounded polytope.

[6]also known as Affine Forms, G-intervals, ...

When the affine transformation preserves dimensionality, $\mathcal{Z}$ is restricted to be a parallelotope. But the class of zonotopes is richer, allowing affine transformations going from $\mathbb{R}^n$ to $\mathbb{R}^d$, where n can be smaller than, equal to, or bigger than d.

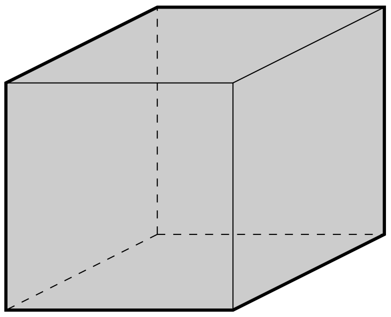


Figure 2.6: *Ceci n'est pas un cube.*

An example, simple to visualize, is the usual drawing of a 3 dimensional cube. In order for it to fit on a 2 dimensional piece of paper one usually applies an affine transformation going from $\mathbb{R}^3$ to $\mathbb{R}^2$, as in Figure 2.6. In order to see more examples of zonogons constructed from a cube in $\mathbb{R}^3$, just take any *almost* cubic object (like a pack of cigarettes) and let the sun apply an affine transformation for you by casting a shadow in the shape of a zonogon.

There is another characterization for zonotopes which is less intuitive but, in our context, more useful. This characterization uses the notion of Minkowski sums.

Definition 2.9 (Zonotope)**.** *A zonotope $\mathcal{Z}$ is the finite Minkowski sum of line segments.*

$$\mathcal{Z} = \left\{ c + \sum_{i=0}^{n-1} \alpha_i g_i : \forall i,\ -1 \le \alpha_i \le 1 \right\}$$

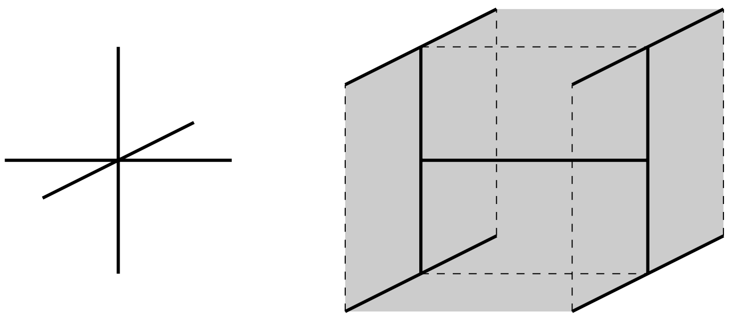


Figure 2.7: A Minkowski sum of line segments.

To better understand this second definition we will link it to the first one by expressing the unit cube as a sum of line segments.

The unit cube [Zon05] $\mathcal{B}_\infty$ in $\mathbb{R}^n$ is the product of n intervals, $[-1;1]^n$. There is a relation between products of sets and Minkowski sums. Indeed if $\mathcal{X}$ and $\mathcal{Y}$ are two subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively then:

$$\mathcal{X} \times \mathcal{Y} = (\mathcal{X} \times \{0_m\}) \oplus (\{0_n\} \times \mathcal{Y})$$

where 0_k is the origin of $\mathbb{R}^k$. Thus we can express the unit cube as:

$$\mathcal{B}_\infty = \bigoplus_{i=0}^{n-1} \mathcal{L}_i \text{ where } \mathcal{L}_i = \{0_i\} \times [-1;1] \times \{0_{n-i-1}\}$$

Thus, a zonotope is a Minkowski sum of line segments.

$$\mathcal{Z} = A\mathcal{B}_\infty \oplus \{c\} = \left(\bigoplus_{i=0}^{n-1} A\mathcal{L}_i \right) \oplus \{c\}$$

If we denote by g_i the i^{th} column of the matrix A, then:

$$A\mathcal{L}_i = \{\alpha g_i : \alpha \in [-1;1]\}$$

and the line segment $A\mathcal{L}_i$ is said to be generated by the vector g_i. Thus a zonotope can be represented by its center c and the list of its generators g_0, …, g_{n-1}. Such a zonotope is denoted by $\langle c | g_0, \ldots, g_{n-1} \rangle$.

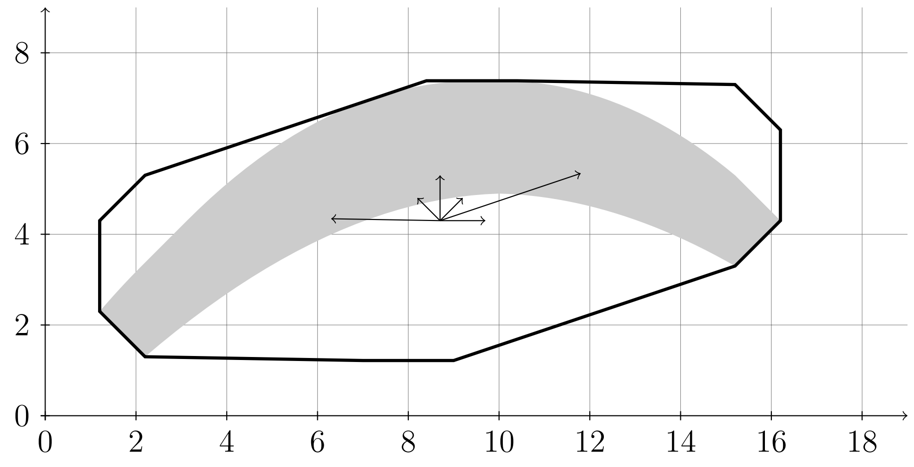


Figure 2.8: A set $\mathcal{S}$ and an enclosing zonotope with its center and generators.

As with polytopes, we have a variable representation size; more generators may help to improve an approximation. The ratio $\frac{n}{d}$, the number of generators over the dimension of the zonotope is called the order of the zonotope. One can see on Figure 2.8 an order 3 approximation of a 2 dimensional non-convex set (thus having 6 generators). However, contrary to polytopes, it has the same limitation as interval products and ellipsoids, in that it has a center of symmetry.

Nevertheless, not all centrally symmetric polytopes are zonotopes. A zonotope is a centrally symmetric polytope whose facets of any dimension are also centrally symmetric polytopes. Note that the condition on central symmetry of facets is very restrictive as illustrated in Figure 2.9. Indeed, in a two dimensional plane any centrally symmetric convex polygon is of course a zonogon[7], but for dimensions higher than 2 very few centrally symmetric convex polytopes are zonotopes. That is why some operations involving zonotopes, for example enclosing a set of points by a zonotope [GNZ03], are much easier in the plane.

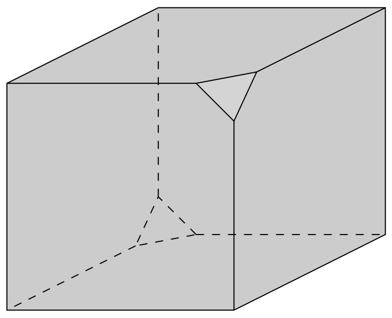


Figure 2.9: A centrally symmetric set which is not a zonotope.

As we will see in this thesis, the main interest of zonotopes in our context is that it is expressed as a “sum of things”. Most of our algorithms can be easily extended to sums of geometrical objects more complex than line segments, like triangles [Rou96], ellipsoids, or any mix of objects for which a linear transformation can be easily computed.

2.2.5 Support Functions

All the sets we have seen so far are convex and represented by a set of parameters. Now we will represent convex sets by a function.

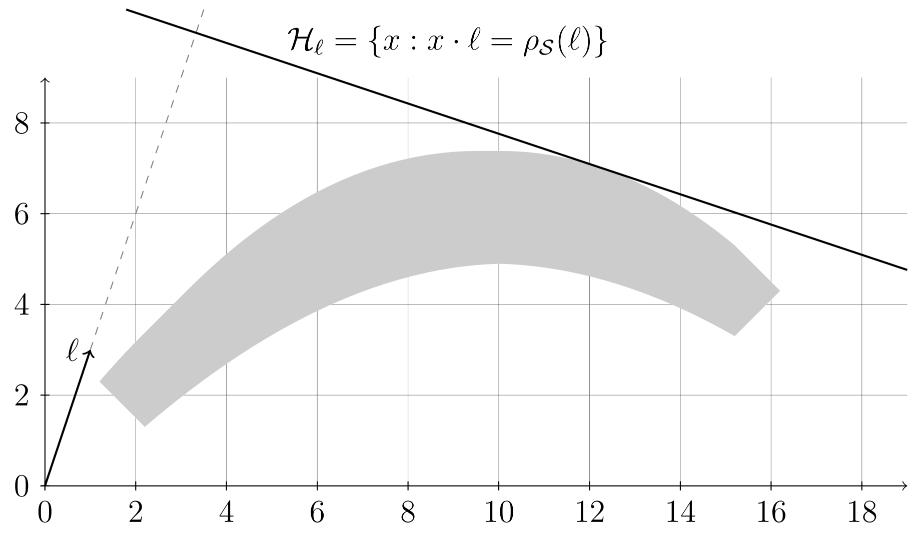
Definition 2.10 (Support function)**.** *The* support function *of a set $\mathcal{S}$, denoted $\rho_\mathcal{S}$ is defined by:*

$$\begin{array}{rccl} \rho_\mathcal{S} : & \mathbb{R}^d & \to & \mathbb{R} \cup \{-\infty, \infty\} \\ & \ell & \mapsto & \sup_{x \in \mathcal{S}} x \cdot \ell \end{array}$$

A point x of $\mathcal{S}$ such that $x \cdot \ell = \rho_\mathcal{S}(\ell)$ is called a support vector *of $\mathcal{S}$ in direction ℓ. We will denote by $\nu_\mathcal{S}(\ell)$ the set of support vectors of $\mathcal{S}$ in direction ℓ, and by $\nu_\mathcal{S}^\ell$ an element of $\nu_\mathcal{S}(\ell)$.*

Given a vector ℓ, $\rho_\mathcal{S}(\ell)$ is the solution to the following linear optimization problem: maximize the linear function $x \cdot \ell$ for x in $\mathcal{S}$. Intuitively the value $\rho_\mathcal{S}(\ell)$ tells you where to place a hyperplane orthogonal to ℓ so that is touches $\mathcal{S}$ as illustrated on Figure 2.10, such a hyperplane is called a *supporting* hyperplane.

[7]A zonogon is a zonotope in dimension 2



$\mathcal{H}_\ell = \{x : x \cdot \ell = \rho_\mathcal{S}(\ell)\}$
ℓ
0
2
4
6
8
10
12
14
16
18

Figure 2.10: A set $\mathcal{S}$ and its supporting hyperplane in direction ℓ: $\mathcal{H}_\ell$.

To any set $\mathcal{S}$, even non-convex, we can associate its support function[8], but from a support function we can only deduce the closed convex hull of this set. Indeed:

$$\mathrm{CH}(\mathcal{S}) = \bigcap_{\ell \in \mathbb{R}^d} \{x : x \cdot \ell \leq \rho_\mathcal{S}(\ell)\} \tag{2.1}$$

Thus, if $\mathcal{S}$ is convex and closed, it is uniquely determined by its support function, and it is equal to the intersection of the infinite set of half-spaces with normal vectors $\ell \in \mathbb{R}^d$ and distance values $\rho_\mathcal{S}(\ell)$. That is why support functions play an important role in convex analysis (see e.g. [BNO03, BV04, RW98]).

This representation can be seen as a $\mathcal{H}$-polytope with an uncountable number of constraints.

Like with any other representation, a transformation on $\mathcal{S}$ is translated into an operation on $\rho_\mathcal{S}$. And properties of $\mathcal{S}$ are computed using one or several calls to $\rho_\mathcal{S}$. Of course in order to manipulate this function on a computer it will be represented by an algorithm computing its values.

For the unit balls associated with the usual norms, the support function can be computed efficiently.

Proposition 2.2. *For any $k > 1$, the unit ball for the k-norm in dimension d is:*

$$\mathcal{B}_k = \left\{ x \in \mathbb{R}^d : \ \|x\|_k = \left(\sum_{i=0}^{d-1} |x_i|^k \right)^{1/k} \leq 1 \right\}$$

its support function is given by:

$$\rho_{\mathcal{B}_k}(\ell) = \|\ell\|_{\frac{k}{k-1}}$$

[8]This does not mean that for any set $\mathcal{S}$ we can easily find an algorithm computing its support function.

For $\ell \neq 0$, an associated support vector is given by:

$$\nu^{\ell}_{\mathcal{B}^d_k} = \frac{v}{\|v\|_k}$$

where v is such that $v_i \ell_i = |\ell_i|^{\frac{k}{k-1}}$ for all i.

Proof. (see A.1.1 on page 125) □

Passing to the limit, we can also deduce the support function for the unit balls $\mathcal{B}_1$and $\mathcal{B}_\infty = \left\{x \in \mathbb{R}^d : \|x\|_\infty = \max_{0 \le i < d} |x_i| \le 1\right\}$.

$$\begin{aligned} \rho_{\mathcal{B}_1}(\ell) &= \|\ell\|_\infty \\ \rho_{\mathcal{B}_\infty}(\ell) &= \|\ell\|_1 \end{aligned}$$

More complex sets can be considered using operations on these elementary convex sets, such as linear transformations, homothety, Minkowski sum, or convex hull of union of sets.

The support function of sets defined using these operations can be computed using the following properties:

Proposition 2.3. *For all compact convex sets $\mathcal{X}$, $\mathcal{Y} \subseteq \mathbb{R}^d$, for all matrices A, all positive scalars λ, and all vectors $\ell \in \mathbb{R}^d$, we have:*

$$\begin{aligned} \rho_{A\mathcal{X}}(\ell) &= \rho_{\mathcal{X}}(A^\top \ell) \\ \rho_{\lambda\mathcal{X}}(\ell) &= \rho_{\mathcal{X}}(\lambda\ell) \;=\; \lambda\rho_{\mathcal{X}}(\ell) \\ \rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) &= \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell) \\ \rho_{CH(\mathcal{X}\cup\mathcal{Y})}(\ell) &= \max(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)) \end{aligned}$$

Proof. A proof can be easily derived from the properties of the dot product. □

With just the first of these properties, and the support function of $\mathcal{B}_\infty$ and $\mathcal{B}_2$ respectively, we can already deduce the support function of a zonotope $\mathcal{Z}$ whose generators g_0, …, g_{n-1} are the columns of a matrix A, or an ellipsoid $\mathcal{E}$ with shape matrix $Q = BB^\top$:

$$\begin{aligned} \rho_{\mathcal{Z}}(\ell) = \rho_{A\mathcal{B}_\infty}(\ell) &= \|A^\top \ell\|_1 = \sum_{j=0}^{n-1} |g_j \cdot \ell| \\ \rho_{\mathcal{E}}(\ell) = \rho_{B\mathcal{B}_2}(\ell) &= \|B^\top \ell\|_2 = \sqrt{\ell^\top Q \ell} \end{aligned}$$

As for polytopes, computing the support function is exactly solving a linear program. Indeed, for a convex $\mathcal{H}$-polytope represented by a set of constraints

$Ax \le b$, the support function at ℓ is the solution to the following optimization problem:

$$\begin{aligned} &\text{maximize } x \cdot \ell \\ &\text{subject to } Ax \le b \end{aligned}$$

Thus, support functions can be computed efficiently for a quite large class of sets. Moreover, it is often possible to deduce one support vector in a given direction from the evaluation of the support function in that direction. Support vectors for unit balls can be deduced from the proof of Proposition 2.2, and an equivalent to Proposition 2.3 can be easily formulated.

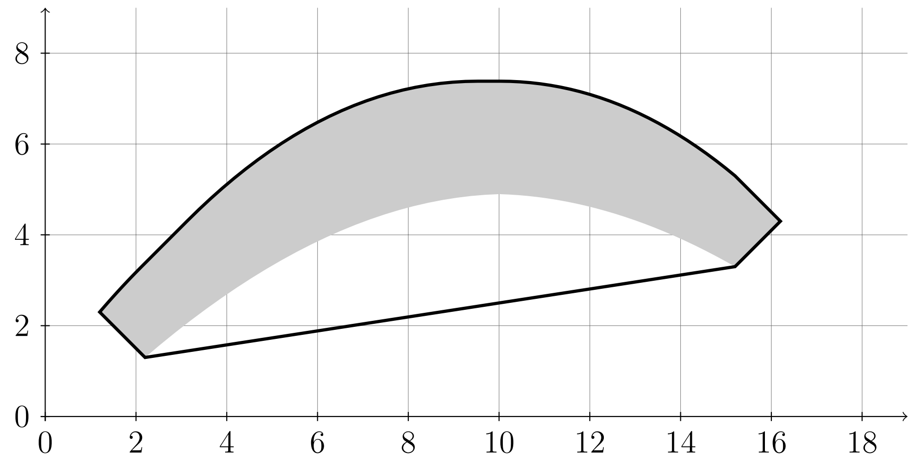


Figure 2.11: A set $\mathcal{S}$ and its convex hull $\mathrm{CH}(\mathcal{S})$.

The set $\mathcal{S}$ shown in Figures 2.3, 2.4, 2.5, 2.8, and 2.11 is:

$$\mathcal{S} = \left\{ \begin{pmatrix} 2.2 \\ 2.3 \end{pmatrix} \right\} \oplus \bigcup_{\lambda \in [0;1]} \big((1-\lambda)\mathcal{P}_0 \oplus \lambda \mathcal{P}_1 \oplus \lambda(1-\lambda)\mathcal{P}_2 \big)$$

where

$$\mathcal{P}_0 = \mathcal{B}_1 \qquad \mathcal{P}_1 = \left\{ \begin{pmatrix} 13 \\ 2 \end{pmatrix} \right\} \oplus \mathcal{B}_1 \qquad \mathcal{P}_2 = \left\{ \begin{pmatrix} 0 \\ 11 \end{pmatrix} \right\} \oplus \mathcal{B}_\infty$$

Its support function is rather easy to compute:

$$\rho_{\mathcal{S}}(\ell) = \max_{\lambda \in [0;1]} \left(\begin{pmatrix} 2.2 \\ 2.3 \end{pmatrix} \cdot \ell + \|\ell\|_\infty + \lambda \begin{pmatrix} 13 \\ 2 \end{pmatrix} \cdot \ell + \lambda(1-\lambda) \left(\begin{pmatrix} 0 \\ 11 \end{pmatrix} \cdot \ell + \|\ell\|_1 \right) \right)$$

Thus, only a few dot products and maximizing a degree 2 polynomial in one variable on an interval are needed to compute $\rho_{\mathcal{S}}(\ell)$, which allows us to represent the convex hull of $\mathcal{S}$ efficiently.

Support functions are not always that simple. One way to simplify one is to over-approximate the set it represents by a polytope, and then represent that polytope by its support function.

From equation (2.1), it is easy to see that polyhedral over-approximation of an arbitrary compact convex set can be obtained by *sampling* its support function.

Proposition 2.4. *Let $\mathcal{S}$ be a compact convex set and $\ell_1, \dots, \ell_r \in \mathbb{R}^d$ be arbitrarily chosen vectors; let us define the following halfspaces:*

$$H_k = \{x \in \mathbb{R}^d : \ell_k \cdot x \leq \rho_{\mathcal{S}}(\ell_k)\},\ k = 1, \dots, r$$

and the polyhedron

$$\mathcal{P} = \bigcap_{k=1}^{r} H_k.$$

Then, $\mathcal{S} \subseteq \mathcal{P}$. Moreover, we say that this over-approximation is tight as $\mathcal{S}$ touches all the facets of $\mathcal{P}$.

An example of such polyhedral over-approximation of a convex set can be seen in Figure 2.5. Indeed, approximating tightly a set by a polytope is equivalent to sampling its support function.

Furthermore, support vectors can be used to compute an under-approximation of an arbitrary compact convex set.

Proposition 2.5. *Let $\mathcal{S}$ be a compact convex set and $\ell_1, \dots, \ell_r \in \mathbb{R}^d$ be arbitrarily chosen non zero vectors; let us take one point x_k in each set $\nu_{\mathcal{S}}(\ell_k)$ and define the polyhedron*

$$\mathcal{P} = CH\left(\bigcup_{k=1}^{r} \{x_k\}\right).$$

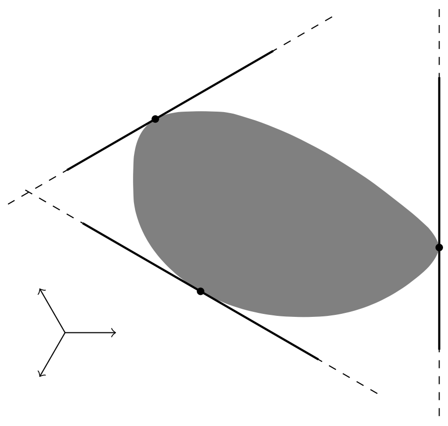
Then, $\mathcal{P} \subseteq \mathcal{S}$. Moreover, we say that this under-approximation is tight as $\mathcal{P}$ touches the boundary of $\mathcal{S}$.

Figure 2.12 shows how a compact convex set can be under- and over-approximated by two polytopes using its support vectors and support function respectively. This figures also illustrate how the normal to the facets of the under-approximation can be used to improve both approximations.

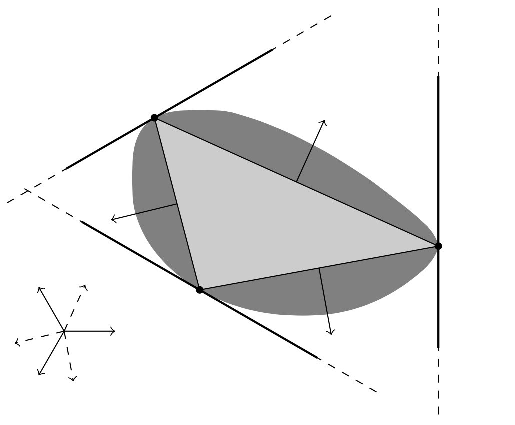
2.2.6 Other Representations

Other representations of sets have been used in the context of reachability analysis. Notably some that can represent non-convex sets. We will not present them but only give a few references to some of the main classes of sets.

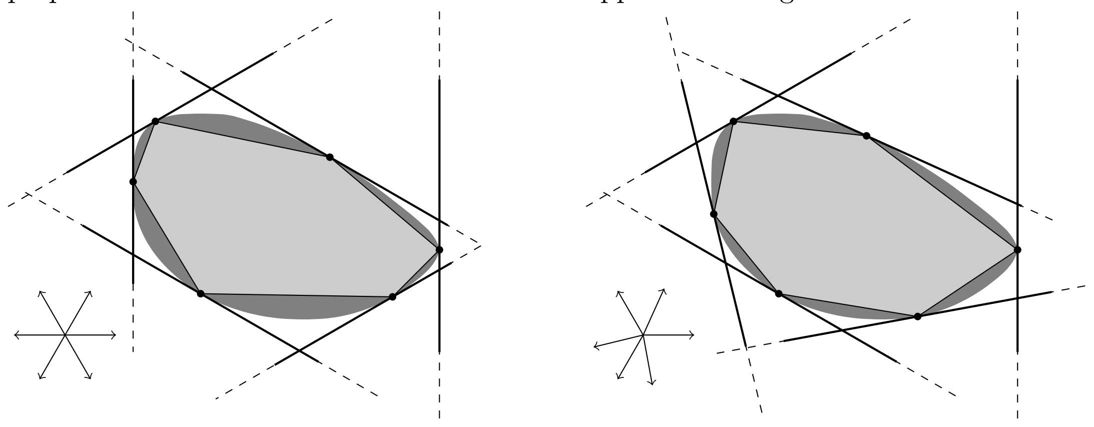
- Taylor models [Neu03] are based on Taylor expansions and interval products. They can be seen as a generalization of interval products, similar to order 0 Taylor models, and zonotopes, similar to order 1 Taylor models.



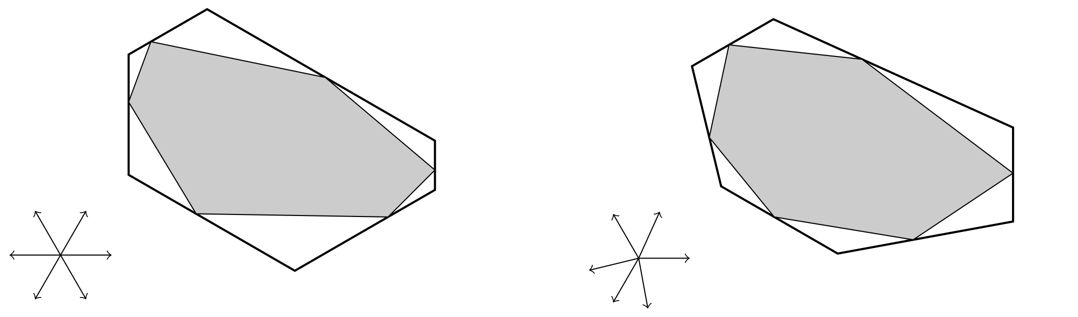
(a) A set, 3 directions, and their associated support vectors and supporting hyperplane.



(b) Normals to the facets of the under-approximation give 3 new directions.



(c) Improvement of the approximation with evenly distributed directions (left) or directions given by previous under-approximation (right).



(d) Resulting over- and under-approximations for two different sets of directions.

Figure 2.12: Refining under- and over-approximation of a closed convex set represented by its support function.

- In the level sets [Set99, MT00] framework, a set is represented by the zeros, or negative values, of a multivariate function.
- Projectagons [GM99] are non convex polytopes represented by their projections on two dimensional spaces.

2.3 Operations on Sets

In the previous section we have described how we represent sets in this thesis. In the next chapters we will manipulate these representations in order to approximate the reachable sets of linear and hybrid systems.

The main operations we will need are linear transformation, Minkowski sum, convex union, and intersection with a hyperplane. Table 2.1 informally summarizes the difficulty of performing these operations in each of the classes presented in the previous section.

	Size	$A\cdot$	$\cdot\oplus\cdot$	$\mathrm{CH}(\cdot\cup\cdot)$	$\cdot\cap\mathcal{H}$
Boxes	$2d$		$+$		
Ellipsoids	d^2+d	$+$			$+$
$\mathcal{H}$-Polytopes	$kd+k$	$*$	$-$	$-$	$+$
$\mathcal{V}$-Polytopes	kd	$+$	$-$	$+$	$-$
Zonotopes	$kd+d$	$+$	$+$		
Support Functions	NA	$+$	$+$	$+$	$-$

Table 2.1: Performing linear transformation, Minkowski sum, convex union, and intersection with a hyperplane using different classes of sets together with their representation is either easy ($+$), easy most of the time ($*$), hard ($-$), or impossible (no sign) in the class considered. Representation size is expressed as a function of d, the dimension of the system, and k, the number of constraints ($\mathcal{H}$-Polytopes), vertices ($\mathcal{V}$-Polytopes), or generators (Zonotopes).

2.3.1 Boxes

Boxes are very simple sets with a compact representation, their main disadvantage is that very few operations result in a box. Amongst the four operations considered, this class of sets is only closed by Minkowski sum.

This operation can be applied very easily by taking the sum of each interval:

$$[\underline{x};\overline{x}]\oplus[\underline{y};\overline{y}]=[\underline{x}+\underline{y};\overline{x}+\overline{y}]$$

2.3.2 Ellipsoids

An ellipsoid is the image by an invertible linear transformation of an Euclidean ball. We can extend this definition to non-invertible linear transformation using the support function of ellipsoids. Indeed, Given a point c and a positive semi-definite[9] matrix Q we can define the (possibly degenerate) ellipsoid $\mathcal{E}(c, Q)$ of center c and shape matrix Q as the set whose support function is:

$$\rho_{\mathcal{E}(c,Q)}(\ell) = c \cdot \ell + \sqrt{\ell^\top Q \ell}$$

Now that we can manipulate degenerate ellipsoids it is clear that for any linear transformation A, the image of an ellipsoid of center c and shape matrix Q by A is an ellipsoid, and its center is Ac and its shape matrix $AQA^\top$.

The class of ellipsoids is also closed by intersection with a hyperplane. Indeed, if $\mathcal{E}$ is the image of a Euclidean ball by a linear transformation A, the pre-image of $\mathcal{E}$ and any hyperplane of $\mathbb{R}^d$ by A are an Euclidean ball and a hyperplane in the pre-image of $\mathbb{R}^d$. Since the intersection of an Euclidean ball and a hyperplane is an Euclidean ball of smaller dimension, its image by A is an ellipsoid.

Unfortunately this class of sets is not closed under Minkowski sum or convex union, as illustrated on Figure 2.13.

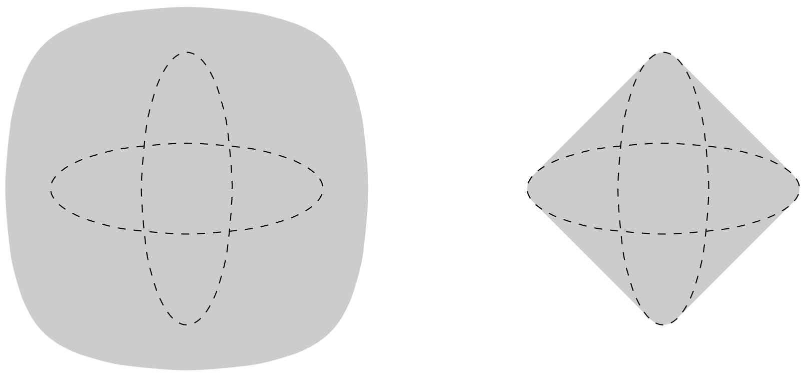


Figure 2.13: Minkowski sum (left) and Convex union (right) of two ellipsoids (dashed).

2.3.3 Polytopes

The class of polytopes is closed under all four operations we are interested in. And most of this operations can be performed efficiently on at least one of the representation if we do not impose to be in canonical form.

Intersection with a hyperplane can be done by just adding two constraints to a $\mathcal{H}$-polytope.

[9]A matrix Q is positive semi-definite iff it is equal to its transpose $Q^\top$, and $x \cdot Qx \geq 0$ for all x.

Convex union of two $\mathcal{V}$-polytopes can be done by taking the union of their vertices.

Applying a linear transformation to a $\mathcal{V}$-polytope can be done by applying it to each of its vertices. For $\mathcal{H}$-polytope it is almost as easy, the linear transformation must be applied to each of the constraints:

$$A\{x : x \cdot n \leq \gamma\} = \{x : x \cdot (A^{-1})^\top n \leq \gamma\}$$

If A is not invertible, the problem becomes much harder.

As for Minkowski sum, both representations not only suffer from the lack of efficient algorithm [Wei07, Tiw08], but also from the dramatic increase of the representation size.

2.3.4 Zonotopes

This combinatorial explosion of Minkowski sum is hidden for zonotopes represented by their generators. Indeed the sum of two zonotopes $\mathcal{Z}_1 = \langle c|g_0, \ldots, g_{n-1}\rangle$ and $\mathcal{Z}_2 = \left\langle c'|g'_0, \ldots, g'_{m-1}\right\rangle$ is:

$$\mathcal{Z}_1 \oplus \mathcal{Z}_2 = \left\langle c + c'|g_0, \ldots, g_{n-1}, g'_0, \ldots, g'_{m-1}\right\rangle$$

It can be computed in $\mathcal{O}(d)$, and the size of the output is only the sum of the size of the inputs.

A generic zonotope of order p, though it is encoded by only $pd^2 + d$ numbers, has more than $(2p)^{d-1}/\sqrt{d}$ vertices [Zas75]. Hence, zonotopes are perfectly suited for the representation of high dimensional sets.

Linear transformation is not harder, thanks to the distributivity of linear transformation over Minkowski sum. The image of a zonotope $\mathcal{Z} = \langle c|g_0, \ldots, g_{n-1}\rangle$ by a linear transformation A is:

$$A\mathcal{Z} = \langle Ac|Ag_0, \ldots, Ag_{n-1}\rangle$$

Then, the computational complexity of a linear transformation applied to a zonotope is $\mathcal{O}(p\mathfrak{L}_d)$, where $\mathfrak{L}_d$ is the complexity of the multiplication of two $d \times d$ matrices. Using standard matrix multiplication the computational complexity of the linear transformation is[10] $\mathcal{O}(pd^3)$. In comparison, if the zonotope $\mathcal{Z}$ was to be represented by its vertices, the linear transformation would require at least $(2p)^{d-1}d^{3/2}$ operations.

Unfortunately, this class of sets is not closed under convex union or intersection with a hyperplane. Indeed, both operations break central symmetry; convex union for obvious reasons, and an example with intersection with a hyperplane is illustrated on Figure 2.9, where the intersection between a cube and a plane is a triangle.

[10]Note that, theoretically, the complexity can be further reduced down to $\mathcal{O}\left(pd^{2.376}\right)$ by using a more sophisticate matrix multiplication algorithm [CW90].

2.3.5 Support Functions

As we already know from Proposition 2.3, most of these operations can be performed very easily on convex sets represented by their support functions.

Computing the value of the support function of the intersection of two sets $\mathcal{X}$ and $\mathcal{Y}$ in one direction, using only their support functions is much harder, but we can at least bound this value:

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) \leq \min\left(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)\right)$$

We even can use this approximation to represent $\mathcal{X} \cap \mathcal{Y}$ since:

$$\mathcal{X} \cap \mathcal{Y} = \bigcap_{\ell \in \mathbb{R}^d} \left\{x : x \cdot \ell \leq \min\left(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)\right)\right\}$$

Unfortunately this property is lost as soon as Minkowski sum is involved, indeed $\min\left(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)\right) + \rho_{\mathcal{Z}}(\ell)$ does not represent $\mathcal{X} \cap \mathcal{Y} \oplus \mathcal{Z}$ but $(\mathcal{X} \oplus \mathcal{Z}) \cap (\mathcal{Y} \oplus \mathcal{Z})$. Moreover, if $\mathcal{Y}$ is a hyperplane, then $\rho_{\mathcal{Y}}$ will be equal to infinity almost everywhere.

There is an exact relation [RW98] linking $\rho_{\mathcal{X}\cap\mathcal{Y}}$, $\rho_{\mathcal{X}}$, and $\rho_{\mathcal{Y}}$; unfortunately, this relation is not effective from the computational point of view.

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) = \inf_{w \in \mathbb{R}^d} \left(\rho_{\mathcal{X}}(\ell - w) + \rho_{\mathcal{Y}}(w)\right)$$

If $\mathcal{Y}$ is a hyperplane, $\mathcal{Y} = \{x : x \cdot n = \gamma\}$, then the problem can be reduced to the minimization of a unimodal function in one variable:

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) = \inf_{\lambda \in \mathbb{R}} \left(\rho_{\mathcal{X}}(\ell - \lambda n) + \lambda\gamma\right)$$

The case of intersection with a hyperplane will be investigated in detail in Section 8.3, where we will reduce the problem to the minimization of a slightly different function.

In Table 2.1, the representation size of support functions is not indicated. In fact, it can be almost anything and is not relevant. What is relevant is the complexity of its algorithm.

Part I

Linear Systems

Chapter 3

State of the Art in Linear Systems Reachability

Résumé : *Nous présentons ici quelques méthodes pour le calcul de l'ensemble atteignable des systèmes linéaires sur un temps borné. Ces méthodes sont toutes basées sur la discrétisation de l'équation différentielle $\dot{x} = Ax(t)+u(t)$ en une relation de récurrence sur des ensembles : $\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$. Chaque Ω_i contient l'ensemble $\mathcal{R}_{[i\delta;(i+1)\delta]}$ où δ est le pas de discrétisation. L'ensemble atteignable par le système sur $[0; N\delta]$ peut alors être sur-approximé par les N premiers termes de cette séquence.*
Malheureusement les Ω_i sont en général difficilement calculables, c'est pourquoi la plupart des outils produisent une approximation $\widetilde{\Omega}_i$ de Ω_i à chaque étape. L'accumulation et la propagation de ces approximations peuvent rendre les $\widetilde{\Omega}_i$ inutilisables.

In this chapter we present some approaches, related to ours, for computing the set of points reachable by a linear system within a time horizon T.

As stated in Chapter 1, the approach we take is related to simulation, but instead of having points evolving in the state space we have sets. It is sometimes possible to transform the differential equation on the state variables in an equation on the parameters of the set representation [KV06]. Integrating the latter will result in a sequence of sets approximating each of the $\mathcal{R}_{t_i}(\mathcal{X}_0)$, where the t_i are the times corresponding to the integration steps. Then, one may miss some interesting behaviors occurring during any $]t_i; t_{i+1}[$.

Instead, the original differential equation (1.1) is discretized and the reachable set $\mathcal{R}_{[0;T]}$ is over-approximated by the union of a finite number N of sets from the chosen class, that is why we also speak about reachable sets in plural. Each of these sets is an approximation of $\mathcal{R}_{[i\delta;(i+1)\delta]}$, where i is an integer between 0 and $N-1$, and $\delta = \frac{T}{N}$ is the time step. As for simulation, $\mathcal{R}_{[0;t]}$ is easier to compute accurately if t is small.

In order to compute these sets we use the fact that the composition of two reachability operators $\mathcal{R}_{[t_0;t_1]}$ and $\mathcal{R}_{[t'_0;t'_1]}$ behave isomorphically[1] to the addition of intervals, that is:

$$\mathcal{R}_{[t_0;t_1]}\left(\mathcal{R}_{[t'_0;t'_1]}(\mathcal{Y})\right) = \mathcal{R}_{[t_0+t'_0;t_1+t'_1]}(\mathcal{Y})$$

In particular, for any i in $\mathbb{N}$:

$$\mathcal{R}_{\delta}\left(\mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{Y})\right) = \mathcal{R}_{[(i+1)\delta;(i+2)\delta]}(\mathcal{Y})$$

This is how Algorithm 3.1 computes the reachable sets for time-invariant (not necessarily linear) systems.

Algorithm 3.1 Algorithmic scheme for reachability analysis.

```
Input: $\mathcal{X}_0$, $T$ the time horizon, $N$ the number of steps.
Output: $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$ as a union of $N$ sets.
1: $\delta \leftarrow \frac{T}{N}$
2: $\Omega_0 \leftarrow \mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$
3: for $i$ from 0 to $N-2$ do
4:     $\Omega_{i+1} \leftarrow \mathcal{R}_{\delta}(\Omega_i)$
5: end for
6: return $\Omega_0 \cup \cdots \cup \Omega_{N-1}$
```

Then we need a way to approximate $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$ by a set in the chosen class and a map $\overline{\mathcal{R}}_{\delta}$ operating on this class such that $\mathcal{R}_{\delta}(\mathcal{Y}) \subseteq \overline{\mathcal{R}}_{\delta}(\mathcal{Y})$ such maps are called extensions of $\mathcal{R}_{\delta}$ over a particular class.

We will present here some of the classical methods, computing the bounded time reachable set of a continuous linear system, based on Algorithm 3.1. We will first show how time discretization can be done for autonomous system and non-autonomous systems without specifying the class of sets used. Then the case of each of the classes of sets defined in the previous chapter will be explored in more detail.

3.1 Autonomous System

We are here interested in the reachable sets of a dynamical system given by an autonomous *linear* ordinary differential equation (ODE)[2].

$$\dot{x}(t) = Ax(t) \tag{3.1}$$

[1]If the system is time variant, $\dot{x} = f(x, u, t)$, a similar approach is possible but slightly less readable. But any time varying system can be transformed into a time-invariant one by adding one variable, representing time, to the state space.

[2]Some autonomous systems are expressed as $\dot{x}(t) = Ax(t) + b$, they can be rewritten using one additional state variable with null derivative as $\dot{x}(t) = A'x(t)$

Solutions for equation (3.1) have the following form:

$$x(t) = e^{tA}x_0$$

From this expression, it is easy to deduce that for any set $\mathcal{Y}$, $\mathcal{R}_\delta(\mathcal{Y})$ is $e^{\delta A}\mathcal{Y}$. Thus the sequence of reachable sets we want to compute is defined by the following recurrence relation:

$$\Omega_{i+1} = e^{\delta A}\Omega_i$$

Since the class of boxes is not closed under linear transformations it can not be used to compute the terms of this sequence, in contrast to ellipsoids, polytopes, zonotopes and convex sets represented by their support functions.

But first, $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$ must be approximated by a set Ω_0. Indeed this set is not convex and we only consider here classes of convex sets. We will here briefly outline a classical method used to perform this approximation [Dan00, Gir04].

Our main interest is in over-approximation: Ω_0 must contain $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, and in particular it must contain $\mathcal{R}_0(\mathcal{X}_0)$ and $\mathcal{R}_\delta(\mathcal{X}_0)$. Thus, we will start from the convex hull of $\mathcal{R}_0(\mathcal{X}_0) \cup \mathcal{R}_\delta(\mathcal{X}_0)$. Let us call this set $\widetilde{\Omega}$.

Then an over-approximation Ω_0 can be computed by finding an upper-bound α_δ for the Hausdorff distance between $\widetilde{\Omega}$ and $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, and using Proposition 2.1.

$$\mathcal{R}_{[0;\delta]}(\mathcal{X}_0) \subseteq \mathrm{CH}\left(\mathcal{R}_0(\mathcal{X}_0) \cup \mathcal{R}_\delta(\mathcal{X}_0)\right) \oplus \mathcal{B}(\alpha_\delta) = \Omega_0$$

where $\mathcal{B}(\alpha_\delta)$ is the ball of radius α_δ. The reader is referred to [Gir04] or to Lemma 5.1 for more details on different ways to find such a bound.

An alternative way to compute a polyhedral over-approximation of $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, described in [CK98, Chu99], is to solve several non-linear optimization problems over it, in other words, to sample its support function.

3.2 Non-Autonomous System

We consider now a non-autonomous linear system[3]:

$$\dot{x}(t) = Ax(t) + u(t) \tag{3.2}$$

with $x(0)$ in $\mathcal{X}_0$ and $u(t)$ in $\mathcal{U}$. We assume that $\mathcal{U}$ is compact and convex, not necessarily full-dimensional, and only consider measurable input functions u.

Solutions for equation (3.2) have the following form:

$$x(t) = e^{tA}x_0 + \int_0^t e^{(t-s)A}u(s)\ ds \tag{3.3}$$

If we remark that $\int_0^t e^{(t-s)A}u(s)\ ds$ is equal to $\mathcal{R}_t(\{0\})$, we can decompose the operator $\mathcal{R}_\delta$ into:

$$\mathcal{R}_\delta : \mathcal{Y} \mapsto e^{\delta A}\mathcal{Y} \oplus \mathcal{R}_\delta(\{0\})$$

[3]non-autonomous systems are often expressed as $\dot{x}(t) = Ax(t) + Bu(t)$, they can be rewritten as $\dot{x}(t) = Ax(t) + v(t)$ by constraining $v(t)$ to be in $B\mathcal{U}$.

and express the set of points reachable during $[0;\delta]$ by:

$$\mathcal{R}_{[0;\delta]}(\mathcal{X}_0) = \bigcup_{s\in[0;\delta]} \left(e^{sA}\mathcal{X}_0 \oplus \mathcal{R}_s(\{0\})\right)$$

Two things remain to be done:

- Find a way to express, or over-approximate, $\mathcal{R}_\delta(\{0\})$ by a set $\mathcal{V}$ in the class of sets chosen for computation. Leading to the set-valued discretized system:

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V} \tag{3.4}$$

  with $\Phi = e^{\delta A}$

- Initialize this sequence of sets by over-approximating $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$ by a set Ω_0.

We will then have to compute the N first sets of the sequence defined by (3.4) and Ω_0 in order to over-approximate the reachable set of the original system between times 0 and $N\delta$.

There are several ways to approximate $\mathcal{R}_\delta(\{0\})$, and $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$. In the following we summarize one such approximation scheme taken from [Gir05].

Lemma 3.1 ([Gir05])**.** *Given a norm $\|\cdot\|$, let $\mu = \sup_{u\in\mathcal{U}}\|u\|$ then:*

$$\mathcal{R}_\delta(\{0\}) \subseteq \mathcal{V} = \mathcal{B}(\beta_\delta)$$

where $\mathcal{B}(\beta_\delta)$ is the ball of radius β_δ associated to $\|\cdot\|$, and $\beta_\delta = \frac{e^{\delta\|A\|-1}}{\|A\|}\mu$.

Furthermore, if $\mathcal{U} = \mathcal{B}(\mu)$, for any set $\mathcal{S}$:

$$d_H\left(\mathcal{R}_\delta(\mathcal{S}), \Phi\mathcal{S}\oplus\mathcal{V}\right) \le \mu\|A\|e^{\delta\|A\|}\delta^2$$

In order to over-approximate $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, one can use the same kind of techniques used for the autonomous case.

Lemma 3.2 ([Gir05][4])**.** *Given a norm $\|\cdot\|$, let $\mu = \sup_{u\in\mathcal{U}}\|u\|$ and $\nu = \sup_{x\in\mathcal{X}_0}\|x\|$ then:*

$$\mathcal{R}_{[0;\delta]}(\mathcal{X}_0) \subseteq \Omega_0 = CH(\mathcal{X}_0 \cup \Phi\mathcal{X}_0) \oplus \mathcal{B}(\alpha_\delta + \beta_\delta)$$

where $\alpha_\delta = \left(e^{\delta\|A\|} - 1 - \delta\|A\|\right)\nu$.

Furthermore, if $\mathcal{U} = \mathcal{B}(\mu)$:

$$d_H\left(\mathcal{R}_{[0;\delta]}(\mathcal{X}_0), CH(\mathcal{X}_0\cup\Phi\mathcal{X}_0)\oplus\mathcal{B}(\alpha_\delta+\beta_\delta)\right) \le \delta\|A\|e^{\delta\|A\|}\left(\frac{\mu}{\|A\|} + \left(\frac{1}{2}+\delta\right)\nu\right)$$

Using these two lemmata, one can initialize the sequence defined by equation (3.4) and over-approximate the reachable set $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$ by $\bigcup_{i=0}^{N-1}\Omega_i$. Furthermore, the following theorem tells us that the distance between these two sets vanishes when δ tends toward 0.

[4] In [Gir05], this lemma is stated for a zonotope representation and thus use an over-approximation of $\mathrm{CH}(\mathcal{X}_0 \cup \Phi\mathcal{X}_0)$ while ensuring the convergence of the approximation.

Theorem 3.1 ([Gir05]). *Let Ω_0, ..., Ω_{N-1}, be the N first terms of the sequence defined by (3.4) using $\mathcal{V}$ and Ω_0 as defined in Lemmata 3.1 and 3.2 respectively. Then:*

$$\mathcal{R}_{[0;T]}(\mathcal{X}_0) \subseteq \bigcup_{i=0}^{N-1} \Omega_i$$

Furthermore, if $\mathcal{U} = \mathcal{B}(\mu)$:

$$d_H\left(\mathcal{R}_{[0;T]}(\mathcal{X}_0), \bigcup_{i=0}^{N-1} \Omega_i\right) \leq \delta\|A\|e^{T\|A\|}\left(\frac{2\mu}{\|A\|} + \left(\frac{1}{2} + \delta\right)\nu\right)$$

3.3 Reachability of the Discretized System

We now have to compute the N first sets of sequence (3.4):

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

Algorithm 3.2 makes a direct use of this recurrence relation.

Algorithm 3.2 Reachability of linear systems with input.

```
Input: The matrix $\Phi$, the sets $\Omega_0$ and $\mathcal{V}$, an integer $N$.
Output: The first $N$ terms of the sequence defined in equation (3.4).
1: for $i$ from 0 to $N-2$ do
2:     $\Omega_{i+1} \leftarrow \Phi\Omega_i \oplus \mathcal{V}$
3: end for
4: return $\{\Omega_0, \ldots, \Omega_{N-1}\}$
```

We do not worry here about the discretization procedure which induces an inevitable over-approximation (which can be provably reduced by reducing the time step). That is why the only operations we need for this algorithm to work are linear transformations and Minkowski sum. Among the classes of sets presented in Chapter 2, only polytopes, zonotopes, and convex sets represented by their support functions are closed under these operations; hence boxes and ellipsoids are excluded (see Table 2.1).

Polytopes One implementation of this method is given in the *Multi-Parametric Toolbox* (MPT) [KGBM04, KGB04]. The matrix Φ is $e^{\delta A}$. Thus it is invertible and the linear transformation can be computed computed linearly in the representation size of Ω_i, whether $\mathcal{H}$-polytopes or $\mathcal{V}$-polytopes are used. Unfortunately, the other operation involved, Minkowski sum, is hard to compute for both representations. Moreover, Minkowski sum increases the number of constraints or vertices defining a polytope, and makes linear transformation progressively harder to compute as we proceed in the iteration.

Indeed, if both Ω_0 and $\mathcal{V}$ are cubes, after k discrete steps Ω_{k-1} may have more than $\frac{(2k)^{d-1}}{\sqrt{d}}$ vertices and more than $2\binom{kd}{d-1}$ facets of dimension $d-1$. Hence, except for small dimension and small discrete-time horizon, or equivalently, large time step, Algorithm 3.2 can not be reasonably used with polytopes.

Zonotopes Unlike polytopes, Minkowski sum is easy to compute for zonotopes and does not dramatically increase the number of generators. If Ω_0 and $\mathcal{V}$ are of order p and q respectively, then after k discrete steps the representation size of Ω_{k-1} is of order $p+(k-1)q$.

Thus computing the N first sets of sequence (3.4) using zonotopes can be done in time $\mathcal{O}\left((Np+N^2q)\mathfrak{L}_d\right)$, with space complexity $\mathcal{O}\left((Np+N^2q)d^2\right)$. Except for very large discrete-time horizons, or very small time steps, this computation is tractable. Unfortunately, using zonotopes, the output of Algorithm 3.2 consists of a list of high order zonotopes which are hard to handle for further use.

Support Functions The case of support functions is special. Using this representation, Algorithm 3.2 returns a list of N functions. Computing their evaluation tree can be done in $\mathcal{O}(N)$ time and memory.

Support functions used in this way can be seen as *lazy* computation. All will be done when we will need the value of $\rho_{\Omega_k}(\ell)$ which has time complexity of $\mathcal{O}\left(k(d^2+\mathfrak{C}^t_{\rho_\mathcal{V}})+\mathfrak{C}^t_{\rho_{\Omega_0}}\right)$, where $\mathfrak{C}^t_{\rho_\mathcal{V}}$ and $\mathfrak{C}^t_{\rho_{\Omega_0}}$ are the time complexity of $\rho_\mathcal{V}$ and ρ_{Ω_0} respectively. And the time complexity of evaluating all the ρ_{Ω_k} in one direction ℓ is $\mathcal{O}\left(N^2(d^2+\mathfrak{C}^t_{\rho_\mathcal{V}})+N\mathfrak{C}^t_{\rho_{\Omega_0}}\right)$.

3.4 Approximate Reachability of the Discretized System

The fact that some classes of sets are not closed under both linear transformation and Minkowski sum, or that the application of these operations produces progressively more complex sets, motivates the use of over-approximation (*wrapping*), replacing Ω_i by a superset $\overline{\Omega}_i$ which is less complex. Recurrence (3.4) is thus replaced by

$$\overline{\Omega}_{i+1} = \overline{\Phi\overline{\Omega}_i \oplus \mathcal{V}} \tag{3.5}$$

where the overline denotes an over-approximation.

To realize (3.5) we need an approximation procedure $\text{Approx}_{\Phi\cdot\oplus\cdot}()$ which takes as input two sets $\mathcal{X}$ and $\mathcal{Y}$ and returns an over-approximation of $\Phi\mathcal{X}\oplus\mathcal{Y}$. The same can be done for under-approximation.

Before briefly explaining some of the most common approximation procedures, we will investigate the implication of the use of an approximation procedure in Algorithm 3.3.

Algorithm 3.3 Approximate Reachability of linear systems with input.

Input: The matrix Φ, the sets Ω_0 and $\mathcal{V}$, an integer N.
Output: The first N terms of the sequence defined in equation (3.5).

```
1: for $i$ from 0 to $N-2$ do
2:     $\widetilde{\Omega}_{i+1} \leftarrow \text{Approx}_{\Phi\cdot\oplus\cdot}\left(\widetilde{\Omega}_i, \mathcal{V}\right)$
3: end for
4: return $\{\widetilde{\Omega}_0, \ldots, \widetilde{\Omega}_{N-1}\}$
```

Remark. *What is important here is the* shape *of the sets considered, not their position in the state space. That is why in the following, figures of individual sets and their approximation might not be given together with a reference system.*

3.4.1 Wrapping Effect

In Algorithm 3.3, $\widetilde{\Omega}_{i+1}$, the approximation of Ω_{i+1} is computed using $\text{Approx}_{\Phi\cdot\oplus\cdot}()$ from $\widetilde{\Omega}_i$ which is already an approximation of Ω_i. This accumulation and propagation of approximation errors can lead, if uncontrolled, to an exaggerated growth of the enclosures.

Because of these successive *wrappings*, even for stable systems, Algorithm 3.3 can produce a sequence of exponentially growing sets. Moreover, since the number of discrete sets, N, is proportional to the inverse of the timestep, δ, Theorem 3.1 is lost, and reducing the timestep does not necessarily reduce the approximation error.

This effect was first observed in the early 1960s with the use of interval methods in the field of validated integration of ODEs. Since then, various strategies have been designed to try to control it.

Unfortunately, the error introduce by an approximation method is often very hard to evaluate, or even define. One good criterion would be the Hausdorff distance between the exact set Ω_i and its enclosure $\widetilde{\Omega}_i$ normalized by the diameter of Ω_i, where the diameter of a set $\mathcal{S}$ is defined to be $D_{\mathcal{S}} = \sup_{x,y\in\mathcal{S}^2} \|y - x\|$. But this value is very hard to bound in a useful way, mainly because of the linear transformation.

3.4.2 Approximation Strategies

In this section we investigate which approximation functions, $\text{Approx}_{\Phi\cdot\oplus\cdot}()$, have been used for reachability analysis with each class of sets presented in Chapter 2.

3.4.2.1 Interval Hulls

The class of boxes is a very simple class, as such it has some advantages: the *best* over-approximation of a set by a box can be easily defined as the intersection of all possible box over-approximations, in other words, its interval hull.

The ultimate $\text{Approx}_{\Phi\cdot\oplus\cdot}()$ function is the function that to $\mathcal{X}$ and $\mathcal{Y}$ associates $\Box(\Phi\mathcal{X} \oplus \mathcal{Y})$.

Let us analyse how it behaves on a very simple two dimensional example:

$$\begin{aligned} \Phi &= \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{pmatrix} \\ \Omega_0 &= \mathcal{B}_\infty = [-1;1] \times [-1;1] \\ \mathcal{V} &= \{0\} \end{aligned}$$

Φ is just the matrix of rotation by an angle $(\pi/4)$. For all k, Ω_{2k} is Ω_0, and Ω_{2k+1} is $\Phi\Omega_0$.

Using Algorithm 3.3 with the best possible approximation function we get:

$$\overline{\Omega}_k = 2^{k/2}\Omega_0$$

as illustrated on Figure 3.1.

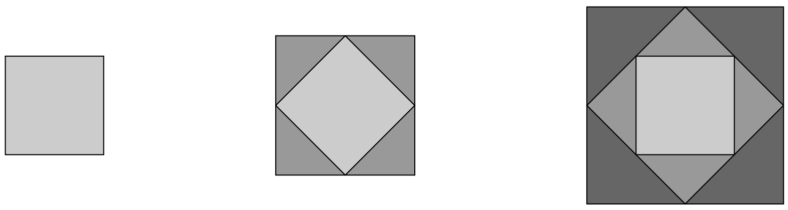


Figure 3.1: Successive wrapping induced by a rotation of $\frac{\pi}{4}$. From left to right, in light grey: Ω_0, Ω_1, and Ω_2; in grey: $\widetilde{\Omega}_1$ and $\Phi\widetilde{\Omega}_1$; in dark grey $\widetilde{\Omega}_2$.

This figure also illustrates the fact that the over-approximation at one step can be induced by the propagation of the approximation error introduced at the previous step only.

The analysis performed with interval hulls is closely linked to the choice of the direction of the axis. One improvement is to perform an orthonormal change of variables at each step in order to reduce the approximation error. This pre-conditioning of the class of boxes leads to the class of *oriented rectangular hulls* (ORH) [SK03].

Closure by Minkowski sum is lost, but a good orientation strategy can dramatically improve the analysis. The QR-method [Loh88] uses a QR decomposition of Φ and is very effective for pure rotations, but fails exponentially on some examples.

3.4.2.2 Ellipsoids

Contrary to the class of Boxes, the class of Ellipsoids is closed under linear transformation but not Minkowski sum.

Various ellipsoidal calculi [RST02, KV06] overcome this problem, but choosing the best ellipsoidal enclosure with respect to Hausdorff distance at each step gives no guarantees on the quality of the over-approximation after several steps. Instead, one[5] can ensure that the boundary of each $\widetilde{\Omega}_i$ touches the boundary of the set it over- or under-approximate in at least one direction, the approximation is then said to be *tight*.

Indeed if one can tightly approximate the sum of two ellipsoids in any given direction, and if $\widetilde{\Omega}_i$ is a tight approximation of Ω_i in direction ℓ_i, then one only has to tightly approximate the sum of $\Phi\widetilde{\Omega}_i$ and $\mathcal{V}$ in direction $(\Phi^{-1})^\top \ell_i$ to ensure the tightness of $\widetilde{\Omega}_{i+1}$.

Tightness might be an interesting property, but it is not always relevant, even more if it is not always in the same direction. Indeed tightness of each individual $\widetilde{\Omega}_i$ guarantees the tightness of the reachable tube but gives no information about its projection on the state space: the reachable set.

Moreover, since ellipsoids have a constrained shape, forcing tightness in one direction may lead to rough over-approximation in other directions. In practice, approximating tightness in one random direction instead of ℓ_i at each step, may produce better (with respect to Hausdorff distance) approximations, as illustrated on Figure 3.2.

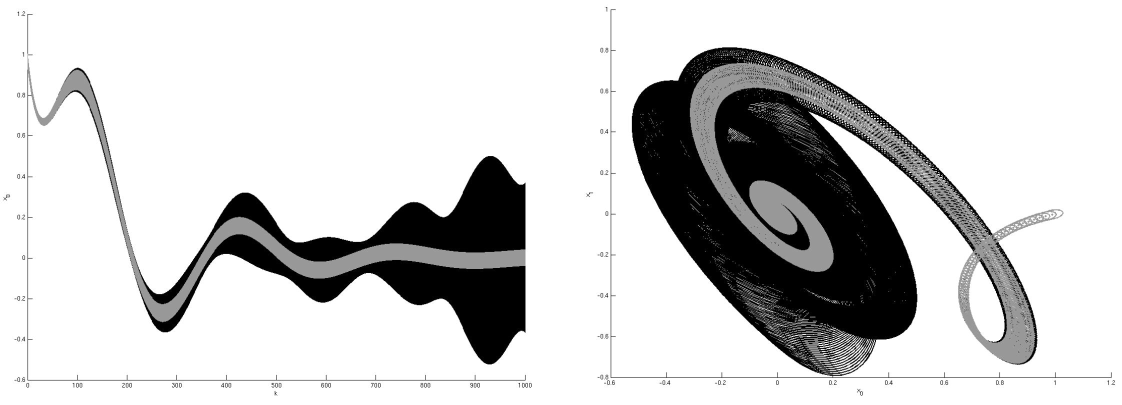


Figure 3.2: Over-approximation of the reachable set of a 5 dimensional system using tight (in black) or non-tight (in grey) approximations, obtained with the Ellipsoidal Toolbox [KV06] and a modified version for the non-tight approximation. On the left: range of the first variable as a function of the step k. On the right: projection of the reachable set on the first two variables. For more details see Appendix B.

3.4.2.3 Polytopes

Here the approximation procedure is not introduced to overcome the lack of closure under one operation of the class considered, but to avoid the uncontrolled growth of the number of parameters needed to represent the sets involved.

[5]This is what is done in the Ellipsoidal Toolbox, which exists as a standalone [KV06] or as part of the MPT [KGB04].

The most common technique is to over-approximate $\Phi\widetilde{\Omega}_i \oplus \mathcal{V}$ by a polytope whose facets have the same normals has the facets of $\Phi\widetilde{\Omega}_i$. We say that we *push* or *lift* the facets of $\Phi\widetilde{\Omega}_i$ until it includes $\Phi\widetilde{\Omega}_i \oplus \mathcal{V}$. One simple way to do it, is to represent $\mathcal{V}$ by its support function $\rho_{\mathcal{V}}$ and to replace every facet of the form:

$$\{x : x \cdot n \leq \gamma\}$$

by

$$\{x : x \cdot n \leq \gamma + \rho_{\mathcal{V}}(n)\}$$

It is also possible to perform an exact computation and then remove some of the constraints, or vertices, of the resulting polytope [Fre08].

Another solution is to restrict the $\widetilde{\Omega}_i$ to be in a smaller class. As an example, using template polyhedra, or only considering initial sets of small dimension [Han05].

3.4.2.4 Zonotopes

From Section 3.3 we know that zonotopes allow for a quite efficient implementation of the exact scheme. But the number of generators of Ω_i grow linearly with the i, which leads to a time complexity quadratic in N, the discrete time horizon.

In order to tackle this problem one must limit the number of generators. Over-approximating a zonotope by reducing its number of generators is a hard problem. Various strategies [Küh98, Gir05, ASB09] use the same kind of ideas, illustrated on Figure 3.3:

- The set $\mathcal{G}$ of generators is partitioned into a set $\mathcal{K}$ of generators that will be kept, and a set $\mathcal{A}$. So that the zonotope $\mathcal{Z}_{\mathcal{G}}$ generated by $\mathcal{G}$ is the sum of $\mathcal{Z}_{\mathcal{K}}$ and $\mathcal{Z}_{\mathcal{A}}$, generated respectively by $\mathcal{K}$ and $\mathcal{A}$.

- The zonotope $\mathcal{Z}_{\mathcal{A}}$ is over-approximated by an order one zonotope $\mathcal{Z}_{\text{new}}$, which can be either a box or with directions chosen from $\mathcal{K}$.

- Then $\mathcal{Z}_{\mathcal{G}}$ is over-approximated by $\mathcal{Z}_{\mathcal{K}} \oplus \mathcal{Z}_{\text{new}}$.

These approaches have some serious limitations.

Quality of Approximation As already stated, reducing the number of generators without introducing too much error is a hard problem. Even defining what the *best* approximation is can be challenging. Still, one can expect some kind of local optimality. But when approximating a zonotope $\mathcal{Z}$ by a zonotope $\overline{\mathcal{Z}}$ using one of the methods described previously, there is no guarantee that there is no other zonotope $\widetilde{\mathcal{Z}}$, whose generators have the same directions as the generators of $\overline{\mathcal{Z}}$ but with different lengths, such that $\mathcal{Z} \subset \widetilde{\mathcal{Z}} \subset \overline{\mathcal{Z}}$. Indeed if there is a generator h in $\mathcal{A}$ and a generator g in $\mathcal{K}$ such that h and g are in the same orthant[6], the overall approximation can be improved by using g in the approximation of h as illustrated on Figure 3.4 and 3.5.

[6]If $\mathcal{Z}_{\text{new}}$ is not a cube, one can use the same idea with one change of variables.

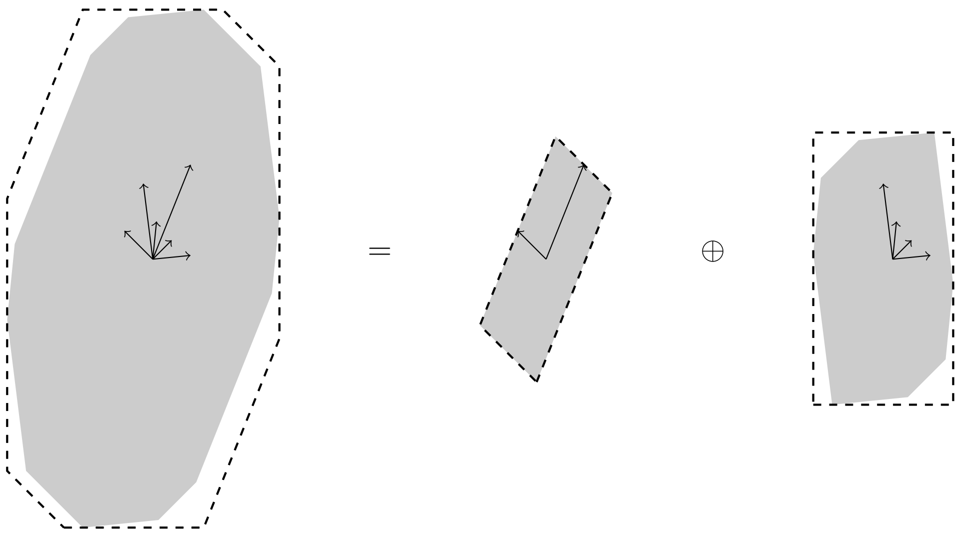


Figure 3.3: Over-approximation of a zonotope by a lower order zonotope.

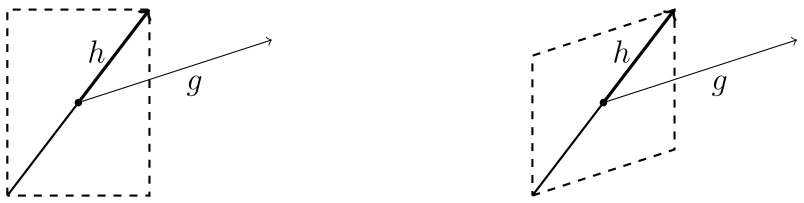


Figure 3.4: A segment generated by vector h is approximated by its interval hull (left). Using a vector g in the same orthant leads to a strictly better approximation (right).

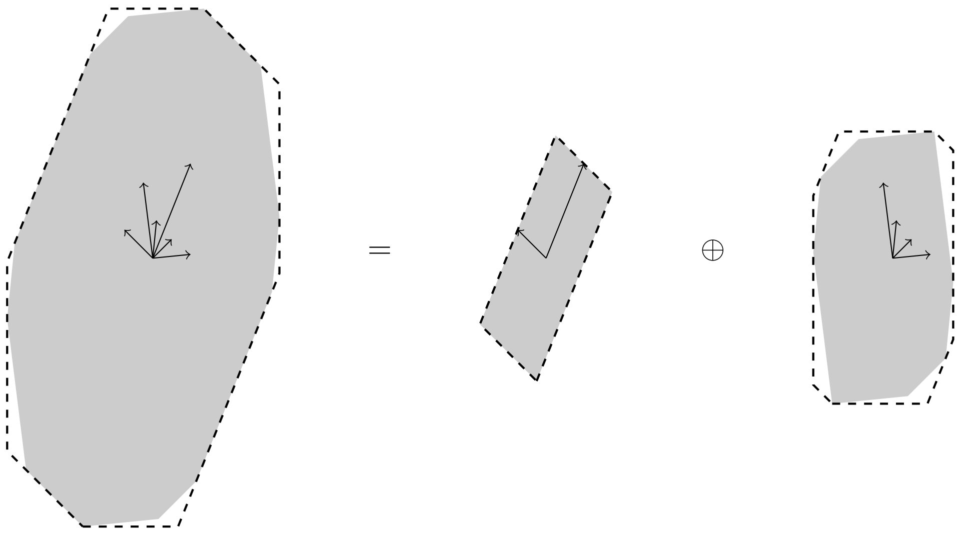
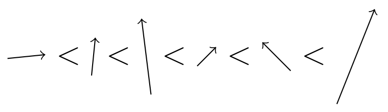


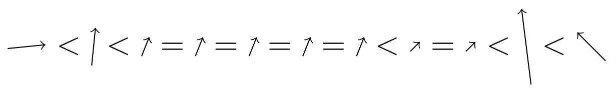
Figure 3.5: Improving the approximation in Figure 3.3.

Robustness The generators in $\mathcal{A}$ are chosen for individual properties, without looking at the whole set of generators. If one replaces one generator g in a zonotope $\mathcal{Z}$ by $\frac{1}{n}g+\epsilon_1, \ldots, \frac{1}{n}g+\epsilon_n$, where the ϵ_i are random vectors whose norm is very small, it will not change $\mathcal{Z}$ a lot but may dramatically change its over-approximation.

As an example, in [Gir05], the generators are sorted with respect to $\|g\|_1 - \|g\|_\infty$, only the biggest vectors are kept. Using the zonotope in Figure 3.5, we get:



By *cutting* some generators in several parts, we can reorder them arbitrarily.



Even if both sets of vectors generate the same zonotopes, we get two very different over-approximation as illustrated on Figure 3.6.

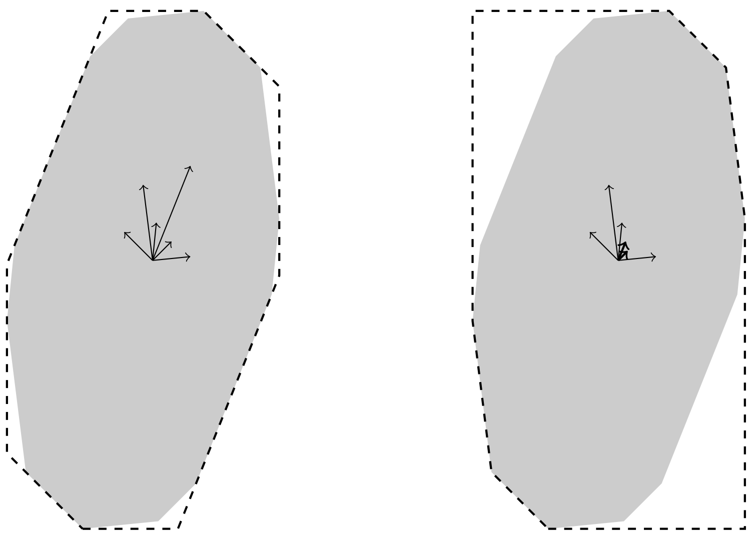


Figure 3.6: Splitting some generators of a zonotope may dramatically change its over-approximation, without changing its shape.

In order to avoid this problem one might chose the generators independently from their length. As an example, instead of sorting the generators with respect to $\|g\|_1 - \|g\|_\infty$, one might sort them with respect to $\frac{\|g\|_1 - \|g\|_\infty}{\|g\|_2}$; but if this increase robustness it might decrease the quality of approximation in general.

It might be more interesting, but much more costly, to form clusters of generators with similar directions.

To the best of the author's knowledge no approximation strategy solves any of these issues. Since zonotopes are involved in numerous fields it would be interesting to try to solve them.

3.4.2.5 Support Functions

Support functions have been mainly used to represent the initial set Ω_0 or $\mathcal{V}$, sometimes without being named [CK98, Chu99, Dan00].

Reachability analysis of linear systems based on the use of support functions was explicitly proposed in [Var98]. The support functions of the reachable sets are computed recursively using the relation:

$$\rho_{\Omega_{i+1}}(\ell) = \rho_{\Omega_i}(\Phi^\top \ell) + \rho_{\mathcal{V}}(\ell).$$

Then, a polyhedral over-approximation $\overline{\Omega}_i$ is computed by first choosing a sequence $\ell_{0,0}, \dots \ell_{0,r-1}$ of initial directions, then for each direction, we have:

$$\begin{aligned} \ell_{i,k} &= \left(\Phi^\top\right)^{-1} \ell_{i-1,k} & \\ \rho_{i,k} &= \rho_{i-1,k} + \rho_{\mathcal{V}}(\ell_{i,k}) & \qquad \rho_{0,k} = \rho_{\Omega_0}(\ell_{0,k}) \end{aligned}$$

And we get the following over-approximations:

$$\overline{\Omega}_i = \bigcap_{k=0}^{r-1} \left\{ x \in \mathbb{R}^d : \ell_{i,k} \cdot x \le \rho_{i,k} = \rho_{\Omega_i}(\ell_{i,k}) \right\}$$

where $\ell_{i,k} = ((\Phi_\delta{}^\top)^{-1})^i \ell_{0,k}$. Note that $\overline{\Omega}_i$ is defined using the exact values of the support function of Ω_i in the directions $\ell_{i,k}$. As a consequence, Ω_i touches all the faces of $\overline{\Omega}_i$, and the approximation is said to be *tight*.

Then, if a support function is needed one can derive one from the polyhedral representation of $\overline{\Omega}_i$ by solving a linear program.

But the quality of these over-approximations might suffer from the evolution of the directions $\ell_{i,k}$. Let us fix $\ell_{0,k}$. The directions used for the approximation of Ω_i are then $\ell_{i,k} = ((\Phi^\top)^{-1})^i \ell_{0,k}$. For simplicity, we assume that the eigenvalue of $(\Phi^\top)^{-1}$ with largest modulus is real and denote by ℓ^* the associated eigenvector. Then all the vectors $\ell_{i,1}, \dots, \ell_{i,r}$ tend to point towards the direction of ℓ^* when i grows. This means that the polyhedral over-approximation $\overline{\Omega}_i$ is likely to be ill-conditioned for large values of i

3.4.3 Curse of Dimensionality

Working in high dimension, or just in not small dimension poses a number of problems that makes everything more difficult.

First there is a combinatorial problem. As an example, the Minkowski sum of two polytopes introduce many more vertices and facets in high dimension.

Secondly, and more importantly, the quality of any approximation decreases dramatically. Consider the unit cube $\mathcal{B}_\infty$, and the euclidean unit ball $\mathcal{B}_2$, and their best possible enclosure, with respect to Hausdorff distance, by an ellipsoid or a parallelotope respectively. In dimension d the best enclosures are an euclidean ball of radius $\sqrt{d}$ and a unit cube respectively. Their error in terms of Hausdorff distance will be of the order $\sqrt{d}$, as illustrated on Figure 3.7, although $\mathcal{B}_2$ touches $\mathcal{B}_\infty$ in $2d$ points, and $\mathcal{B}_\infty$ touches $\sqrt{d}\mathcal{B}_2$ in 2^d points.

This phenomenon is illustrated on Figure 3.7. A 100-dimensional unit Euclidean ball is tightly approximated by the unit cube in 200 directions

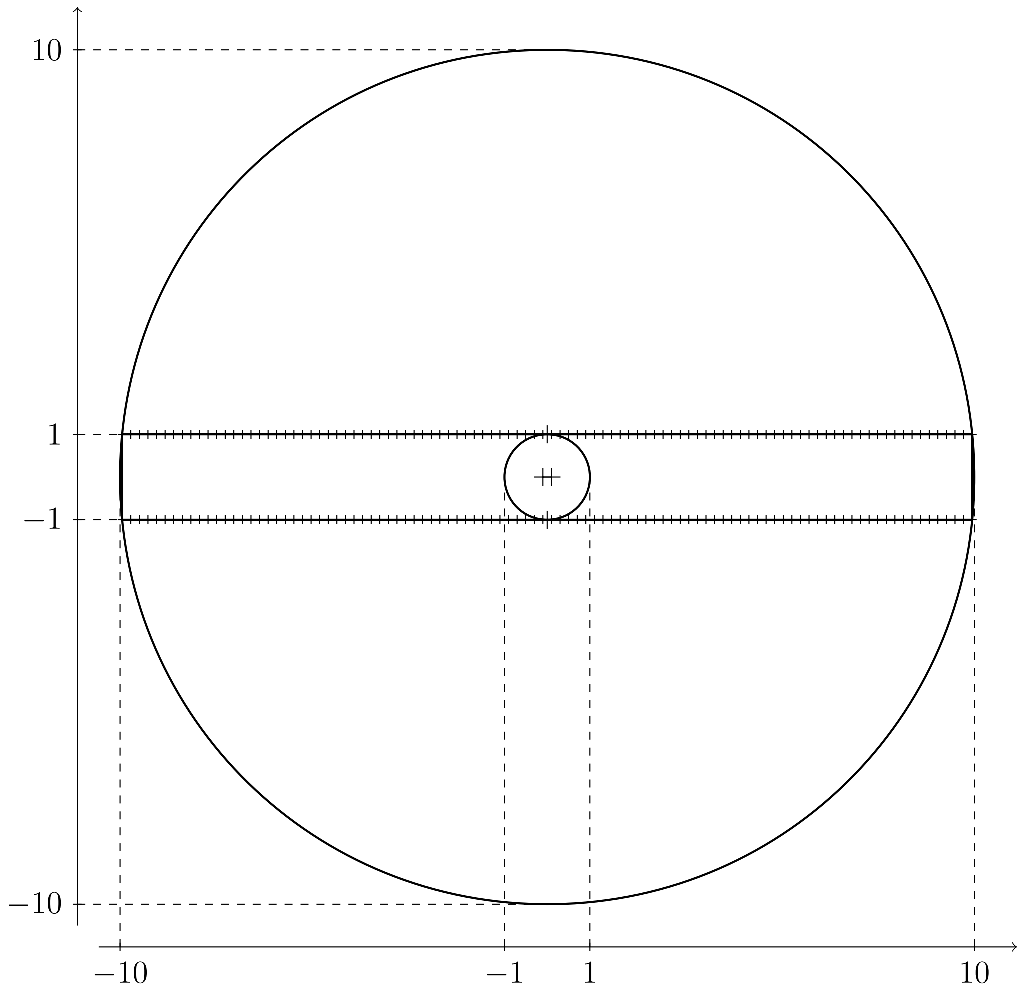


Figure 3.7: In dimension 100, projected on the plane generated by $(0, 1, \dots, 1)$ and $(1, 0, \dots, 0)$, the unit Euclidean ball is tightly approximated by the unit cube in 200 directions, leading to 200 intersection points between their respective boundaries. The two central crosses correspond to 99 intersection points each.
The unit cube is tightly approximated by a ball of radius 10 in 2^{100} directions leading to 2^{100} intersection points between their respective boundaries. Each cross is the projection of between 1 and $\binom{99}{49}$ intersection points.

Chapter 4

A New Scheme

Résumé : *Nous présentons dans ce chapitre la principale contribution de cette thèse, à la fois simple et efficace : un nouveau schéma algorithmique pour l'analyse d'atteignabilité des systèmes linéaires invariants. Nous décomposons la relation de récurence $\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$ en trois relations :*

$$\begin{array}{rclrcl} \mathcal{A}_0 & = & \Omega_0, & \mathcal{A}_{i+1} & = & \Phi\mathcal{A}_i, \\ \mathcal{V}_0 & = & \mathcal{V}, & \mathcal{V}_{i+1} & = & \Phi\mathcal{V}_i, \\ \mathcal{S}_0 & = & \{0\}, & \mathcal{S}_{i+1} & = & \mathcal{S}_i \oplus \mathcal{V}_i. \end{array}$$

Ω_i est alors égal à $\mathcal{A}_i \oplus S_i$. La séparation de la transformation linéaire et la somme de Minkowski nous permet de calculer efficacement la séquence des Ω_i en utilisant des zonotopes. Ces zonotopes ayant beaucoup de générateurs, nous développons également une version qui approxime les Ω_i sans souffrir de l'effet d'emballage.

The major contribution of this chapter is a new implementation scheme for the recurrence relation $\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$ which improves significantly (both theoretically and empirically) the computation of the reachable sets of linear time-invariant (LTI) systems with bounded inputs. A version of this algorithm based on *zonotopes* decisively outperforms related algorithms. In addition, algorithms for the computation of over- and under-approximations of the reachable sets are proposed. These algorithms are not subject to the *wrapping effect*. Most of the results in this chapter have been published in [GLGM06]

This chapter is organized as follows: in Section 4.1 we describe the simplest and most powerful contribution of this thesis, a new, more efficient algorithm, and discuss its implementations using zonotopes and support functions while in Section 4.2 we describe a general approximation scheme aiming to replace Algorithm 3.3 that gives good results even when the approximation function is the interval hull, then in Section 4.3 we improve the transformation of the original continuous-time system into an approximating discrete-time system. In the last section several implementations of both algorithms are tested on some examples.

Algorithm 4.1 Reachability of discrete linear time-invariant systems.

Input: The matrix Φ, the sets Ω_0 and $\mathcal{V}$, an integer N.
Output: The first N terms of the sequence defined in equation (3.4).

```
 1: $\mathcal{A}_0 \leftarrow \Omega_0$
 2: $\mathcal{V}_0 \leftarrow \mathcal{V}$
 3: $\mathcal{S}_0 \leftarrow \{0\}$
 4: for $i$ from 1 to $N-1$ do
 5:     $\mathcal{A}_i \leftarrow \Phi\mathcal{A}_{i-1}$                      ▷ $\mathcal{A}_i = \Phi^i\Omega_0$
 6:     $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \oplus \mathcal{V}_{i-1}$   ▷ $\mathcal{S}_i = \bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$
 7:     $\mathcal{V}_i \leftarrow \Phi\mathcal{V}_{i-1}$                      ▷ $\mathcal{V}_i = \Phi^i\mathcal{V}$
 8:     $\Omega_i \leftarrow \mathcal{A}_i \oplus \mathcal{S}_i$              ▷ $\Omega_i = \Phi^i\Omega_0 \oplus \bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$
 9: end for
10: return $\{\Omega_0, \dots, \Omega_{N-1}\}$
```

4.1 A New Scheme

We have a sequence of sets defined by equation (3.4):

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

Applying this recurrence relation leads to a time consuming algorithm. Applying an approximate version of this relation leads to an accumulation of errors hard to quantify because of the alternation between linear transformations and Minkowski sums.

Let us remark that the closed form of Ω_i is:

$$\Omega_i = \Phi^i\Omega_0 \oplus \bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$$

Then, let us define the auxiliary sequences of sets:

$$\begin{array}{rclrcl} \mathcal{A}_0 &=& \Omega_0, & \mathcal{A}_{i+1} &=& \Phi\mathcal{A}_i, \\ \mathcal{V}_0 &=& \mathcal{V}, & \mathcal{V}_{i+1} &=& \Phi\mathcal{V}_i, \\ \mathcal{S}_0 &=& \{0\}, & \mathcal{S}_{i+1} &=& \mathcal{S}_i \oplus \mathcal{V}_i. \end{array} \tag{4.1}$$

Equivalently, we have

$$\mathcal{A}_i = \Phi^i\Omega_0,\ \mathcal{V}_i = \Phi^i\mathcal{V} \text{ and } S_i = \bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$$

Therefore, $\Omega_i = \mathcal{A}_i \oplus S_i$ where $\mathcal{A}_i$ is the reachable set of the autonomous system from the set of initial states Ω_0, and $\mathcal{S}_i$ is the reachable set of the system with inputs from the initial set $\{0\}$. Note that the decomposition of the linear transformation and the Minkowski sum in the computation of $\mathcal{S}_{i+1}$ is possible only because the system is time-invariant. Algorithm 4.1 implements the reachable set computation based on the recurrence relations (4.1).

The main advantage of this algorithm is that the linear transformations are applied to sets whose representation size *does not increase*[1] at each iteration and this constitutes a significant improvement over existing algorithmic realizations of the recurrence relation (3.4).

Theorem 4.1. *The time complexity of Algorithm 4.1 is bounded by*

$$\mathcal{O}\left(N\mathfrak{L}(n_{in}) + N\mathfrak{S}(n_{out})\right)$$

where $\mathfrak{L}$ is the complexity of performing a linear transformation, $\mathfrak{S}$ is the complexity of performing a Minkowski sum, n_{in} bounds the representation size of Ω_0 and $\mathcal{V}$, and n_{out} bounds the representation size of Ω_{N-1}.

These parameters depend obviously on the class of sets chosen for the representation.

Due to the Minkowski sum, the size of the output may actually be very large. Hence, for an efficient implementation of Algorithm 4.1, the class of sets used for the representation of the reachable sets has to satisfy one of the following properties. Either the representation size of the Minkowski sum of two sets equals the representation size of the operands, or the computational complexity of the Minkowski sum is independent of the size of the operands.

General polytopes, for example, satisfy neither of these requirements. As far as we know, there is no reasonable representation satisfying the first property which is closed under Minkowski sum and linear transformations. The second property is satisfied by the class of zonotopes and support functions for which the complexity of Minkowski sum does not depend on the description complexity of the sets.

In the following section, the implementation of Algorithm 4.1 using zonotopes is discussed.

4.1.1 Implementation using Zonotopes

From Section 2.3.4, we know that the complexity of applying a linear transformation to a d-dimensional zonotope of order p is $\mathcal{O}\left(p\mathfrak{L}_d\right)$, where $\mathfrak{L}_d$ is the complexity of the multiplication of two $d \times d$ matrices. For Minkowski sum, the complexity is only $\mathcal{O}\left(d\right)$ because we only have to sum the centers of the two operands and concatenate their list of generators. We thus get the following corollary to Theorem 4.1:

Corollary 4.1. *The time complexity of Algorithm 4.1 using zonotopes to represent sets is*[2]*:*

$$\mathcal{O}\left(N(p+q)d^3\right)$$

[1]If we make the reasonable assumption that a linear transformation does not increase the representation size of a set. This is not verified for support functions, but then applying a linear transformation has constant cost $\mathcal{O}\left(1\right)$

[2]Note that, theoretically, the complexity can be further reduced down to $\mathcal{O}\left(N(p+q)d^{2.376}\right)$ by using a more sophisticate matrix multiplication algorithm [CW90].

where p and q are the order of the zonotopes Ω_0 and $\mathcal{V}$ respectively.

Moreover, since most of the generators of $\mathcal{S}_i$ are already stored in memory as the generators of $\mathcal{S}_{i-1}$, the additional cost, in term of memory, of Ω_i after the computation of $\Omega_1, \ldots, \Omega_{i-1}$ is only $\mathcal{O}\left((p+q)d^2\right)$, the space needed to store the generators of $\mathcal{A}_i$ and $\mathcal{V}_i$. Therefore, the space complexity of a zonotope implementation of Algorithm 4.1 is $\mathcal{O}\left(N(p+q)d^2\right)$.

We now have a very efficient exact algorithm for computing the N first terms of the sequence defined in Equation (3.4). Both its time and space complexity are linear in N. Unfortunately its output consists of zonotopes of order up to $p + Nq$. Because of the combinatorial nature of zonotopes, it seems that not much can be done with this output. Another limitation of this algorithm is that Ω_0 and $\mathcal{V}$ must be given as zonotopes.

In order to solve this last problem one can choose to extend this algorithm to sums of arbitrary sets. Remember that a zonotope is a Minkowski sum of a set of segments, each segment being represented by its generator. Then, instead of a list of vectors, $\mathcal{S}_i$ will be a list of sets.

But this does not solve the problem of the combinatorial nature of the Minkowski sum. Fortunately, some operations do not need to be performed in full dimension and one can reduce the dimension of the state space. This is possible because linear transformation, and projection in particular, is distributive with respect to Minkowski sum. As an example, if one want to plot the reachable set, considering that most displays are in $2D$, one can first project all the generators on the screen, whether they are segments or more complex sets, and then compute the sum in $2D$. More details will be given in Section 8.2.

4.1.2 Implementation using Support Functions

It is clear that, as for Algorithm 3.2, support functions allow an efficient implementation of Algorithm 4.1. The time complexity does not change, $\mathcal{O}(N)$, neither does the time complexity of the returned support functions, $\mathcal{O}\left(Nd^2 + N\mathfrak{C}^t_{\rho_\mathcal{V}} + \mathfrak{C}^t_{\rho_{\Omega_0}}\right)$.

The problem is that when analyzing the reachable set $\bigcup_{i=0}^{N-1} \Omega_i$, one will need to evaluate the support function of all the Ω_i, the complexity of this elementary task is $\mathcal{O}\left(N^2d^2 + N^2\mathfrak{C}^t_{\rho_\mathcal{V}} + N\mathfrak{C}^t_{\rho_{\Omega_0}}\right)$. In order to have an efficient implementation of this algorithm using support functions, one must, instead of a sequence of support functions, consider a single function that returns a sequence of values:

$$\ell \mapsto \left(\rho_{\Omega_0}(\ell), \ldots, \rho_{\Omega_{N-1}}(\ell)\right)$$

and use the redundancies of the system to get an efficient way of computing this function. That is what we will do in Chapter 5.

4.2 Approximation without Wrapping Effect

Algorithm 4.1 implemented with zonotopes makes it possible to compute *exactly* the sets defined by the recurrence relation (3.4):

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

The output of this algorithm consists in a sequence of high order zonotopes with many generators in common. Although we can use it for some operations as we will see in Chapter 8, it is often preferable to have an output that is easier to manipulate, even at the expense of approximations. Moreover the sequence is computed exactly only if Ω_0 and $\mathcal{V}$ can be expressed as zonotopes, which is not always the case. The extension to more general classes of sets evoked in Section 4.1.1 renders the manipulation of the output even more complicated.

That is why we investigate an approximation variant of Algorithm 4.1. Before going further, let us recall a few facts from Chapter 3.

There is no efficient implementation of Algorithm 3.2. This is due to this instruction: $\Omega_{i+1} \leftarrow \Phi\Omega_i \oplus \mathcal{V}$. The complexity of Ω_i grows because of the Minkowski sum, and $\Phi\Omega_i \oplus \mathcal{V}$ becomes harder and harder to compute. In order to deal with this problem, researchers in the field decided to replace this instruction by $\Omega_{i+1} \leftarrow \text{Approx}_{\Phi\cdot\oplus\cdot}(\Omega_i, \mathcal{V})$, which led to Algorithm 3.3. The challenge was then to find a *good* approximation function. But the error introduced by this sequence of approximations is hard to control or analyze, and no implementation provides a reasonable bound in terms of Hausdorff distance.

Here, Ω_i is computed from three sequences of sets. $\mathcal{A}_i$ and $\mathcal{V}_i$ are easy to compute, the only requirement is that they are in a class of sets closed by linear transformation. The difficulty comes from $\mathcal{S}_i$. Thus instead of computing it exactly, we will compute an approximation $\widetilde{\mathcal{S}}_i$ with the following recurrence relation:

$$\widetilde{\mathcal{S}}_0 = \{0\}, \text{ and } \widetilde{\mathcal{S}}_i = \text{Approx}_{\oplus}\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right)$$

We replace $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \oplus \mathcal{V}_{i-1}$ by $\widetilde{\mathcal{S}}_i \leftarrow \text{Approx}_{\oplus}\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right)$ in Algorithm 4.1 to get its approximation variant, Algorithm 4.2.

The challenge is now to find a good approximation function for Minkowski sum. This is much easier than finding a good approximation function for linear transformation and Minkowski sum $\Phi\Omega_i \oplus \mathcal{V}$. In fact, we can reuse all (over- or under-) approximation functions designed for Algorithm 3.3 with identity as a linear transformation. Indeed choosing

$$\text{Approx}_{\oplus}\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right) = \text{Approx}_{I\cdot\oplus\cdot}\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right)$$

is one possibility.

Moreover the error in terms of Hausdorff distance is much easier to analyze since:

$$d_H\left(\widetilde{\mathcal{S}}_i, \mathcal{S}_i\right) \leq d_H\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{S}_{i-1}\right) + d_H\left(\text{Approx}_{\oplus}\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right), \widetilde{\mathcal{S}}_{i-1} \oplus \mathcal{V}_{i-1}\right)$$

Algorithm 4.2 Approximate reachability of discrete linear time-invariant systems.

Input: The matrix Φ, the sets Ω_0 and $\mathcal{V}$, an integer N.
Output: Approximations of the first N terms of the sequence defined in equation (3.4).

```
 1: $\mathcal{A}_0 \leftarrow \Omega_0$
 2: $\mathcal{V}_0 \leftarrow \mathcal{V}$
 3: $\widetilde{\mathcal{S}}_0 \leftarrow \{0\}$
 4: for $i$ from 1 to $N-1$ do
 5:     $\mathcal{A}_i \leftarrow \Phi\mathcal{A}_{i-1}$
 6:     $\widetilde{\mathcal{S}}_i \leftarrow \text{Approx}_\oplus\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right)$
 7:     $\mathcal{V}_i \leftarrow \Phi\mathcal{V}_{i-1}$
 8:     $\widetilde{\Omega}_i \leftarrow \text{Approx}_\oplus\left(\widetilde{\mathcal{S}}_i, \mathcal{A}_i\right)$
 9: end for
10: return $\{\widetilde{\Omega}_0, \dots, \widetilde{\Omega}_{N-1}\}$
```

The error introduced by Algorithm 4.2 in terms of Hausdorff distance is smaller than the sum of the errors introduced at each step:

$$\begin{aligned} d_H\left(\widetilde{\Omega}_i, \Omega_i\right) \quad \leq \quad & d_H\left(\text{Approx}_\oplus\left(\widetilde{\mathcal{S}}_i, \mathcal{A}_i\right), \widetilde{\mathcal{S}}_i \oplus \mathcal{A}_i\right) \\ & + \sum_{j=0}^{i-1} d_H\left(\text{Approx}_\oplus\left(\widetilde{\mathcal{S}}_j, \mathcal{V}_j\right), \widetilde{\mathcal{S}}_j \oplus \mathcal{V}_j\right) \end{aligned}$$

which results in a limited wrapping effect.

Another possibility is to use two classes of sets closed under linear transformation for $\mathcal{A}_i$ and $\mathcal{V}_i$, and a class of sets closed under Minkowski sum with constant representation size for $\mathcal{S}_i$. We then only need an approximation function that transforms a set of any of the first two classes into a set of this last class. Then:

$$\text{Approx}_\oplus\left(\widetilde{\mathcal{S}}_{i-1}, \mathcal{V}_{i-1}\right) = \widetilde{\mathcal{S}}_{i-1} \oplus \text{Approx}\left(\mathcal{V}_{i-1}\right)$$

Let us remark that here, approximations occur only when $\text{Approx}(\cdot)$ is invoked. Note that $\text{Approx}(\cdot)$ is always applied to *exact* sets and that other operations are computed exactly. This is the main difference with Algorithm 3.3, the approximation $\widetilde{\Omega}_i$ of Ω_i is not used in the computation of $\widetilde{\Omega}_{i+1}$. Thus, approximation errors do not propagate further through the computations and Algorithm 4.2 does not suffer from the wrapping effect.

In this case the error becomes:

$$d_H\left(\widetilde{\Omega}_i, \Omega_i\right) \quad \leq \quad d_H\left(\text{Approx}\left(\mathcal{A}_i\right), \mathcal{A}_i\right) + \sum_{j=0}^{i-1} d_H\left(\text{Approx}\left(\mathcal{V}_j\right), \mathcal{V}_j\right)$$

And if $\text{Approx}(\cdot)$ is distributive with respect to Minkowski sum, $\widetilde{\Omega}_i$ is equal to $\text{Approx}(\Omega_i)$. As a consequence we have the following important proposition.

Proposition 4.1. *If the $\widetilde{\Omega}_i$ are computed using Algorithm 4.2 with an approximation function distributive with respect to Minkowski sum, then:*

$$\lim_{\delta\to 0}\bigcup_{i=0}^{\lfloor T/\delta\rfloor}\widetilde{\Omega}_i = \bigcup_{t\in[0;T]} Approx\left(\mathcal{R}_t(\mathcal{X}_0)\right)$$

This proposition guarantees that reducing the time step improves the approximation of the reachable set of the original continuous system, which is not the case for Algorithm 3.3.

In the following sections we will use some approximation functions such that $\mathrm{Approx}_\oplus\left(\widetilde{\mathcal{S}}_i,\mathcal{V}_i\right)\subseteq\widetilde{\mathcal{S}}_i\oplus\mathcal{V}_i$ or $\widetilde{\mathcal{S}}_i\oplus\mathcal{V}_i\subseteq\mathrm{Approx}_\oplus\left(\widetilde{\mathcal{S}}_i,\mathcal{V}_i\right)$ leading to over- and under-approximation algorithms.

4.2.1 Over-Approximations

The worst approximation function we have presented in Chapter 3 is the interval hull. But it behaves really well with Algorithm 4.2. If we define $\mathrm{Approx}_\oplus(\cdot,\cdot)$ as:

$$\mathrm{Approx}_\oplus\left(\overline{\mathcal{S}}_{i-1},\mathcal{V}_{i-1}\right)=\overline{\mathcal{S}}_{i-1}\oplus\Box\left(\mathcal{V}_{i-1}\right)$$

we will have $\overline{\Omega}_i=\Box\left(\Omega_i\right)$, since $\Box(\cdot)$ is distributive with respect to Minkowski sum.

Using interval hulls with Algorithm 3.3 led to an over-approximation that could grow exponentially with the number of steps, even for a contracting systems. Here, since $\overline{\Omega}_i=\Box\left(\Omega_i\right)$, we have:

$$d_H\left(\overline{\Omega}_i,\Omega_i\right)\le\frac{D_{\Omega_i}}{2}$$

where $D_{\Omega_i}=\sup_{x,y\in\Omega_i}\|x-y\|$. Moreover each face of $\overline{\Omega}_i$ has at least one common point with the set Ω_i, the over-approximations $\overline{\Omega}_0,\ldots,\overline{\Omega}_{N-1}$ are *tight*.

Moreover, the Minkowski sum of two interval hulls is very easy to compute, and they can be represented with only $2d$ values. Thus the time complexity of Algorithm 4.2 implemented with interval hulls is

$$\mathcal{O}\left(N(\mathfrak{L}_\mathcal{A}+\mathfrak{L}_\mathcal{V}+\mathfrak{C}_{\Box(\mathcal{A})}+\mathfrak{C}_{\Box(\mathcal{V})})\right)$$

and its space complexity is:

$$\mathcal{O}\left(\mathfrak{C}_\mathcal{A}+\mathfrak{C}_\mathcal{V}+Nd\right)$$

One way to compute the interval hull of a set is to evaluate its support function in the directions given by the canonical basis of $\mathbb{R}^d$, e_0, …, e_{d-1} and their opposites $-e_0$, …, $-e_{d-1}$. The time complexity of Algorithm 4.2 implemented with interval hulls is then:

$$\mathcal{O}\left(N(\mathfrak{L}_\mathcal{A}+\mathfrak{L}_\mathcal{V}+d(\mathfrak{C}^t_{\rho_\mathcal{A}}+\mathfrak{C}^t_{\rho_\mathcal{V}}))\right)$$

In fact we do not have to limit ourselves to directions e_0, …, e_{d-1}, $-e_0$, …, $-e_{d-1}$. Instead, we can chose any finite set of arbitrarily chosen directions, the resulting approximation is called a template polyhedron.

Definition 4.1 (Template polyhedron). *For any set $\mathcal{L} = \{\ell_0, \dots, \ell_{k-1}\}$ of vectors in $\mathbb{R}^d$, and any vector v in $(\mathbb{R} \cup \{\infty\})^k$, we define the polytope:*

$$[v]_{\mathcal{L}} = \{x : \forall i \in [0; k-1],\, x \cdot \ell_i \leq v_i\}$$

For any set $\mathcal{L}$, we define the class $\mathcal{P}_{\mathcal{L}}$ of $\mathcal{L}$-polytopes as the set of all $[v]_{\mathcal{L}}$.

Note that $[v]_{\mathcal{L}} = [u]_{\mathcal{L}}$ does not imply $v = u$.

Let us remark that interval products constitute the class of $\mathcal{L}$-polytopes where $\mathcal{L}$ is the set $\{e_0, \dots, e_{d-1}, -e_0, \dots, -e_{d-1}\}$, and that parallelepipeds are obtained when $\mathcal{L}$ is a set of d linearly independent vectors and their opposite. For a given set of vectors $\mathcal{L}$, the closure of the class of $\mathcal{L}$-polytopes under Minkowski sum is easy to show:

$$[v]_{\mathcal{L}} \oplus [u]_{\mathcal{L}} = [v + u]_{\mathcal{L}}.$$

In order to use $\mathcal{L}$-polytopes in Algorithm 4.2, we need an over-approximation function which maps a set to its smallest enclosing $\mathcal{L}$-polytope.

Proposition 4.2. *Let $\mathcal{A}$ be a subset of $\mathbb{R}^d$, then the $\mathcal{L}$-polytope $\square_{\mathcal{L}}(\mathcal{A}) = [v]_{\mathcal{L}}$ given by*[3]

$$v_i = \rho_{\mathcal{A}}(\ell_i)$$

is the smallest enclosing $\mathcal{L}$-polytope of $\mathcal{A}$. Moreover, each face of $\square_{\mathcal{L}}(\mathcal{A})$ has at least one common point with $\mathcal{A}$.

The distributivity of $\square_{\mathcal{L}}(\cdot)$ with respect to Minkowski sum is straightforward. Hence, it can be efficiently used as an approximation function for Algorithm 4.2. The time complexity of the algorithm becomes

$$\mathcal{O}\left(N(\mathfrak{L}_{\mathcal{A}} + \mathfrak{L}_{\mathcal{V}} + k(\mathfrak{C}^t_{\rho_{\mathcal{A}}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}}))\right)$$

and its space complexity is:

$$\mathcal{O}\left(\mathfrak{C}_{\mathcal{A}} + \mathfrak{C}_{\mathcal{V}} + Nk\right)$$

where k is the cardinality of $\mathcal{L}$.

Moreover, since $\overline{\Omega}_i = \square_{\mathcal{L}}(\Omega_i)$, we have:

Proposition 4.3. *The ratio between the Hausdorff distance between $\overline{\Omega}_i$ and Ω_i and the radius of Ω_i is bounded by a constant depending only on $\mathcal{L}$:*

$$\frac{d_H\left(\overline{\Omega}_i, \Omega_i\right)}{R_{\Omega_i}} \leq H_{\mathcal{L}} + 1$$

where $H_{\mathcal{L}}$ is the Hausdorff distance between the unit ball and its smallest enclosing by an $\mathcal{L}$-polytope, and R_{Ω_i} is the radius of Ω_i:

$$R_{\Omega_i} = \inf_{c \in \mathbb{R}^d} \sup_{x \in \Omega_i} \|x - c\|$$

[3]Note that we do not require here $\mathcal{A}$ to be represented by its support function, we just need to be able to compute it.

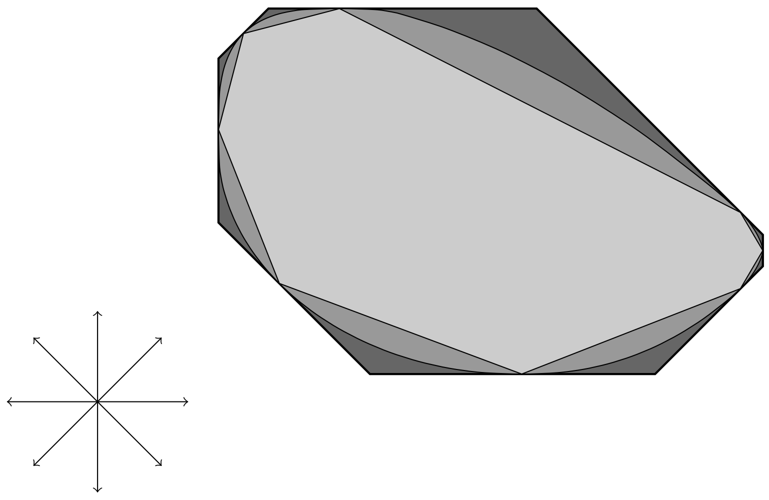


Figure 4.1: A set (grey) and its over- (dark grey) and under- (light grey) approximations in the directions given by $\mathcal{L}$.

Proof. $\overline{\Omega}_i$ is the smallest enclosing $\mathcal{L}$-polytope of Ω_i, and by definition, Ω_i is included in a ball of radius R_{Ω_i}, $\mathcal{B}(R_{\Omega_i})$, thus we have: $\Omega_i \subseteq \overline{\Omega}_i \subseteq \square_{\mathcal{L}}(\mathcal{B}(R_{\Omega_i}))$ and:

$$\begin{aligned} d_H\left(\overline{\Omega}_i, \Omega_i\right) &\leq d_H\left(\square_{\mathcal{L}}(\mathcal{B}(R_{\Omega_i})), \Omega_i\right) \\ &\leq d_H\left(\square_{\mathcal{L}}(\mathcal{B}(R_{\Omega_i})), \mathcal{B}(R_{\Omega_i})\right) + d_H\left(\mathcal{B}(R_{\Omega_i}), \Omega_i\right) \\ &\leq H_{\mathcal{L}} R_{\Omega_i} + R_{\Omega_i} \end{aligned}$$

which proves the proposition. □

To the best of the author's knowledge, no other method offers a constant bound on the ratio between the approximation error and the radius of Ω_i.

Remark. *Since we do not use $\overline{\Omega}_i$ to compute $\overline{\Omega}_{i+1}$ we lose the following property, common to Algorithms 3.2, 3.3, and 4.1:*

$$\Phi\overline{\Omega}_i \oplus \mathcal{V} \subseteq \overline{\Omega}_{i+1}$$

4.2.2 Under-Approximations

The only classes of sets with constant representation size closed under Minkowski sum we have seen so far are the classes of $\mathcal{L}$-polytopes. Since it might be hard to compute for an arbitrary set its largest enclosed $\mathcal{L}$-polytope, if it exists, we will not be able to decompose $\text{Approx}_{\oplus}(\underline{\mathcal{S}}_i, \mathcal{V}_i)$ into $\underline{\mathcal{S}}_i \oplus \text{Approx}(\mathcal{V}_i)$ when considering under-approximations.

The computation of an over-approximation of a set $\mathcal{A}$ by an $\mathcal{L}$-polytope involves the support function of $\mathcal{A}$. If $\mathcal{A}$ is closed and convex, we can use its support vectors in order to under-approximate it. Indeed for any set $\mathcal{L} = \{\ell_0, \ldots, \ell_{k-1}\}$ of vectors in $\mathbb{R}^d$, the set of support vectors of $\mathcal{A}$ in direction ℓ_i, $\nu_{\mathcal{A}}(\ell_i)$, is included in $\mathcal{A}$. Since $\mathcal{A}$ is convex, we have:

$$\mathrm{CH}\left(\bigcup_{\ell\in\mathcal{L}} \nu_{\mathcal{A}}(\ell)\right) \subseteq \mathcal{A}$$

Let us remark that linear transformation and Minkowski sum preserve convexity, thus if Ω_0 and $\mathcal{V}$ are convex, so are all the Ω_i.

Let $\mathcal{L}$ be a set of directions, instead of working with the sets $\nu_{\Omega_i}(\ell)$, we will compute one point $\nu^{\ell}_{\Omega_i}$ in each $\nu_{\Omega_i}(\ell)$ and under-approximate Ω_i by the convex hull of $\{\nu^{\ell}_{\Omega_i} : \ell \in \mathcal{L}\}$. In order to do so, we will represent $\underline{S}_i$ in Algorithm 4.2 by a set of vectors indexed by vectors in $\mathcal{L}$:

$$\underline{S}_i = \mathrm{CH}\left(\{g_\ell : \ell \in \mathcal{L}\}\right)$$

the computation of $\underline{S}_{i+1}$ can then be done by:

$$\mathrm{Approx}_{\oplus}\left(\underline{S}_i, \mathcal{V}_i\right) = \mathrm{CH}\left(\{g_\ell + \nu^{\ell}_{\mathcal{V}_i} : \ell \in \mathcal{L}\}\right) \tag{4.2}$$

The time complexity of Algorithm 4.2 using this approximation of the Minkowski sum is

$$\mathcal{O}\left(N(\mathfrak{L}_{\mathcal{A}} + \mathfrak{L}_{\mathcal{V}} + k(\mathfrak{C}^t_{\rho_{\mathcal{A}}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}}))\right)$$

and its space complexity is:

$$\mathcal{O}\left(\mathfrak{C}_{\mathcal{A}} + \mathfrak{C}_{\mathcal{V}} + Nkd\right)$$

where k is the cardinal of $\mathcal{L}$.

Moreover, since $\underline{\Omega}_i = \mathrm{CH}\left(\{\nu^{\ell}_{\Omega_i} : \ell \in \mathcal{L}\}\right)$, we have:

$$d_H\left(\underline{\Omega}_i, \Omega_i\right) \leq d_H\left(\underline{\Omega}_i, \square_{\mathcal{L}}\left(\underline{\Omega}_i\right)\right) \leq \left(H_{\mathcal{L}} + 1\right) R_{\Omega_i}$$

Remark. *It is also possible to work with a convex hull of a set of ellipsoids instead of a set of points by using an under-approximation for the Minkowski sum [KV06].*

4.3 Time Discretization

In Section 4.1 and 4.2 we showed how to compute, or approximate, the N first sets defined by the recurrence relation $\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$. This discrete sequence of sets is obtained from the time discretization of the following differential equation:

$$\dot{x}(t) = Ax(t) + u(t)$$

with $x(0) \in \mathcal{X}_0$ and $u(t) \in \mathcal{U}$. With a well chosen Ω_0 and $\mathcal{V}$, we can over-approximate the reachable set of the continuous system, $\mathcal{R}_{[0,T]}(\mathcal{X}_0)$, by $\bigcup_{i=0}^{N-1} \Omega_i$. Theorem 3.1 guarantees that the error introduced by this time discretization is proportional to the time step $\delta = T/N$.

In this section we improve the precision of the time discretization procedure, but we will first show that there is little hope to improve asymptotically the bounds on the error given by Theorem 3.1.

Proposition 4.4. *The Hausdorff distance between $\mathcal{R}_{[i\delta;(i+1)\delta]}$ and its convex hull can be proportional to δ.*

Proof. (see A.2.1 on page 126) □

This proposition tells us that using convex sets it is not possible to have a discretization scheme giving a better asymptotic bound on the error than $\mathcal{O}(\delta)$. Trying to improve the bound of Theorem 3.1 from this point of view is hopeless.

But from a practical point of view it can be improved in several ways. The most obvious one is to use the superposition principle and to decompose $\mathcal{X}_0$ and $\mathcal{U}$ into $\{x_0\} \oplus \mathcal{X}_0'$ and $\{b\} \oplus \mathcal{U}'$ respectively, where x_0 and b are chosen so that $\sup_{x\in\mathcal{X}_0'} \|x\|$ and $\sup_{u\in\mathcal{U}'} \|u\|$ are minimal.

The autonomous system $\dot{x}(t) = Ax(t) + b$ can then be discretized into

$$\omega_{k+1} = \Phi\omega_k \oplus \left\{ \int_0^\delta e^{(\delta-\tau)A} b d\tau \right\}$$

with ω_0 computed from x_0 with a guaranteed orbit algorithm and a smaller time step.

The non-autonomous system $\dot{x}(t) = Ax(t)+u(t)$, with $x(0) \in \mathcal{X}_0'$ and $u(t) \in \mathcal{U}'$, is discretized into:

$$\Omega_{i+1}' = \Phi\Omega_i' \oplus \mathcal{V}'$$

this discretization involves $R_{\mathcal{X}_0'}$ and $R_{\mathcal{U}'}$, the radii of $\mathcal{X}_0'$ and $\mathcal{U}'$ respectively, which are much smaller than $R_{\mathcal{X}_0}$ and R_U.

We can now superpose these two systems in order to get a recurrence relation allowing us to over-approximate the reachable set of the original system.

$$\Omega_{i+1} = \Phi\Omega_i \oplus \left\{ \int_0^\delta e^{(\delta-\tau)A} b d\tau \right\} \oplus \mathcal{V}' \text{ with } \Omega_0 = \omega_0 \oplus \Omega_0'$$

In the following, our approximations of the discretized flow and initial set involve $\boxdot(\mathcal{X}_0)$ and $\boxdot(\mathcal{U})$, it might thus be interesting to choose x_0 and b such that $\square(\mathcal{X}_0)$ and $\square(\mathcal{U})$ are already centrally symmetric. We drop the $'$ for readability.

For the next improvements we need the following operator: the extension of absolute value to vectors and matrices, component wise.

Definition 4.2. *Let $M = (m_{i,j})$ be a matrix, and $v = (v_i)$ a vector. We define $|M|$ and $|v|$ the absolute values of M and V:*

$$|M| = (|m_{i,j}|) \textit{ and } |v| = (|v_i|)$$

We can now use this operator to have a more precise approximation of the flow.

Given a subset $\Omega \subseteq \mathbb{R}^d$, the following lemma provides us with an over-approximation of $\mathcal{R}_\delta(\Omega)$:

Lemma 4.1. *Let $\Omega \subseteq \mathbb{R}^d$, let Ω' be the set defined by :*

$$\Omega' = e^{\delta A}\Omega \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}} \tag{4.3}$$

where $\mathcal{E}_{\mathcal{U}} = \square\left(|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) \boxdot (A\mathcal{U})\right)$

Then, $\mathcal{R}_\delta(\Omega) \subseteq \Omega'$ and, if $\|\cdot\|$ is the infinity norm:

$$d_H\left(\Omega', \mathcal{R}_\delta\left(\Omega\right)\right) \leq 2R_{\mathcal{E}_{\mathcal{U}}} \leq 2\left(e^{\delta\|A\|} - 1 - \delta\|A\|\right)\frac{R_{\mathcal{U}}}{\|A\|} \tag{4.4}$$

Proof. (see A.2.2 on page 127) □

We now have discretized the differential equation, but we still have to over-approximate the first element of the sequence $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$.

Most classes of sets impose strong restrictions on our ability to over-approximate $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$ efficiently. The most flexible class of set is the class of convex sets represented by their support function. The next chapter is devoted to this class, and Lemma 5.1 describes such an over-approximation.

We can deduce from this lemma an over-approximation in other classes of sets. If we denote:

$$\begin{aligned}\mathcal{E}_{\mathcal{X}_0} =& \square\left(|A|^{-1}\left(e^{\delta|A|} - I\right) \boxdot \left(A(I - e^{\delta A})\mathcal{X}_0\right)\right)\\ &\oplus \square\left(|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) \boxdot \left(A^2 e^{\delta A}\mathcal{X}_0\right)\right),\end{aligned}$$

since $\mathcal{E}_{\mathcal{X}_0}$ and $\mathcal{E}_{\mathcal{U}}$ are boxes, we can easily compute $\mathcal{E}_{\mathcal{X}_0} \ominus \mathcal{E}_{\mathcal{U}}$, the smallest set $\mathcal{E}$ such that $\mathcal{E}_{\mathcal{X}_0} \subseteq \mathcal{E}_{\mathcal{U}} \oplus \mathcal{E}$.

Then, Ω_0 can be over-approximated by:

$$\Omega_0 \subseteq \mathrm{CH}\left(\mathcal{X}_0, e^{\delta A}\mathcal{X}_0 \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}}\right) \oplus \frac{1}{4}\mathcal{E}$$

Or, if the convex hull is not representable or too hard to compute:

$$\Omega_0 \subseteq \overline{\mathrm{CH}\left(\mathcal{X}_0, e^{\delta A}\mathcal{X}_0\right)} \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}} \oplus \frac{1}{4}\mathcal{E}$$

Since δ is small, $\mathcal{X}_0$ and $e^{\delta A}\mathcal{X}_0$ are almost the same, and their convex hull can be approximated more easily.

Remark. *$|A|$ need not to be invertible. Indeed:*

$$|A|^{-1}\left(e^{\delta|A|} - I\right) = \sum_{i=0}^{\infty}\frac{\delta^{i+1}}{(i+1)!}|A|^i$$

$$|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) = \sum_{i=0}^{\infty}\frac{\delta^{i+2}}{(i+2)!}|A|^i$$

They can be computed by taking the matrix exponential of $3d \times 3d$ block matrix:

$$\exp\left(\delta\begin{pmatrix}|A| & I & 0\\ 0 & 0 & I\\ 0 & 0 & 0\end{pmatrix}\right) = \begin{pmatrix}e^{\delta|A|} & \sum_{i=0}^{\infty}\frac{\delta^{i+1}}{(i+1)!}|A|^i & \sum_{i=0}^{\infty}\frac{\delta^{i+2}}{(i+2)!}|A|^i\\ 0 & I & \delta I\\ 0 & 0 & I\end{pmatrix}$$

Now that we have an over-approximation of Ω_0 and an extensions of $\mathcal{R}_\delta$ we can over-approximate $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$ by the N first sets of the sequence :

$$\Omega_{i+1} = e^{\delta A}\Omega_i \oplus \delta\mathcal{U} \oplus \mathcal{E}_\mathcal{U} \tag{4.5}$$

Remark. *If instead we want to under-approximate $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$ we first have to take $\Omega_0 \subseteq \mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$. One obvious way to get this is to take $\Omega_0 = \mathcal{X}_0$. We can then consider that the input u is constant on all the intervals $[i\delta;(i+1)\delta]$. If we define the sequence Ω_i as:*

$$\Omega_{i+1} = e^{\delta A}\Omega_i \oplus A^{-1}\left(e^{\delta A} - I\right)\mathcal{U}$$

then for all i, $\Omega_i \subseteq \mathcal{R}_{i\delta}(\mathcal{X}_0) \subseteq \mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{X}_0)$ and:

$$\bigcup_{i\in[0;N]} \Omega_i \subseteq \mathcal{R}_{[0;T]}(\mathcal{X}_0)$$

4.4 Experimental Results

Most of the algorithms in this chapter have been implemented into a prototype tool in OCaml. We use the GNU Scientific Library for linear algebra and the GNU Linear Programming Kit as an LP solver. For the sake of comparison, we have also implemented the zonotope-based reachability algorithm presented in [Gir05]. At each step, in order to avoid computational explosion, it reduces the complexity of the reachable set by over-approximating it by a zonotope of fixed order p as explained in Section 3.4.2.4. In the following, we refer to this algorithm by Zono-p. Set representations and linear algebra operations were implemented in separate modules so that all algorithms use the same subroutines. All computations were performed on a Pentium IV 3.2GHz with 1GB RAM.

4.4.1 A Five-Dimensional Linear System

As a first benchmark consider the following five-dimensional example borrowed from [Gir05].

Example 4.1. *Compute the set of points reachable for time t in $[0;5]$ by the following system:*

$$\dot{x} = Ax(t) + u(t)$$

where $A = PDP^{-1}$ is the image of the block diagonal matrix D after a change of variables P.

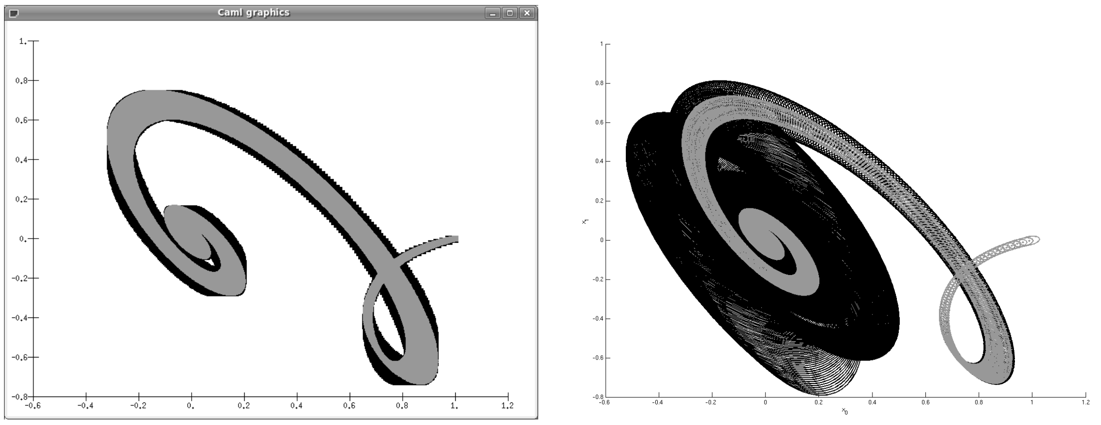
$$D = \begin{pmatrix} -1 & -4 & 0 & 0 & 0 \\ 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & -3 & 1 & 0 \\ 0 & 0 & -1 & -3 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \qquad P = \begin{pmatrix} 0.6 & -0.1 & 0.1 & 0.7 & -0.2 \\ -0.5 & 0.7 & -0.1 & -0.8 & 0 \\ 0.9 & -0.5 & 0.3 & -0.6 & 0.1 \\ 0.5 & -0.7 & 0.5 & 0.6 & 0.3 \\ 0.8 & 0.7 & 0.6 & -0.3 & 0.2 \end{pmatrix}$$

For all time t the input u must belong to a set $\mathcal{U}$ equal to the Euclidean ball of radius 0.01 *centered at the origin; the initial set $\mathcal{X}_0$ is defined as the cube of side* 0.02 *centered at* $(1, 0, 0, 0, 0)$*:*

$$\mathcal{X}_0 = \left\{(1, 0, 0, 0, 0)^\top\right\} \oplus \mathcal{B}_\infty(0.01) \qquad\qquad \mathcal{U} = \mathcal{B}_2(0.01)$$

4.4.1.1 Comparison with the Ellipsoidal Toolbox

A first difficulty of this example is that the initial set $\mathcal{X}_0$ and the set of inputs $\mathcal{U}$ do not belong to the same class of sets, the first one is a box, and thus can also be expressed as a polytope or zonotope, and the second one is an ellipsoid, the only class of sets that can represent them both is the class of convex sets represented by their support function, this will be the subject of Chapter 5.



(a) Using Algorithm 4.1 (grey) and Algorithm 4.2 with boxes (black).

(b) Using the Ellipsoidal Toolbox (black) and a modified version (grey).

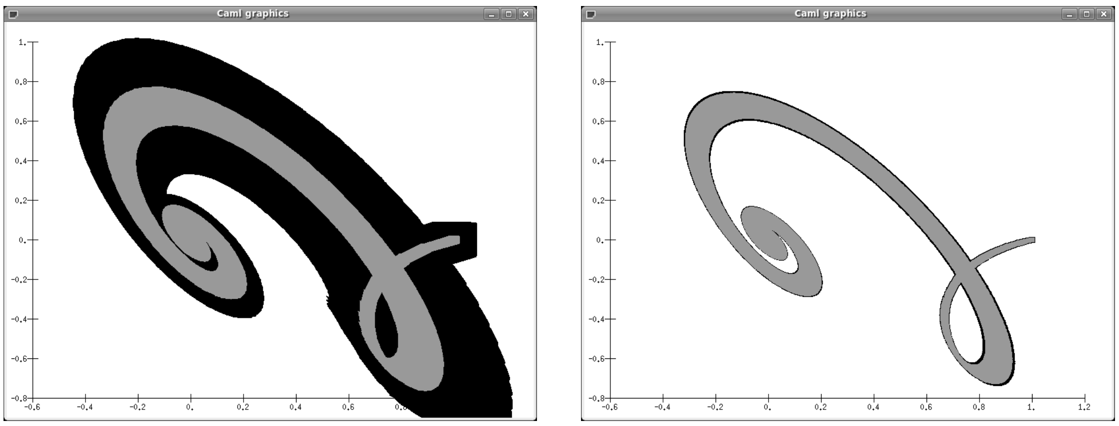
Figure 4.2: Projection on the first two variables of the reachable set of Example 4.1 computed using the discretization procedure of Section 3.2 with $N = 1000$.

On Figure 3.2 (reproduced on Figure 4.2b) we used the Ellipsoidal Toolbox, and a modified version of it, to compute the reachable set of Example 4.1. Since this tool deals with ellipsoids, we defined $\mathcal{X}_0$ to be the euclidean ball of radius 0.01 centered at $(1, 0, 0, 0, 0)$ in order not to advantage our algorithms. The resulting reachable set is smaller; remember that because of the curse of dimensionality, the ration between the radii of the unit ball and the smallest ball enclosing the unit cube is $\sqrt{5} > 2.236$. Our implementations of Algorithms 4.1 and 4.2 deal with zonotopes. In order to be conservative we have to convert ellipsoids to their interval hull[4]. Converting $\mathcal{U}$ to $\square(\mathcal{U})$ directly is not a good idea. Instead we perform the time discretization with $\mathcal{U}$ as an ellipsoid and then we convert $\mathcal{V}$ to a box.

[4]Note that an implementation of Algorithms 4.2 mixing zonotopes and ellipsoids is possible: we either convert the $\mathcal{A}_i$ to ellipsoids or the $\mathcal{V}_i$ to $\mathcal{L}$-polytopes whether we want Ω_i to be expressed as an ellipsoid or an $\mathcal{L}$-polytope respectively.

For a fair comparison between both techniques we use the same discretization procedure, described in Section 3.2, for the Ellipsoidal Toolbox and our algorithms. The reachable sets on Figure 4.2 are not surprising. We already know that Algorithm 4.1 computes the exact reachable set up to initial discretization errors, and that Algorithm 4.2 implemented with boxes returns the interval hulls of the exact Ω_i.

4.4.1.2 Improved Time Discretization



(a) $N = 200$, $\delta = T/N = 0.025$. (b) $N = 1000$, $\delta = T/N = 0.005$.

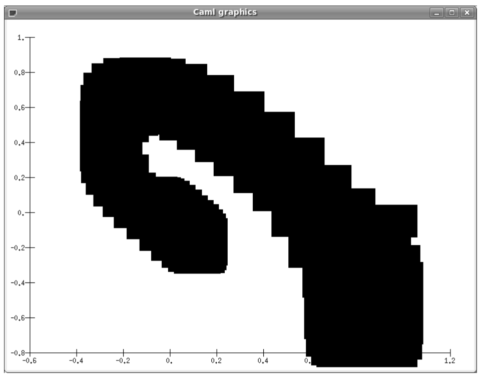
Figure 4.3: Projection on the first two variables of the reachable set of Example 4.1 computed with Algorithm 4.1 using the discretization procedure from Section 3.2 (black) and 4.3 (grey).

On Figure 4.3 we compare the error introduced by the discretization procedure from Section 3.2 with the error introduced by ours, described in Section 4.3. In order to do so, we use Algorithm 4.1. This algorithm computes the exact sequence of Ω_i. Thus the differences between both reachable sets is only due to the time discretization procedure used. With a small time step the difference is barely visible since both methods converge towards the exact reachable set. But our method makes it possible to take a bigger time step with a reasonable error. There are two main advantages, both coming from the fact that N is smaller:

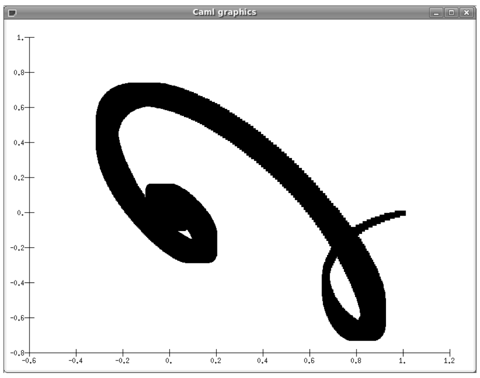
- Computing the N first sets of the sequence of Ω_i is faster if N is smaller.
- Manipulating the reachable set is much easier, it is composed of fewer sets, and, if computed exactly using Algorithm 4.1, they are zonotopes of lower order.

4.4.1.3 Varying the Time Step

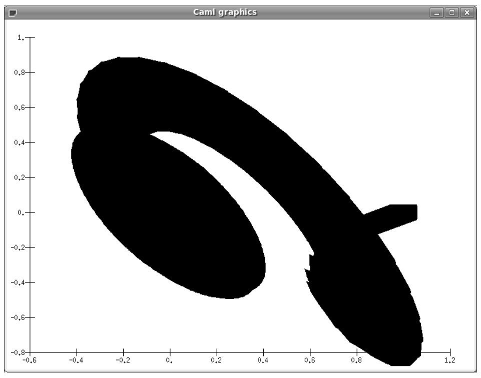
Our improved time discretization procedure allows to converge faster toward the exact reachable set, and reducing the time step improves the quality of the approximation as illustrated on Figures 4.4a and b. But this is not true for all algorithms. Indeed, for at least some of the algorithms that are subject to the wrapping effect, which is not the case of Algorithms 4.1 and 4.2, reducing the time step increase the wrapping effect because it increases the number of iterations required to cover the same time interval. As we can see on Figures 4.4c and d, reducing the time step actually decreases the quality of the over-approximations obtained by Algorithm Zono-20, which breaks the convergence of the overall analysis.



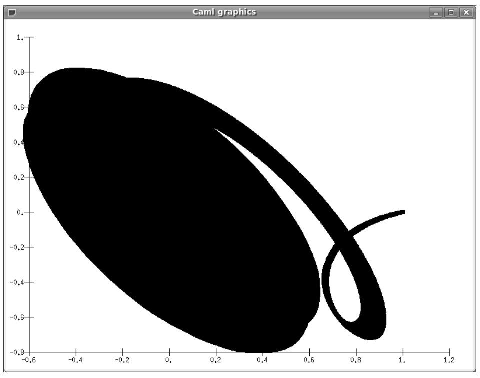
(a) $N = 100$, $\delta = T/N = 0.05$.



(b) $N = 1000$, $\delta = T/N = 0.005$.



(c) $N = 100$, $\delta = T/N = 0.05$.



(d) $N = 1000$, $\delta = T/N = 0.005$.

Figure 4.4: Projection on the first two variables of the reachable set of Example 4.1 computed with Algorithm 4.2 using interval hulls (top) or Algorithm Zono-20 (bottom).

4.4.1.4 Time and Memory Consumption

The efficiency of our algorithms in term of time and memory consumption is confirmed by Tables 4.1 and 4.2. The shading denotes a qualitative interpretation of the values: lighter is better. We can see that Algorithms 4.1 and 4.2 are fast and require much less memory. Algorithm 4.2, which computes interval-hull approximations, is at least 30 times faster and needs more than 35 times less memory than Algorithm Zono-20, while producing approximations of higher quality.

$N =$	100	200	400	600	800	1000
Algorithm 4.1	< 0.01	< 0.01	< 0.01	0.01	0.01	0.02
Algorithm 4.2	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.01
Zono-20	0.02	0.04	0.11	0.16	0.22	0.29
Zono-40	0.05	0.12	0.24	0.35	0.47	0.59

Table 4.1: Execution time (in seconds) of the reachability analysis of Example 4.1 with different values of N.

$N =$	100	200	400	600	800	1000
Algorithm 4.1	0.234	0.468	0.703	1.171	1.640	1.875
Algorithm 4.2	0.234	0.234	0.234	0.234	0.234	0.234
Zono-20	0.937	1.640	3.281	4.921	6.562	8.203
Zono-40	1.640	3.046	6.328	9.609	13.125	15.937

Table 4.2: Memory consumption (in MB) of the reachability analysis of Example 4.1 with different values of N.

$N =$	100	200	400	600	800	1000
Algorithm 4.1	0.09	0.40	1.63	3.74	6.94	11.03
Algorithm 4.2	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
Zono-20	0.04	0.10	0.20	0.31	0.41	0.53
Zono-40	0.09	0.20	0.44	0.65	0.88	1.13

Table 4.3: Time (in seconds) required for plotting the computed sets.

We also report on Table 4.3 the time required for plotting the output of each algorithm. Not surprisingly, interval hulls are very easy to manipulate, which confirms the usefulness of Algorithm 4.2 implemented with interval hulls.

4.4.2 High-Dimensional Linear Systems

Computation times and memory consumptions for linear systems of several dimensions d are reported in Table 4.5 and 4.6.

These systems are randomly generated according to the following procedure: the matrix A is chosen at random and then normalized for the infinity norm and the inputs are cubes of side 0.02 with a random center. We compute the set of points reachable during $[0; 1]$. The time needed for the discretization of the system

$d =$	10	20	50	100	200	500
Section 3.2 (s)	< 0.01	< 0.01	< 0.01	< 0.01	0.12	2.45
Section 3.2 (MB)	0.234	0.234	0.234	0.703	2.097	13.23
Section 4.3 (s)	< 0.01	< 0.01	< 0.01	< 0.01	1.54	30.84
Section 4.3 (MB)	0.234	0.234	0.234	0.937	3.406	20.203

Table 4.4: Time and memory consumption of the discretization procedures described in Sections 3.2 and 4.3.

$d =$	10	20	50	100	200	500
Algorithm 4.1	< 0.01	0.02	0.24	1.18	11.6	132.
Algorithm 4.2	< 0.01	0.02	0.19	1.20	11.9	133.
Zono-20	0.07	0.26	1.60	8.10	65.1	> 300
Zono-40	0.12	0.46	2.77	14.0	117.	> 300

Table 4.5: Execution time (in seconds) of the reachability analysis of randomly generated systems of different dimensions.

is reported on Table 4.4. We use a time step of 0.01 and thus need to compute the 100 first Ω_i.

Algorithms 4.1 and 4.2 appear to be scalable in terms of both time and space, which confirms the theoretical complexity estimations. Let us remark that using Algorithm 4.2, we can compute a tight over-approximation of the reachable set of a 100-dimensional system after 100 time steps using less than 1MB memory in a little more than 1s.

$d =$	10	20	50	100	200	500
Algorithm 4.1	0.703	1.640	8.671	33.04	128.4	794.1
Algorithm 4.2	0.234	0.234	0.468	0.937	3.269	18.38
Zono-20	2.578	7.968	43.35	166.1	652.2	
Zono-40	4.218	13.82	75.23	286.6	1119.	

Table 4.6: Memory consumption (in MB) of the reachability analysis of randomly generated systems of different dimensions.

Chapter 5

Going Further with Support Functions

Résumé : *Nous avons présenté jusqu'alors des algorithmes indépendants de la représentation choisie. Nous avons ensuite montré comment instancier ces algorithmes. Ici nous nous intéressons exclusivement à la représentation des ensembles convexes par leur fonction support. L'algorithme que nous proposons prend en entrée Φ, $\rho_{\mathcal{V}}$, ρ_{Ω_0}, et N et renvoie une fonction f :*

$$f : \ell \mapsto (\rho_{\Omega_0}(\ell), \ldots, \rho_{\Omega_{N-1}}(\ell))$$

Cette algorithme accélère significativement l'analyse quand on ne s'intéresse qu'à un sous-espace de l'espace d'états. Par ailleurs, la polyvalence des fonctions support permet d'améliorer considérablement le calcul de Ω_0 et $\mathcal{V}$ lors de la discrétisation du système original.

In this chapter we focus exclusively on the class of convex sets represented by their support functions. As stated in Sections 3.3 and 4.1.2 this representation demands an adaptation of the regular computational scheme. Moreover, its expressiveness allows for a better approximation of $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, the initial set for the discretized system. Most of the results in this chapter have been published in [GLG08a, LGG09b].

5.1 Discrete Reachability Algorithm

We have a sequence of sets defined by equation (3.4):

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

We want to compute a representation for the N first sets of this sequence.

As stated before, returning a list of support functions is not satisfying, because we want to exploit the redundancies of the system. Instead we return a function,

f, taking as input a vector ℓ, and returning the values of each support function in this direction:

$$f : \ell \mapsto (\rho_{\Omega_0}(\ell), \dots, \rho_{\Omega_{N-1}}(\ell)) \tag{5.1}$$

Thus our reachability algorithm takes as input Φ, $\rho_{\mathcal{V}}$, ρ_{Ω_0}, and N and returns this function. This is, in fact, the partial evaluation of Algorithm 5.1.

Algorithm 5.1 Evaluation of $\rho_{\Omega_0}(\ell), \dots, \rho_{\Omega_{N-1}}(\ell)$.

```
Input: The matrix $\Phi$, the support functions of the sets $\mathcal{V}$ and $\Omega_0$, an integer $N$,
    and a vector $\ell$.
Output: $\rho_i = \rho_{\Omega_i}(\ell)$ for $i$ in $\{0, \dots, N-1\}$
1: $r_0 \leftarrow \ell$
2: $s_0 \leftarrow 0$
3: $\rho_0 \leftarrow \rho_{\Omega_0}(r_0)$
4: for $i$ from 1 to $N-1$ do
5:     $r_i \leftarrow \Phi^\top r_{i-1}$                      $\triangleright\ r_i = \left(\Phi^\top\right)^i \ell$
6:     $s_i \leftarrow s_{i-1} + \rho_{\mathcal{V}}(r_{i-1})$     $\triangleright\ s_i = \sum_{j=0}^{i-1} \rho_{\mathcal{V}}\left(\left(\Phi^\top\right)^j \ell\right)$
7:     $\rho_i \leftarrow \rho_{\Omega_0}(r_i) + s_i$     $\triangleright\ \rho_i = \rho_{\Omega_0}\left(\left(\Phi^\top\right)^i \ell\right) + \sum_{j=0}^{i-1} \rho_{\mathcal{V}}\left(\left(\Phi^\top\right)^j \ell\right)$
8: end for
9: return $\{\rho_0, \dots, \rho_{N-1}\}$
```

Proposition 5.1. *Algorithm 5.1 returns $\rho_{\Omega_0}(\ell), \dots, \rho_{\Omega_{N-1}}(\ell)$, where the Ω_i are defined by equation* (3.4).

Its time complexity is:

$$\mathcal{O}\left(N\left(d^2 + \mathfrak{C}^t_{\rho_{\Omega_0}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}}\right)\right)$$

Its space complexity is:

$$\mathcal{O}\left(Nd + d^2 + \mathfrak{C}^s_{\rho_{\Omega_0}} + \mathfrak{C}^s_{\rho_{\mathcal{V}}}\right)$$

where $\mathfrak{C}^t_\rho$ and $\mathfrak{C}^s_\rho$ are, respectively, the time and space complexity of ρ.

Proof. A proof can be easily deduce from Proposition 2.3. □

5.1.1 Comparison with Related Approaches

As already stated in Section 3.4.2.5, reachability analysis of linear systems based on the use of support functions has already been proposed in [Var98]. We would like to discuss here the differences between the two approaches. In [Var98], the support functions of the reachable sets are computed recursively using the relation:

$$\rho_{\Omega_{i+1}}(\ell) = \rho_{\Omega_i}(\Phi^\top \ell) + \rho_{\mathcal{V}}(\ell).$$

Given a vector ℓ, an algorithm based on this relation returns:

$$\left(\rho_{\Omega_0}\left(\left(\Phi^\top\right)^{N-1}\ell\right), \rho_{\Omega_1}\left(\left(\Phi^\top\right)^{N-2}\ell\right), \dots, \rho_{\Omega_{N-2}}\left(\Phi^\top\ell\right), \rho_{\Omega_{N-1}}(\ell)\right)$$

Thus, using the same idea of partial evaluation, we can deduce an algorithm, with the same complexity as Algorithm 5.1, taking as input Φ, $\rho_{\mathcal{V}}$, ρ_{Ω_0}, and N and returning the function:

$$g : \ell \mapsto \left(\rho_{\Omega_0}\left(\left(\Phi^\top\right)^{N-1}\ell\right), \dots, \rho_{\Omega_{N-1}}(\ell)\right) \tag{5.2}$$

When Φ is invertible (and it is if it comes from the discretization of a continuous system) all the Ω_i can be deduced from g.

f and g, as defined by equations (5.1) and (5.2) respectively, are thus two representations of the sequence of sets (Ω_i), and they have the same time complexity. But g does not return the values of all the ρ_{Ω_i} in the same direction, and this has some serious consequences on its usability as a representation.

First, computing the support function of the reachable set $\rho_{\cup\Omega_i}$ can be done with only one call to f and taking the maximum amongst N values, but requires N calls to g and N linear transformations.

Second, if one wants to plot the reachable set on a $2d$ screen, or even in $3d$, one must first approximate the projection of every Ω_i in the considered low dimensional subspace by a polytope, with a precision depending on the resolution of the screen. Sampling f in directions that are in the considered subspace is enough, but, because of Φ, calling g might give a constraints in this subspace only for Ω_{N-1}. More generally, this problem occurs whenever one is interested in the projection of the reachable sets on an output subspace. Let us consider, for instance the single output system:

$$\begin{cases} \dot{x}(t) &=& Ax(t) + Bu(t),\ u(t) \in \mathcal{U},\ x(0) \in \mathcal{X}_0 \\ y(t) &=& cx(t) \end{cases}$$

where $c^\top \in \mathbb{R}^d$. Then, in order to compute an over-approximation of the interval reachable by $y(t)$ it is sufficient to run Algorithm 5.1 with $\ell = c^\top$.

Lastly, computing polyhedral over-approximations of the Ω_i by sampling f and h leads to very different results. Sampling f with vectors in a set $\mathcal{L}$ produces $\mathcal{L}$-polyhedral approximations. From Proposition 4.3, we know that the ratio between the approximation error and the radius of Ω_i depends only on $\mathcal{L}$. On the contrary, sampling g with vectors in a set $\mathcal{L}$ produces over-approximations that can be ill-conditioned because all vectors in $\left(\Phi^\top\right)^{N-1}\mathcal{L}$ might tend towards the same vector as already explained in section3.4.2.5.

5.1.2 Improvements of the Algorithm

Algorithm 5.1 allows a functional representation of the sequence of sets (Ω_i). The directions in which f, as defined in equation (5.1), will be called are guided by the transformation we want to apply, or the property we want to check, on the reachable set. A side effect of these calls is the creation of a polyhedral approximation of each Ω_i.

Sometimes, at each step of the analysis of the reachable set, there is only one direction ℓ of interest, as we will see in Section 8.3, after the evaluation of $f(\ell)$ we either stop the analysis or deduce a new direction of interest.

More often, there are several directions of interest, in particular if there are several properties to check, as we will see in Section 8.1. In this case, one can use the fact that f can be computed independantly for each direction. Then, with α processors, the support function of every Ω_i can be computed in r directions with complexity:

$$\mathcal{O}\left(\left\lceil \frac{r}{\alpha} \right\rceil N(d^2 + \mathfrak{C}^t_{\rho_{\Omega_0}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}})\right)$$

This is of particular interest if one wants to compute a polyhedral approximation of the Ω_i, leading to an efficient parallel variant of the $\mathcal{L}$-polytopes implementation of Algorithm 4.2.

When computing such approximations, the choice of some directions is guided by the system, and others are chosen we a relative freedom. In this case, we can do much better than just sampling f.

Let us assume that the different directions of approximation $\ell_1, \dots, \ell_r$ have been chosen such that:

$$\ell_k = (\Phi^\top)^{p_k}\ell,\ k = 1, \dots, r \text{ where } p_1 < \dots < p_r.$$

One could argue that such a choice of directions suffers from the problem already evoked in Section 5.1.1. Indeed, given a direction ℓ, the vectors $({\Phi_\delta}^\top)^p\ell$ will point towards the same direction as p grows. For that reason the indices $p_1, \dots, p_r$ must be chosen carefully.

As an example, one can choose the indices p_k iteratively, taking p_{k+1} such that the angle between $(\Phi^\top)^{p_{k+1}}\ell$ and the vectors $(\Phi^\top)^{p_i}\ell$ is bigger than some value, or:

$$\forall i \leq k,\ \left| \frac{(\Phi^\top)^{p_{k+1}}\ell \cdot (\Phi^\top)^{p_i}\ell}{\|(\Phi^\top)^{p_{k+1}}\ell\|\|(\Phi^\top)^{p_i}\ell\|} \right| \leq \varepsilon.$$

Then, it follows that for all $i = 0, \dots, N-1$, $k = 1, \dots, r$:

$$\begin{aligned}
\rho_{\Omega_i}(\ell_k) &= \rho_{\Omega_0}\left(({\Phi_\delta}^\top)^i({\Phi_\delta}^\top)^{p_k}\ell\right) + \sum_{j=0}^{i-1} \rho_{W_\delta}\left(({\Phi_\delta}^\top)^j({\Phi_\delta}^\top)^{p_k}\ell\right) \\
&= \rho_{\Omega_0}\left(({\Phi_\delta}^\top)^{i+p_k}\ell\right) + \sum_{j=0}^{i-1} \rho_{W_\delta}\left(({\Phi_\delta}^\top)^{j+p_k}\ell\right) \\
&= \rho_{\Omega_0}\left(({\Phi_\delta}^\top)^{i+p_k}\ell\right) + \sum_{j=p_k}^{i+p_k-1} \rho_{W_\delta}\left(({\Phi_\delta}^\top)^{j}\ell\right) \\
&= \rho_{\Omega_0}(r_{i+p_k}) + s_{i+p_k} - s_{p_k}.
\end{aligned}$$

Thus, it is sufficient to compute the sequences $r_0, \dots, r_{N+p_r-1}$ and $s_0, \dots, s_{N+p_r-1}$. Then, the complexity of the reachability analysis drops to

$$\mathcal{O}\left((N+p_r)(d^2 + \mathfrak{C}^t_{\rho_{\Omega_0}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}} + r)\right)$$

If ℓ is an eigenvector of $\Phi^\top$ associated to a real eigenvalue, it is clear that the vectors $\ell_k = (\Phi^\top)^{p_k}\ell$ will be colinear. In this case, the previous improvement cannot be used. However, evaluating the support function in the direction of an eigenvector ℓ can be interesting as it can be done very efficiently. If $\Phi^\top\ell = \lambda\ell$, with $\lambda \geq 0$ then:

$$\begin{aligned} \rho_{\Omega_i}(\ell) &= \rho_{\Omega_0}(\lambda^i\ell) + \sum_{j=0}^{i-1} \rho_{W_\delta}(\lambda^j\ell) \\ &= \lambda^i\rho_{\Omega_0}(\ell) + \rho_{W_\delta}(\ell)\sum_{j=0}^{i-1}\lambda^j. \end{aligned}$$

Then, only the evaluation of $\rho_{\Omega_0}(\ell)$ and $\rho_{\mathcal{V}}(\ell)$ are needed. Hyperplanes bounding the reachable sets, in the direction given by an eigenvector are computed in only $\mathcal{O}\left(N + \mathfrak{C}^t_{\rho_{\Omega_0}} + \mathfrak{C}^t_{\rho_{\mathcal{V}}}\right)$. Similarly, if $\lambda < 0$, $\rho_{\Omega_i}(\ell)$ can be computed from $\rho_{\Omega_0}(\ell)$, $\rho_{\Omega_0}(-\ell)$, $\rho_{\mathcal{V}}(\ell)$ and $\rho_{\mathcal{V}}(-\ell)$.

5.1.3 Computing Support Vectors

Now that we know how to evaluate the support function of the Ω_i in one direction, it would be also interesting to have an associated support vector in that direction. Interesting applications include the production of a polyhedral under-approximation of Ω_i expressed as the convex hull of a set of points.

The closed form of Ω_i is

$$\Phi^i\Omega_0 \oplus \bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$$

In order to evaluate its support function in direction ℓ, we use a sequence $r_j = \left(\Phi^\top\right)^j \ell$, and evaluate the support functions of Ω_0 and $\mathcal{V}$ in the directions r_j.

$$\rho_{\Omega_i} = \rho_{\Omega_0}\left(r_i\right) + \sum_{j=0}^{i-1} \rho_{\mathcal{V}}\left(r_j\right)$$

Together with the $\rho_{\Omega_0}(r_j)$ and $\rho_{\mathcal{V}}(r_j)$, it is possible to compute some support vectors, denoted as $\nu^{r_j}_{\Omega_0}$ and $\nu^{r_j}_{\mathcal{V}}$. Now we can use the following proposition to find a support vector for Ω_i.

Proposition 5.2. *For any sets $\mathcal{X}$ and $\mathcal{Y}$, any linear transformation A, and any vector ℓ, we have:*

$$\nu_{A\mathcal{X}}(\ell) = A\nu_{\mathcal{X}}\left(A^\top \ell\right)$$
$$\nu_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \nu_{\mathcal{X}}(\ell) \oplus \nu_{\mathcal{Y}}(\ell)$$

Proof. A proof can be easily derived from the properties of the dot product. □

Then the set of support vectors for Ω_i in direction ℓ is:

$$\nu_{\Omega_i}(\ell) = \Phi^i \nu_{\Omega_0}(r_i) \oplus \bigoplus_{j=0}^{i-1} \Phi^j \nu_{\mathcal{V}}(r_j)$$

and since $\nu_{\Omega_0}^{r_j}$ and $\nu_{\mathcal{V}}^{r_j}$ belongs to $\nu_{\Omega_0}(r_j)$ and $\nu_{\mathcal{V}}(r_j)$ respectively, we can compute an element of $\nu_{\Omega_i}(\ell)$:

$$\Phi^i \nu_{\Omega_0}^{r_i} + \sum_{j=0}^{i-1} \Phi^j \nu_{\mathcal{V}}^{r_j} \in \nu_{\Omega_i}(\ell)$$

This point is in fact the i^{th} element defined by $x_{k+1} = \Phi x_k + v_k$ with $x_0 = \nu_{\Omega_0}^{r_i}$ and $v_k = \nu_{\mathcal{V}}^{r_{i-k-1}}$.

Contrary to what was done in Section 4.2.2, we do not get directly support vectors of the Ω_i, but the inputs that drive the system to these support vectors, which is interesting from the control synthesis point of view, but if we want these support vectors, additional computations are required. Indeed we can get $\nu_{\Omega_N}^{\ell}$ by simulating the system with $x_0 = \nu_{\Omega_0}^{r_i}$ and $v_k = \nu_{\mathcal{V}}^{r_{i-k-1}}$: $\mathcal{O}\left(Nd^2\right)$ operations are required. If we get support vectors for all i using simulations, then it will require $\mathcal{O}\left(N^2 d^2\right)$ operations. Instead, if N is bigger than d, it is more efficient to first compute all the Φ^j in $\mathcal{O}\left(Nd^3\right)$ operations and then all the $\Phi^j \nu_{\mathcal{V}}^{r_j}$ in $\mathcal{O}\left(Nd^2\right)$. Finally, the sums $\sum_{j=0}^{i-1} \Phi^j \nu_{\mathcal{V}}^{r_j}$ can be computed using:

$$\sum_{j=0}^{i} \Phi^j \nu_{\mathcal{V}}^{r_j} = \left(\sum_{j=0}^{i-1} \Phi^j \nu_{\mathcal{V}}^{r_j}\right) + \Phi^i \nu_{\mathcal{V}}^{r_i}$$

Getting support vectors is thus rather costly compared to evaluating the support functions. From Proposition 5.1, we know that Algorithm 5.1 computes $\rho_{\Omega_0}(\ell), \ldots, \rho_{\Omega_{N-1}}(\ell)$, with time complexity:

$$\mathcal{O}\left(N\left(d^2 + \mathfrak{C}_{\rho_{\Omega_0}}^t + \mathfrak{C}_{\rho_{\mathcal{V}}}^t\right)\right)$$

Even if we can get support vectors for free from the evaluation of ρ_{Ω_0} and $\rho_{\mathcal{V}}$, $\mathcal{O}\left(Nd^3\right)$ operations remains to be done.

Remark. *It is possible to do a little bit better if d is bigger than $\log N$. We can first compute the sequence of Φ^{2^k} in $\mathcal{O}\left(\log N d^3\right)$ operations, then each $\Phi^j \nu_{\mathcal{V}}^{r_j}$ requires a number of linear transformations equal to the number of 1's in the base 2 representation of j. Computing the support vectors of all the Ω_i in one direction from $\nu_{\Omega_0}^{r_j}$ and $\nu_{\mathcal{V}}^{r_j}$ requires $\mathcal{O}\left((N+d)d^2 \log N\right)$ operations.*

5.2 Improved Time Discretization

In this section we take advantage of the expressiveness of support functions to get a better over-approximation Ω_0 of $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$.

As in the two previous chapters, we have a linear ordinary differential equation:

$$\dot{x}(t) = Ax(t) + u(t)$$

with $x(0) \in \mathcal{X}_0$ and $u(t) \in \mathcal{U}$. And we want to compute $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$, the set of points reachable from $\mathcal{X}_0$ in time less than T.

We do it by covering $\mathcal{R}_{[0;T]}(\mathcal{X}_0)$ by N convex sets defined by the recurrence relation:

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

where Φ is $e^{\delta A}$, $\mathcal{V}$ is an over-approximation of $\mathcal{R}_\delta(\{0\})$ and Ω_0 an over-approximation of $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$, with $\delta = T/N$.

Lemma 4.1, on page 56, already gives us an over-approximation of $\mathcal{R}_\delta(\{0\})$, as the Minkowski sum of $\delta\mathcal{U}$ and a box. The following lemma gives us an over-approximation of $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0)$ by a set with an easy-to-compute support function.

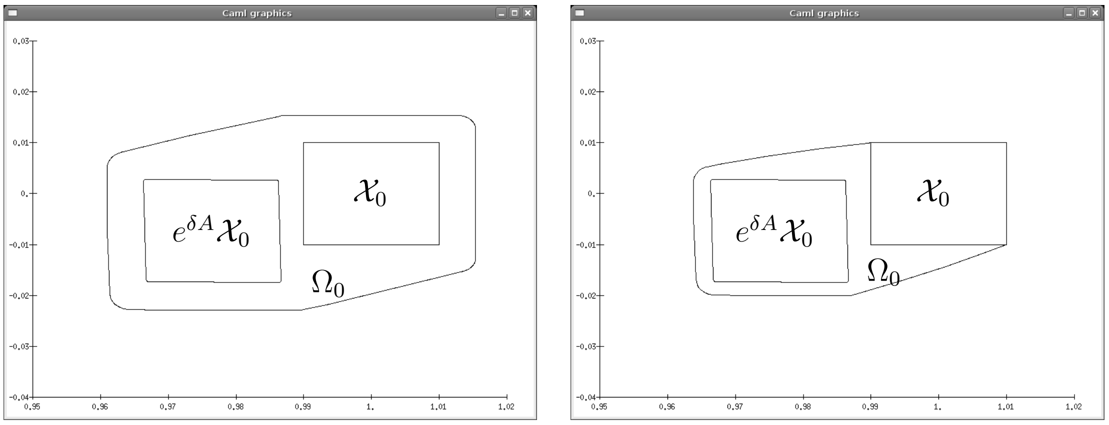


Figure 5.1: Approximating $\mathcal{R}_{[0;\delta]}$ for Example 4.1 with $\mu = 0.5$ using the discretization procedure from Section 3.2 (left) or 5.2 (right).

Lemma 5.1. *Let $\lambda \in [0;1]$, and $\Omega_{0,\lambda}$ be the convex set defined by :*

$$\begin{aligned}\Omega_{0,\lambda} = (1-\lambda)\mathcal{X}_0 \oplus \lambda e^{\delta A}\mathcal{X}_0 \oplus \lambda(1-\lambda)\mathcal{E}_{\mathcal{X}_0} \\ \oplus \lambda\delta\mathcal{U} \quad \oplus \lambda^2\mathcal{E}_{\mathcal{U}}\end{aligned} \tag{5.3}$$

where

$$\begin{aligned}\mathcal{E}_{\mathcal{X}_0} =& \square\left(|A|^{-1}\left(e^{\delta|A|} - I\right) \boxdot \left(A(I - e^{\delta A})\mathcal{X}_0\right)\right) \\ &\oplus \square\left(|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) \boxdot \left(A^2 e^{\delta A}\mathcal{X}_0\right)\right) \\ \textit{and } \mathcal{E}_{\mathcal{U}} =& \square\left(|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) \boxdot (A\mathcal{U})\right)\end{aligned}$$

If we define Ω_0 *as:*

$$\Omega_0 = CH\left(\bigcup_{\lambda\in[0;1]} \Omega_{0,\lambda}\right) \tag{5.4}$$

then, $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0) \subseteq \Omega_0$ *and*

$$\begin{aligned} d_H\left(\Omega_0, \mathcal{R}_{[0;\delta]}(\mathcal{X}_0)\right) &\leq \tfrac{1}{4}\left(e^{\delta\|A\|}-1\right)D_{\mathcal{X}_0} + \tfrac{1}{2}R_{\mathcal{E}_{\mathcal{X}_0}} + 2R_{\mathcal{E}_{\mathcal{U}}} \\ &\leq \tfrac{1}{4}\left(e^{\delta\|A\|}-1\right)D_{\mathcal{X}_0} + 2\left(e^{\delta\|A\|}-1\right)^2\left(\tfrac{R_{\mathcal{X}_0}}{2} + \tfrac{R_{\mathcal{U}}}{\|A\|}\right) \end{aligned} \tag{5.5}$$

Proof. (see A.3.1 on page 129) □

One feature of this approximation is that $\Omega_{0,1}$, the part of $\Omega_0 = \overline{\mathcal{R}}_{[0;\delta]}$ corresponding to time δ, is equal to $\Phi\Omega_{0,0} \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}}$, which is the part of $\Omega_1 = \Phi\Omega_0 \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}}$ corresponding to time δ. This allows a smoother approximation of the reachable set, as illustrated on Figures 5.1 and 5.2.

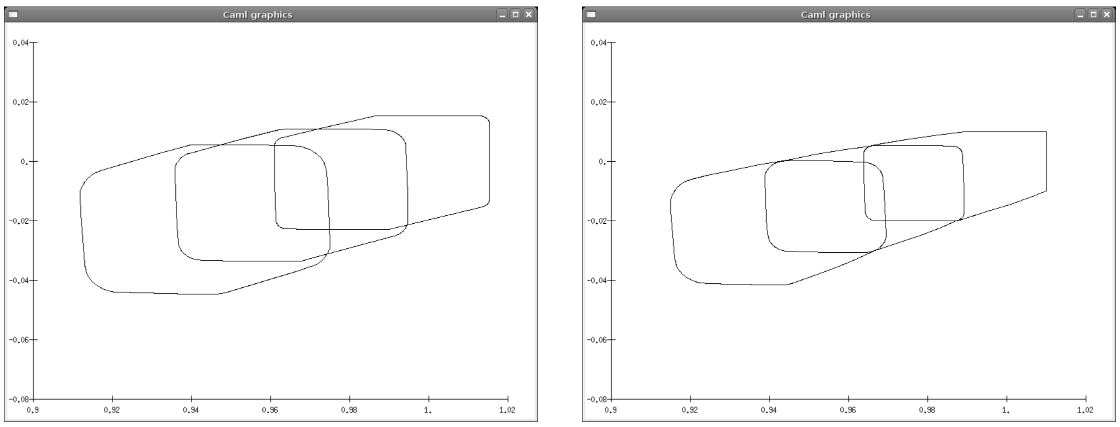


Figure 5.2: Ω_0, Ω_1, and Ω_2, as defined by Example 4.1 with $\mu = 0.5$ using the discretization procedure from Section 3.2 (left) or 5.2 (right).

Ω_0, as defined in this lemma, might seem hard to represent. In fact, its support function is not harder to compute than the one of $\mathcal{X}_0$ and $\mathcal{U}$. Since the signs of λ, $(1-\lambda)$, λ^2, and $\lambda(1-\lambda)$ do not change on $[0;1]$, we have:

$$\begin{aligned} \rho_{\Omega_0}(\ell) = \sup\nolimits_{\lambda\in[0;1]}\big((1-\lambda)\rho_{\mathcal{X}_0}(\ell) &+ \lambda\rho_{\mathcal{X}_0}((e^{\delta A})^\top\ell) + \lambda(1-\lambda)\rho_{\mathcal{E}_{\mathcal{X}_0}}(\ell) \\ &+ \lambda\delta\rho_{\mathcal{U}}(\ell) \qquad + \lambda^2\rho_{\mathcal{E}_{\mathcal{U}}}(\ell)\big) \end{aligned}$$

which means that we only have to maximize a polynomial in one variable of degree 2 on $[0;1]$ after the evaluation of the support function of the sets involved.

$\mathcal{E}_{\mathcal{X}_0}$ and $\mathcal{E}_{\mathcal{U}}$ will be first expressed as boxes using $2d$ calls to $\rho_{\mathcal{X}_0}$ and $\rho_{\mathcal{U}}$ respectively and a few linear transformations. Then these boxes will be represented by their support functions.

Proposition 5.3. *The time complexity of ρ_{Ω_0} as defined in equation* (5.4) *is*

$$\mathfrak{C}^t_{\rho_{\Omega_0}} = \mathcal{O}\left(d^2 + \mathfrak{C}^t_{\rho_{\mathcal{X}_0}} + \mathfrak{C}^t_{\rho_{\mathcal{U}}}\right)$$

We already know from Proposition 4.4 that we can not improve asymptotically the bounds on the error given by Theorem 3.1, but we can state a similar result.

Theorem 5.1. *Consider the sequence of sets defined by equations* (5.4) *and* (4.3)*. Then, for all $i \in \mathbb{N}$, we have $\mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{X}_0) \subseteq \Omega_i$ and*

$$d_H\left(\Omega_i, \mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{X}_0)\right) \leq \delta e^{T\|A\|}\left(\frac{\|A\|}{4} D_{\mathcal{X}_0} + \frac{3}{4}\delta\|A\|^2 e^{\delta\|A\|} R_{\mathcal{X}_0} + e^{\delta\|A\|} R_{\mathcal{U}}\right)$$

Proof. (see A.3.2 on page 132) □

From this point of view, there is no significant improvement between this theorem and Theorem 3.1. From a practical point of view, we already know that it might produce a much smaller over-approximation as illustrated on Figure 4.3a.

5.3 Continuous Reachability Algorithm

As explained in the introduction of Chapter 3, it is possible to avoid time discretization and work directly with the continuous equation by translating the differential equation on the state variable into a differential equation on the parameters of the set representation.

We want to compute the reachable set of:

$$\dot{x}(t) = Ax(t) + u(t)$$

with $x(0) \in \mathcal{X}_0$ and $u(t) \in \mathcal{U}$.

We are interested in $\mathcal{R}_t$ for all t in $[0;T]$. Similar to the discrete-time case, we want to express $\rho_{\mathcal{R}_t}$ as a function of $\rho_{\mathcal{X}_0}$ and $\rho_{\mathcal{U}}$.

Proposition 5.4. *For all $t \in \mathbb{R}^+$,*

$$\rho_{\mathcal{R}_t}(\ell) = \rho_{\mathcal{X}_0}\left(e^{tA^\top}\ell\right) + \int_0^t \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right) d\tau. \quad (5.6)$$

Proof. (see A.3.3 on page 133) □

For the practical computation of the support function of the reachable sets, we introduce, similar to the discrete-time case, auxiliary functions $r : \mathbb{R}^+ \to \mathbb{R}^d$ and $s : \mathbb{R}^+ \to \mathbb{R}$ defined by the differential equations:

$$\dot{r}(t) = A^\top r(t) \qquad r(0) = \ell, \quad (5.7)$$
$$\dot{s}(t) = \rho_{\mathcal{U}}(r(t)) \qquad s(0) = 0. \quad (5.8)$$

Equivalently, we have

$$r(t) = e^{tA^\top}\ell \text{ and } s(t) = \int_0^t \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right) d\tau$$

Using Proposition 5.4, it follows that the support function of the reachable set $\mathcal{R}_t$ can be computed using the following equation

$$\rho_{\mathcal{R}_t}(\ell) = \rho_{\mathcal{X}_0}(r(t)) + s(t).$$

Hence, the evaluation of the support function of the reachable sets can be done by simulating the differential equations (5.7) and (5.8). Naturally in order to guarantee a conservative approximation, one must use guaranteed integration, and take care about what happens between the integration steps.

Most of the remarks done for discrete systems in Section 5.1.1 and 5.1.2 can be adapted to the continuous case.

5.4 Experimental Results

In this section, we show the effectiveness of our approach on some examples. Algorithm 5.1 has been implemented in OCaml, without any of the improvements proposed in Section 5.1.2. All computations were performed on a Pentium IV 3.2GHz with 1GB RAM.

5.4.1 RLC Model of a Transmission Line

The first example we consider is a verification problem for a transmission line borrowed from [Han05]. The goal is to check that the transient behavior of a long transmission line is acceptable both in terms of overshoot and of response time. Figure 5.3 shows a model of the transmission line, which consists of a number of RLC components (R: resistor, L: inductor, C: capacitor) modelling segments of the line. The left side is the sending end and the right side is the receiving end of the transmission line.

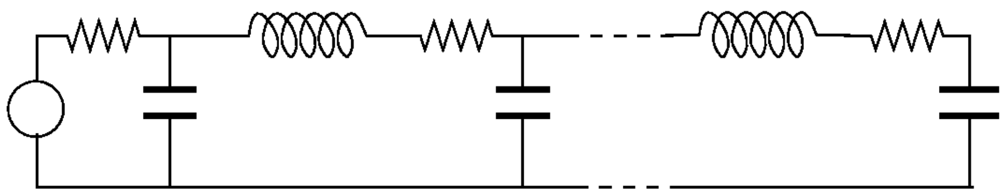


Figure 5.3: RLC model of a transmission line

The dynamics of the system are given by the single-input single-output linear dynamical system

$$\begin{cases} \dot{x}(t) &= Ax(t) + bu_{in}(t), \quad u_{in}(t) \in \mathcal{U},\ x(0) \in \mathcal{X}_0, \\ u_{out}(t) &= cx(t) \end{cases}$$

where $x(t) \in \mathbb{R}^d$ with $d = 81$ is the state vector containing the voltage of the capacitors and the current of the inductors and $u_{in}(t) \in \mathcal{U} \subseteq \mathbb{R}$ is the voltage at the sending end. The output of the system is the voltage $u_{out}(t) \in \mathbb{R}$ at the receiving end, since $u_{out}(t) = cx(t)$ we will take $\ell = c^\top$.

Initially, the system is supposed to be in an ε-neighborhood (with $\varepsilon = 0.01$) of the set of steady states for an input voltage inside $[-0.2; 0.2]$. Then, at time $t = 0$, the input voltage is switched to a value in $[0.99; 1.01]$:

$$\mathcal{X}_0 = -A^{-1}b\,[-0.2; 0.2] \oplus \varepsilon\mathrm{B},\ U = [0.99; 1.01].$$

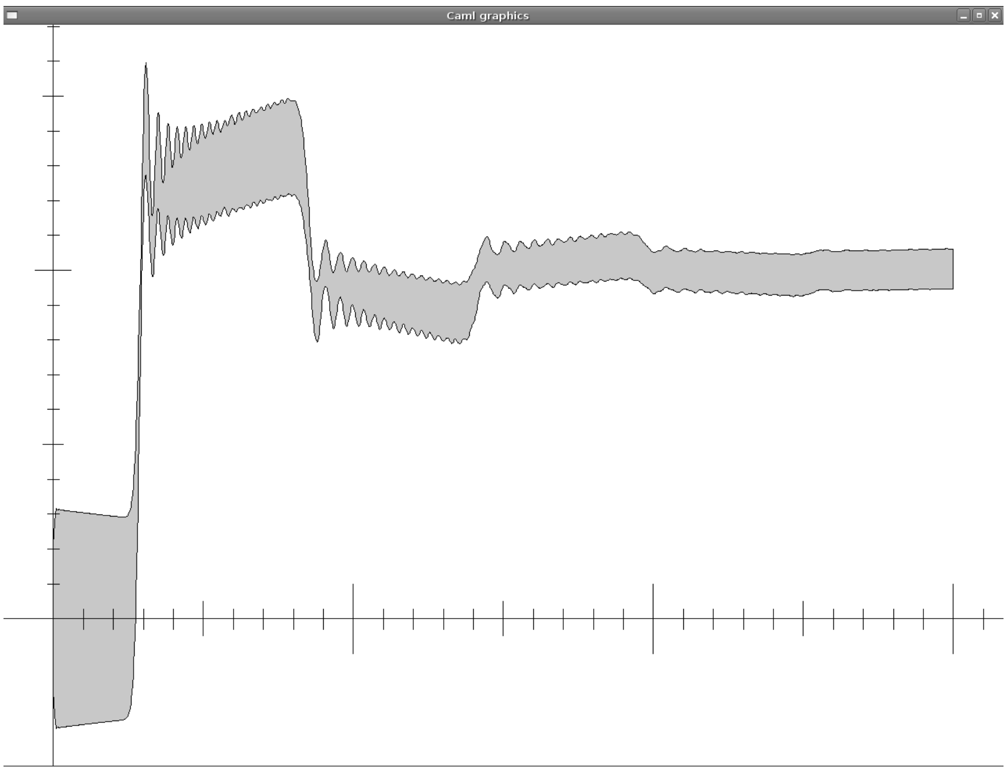


Figure 5.4: Reachable values of $u_{out}(t)$ against time t.

Figure 5.4 shows the reachable values of the output voltage for a time horizon of 3ns. It was computed in 0.10s using 0.234MB.

5.4.2 Extensive Experiments

Our implementation has also been tested on randomly generated examples of different dimension. Tables 5.1 and 5.2 summarize the results of our experimentations. The shading denotes a qualitative interpretation of the values: lighter is better. We computed polyhedral over-approximations of the reachable sets Ω_0, …, Ω_{N-1} with $N = 100$, for random matrices A of dimension d. We either used Algorithm 4.2 (denoted as *direct* in the tables), or sampled Algorithm 5.1 (denoted as

$d =$	10	20	50	100	200	500
direct Z 1	< 0.01	0.01	0.13	1.00	5.44	85.9
sf Z 1	< 0.01	< 0.01	0.01	0.01	0.05	0.28
direct E 1	< 0.01	0.02	0.27	1.71	11.8	
sf E 1	< 0.01	< 0.01	< 0.01	0.02	0.05	0.31
direct Z d	< 0.01	0.02	0.27	1.86	11.4	
sf Z d	< 0.01	0.02	0.23	1.5	11.1	
direct E d	0.01	0.04	0.41	2.82	21.9	
sf E d	< 0.01	0.02	0.19	1.48	8.98	
direct Z d^2	0.04	0.35	7.38	90.6		
sf Z d^2	0.04	0.36	9.83			
direct E d^2	0.03	0.26	6.69			
sf E d^2	0.03	0.32	9.16			

Table 5.1: Execution time (in seconds) for $N = 100$ for several linear systems of different dimensions

$d =$	10	20	50	100	200	500
direct Z 1	0.234	0.234	0.234	0.703	2.258	13.43
sf Z 1	0.234	0.234	0.234	0.469	1.480	8.707
direct E 1	0.234	0.234	0.469	1.172	4.961	
sf E 1	0.234	0.234	0.234	0.703	2.332	12.53
direct Z d	0.234	0.234	0.469	1.172	3.195	
sf Z d	0.234	0.234	0.234	0.703	2.184	
direct E d	0.234	0.469	0.937	3.281	7.332	
sf E d	0.234	0.234	0.234	0.703	3.035	
direct Z d^2	0.703	2.812	18.28	77.81		
sf Z d^2	0.234	0.469	3.75			
direct E d^2	0.703	3.047	18.98			
sf E d^2	0.234	0.469	3.75			

Table 5.2: Memory consumption (in MB) for $N = 100$ for several linear systems of different dimensions

sf in the tables); initial and inputs set were given either as zonotopes of order 1 (Z) or ellipsoids (E); and the computed tight over-approximation consisted in the intersection of 2, $2d$, or $2d^2$ half-spaces. Algorithms are terminated after 90s.

We can see that Algorithm 5.1 has great performances for systems with a single output: it can compute exact bounds on this output for the first 100 timesteps in less than a third of a second for a 500 dimensionnal system.

Part II

Hybrid Systems

Chapter 6

Introduction to Hybrid Systems Reachability

Résumé : *Dans la première partie de cette thèse nous nous sommes intéressé à l'analyse d'atteignabilité des systèmes linéaires. Dans cette seconde partie nous allons adapter nos résultats à l'étude des systèmes hybrides.*
Les systèmes hybrides sont des systèmes qui présentent à la fois une dynamique continue et une dynamique discrète. Par exemple une balle rebondissante peut être modélisée par un système hybride : à chaque rebond la vitesse de la balle est instantanément changée, entre les rebonds la balle est en chute libre et sa position ainsi que sa vitesse évoluent suivant une équation différentielle.
Dans ce chapitre nous décrivons une modélisation classique des systèmes hybrides : les automates hybrides. Nous présentons ensuite le schéma algorithmique classique pour l'analyse d'atteignabilité des automates hybrides.

Hybrid dynamical systems are systems that exhibit both continuous and discrete dynamics. They can arise when a digital controller interacts with a physical system, as in flight control systems. They can also come from the simplification of a sigmoidal behavior by a step function, this is common in biology where some genes can go from an inactive state to a fully expressed state in a relatively short time.

A classical example of a hybrid system, where the discrete dynamics comes from *collision*, is the bouncing ball. The ball is dropped from an initial height and bounces on the floor. With each bounce, modeled as an inelastic collision, the speed of the ball is instantaneously changed; between each bounce, the ball is in free fall, and its speed and position evolve according to a differential equation.

In this chapter, we first describe a classical model for hybrid systems called hybrid automata, then we present a classical algorithmic scheme for reachability analysis of hybrid automata.

6.1 Hybrid Automata

Hybrid automata are formal models that combine discrete automata with continuous variables that change over time [MMP91, ACHH93]. In each location of the discrete automaton, the evolution of the continuous variables is specified by a differential equation. Discrete transitions, or jumps, are triggered by some geometrical condition on the continuous variables, and produce an instantaneous transformation on these variables. For a more formal definition:

Definition 6.1 (Hybrid Automaton)**.** *A* hybrid automaton *H is a tuple*

$$H = (\mathcal{Q}, \mathit{Var}, \mathit{Inv}, \mathit{Flow}, \mathit{Trans}, \mathit{Init})$$

consisting of the following elements:

Locations: *elements of $\mathcal{Q}$ are called* locations*, they are the vertices of a graph whose edges, called* discrete transitions*, are given by Trans;*

Variables: *a* state *of the automaton consists of a location and a value for each variable (formally described as a* valuation *over Var). The set of all states of the automaton is called its* state space*. To simplify the presentation, we assume that the state space is $\mathcal{Q} \times \mathbb{R}^d$, where d is the number of variables. We will also simply write y to denote the name of the variable y or its value according to the context; x sometimes denotes the vector of all variables;*

Invariants: *for each location q, the variables can only take values in a given set called* invariant *and denoted $\mathcal{I}_q$. The invariants are given by $\mathit{Inv} \subseteq \mathcal{Q} \times \mathbb{R}^d$;*

Flow: *the set $\mathit{Flow} \subseteq \mathcal{Q} \times \mathbb{R}^d \times \mathbb{R}^d$ defines the continuous change of the variables over time for each location. In a location q, x can take the values of a function $\xi(t)$ if at each time instant t, $(q, \dot{\xi}(t), \xi(t)) \in \mathit{Flow}$, where $\dot{\xi}(t)$ denotes the derivative of $\xi(t)$ with respect to time;*

Transitions: *the* discrete transitions *$\mathit{Trans} \subseteq \mathcal{Q} \times 2^{\mathbb{R}^d \times \mathbb{R}^d} \times \mathcal{Q}$ specify instantaneous changes of the state of the automaton. A transition (q, μ, q') signifies the system can instantaneously jump from any state (q, x) to any state (q', x') if $(q', x') \in \mathit{Inv}$ and $(x, x') \in \mu$. The relation μ is called the* jump relation[1] *of the transition;*

Initial States: *a set of states $\mathit{Init} \subseteq \mathcal{Q} \times \mathbb{R}^d$ specifies the* initial states *from which all behavior of the automaton begins.*

[1]Also known as *reset map*.

We can now define the hybrid automaton of the bouncing ball. Even though this example has only one location, its behaviors consist of alternating time elapse and discrete transitions that show much of the complexity that hybrid systems can exhibit.

Example 6.1. *Consider a bouncing ball, whose height above ground and vertical speed are measured by continuous variables x and v. We can model this system with the hybrid automaton shown in Figure 6.1, which has only one location q and one transition. The equations of motion of the ball lead directly to a description of the continuous dynamics in the form of a set of ordinary differential equations:*

$$Flow = \{(l, \dot{x}, \dot{v}, x, v) : \dot{x} = v, \dot{v} = -g\},$$

where g is the gravitational constant. If we suppose the ground to be at height $x = 0$, we can model the fact that the ball does not penetrate the ground as an invariant:

$$Inv = \{(l, x, v) | x \geq 0\}.$$

As the ball reaches the ground at $x = 0$, it bounces; we abstract this phenomenon by an instantaneous change of its speed, which changes direction and is reduced by some constant positive factor $c \leq 1$ that captures the loss of energy due to deformation. This instantaneous change is modeled by a transition (q, μ, q), with the jump relation

$$\mu = \{(x, v, x', v') : x = 0 \wedge v < 0 \wedge x' = 0 \wedge v' = -cv\}.$$

Note that we include the constraint $v < 0$ since the ball only bounces when it moves towards the ground. If this constraint is omitted, the model erroneously contains the behavior that when the system reaches $x = 0$ it carries out the discrete transition infinitely often in zero time, v converging towards zero. Assuming that initially the ball is at rest at a height x_0, we define the initial states to be

$$Init = \{(q, x, v) : x = x_0, v = 0\},$$

which is indicated in Figure 6.1 by an incoming arrow to the location.

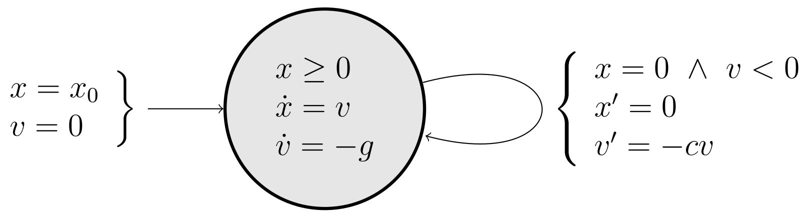


Figure 6.1: Hybrid automaton model of a bouncing ball

The behavior of a hybrid automaton is defined by its *executions*, which are sequences of time elapse and discrete transitions. Formally, an *execution* σ is a finite or infinite sequence of states (q_i, x_i), delays $\delta_i \in \mathbb{R}^{\geq 0}$ and differentiable functions $\xi_i : \mathbb{R} \to \mathbb{R}^d$,

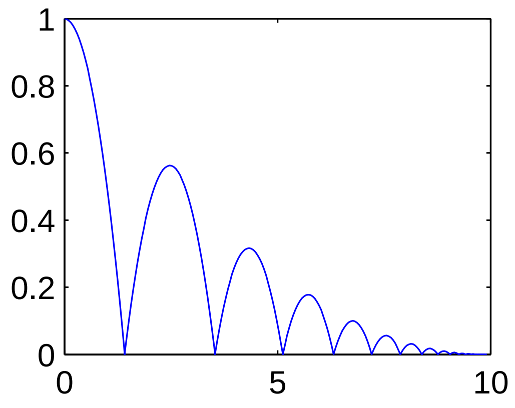
$$\sigma = (q_0, x_0) \xrightarrow{\delta_0,\xi_0} (q_1, x_1) \xrightarrow{\delta_1,\xi_1} (q_2, x_2) \xrightarrow{\delta_2,\xi_2} \cdots \tag{6.1}$$

such that for all $i \geq 0$ the following conditions are satisfied:

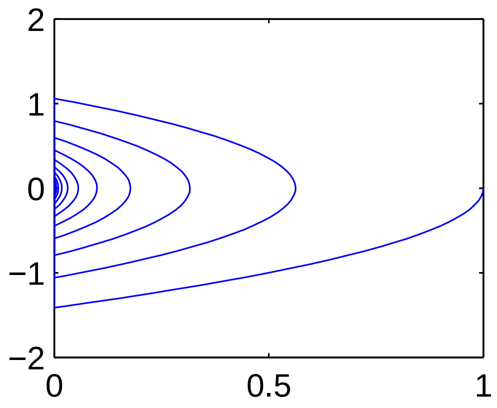
- $(q_0, x_0) \in \mathit{Init}$, $(q_i, x_i) \in \mathit{Inv}$;
- $\xi_i(0) = x_i$ and for all $t, 0 \leq t \leq \delta_i$, $(q_i, \xi_i(t)) \in \mathit{Inv}$,
$$(q_i, \dot{\xi}_i(t), \xi_i(t)) \in \mathit{Flow}$$
- there is a transition (q_i, μ_i, q_{i+1}) such that $(\xi_i(\delta_i), x_{i+1}) \in \mu_i$.

A state (q, x) is *reachable* if there is an executation with $(q, x) = (q_i, \xi_i(t))$, $0 \leq t \leq \delta_i$, for some i.

The continuous component of an execution of the bouncing ball is shown in Figure 6.2a, for constants $g = 1$, $c = 3/4$ and $x_0 = 1$. The states that are reachable from this initial state are shown in Figure 6.2b.



(a) Position x over time t.



(b) Speed v over position x.

Figure 6.2: Behavior of the bouncing ball model.

6.2 Reachability Analysis of Hybrid Automata

The computation of the set of reachable states is generally very costly. For all but the most simple classes of systems, the problem is known to be undecidable [ACH+95, HKPV98]. One therefore often resorts to computing a simpler over-approximation of the set of reachable states.

The basic algorithm for computing the set of reachable states is a simple fixed point calculation. Starting from the set of initial states *Init*, one adds the successor states of time elapse and discrete transitions until no new states are found, and therefore a fixed point is reached. These sucessors states are computed with the following operators:

- The states that are reachable from $\{q\} \times \mathcal{X}$ by letting time elapse are

$$\begin{aligned}\mathcal{R}_{\text{loc}}(q, \mathcal{X}) \;=\; \{(q, x) : \exists \xi, t',\ \xi(0) \in \mathcal{X} \wedge \xi(t') = x \wedge \\ \forall\, 0 \le t \le t',\ \xi(t) \in \mathcal{I}_q \wedge (q, \dot{\xi}(t), \xi(t)) \in \mathit{Flow}\}\end{aligned} \tag{6.2}$$

- Given a set of states $\{q\} \times \mathcal{Y}$, the states that are reachable by taking a discrete transition $e = (q, \mu, q') \in \mathit{Trans}$ are given by

$$\mathcal{R}_{\text{jump}}(e, \mathcal{Y}) = \{(q', x') : \exists x \in \mathcal{Y},\ (x, x') \in \mu\} \tag{6.3}$$

The set of reachable states $\mathcal{R}_H\,(\mathit{Init})$ can be computed with Algorithm 6.1.

Algorithm 6.1 Reachability analysis of a hybrid automaton.

```
Input: A hybrid automaton $H$ in which the set of initial states $Init$ is decomposed
    into $\bigcup_{q \in \mathcal{Q}} \{q\} \times \mathcal{X}_{0,q}$.
Output: $\mathcal{R}_H(Init)$.
 1: $L \leftarrow \{(q, \mathcal{X}_{0,q}) :\ q \in \mathcal{Q}\}$
 2: $\Sigma \leftarrow \emptyset$
 3: $\mathcal{R} \leftarrow \emptyset$
 4: while $L \neq \emptyset$ do
 5:     Pick $(q, \mathcal{X}) \in L$
 6:     $\Sigma \leftarrow \Sigma \cup (\{q\} \times \mathcal{X})$
 7:     $\mathcal{Y} \leftarrow \mathcal{R}_{\text{loc}}(q, \mathcal{X})$            ▷ Reachable set by continuous evolution
 8:     $\mathcal{R} \leftarrow \mathcal{R} \cup (\{q\} \times \mathcal{Y})$
 9:     for $e \in Trans$ of the form $e = (q, \mu, q')$ do
10:         $\mathcal{X}' \leftarrow \mathcal{R}_{\text{jump}}(e, \mathcal{Y})$          ▷ Reachable set by discrete evolution
11:         if $\{q'\} \times \mathcal{X}' \not\subseteq \Sigma$ then
12:             Insert $(q', \mathcal{X}')$ in $L$          ▷ Insert in $L$ if not explored yet
13:         end if
14:     end for
15: end while
16: return $\mathcal{R}$
```

This algorithm, implemented in various tools [CPPAV06, DFGLG08], computes the unbounded time reachable set; even for purely continuous system it is rarely possible to have a useful approximation of the unbounded time reachable set in finite time, thus $\mathcal{R}_{\text{loc}}$ might not terminate. In order to deal with this problem we often resort to bounded time reachability. It can be done by modifying Algorithm 6.1, but it can be cumbersome. Another way is to modify the input hybrid system by adding one variable t whose derivative is 1 in every location, and add transitions to a sink location s when t becomes bigger than the time bound T. If the original hybrid system H is defined by $(\mathcal{Q}, \mathit{Var}, \mathit{Inv}, \mathit{Flow}, \mathit{Trans}, \mathit{Init})$, we analyze H_T defined by:

$$\begin{aligned}
\mathcal{Q}_T &= \mathcal{Q} \cup \{s\} \\
\mathit{Var}_T &= \mathit{Var} \cup \{t\} \\
\mathit{Inv}_T &= \mathit{Inv} \cap (\mathcal{Q} \times \{(x,t) : t \leq T\}) \\
\mathit{Flow}_T &= \{(q, x', 1, x, t) : (q, x', x) \in \mathit{Flow} \wedge 0 \leq t \leq T\} \\
\mathit{Trans}_T &= \big\{(q, \mu_T, q') : \exists \mu,\ (q, \mu, q') \in \mathit{Trans} \wedge \\
&\qquad\qquad \mu_T = \{(x, t, x', t) : (x, x') \in \mu \wedge 0 \leq t \leq T\}\big\} \\
&\quad \cup \big\{(q, \mu, s) : q \in \mathcal{Q} \wedge \mu = \{(x, T, 0, T) : x \in \mathbb{R}^d\}\big\} \\
\mathit{Init}_T &= \mathit{Init} \times \{0\}
\end{aligned}$$

But this is not enough to guarantee the termination of Algorithm 6.1. One problem that can arise is *Zeno* execution: an infinite number of transition in a finite amount of time. An example of such an execution is given by the bouncing ball as defined in Example 6.1. One way around this problem is to impose a bound J on the number of discrete transitions taken in a way similar to the bound on time, then the algorithm computes the set of points reachable in less than time T and with less than J jumps. It is also possible to avoid such an execution by imposing a delay in each discrete transition:

$$\begin{aligned}
\mathit{Trans}_T &= \big\{(q, \mu_T, q') : \exists \mu,\ (q, \mu, q') \in \mathit{Trans} \wedge \\
&\qquad\qquad \mu_T = \{(x, t, x', t+\epsilon) : (x, x') \in \mu \wedge 0 \leq t \leq T\}\big\} \\
&\quad \cup \big\{(q, \mu, s) : q \in \mathcal{Q} \wedge \mu = \{(x, t, 0, T) : x \in \mathbb{R}^d \wedge t \geq T\}\big\}
\end{aligned}$$

One disadvantage is that it changes the semantic of the original system.

6.3 Outline of Part II

In this thesis, we consider hybrid automata with the following restrictions:

- The flow in each location q is a linear time-invariant system $\dot{x}(t) = A_q x(t) + u(t)$ with $u(t) \in \mathcal{U}_q$ and $x(t) \in \mathcal{I}_q$:

$$\mathit{Flow} = \{(q, A_q x + u, x) : q \in \mathcal{Q},\ x \in \mathcal{I}_q \text{ and } u \in \mathcal{U}_q\}$$

- Every discrete transition $e = (q, \mu_e, q')$ in *Trans* is triggered when the continuous variables reach some part of the state space $\mathcal{G}_e$ called the guard of e, an affine transformation is then applied to the continuous variables:

$$\mu_e = \{(x, A_e x + u) : x \in \mathcal{G}_e \text{ and } u \in \mathcal{U}_e\}$$

Thus $\mathcal{R}_{\text{loc}}$ must compute the set of points reachable by a linear time-invariant system within an invariant $\mathcal{I}$, and for any transition $e = (q, \mu_e, q')$, $\mathcal{R}_{\text{jump}}(e, \mathcal{Y})$ is $(q', A_e(\mathcal{Y} \cap \mathcal{G}_e) \oplus \mathcal{U}_e)$.

In Chapters 4 and 5 we have presented several techniques for reachability analysis of linear time-invariant systems. In order to use these algorithms in the context of hybrid systems, we must be able to compute discrete transitions with their outputs. When the Ω_i are approximated, the representation of these approximations are rather classical. They can be ellipsoids or polytopes, as an example. In contrast, when the Ω_i are computed exactly, we use much more complex representations which have never been used in this context. Algorithm 4.1 produces high order zonotopes, with many shared generators. Algorithm 5.1 is based on support function, to a vector ℓ it associates the values of the support function of all the Ω_i.

In the following chapters we will show how theses techniques can be used in the context of hybrid systems reachability analysis. In Chapter 7 we explain some of the implications of the invariant. Then, in Chapter 8, we show how to deal with hyperplanar guards before studying the case of zonotopes and support functions. We do not investigate the numerous other aspects of hybrid systems reachability analysis. Most of the results in these chapters have been published in [GLG08b, LGG09a].

Chapter 7

Staying in the Invariant

Résumé : *Dans ce chapitre nous étudions l'atteignabilité pour le système continu défini dans un état discret d'un automate hybride. Contrairement au cas purement continu, un automate hybride défini un invariant dans chaque état discret ; nous voulons calculer l'ensemble des points atteignables sans sortir de cette invariant. Après discrétisation, la séquence d'ensemble que nous étudions est :*

$$\aleph_{i+1} = (\Phi \aleph_i \oplus \mathcal{V}) \cap \mathcal{I}$$

À cause de l'intersection avec l'invariant nous ne pouvons pas décomposer cette relation de récurrence comme dans la première partie. Nous allons donc étudier une sur-approximation de $\aleph_{i+1}$ *définie par*

$$\aleph_i \subset \aleph_i^{\cap} = \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$$

où Ω_i *est l'ensemble atteignable sans prendre en compte l'invariant et* $\mathcal{I}_j$ *est l'ensemble atteignable à partir de l'invariant.*

In the first part of this thesis we discussed ways to compute the bounded time reachable set of linear time-invariant systems, given by the following equation:

$$\dot{x}(t) = Ax(t) + u(t) \text{ with } u(t) \in \mathcal{U}$$

In order to compute reachable sets for a hybrid system, we must compute reachable sets in each location. And because of the invariant, the differential equation becomes:

$$\dot{x}(t) = Ax(t) + u(t) \text{ with } u(t) \in \mathcal{U} \text{ and } x(t) \in \mathcal{I} \tag{7.1}$$

It can be discretized into:

$$\aleph_{i+1} = (\Phi\aleph_i \oplus \mathcal{V}) \cap \mathcal{I} \tag{7.2}$$

This discretization poses two problems. First, if all the continuous trajectories leave the invariant, then, even the slightest over-approximation of the continuous flow might lead to a huge over-approximation of the reachable set, if some of the discrete trajectories do not leave the invariant. Secondly, it forbids us from using the superposition principle to compute efficiently the exact reachable set for the discretized system.

Indeed, in the purely continuous system, we used the associativity of the Minkowski sum to compute Ω_i as the sum of $\Phi^i\Omega_0$ and $\bigoplus_{j=0}^{i-1} \Phi^j\mathcal{V}$. Now, the closed form of $\aleph_i$ is:

$$\aleph_i = \Big(\big(\ldots\big(\big(\Phi^i\aleph_0 \oplus \Phi^{i-1}\mathcal{V}\big) \cap \Phi^{i-1}\mathcal{I} \oplus \Phi^{i-2}\mathcal{V}\big)\ldots\big) \cap \Phi\mathcal{I} \oplus \mathcal{V}\Big) \cap \mathcal{I}$$

Alas, Minkowski sum is not distributive with respect to intersection, nor is intersection distributive with respect to Minkowski sum. And contrary to what is done in Chapters 4 and 5, if we want an exact representation of $\aleph_{i+1}$ we *must* compute it from $\aleph_i$. Thanks to the following proposition we can compute an over-approximation of $\aleph_{i+1}$ using the efficient algorithms described in the previous chapters.

Proposition 7.1. *Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, be three subsets of $\mathbb{R}^d$, then:*

$$(\mathcal{X} \cap \mathcal{Y}) \oplus \mathcal{Z} \subseteq (\mathcal{X} \oplus \mathcal{Z}) \cap (\mathcal{Y} \oplus \mathcal{Z})$$

Equality can occur, especially when $\mathcal{Z}$ is empty, reduced to one point, or a set of isolated points that are sufficiently further apart. But generally these two sets are different. As an example, consider any two non-empty sets $\mathcal{X}$ and $\mathcal{Y}$ with empty intersection, if we take $\mathcal{Z} = \mathbb{R}^d$, then $(\mathcal{X} \cap \mathcal{Y}) \oplus \mathcal{Z}$ is empty, and $(\mathcal{X} \oplus \mathcal{Z}) \cap (\mathcal{Y} \oplus \mathcal{Z})$ is the whole state space.

Corollary 7.1. *Let $\aleph_i$ be the i^{th} term of the sequence defined by equation* (7.2), *then:*

$$\aleph_i \subseteq \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$$

where Ω_i and the $\mathcal{I}_j$ are defined by:

$$\begin{aligned} \Omega_{k+1} &= \Phi\Omega_k \oplus \mathcal{V} & \qquad \Omega_0 &= \aleph_0 \\ \mathcal{I}_{k+1} &= \Phi\mathcal{I}_k \oplus \mathcal{V} & \qquad \mathcal{I}_0 &= \mathcal{I} \end{aligned}$$

Proof. (see A.4.1 on page 134) □

Thanks to this corollary we have two sequences of sets defined by a recurence relation we already studied extensively in previous chapters. Instead of computing $\aleph_i$ we will over-approximate it by $\Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$, in the following this set is denoted by $\aleph_i^\cap$:

$$\aleph_i^\cap = \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$$

7.1 Working with Algorithm 4.1

Algorithm 7.1 is based on Algorithm 4.1 and computes the sequence of $\aleph_i^\cap$. Note that termination is not guaranteed, and one may have to impose a time bound.

Another problem is posed on lines 15 and 17: the set representations that make it possible to compute the sequences of Ω_i and $\mathcal{I}_i$ exactly, zonotopes and support functions, do not allow intersection, either because the class is not closed under intersection (zonotopes) or because intersection is too hard to compute (support functions).

Algorithm 7.1 Approximate Reachability in one location.

```
Input: The matrix $\Phi$, the sets $\aleph_0$ and $\mathcal{V}$, and the invariant $\mathcal{I}$.
Output: An over-approximation of the non-empty sets of the sequence defined
    by equation (7.2): The first non-empty $\aleph_i^\cap$.
 1: $\mathcal{A}_0^\Omega \leftarrow \aleph_0$
 2: $\mathcal{A}_0^\mathcal{I} \leftarrow \mathcal{I}$
 3: $\mathcal{V}_0 \leftarrow \mathcal{V}$
 4: $\mathcal{S}_0 \leftarrow \{0\}$
 5: $\mathcal{I}_0 \leftarrow \mathcal{I}$
 6: $\mathcal{K}_0 \leftarrow \mathbb{R}^d$
 7: $i \leftarrow 0$
 8: while $\aleph_i^\cap \neq \emptyset$ do
 9:     $i \leftarrow i + 1$
10:     $\mathcal{A}_i^\Omega \leftarrow \Phi \mathcal{A}_{i-1}^\Omega$                 $\triangleright\ \mathcal{A}_i^\Omega = \Phi^i \Omega_0$
11:     $\mathcal{A}_i^\mathcal{I} \leftarrow \Phi \mathcal{A}_{i-1}^\mathcal{I}$                 $\triangleright\ \mathcal{A}_i^\mathcal{I} = \Phi^i \mathcal{I}$
12:     $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \oplus \mathcal{V}_{i-1}$                 $\triangleright\ \mathcal{S}_i = \bigoplus_{j=0}^{i-1} \Phi^j \mathcal{V}$
13:     $\mathcal{V}_i \leftarrow \Phi \mathcal{V}_{i-1}$                 $\triangleright\ \mathcal{V}_i = \Phi^i \mathcal{V}$
14:     $\Omega_i \leftarrow \mathcal{A}_i^\Omega \oplus \mathcal{S}_i$                 $\triangleright\ \Omega_i = \Phi^i \Omega_0 \oplus \bigoplus_{j=0}^{i-1} \Phi^j \mathcal{V}$
15:     $\mathcal{K}_i \leftarrow \mathcal{K}_{i-1} \cap \mathcal{I}_{i-1}$                 $\triangleright\ \mathcal{K}_i = \bigcap_{j=0}^{i-1} \mathcal{I}_j$
16:     $\mathcal{I}_i \leftarrow \mathcal{A}_i^\mathcal{I} \oplus \mathcal{S}_i$                 $\triangleright\ \mathcal{I}_i = \Phi^i \mathcal{I} \oplus \bigoplus_{j=0}^{i-1} \Phi^j \mathcal{V}$
17:     $\aleph_i^\cap \leftarrow \Omega_i \cap \mathcal{K}_i$                 $\triangleright\ \aleph_i^\cap = \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$
18: end while
19: return $\{\aleph_0^\cap, \ldots, \aleph_{i-1}^\cap\}$
```

In order to solve this problem, additional approximations are needed. Similarly to Algorithm 4.2, it is possible to add $\text{Approx}_{\oplus}(\cdot,\cdot)$ and $\text{Approx}_{\cap}(\cdot,\cdot)$ on lines where $\oplus$ and $\cap$ are involved. The good news is that, similarly to Algorithm 4.2, linear transformations are first performed exactly, then Minkowski sums are approximated, and finally intersections are approximated. Operations are performed in a precise order, and sets that have been involved in intersections are never involved in Minkowski sums or linear transformations, this facilitates the avoidance of any wrapping effect.

Another advantage of using such an algorithm is that, if the flow of the hybrid system enters again in *this* location, then we do not need to re-compute the $\mathcal{A}_i^{\mathcal{I}}$, $\mathcal{S}_i$, $\mathcal{K}_i$, and $\mathcal{I}_i$. Moreover, only the last $\mathcal{A}_i^{\mathcal{I}}$ and $\mathcal{I}_i$ needs to be kept in memory, since we only need $\mathcal{S}_i$ and $\mathcal{K}_i$ to compute $\aleph_i^{\cap}$.

7.2 Working with Algorithm 5.1

Algorithm 7.2, adapted from Algorithm 5.1, presents the same advantages. Moreover, even if it only outputs an upper-bound on each value of the support functions of the $\aleph_i^{\cap} = \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$ in direction ℓ, we can still use it to represent $\aleph_i^{\cap}$ because:

$$\aleph_i^{\cap} = \bigcap_{\ell \in \mathbb{R}^d} \left\{ x : x \cdot \ell \leq \min\left(\rho_{\Omega_i}(\ell), \min_{j=0}^{i-1} \rho_{\mathcal{I}_j}(\ell) \right) \right\}$$

Algorithm 7.2 Approximate Reachability in one location using support functions.

```
Input: The matrix $\Phi$, the sets $\aleph_0$ and $\mathcal{V}$, the invariant $\mathcal{I}$, and a direction $\ell$.
Output: Upper-bounds of all $\rho_{\aleph_i^{\cap}}(\ell)$.
1: $r_0 \leftarrow \ell$
2: $s_0 \leftarrow 0$
3: $k_0 \leftarrow \infty$
4: $\rho_0^{\Omega} \leftarrow \rho_{\aleph_0}(\ell)$
5: $\rho_0^{\mathcal{I}} \leftarrow \rho_{\mathcal{I}}(\ell)$
6: $\rho_0 \leftarrow \rho_{\aleph_0}(\ell)$
7: $i \leftarrow 0$
8: while $\rho_i \geq -\rho_{\mathcal{I}}(-\ell)$ do
9:     $i \leftarrow i+1$
10:    $r_i \leftarrow \Phi^{\top} r_{i-1}$                      ▷ $r_i = \left(\Phi^{\top}\right)^i \ell$
11:    $s_i \leftarrow s_{i-1} + \rho_{\mathcal{V}}(r_{i-1})$      ▷ $s_i = \sum_{j=0}^{i-1} \rho_{\mathcal{V}}\left(\left(\Phi^{\top}\right)^j \ell\right)$
12:    $\rho_i^{\Omega} \leftarrow \rho_{\aleph_0}(r_i) + s_i$          ▷ $\rho_i^{\Omega} = \rho_{\Omega_i}(\ell)$
13:    $k_i \leftarrow \min(k_{i-1}, \rho_{i-1}^{\mathcal{I}})$         ▷ $k_i = \min_{j=0}^{i-1} \rho_{\mathcal{I}_j}(\ell)$
14:    $\rho_i^{\mathcal{I}} \leftarrow \rho_{\mathcal{I}}(r_i) + s_i$          ▷ $\rho_i^{\mathcal{I}} = \rho_{\mathcal{I}_i}(\ell)$
15:    $\rho_i = \min(\rho_i^{\Omega}, k_i)$             ▷ $\rho_i = \min\left(\rho_{\Omega_i}(\ell), \min_{j=0}^{i-1} \rho_{\mathcal{I}_j}(\ell)\right)$
16: end while
17: return $\{\rho_0, \ldots, \rho_{i-1}\}$
```

The condition on line 8 in Algorithm 7.2 is sufficient to prove emptiness of $\aleph_i^\cap$. Indeed, the value ρ_i defines a halfspace $\{x : x \cdot \ell \leq \rho_i\}$ containing $\aleph_i^\cap$, and the value of $-\rho_{\mathcal{I}}(-\ell)$ defines a halfspace containing $\mathcal{I}$: $\{x : x \cdot \ell \geq -\rho_{\mathcal{I}}(-\ell)\}$. It is clear that $\rho_i < -\rho_{\mathcal{I}}(-\ell)$ implies that these two halfspaces are empty, and that $\aleph_i^\cap = \aleph_i^\cap \cap \mathcal{I} = \emptyset$.

It is also clear that this condition might never be falsified, even if $\aleph_i^\cap$ becomes empty. That is why Algorithm 7.2 should be called on several directions ℓ concurrently, leading to polyhedral over-approximations of the $\aleph_i^\cap$. These approximations can also be computed using Algorithm 7.1 and template polyhedra to represent $\mathcal{S}_i$ and $\mathcal{K}_i$.

The advantages of Algorithm 7.2 are that all the improvements described in Section 5.1.2 can be used, and that refining the approximations can be done efficiently. The advantage of Algorithm 7.1 is that it is not limited to polyhedral approximations.

7.3 Mixing Algorithms

The two algorithms we have presented here use Algorithms 4.1 and 5.1 respectively to compute the sequence of Ω_i and $\mathcal{I}_i$. One interesting consequence is that we can share some computations between this two sequences. But for better approximations it might be interesting to compute these two sequences by two different algorithms. As an example, one can use a wrapping-effect-free algorithm to compute the sequence of Ω_i, and one algorithm from Chapter 3 to compute the sequence of $\mathcal{I}_i$, or even the sequence of $\mathcal{K}_i$ directly.

Another interesting approach would be to first compute wrapping-effect-free approximations of the Ω_i, and then adapt an algorithm from Chapter 3 to compute a sequence:

$$\overline{\aleph}_{i+1} = \overline{\left(\Phi\overline{\aleph}_i \oplus \mathcal{V}\right) \cap \mathcal{I} \cap \overline{\Omega}_{i+1}}$$

The presence of $\overline{\Omega}_i$ helps avoid the wrapping effect inherent to such a computation scheme. The main advantage is that instead of over-approximating $\aleph_i^\cap$, which is already an over-approximation of $\aleph_i$, we are here over-approximating $\aleph_i$ directly. It is then possible to get an approximation $\overline{\aleph}_i$ that is not a superset of $\aleph_i^\cap$. One drawback is that we do not compute the sequence of $\mathcal{K}_i$ and can not reuse it if hybrid trajectories enter this location again.

Remark. *More generally, when we have several algorithms approximating a sequence of sets, each with its strengths and weaknesses, it is always interesting not only to intersect their outputs, but also to let them help each other.*

If the invariant is a polytope, a useful approximation can be obtained by using $\mathcal{L}$-polytopes, where $\mathcal{L}$ contains the normals to the faces of $\mathcal{I}$, for computing $\overline{\Omega}_i$, and a face-lifting algorithm (see Section 3.4.2.3) for $\overline{\mathcal{I}}_i$.

Once the $\overline{\aleph}_i$ has been computed, they can be used to compute an over-approximation of the intersection between $\overline{\aleph}_i$ and the guards. For numerical reasons it might be interesting to compute this intersection before intersecting $\overline{\aleph}_i$ with $\mathcal{I}$, in that case $\mathcal{K}_i$ should be $\bigcap_{j=1}^{i-1} \mathcal{I}_j$ instead of $\bigcap_{j=0}^{i-1} \mathcal{I}_j$.

The case of the intersection with one hyperplanar guard defining an halfspace as invariant is investigated in Chapter 8. This can be extended to the case of polyhedral invariants.

7.4 Examples

We illustrate the efficiency of our method by adding an invariant to Example 4.1. We used Algorithm 7.2 to compute polyhedral approximation (with 50 constraints) of the sets of points reachable while staying in the invariant.

As a first example we use the halfspace defined by $\{(x_0, x_1, x_2, x_3, x_4) : x_1 \leq 0.64\}$. In a second example we use the cylinder defined by $\{(x_0, x_1, x_2, x_3, x_4) : (x_0 - 0.3)^2 + .(x_1 + 0.025)^2 \leq 0.64\}$. In both examples we use the sequence of Ω_i to add bounds on all variables and improve the analysis, thus our first invariant was expressed as a cube and the second one as the product of a circle with a three dimensional cube. Both examples were analyzed in less than 0.5s and using less than 10MB. Due to poor handling of polytopes in our tool, plotting the computed sets took nearly five minutes.

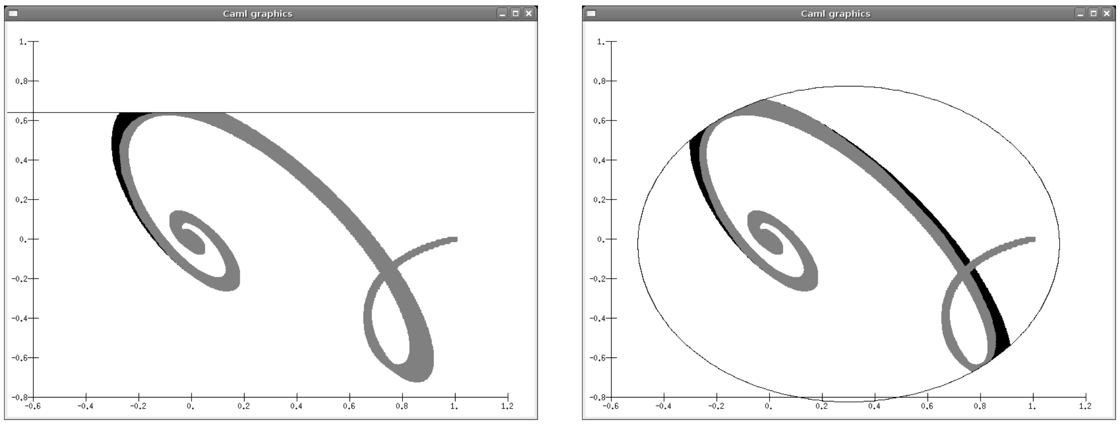


Figure 7.1: Reachability analysis inside an invariant: $\Omega_i \cap \mathcal{I}$ in black and $\overline{\aleph}_i$ in grey.

Chapter 8

Intersection with a Hyperplanar Guard

Résumé : *Dans ce chapitre nous considérons une garde hyperplanaire $\mathcal{G}$ égale à $\{x : x \cdot n = \gamma\}$, l'invariant $\mathcal{I}$ est inclus dans le demi-espace $\{x : x \cdot n \leq \gamma\}$. Nous nous intéressons à l'intersection $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ entre la garde et l'ensemble atteignable par le système continu défini dans l'état discret courant. En appliquant la relation de saut à $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ nous obtiendrons l'ensemble initial dans le nouvel état discret.*
On ne sait pas calculer efficacement la séquence des $\aleph_i$, nous pouvons tout d'abord calculer une première sur-approximation de $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ à partir d'une sur-approximation des $\aleph_i$. Nous pouvons ensuite améliorer cette première approximation grâce à la séquence exacte des Ω_i.
Nous montrons dans ce chapitre comment calculer l'intersection entre un hyperplan et la séquence des Ω_i sous forme de zonotopes ayant beaucoup de générateurs en commun ou sous forme de fonction support.

In this chapter, we consider one hyperplanar guard $\mathcal{G}$ equal to $\{x : x \cdot n = \gamma\}$, the invariant $\mathcal{I}$ is in the halfspace $\{x : x \cdot n \leq \gamma\}$. We are interested in the intersection $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ between the guard and the N first sets of the sequence defined by:

$$\aleph_{i+1} = (\Phi\aleph_i \oplus \mathcal{V}) \cap \mathcal{I}$$

Applying the jump relation to $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ gives us the initial set in the next location. Without loss of generality we suppose here that the jump relation is the identity map; for affine maps, we can use the transformations described in Section 2.3.

Computing the exact sequence of $\aleph_i$ is untractable in general, thus we will over-approximate $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ using two sequences:

- A sequence of sets $\overline{\aleph}_i$ over-approximating the reachable set.
- The N first sets of the sequence defined by:

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$$

Before approximating $\mathcal{Y}_{\mathcal{G}}^{\aleph}$ we first detect for which indices i, $\aleph_i$ may intersect the guard.

8.1 Detecting Intersection

In order to detect for which indices i, $\aleph_i$ intersects the guard we need to know $\rho_{\aleph_i}(n)$, and $\rho_{\aleph_i}(-n)$. Indeed, for any set $\mathcal{X}$:

- If $\rho_{\mathcal{X}}(n) < \gamma$ then $\mathcal{X}$ does not intersect the guard.
- If $-\rho_{\mathcal{X}}(-n) \leq \gamma \leq \rho_{\mathcal{X}}(n)$ then $\mathcal{X}$ does intersect the guard.
- If $\gamma < -\rho_{\mathcal{X}}(-n)$ then $\mathcal{X}$ is outside of the invariant[1].

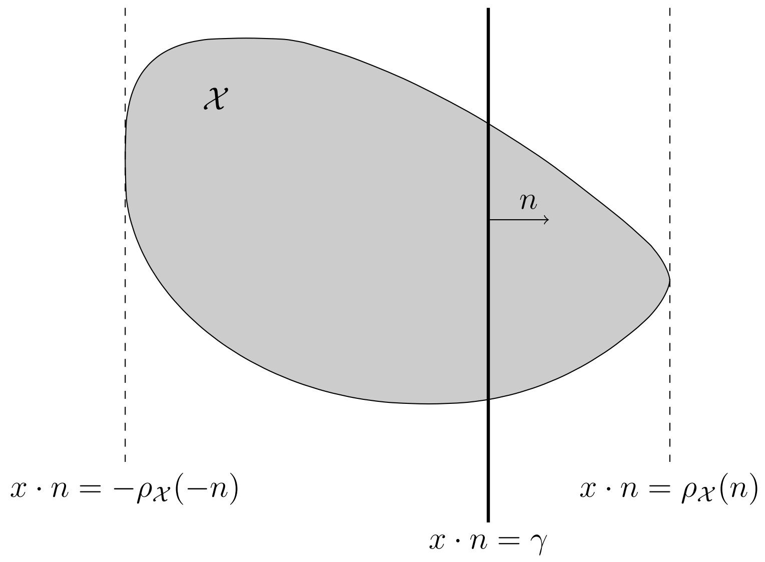

$\mathcal{X}$
n
$x \cdot n = -\rho_{\mathcal{X}}(-n)$
$x \cdot n = \rho_{\mathcal{X}}(n)$
$x \cdot n = \gamma$

Figure 8.1: Checking emptiness of $\mathcal{X} \cap \{x : x \cdot n = \gamma\}$.

Detecting the first index i such that $\aleph_i$ intersects the guard can be done very efficiently. Indeed, for all j smaller than this index, we have $\aleph_j = \Omega_j$, thus i is the first index such that $\gamma \leq \rho_{\Omega_i}(n)$.

Unfortunately, we can not always compute $\rho_{\aleph_i}(n)$ and $\rho_{\aleph_i}(-n)$ efficiently after this first intersection, and only have over-approximations of these values. This is sufficient to reject some indices where we know intersection can not occur. The approximations of $\rho_{\aleph_i}(n)$, and $\rho_{\aleph_i}(-n)$ given by Algorithm 7.2, can be used as a first filter, then one can use $\rho_{\overline{\aleph}_i}(n)$, and $\rho_{\overline{\aleph}_i}(-n)$ on the remaining indices[2].

[1] which means in our context that $\aleph_i$ is empty, and $\rho_{\aleph_i}(-n) = -\infty$.

[2] $\rho_{\overline{\aleph}_i}(n)$ can be smaller than the computed approximation $\rho_{n,i}$ of $\rho_{\aleph_i}(n)$, even if $\overline{\aleph}_i$ is computed by sampling Algorithm 7.2. Indeed, the constraint implied by $\rho_{n,i}$ can be made redundant by some constraints in several other directions.

In the end, we have a set I of indices such that $\overline{\aleph}_i$ intersects $\mathcal{G}$ if i is in I, and $\aleph_i$ does not intersect $\mathcal{G}$ if i is not in I. The set I will be usually partitioned into several sets of consecutive indices. Without loss of generality we consider that I is $[i_{\min} : i_{\max}]$.

Then the intersection between the reachable set and the flow can be over-approximated by $\overline{\aleph}_{i_{\min}} \cap \mathcal{G}$, …, and $\overline{\aleph}_{i_{\max}} \cap \mathcal{G}$. Leading to $i_{\max} - i_{\min} + 1$ initial sets in the next location. In order to avoid an explosion of the number of initial sets to consider, we use instead:

$$\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}} = \mathrm{CH}\left(\bigcup_{i=i_{\min}}^{i_{\max}} \left(\overline{\aleph}_i \cap \mathcal{G}\right)\right)$$

or an approximation of this set.

Remark. *It is sometimes easier to compute instead CH$\left(\left(\bigcup_{i=i_{\min}}^{i_{\max}} \overline{\aleph}_i\right) \cap \mathcal{G}\right)$, but this leads to a coarser approximation.*

Each intersection $\overline{\aleph}_i \cap \mathcal{G}$ can be computed efficiently if $\overline{\aleph}_i$ is an ellipsoid or a polyhedron represented by its faces. Computing their convex union is much harder but it can be approximated. The resulting set might be greatly improved by intersecting it with (an approximation of):

$$\mathcal{Y}_{\mathcal{G}}^{\Omega} = \mathrm{CH}\left(\bigcup_{i=i_{\min}}^{i_{\max}} \left(\Omega_i \cap \mathcal{G}\right)\right)$$

Indeed, if the dot product between the flow and the normal to $\mathcal{G}$ has a constant sign over $\mathcal{Y}_{\mathcal{G}}^{\Omega}$, then $\mathcal{Y}_{\mathcal{G}}^{\Omega}$ is equal to $\mathrm{CH}\left(\bigcup_{i=i_{\min}}^{i_{\max}} (\aleph_i \cap \mathcal{G})\right)$. Even if this is not the case, computing $\mathcal{Y}_{\mathcal{G}}^{\Omega}$ can provide an improvement over $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$.

Remark. *$\mathcal{Y}_{\mathcal{G}}^{\Omega}$ is defined using the indices $i_{\min}$ and $i_{\max}$ computed thanks to the sequence of $\overline{\aleph}_i$. This allows a faster computation and a better estimate since some intersections do not need to be computed.*

In the two following sections, we will show how $\mathcal{Y}_{\mathcal{G}}^{\Omega}$ can be computed, or at least approximated, when the sequence of Ω_i is computed by Algorithm 4.1 using zonotopes, or Algorithm 5.1 using support functions.

8.2 Zonotope/Hyperplane Intersection

An efficient implementation of Algorithm 4.1 using zonotopes outputs a sequence of zonotopes of increasingly high order with a lot of generators in common. For now, we forget about these redundancies and only consider the intersection between one high order zonotope $\mathcal{Z}$ and a hyperplane $\mathcal{G}$ equal to $\{x : x \cdot n = \gamma\}$.

The problem with such an intersection is that the end result might not be a zonotope, as already stated in Section 2.3.4, in fact it can be any polytope.

Proposition 8.1. *Let $\mathcal{G}$ be a hyperplane, and $\mathcal{P}$ a polytope in $\mathcal{G}$. There exists a zonotope $\mathcal{Z}$ such that $\mathcal{P} = \mathcal{Z} \cap \mathcal{G}$.*

Proof. Without loss of generality we will take $\mathcal{G}$ as $\{x : x_0 = 1\}$. If $\mathcal{V}$ is the set of vertices of $\mathcal{P}$ let us define the zonotope $\mathcal{Z}$:

$$\mathcal{Z} = \left\{ \sum_{v \in \mathcal{V}} \alpha_v v : 0 \leq \alpha_v \leq 1 \right\}$$

Let us note that since the first component of every vertices of $\mathcal{P}$ is 1, the first component of $\sum_{v \in \mathcal{V}} \alpha_v v$ is $\sum \alpha_v$, then the intersection of $\mathcal{Z}$ with $\mathcal{G}$ is:

$$\mathcal{Z} \cap \mathcal{G} = \left\{ \sum_{v \in \mathcal{V}} \alpha_v v : 0 \leq \alpha_v \leq 1 \text{ and } \sum \alpha_v = 1 \right\}$$

which is exactly the convex hull of $\mathcal{V}$, in other words:

$$\mathcal{Z} \cap \mathcal{G} = \mathcal{P}$$

□

Thus, if we want to compute this intersection we must work with a class of sets containing at least all convex polytopes. The good news is that computing a $\mathcal{H}$- or $\mathcal{V}$-representation of a zonotope can be done polynomially in the number of its faces [Zas75] or vertices [AF96] respectively, the bad news is that a zonotope with r generators in dimension d might have up to $2\binom{r}{d-1}$ faces and $2\sum_{i=0}^{d-1}\binom{r-1}{i}$ vertices. Even for relatively small zonotopes, this can be prohibitively large. Thus, it is clear that this approach is untractable. Another approach is to over-approximate the zonotope, either via a zonotope of lower order or directly by a polytope, before computing the intersection. However, even if the over-approximation of the zonotope is tight (i.e. the over-approximation touches the zonotope in several points), the over-approximation of the intersection is generally not. Moreover, we want to compute this intersection in order to improve over $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$ which is already computed from an approximation. Instead, we propose a third approach which allows to compute a tight over-approximation of the intersection directly, without approximating or computing a $\mathcal{H}$- or $\mathcal{V}$-representation of $\mathcal{Z}$.

8.2.1 From Dimension d to Dimension 2

We will not over-approximate the intersection by a zonotope directly, but first by a polytope which will then be approximated by, or expressed as, a zonotope. As already discussed in Chapter 2, approximating a set by a polytope is sampling its support function. Moreover, the support function of the intersection between a set $\mathcal{S}$, and a hyperplane $\mathcal{G}$ equal to $\{x : x \cdot n = \gamma\}$, can be expressed as [RW98]:

$$\rho_{\mathcal{S}\cap\mathcal{G}}(\ell) = \inf_{\lambda\in\mathbb{R}} \left(\rho_{\mathcal{S}}(\ell - \lambda n) + \lambda\gamma\right)$$

which can be computed by looking for a vector in the plane (or line) generated by ℓ and n. Instead of using this expression we will use the following proposition.

Proposition 8.2. *Let $\mathcal{G}$ be a hyperplane, $\mathcal{G} = \{x \in \mathbb{R}^d : x \cdot n = \gamma\}$, $\mathcal{S}$ a set, and ℓ in $\mathbb{R}^d$. Let $\Pi_{n,\ell}$ be the following linear transformation:*

$$\begin{array}{rccc} \Pi_{n,\ell}: & \mathbb{R}^d & \to & \mathbb{R}^2 \\ & x & \mapsto & (x \cdot n, x \cdot \ell) \end{array}$$

Then, we have the following equality

$$\{x \cdot \ell : x \in \mathcal{S} \cap \mathcal{G}\} = \{y : (\gamma, y) \in \Pi_{n,\ell}(\mathcal{S})\}$$

Proof. Let y belong to $\{x \cdot \ell : x \in \mathcal{S} \cap \mathcal{G}\}$, then there exists x in $\mathcal{S} \cap \mathcal{G}$ such that $x \cdot \ell = y$. Since $x \in \mathcal{G}$, we have $x \cdot n = \gamma$. Therefore $(\gamma, y) = \Pi_{n,\ell}(x) \in \Pi_{n,\ell}(\mathcal{S})$ because $x \in \mathcal{S}$. Thus, $y \in \{y : (\gamma, y) \in \Pi_{n,\ell}(\mathcal{S})\}$.

Conversely, if $y \in \{y : (\gamma, y) \in \Pi_{n,\ell}(\mathcal{S})\}$, then $(\gamma, y) \in \Pi_{n,\ell}(\mathcal{S})$. It follows that there exists $x \in \mathcal{S}$ such that $x \cdot n = \gamma$ and $x \cdot \ell = y$. Since $x \cdot n = \gamma$, it follows that $x \in \mathcal{G}$. Thus, $y = x \cdot \ell$ with $x \in \mathcal{S} \cap \mathcal{G}$ and it follows that $y \in \{x \cdot \ell : x \in \mathcal{S} \cap \mathcal{G}\}$. □

This proposition states that we can reduce the problem of evaluating the support function of the intersection of a set $\mathcal{S}$ and a hyperplane $\mathcal{G}$ in one direction, to the problem of applying a rank 2 linear transformation, $\Pi_{n,\ell}$, to $\mathcal{S}$ and then computing the intersection of the 2-dimensional set $\Pi_{n,\ell}(\mathcal{S})$ and the line $L_\gamma = \{(x, y) \in \mathbb{R}^2 : x = \gamma\}$. For any set of directions $\mathcal{L}$ computing the $\mathcal{L}$ polytope $\square_{\mathcal{L}}(\mathcal{S})$ can be done by $|\mathcal{L}|$ linear transformation of rank 2 and $|\mathcal{L}|$ intersections in two dimensions. Algorithm 8.1 implements this idea, which is illustrated on Figure 8.2.

Algorithm 8.1 Dimension reduction

Input: A set $\mathcal{S}$, a hyperplane $\mathcal{G} = \{x \in \mathbb{R}^d : x \cdot n = \gamma\}$ and a finite set $\mathcal{L}$ of directions.

Output: $\square_{\mathcal{L}}(\mathcal{S})$.

```
1: for $\ell$ in $\mathcal{L}$ do
2:     $\mathcal{S}^\pi_{n,\ell} \leftarrow \Pi_{n,\ell}(\mathcal{S})$
3:     $\rho_\ell \leftarrow$ BOUND_INTERSECT_2D($\mathcal{S}^\pi_{n,\ell}$, $L_\gamma$)
4: end for
5: return $\{x \in \mathbb{R}^d : \forall \ell \in \mathcal{L}, x \cdot \ell \leq \rho\ell\}$
```

All the $\Pi_{n,\ell}$ involve multiplication by $n^\top$, in order to take advantage of this redundancy, one may apply all the $\Pi_{n,\ell}$ using one $(|\mathcal{L}| + 1) \times d$ matrix. Moreover, if for all ℓ in $\mathcal{L}$, $-\ell$ also belongs to $\mathcal{L}$ then $\Pi_{n,-\ell}(\mathcal{S})$ need not to be computed and $\rho_{-\ell}$ can be computed using $\Pi_{n,\ell}(\mathcal{S})$. Then all the two dimensional intersection problems can be generated using one $(\frac{|\mathcal{L}|}{2} + 1) \times d$ matrix.

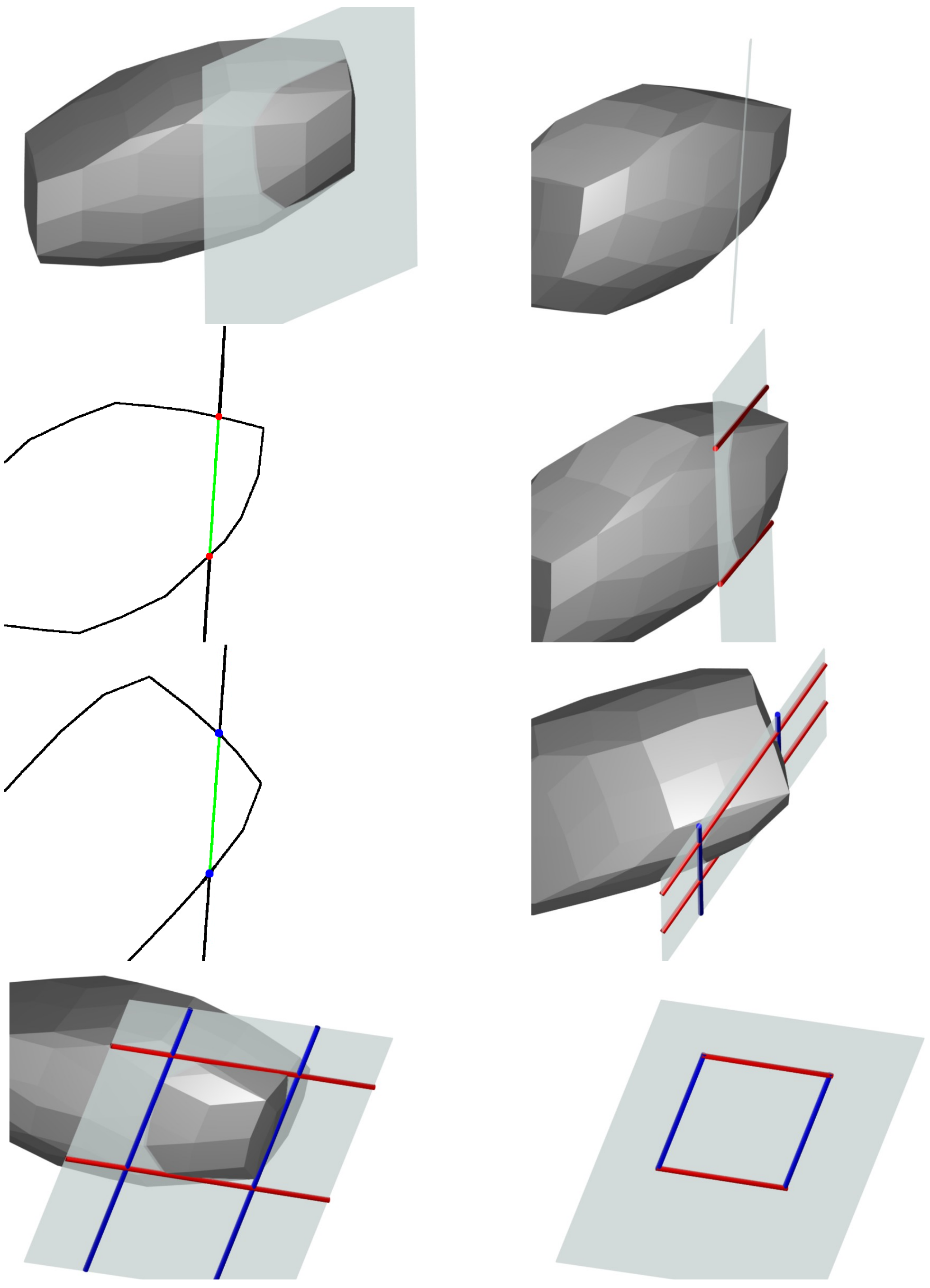


Figure 8.2: Polyhedral approximation of the intersection between a set and a hyperplane can be done using projections and intersections in 2d only.

In our case, the set $\mathcal{S}$ is a zonotope, then the projection $\Pi_{n,\ell}(\mathcal{S})$ is a two-dimensional zonotope, a zonogon, which can be computed efficiently:

$$\Pi_{n,\ell}(\langle c|g_1,\ldots,g_r\rangle) = \langle \Pi_{n,\ell}(c)|\Pi_{n,\ell}(g_1),\ldots,\Pi_{n,\ell}(g_r)\rangle \ .$$

The computation of the intersection of $\Pi_{n,\ell}(\mathcal{S})$ and the line L_γ is investigated in the next subsection where two algorithms are proposed to solve this problem.

Remarks. *One should keep in mind that this dimension reduction technique is not specific to zonotopes but applies to any set. It is only efficient if the image by a linear transformation can be computed easily, which excludes $\mathcal{H}$-polytopes, but includes any implicit Minkowski sum of a combination of $\mathcal{V}$-polytopes and ellipsoids as already evoked in Section 4.1.1.*

Furthermore, it can be easily adapted to the intersection with a halfspace. Another variant would be to allow reduction to any subspace of $\mathbb{R}^d$, whether because efficient intersection algorithms exist in dimension higher than two, or in order to study the intersection with several halfplanes or halfspaces.

8.2.2 Intersection of a Zonogon and a Line

Algorithm 8.1 requires the computation of the support function of the intersection of a two dimensional zonotope with a line in direction $(0,1)$. In a two dimensional space, a zonotope is called a zonogon and its number of vertices, as its number of edges, is two times its number of generators. Thus, it is possible to express a zonogon as a polygon (two dimensional polytope) which can easily be intersected with a line. We now denote by $\mathcal{Z} = \langle c|g_1,\ldots,g_r\rangle$ the zonogon that we want to intersect with $L_\gamma = \{(x,y) : x = \gamma\}$. An extremely naive way of determining the list of vertices of a zonogon is to generate the list of points $\{c + \sum_{i=1}^{r} \alpha_i g_i : \forall i, \alpha_i = -1 \text{ or } \alpha_i = 1\}$ and then to take the convex hull of this set. This is clearly not a good approach since we need to compute a list of 2^r points.

Instead, we will look for the edge of $\mathcal{Z}$ intersecting L_γ with the highest y-coordinate. A first algorithm works by scanning the vertices and is mainly intended to help understand the second algorithm.

Remark. *This problem is very similar to the fractional Knapsack problem. Eppstein suggested in a talk [BE01] that one could maximize a linear function on the intersection of a zonotope and a hyperplane by adapting the greedy algorithm for the fractional Knapsack problem. This is actually what Algorithm 8.2 does. Algorithm 8.3, on the other hand, is similar to Bamas and Zemel's algorithm [BZ80].*

In the following, for any vector v, x_v and y_v will denote the x- and y-coordinates of v respectively.

8.2.2.1 Scanning the Vertices.

It is well known that the facets of a zonotope $\langle c|g_1,\dots,g_r\rangle$ are zonotopes whose generators are taken from the list $\{g_1,\dots,g_r\}$. Then, we can deduce that the edges of a zonogon are segments of the form $[P; P+2g]$ where P is a vertex of the zonogon and g a generator. Therefore, it is sufficient to scan the generators in trigonometric (or anti-trigonometric) order to scan the vertices of the zonogon in a way that is similar to the gift wrapping algorithm [Jar73]. This idea is implemented in Algorithm 8.2.

Algorithm 8.2 BOUND_INTERSECT_2D

Input: A zonogon $\mathcal{Z} = \langle c|g_1,\dots,g_r\rangle$ and a line $L_\gamma = \{(x,y) : x = \gamma\}$ such that $\mathcal{Z} \cap L_\gamma \neq \emptyset$.

Output: $\rho_{\mathcal{Z}\cap L_\gamma}((0,1)^\top)$.

```
 1: $P \leftarrow c$                                          ▷ current position
 2: $j \leftarrow 1$
 3: for $i$ from 1 to $r$ do
 4:     if $y_{g_i} > 0$ or ($y_{g_i} = 0$ and $x_{g_i} > 0$) then      ▷ $g_i = (x_{g_i}, y_{g_i})$
 5:         $g_i \leftarrow -g_i$              ▷ Ensure all generators are pointing downward
 6:     end if
 7:     $P \leftarrow P - g_i$                  ▷ Drives $P$ toward the highest vertex of $\mathcal{Z}$
 8: end for
 9: $g_1,\dots,g_r \leftarrow \mathrm{SORT}(g_1,\dots,g_r)$    ▷ Sort the generators in trigonometric order
10: if $x_P < \gamma$ then
11:     $g_1,\dots,g_r \leftarrow g_r,\dots,g_1$          ▷ Sort the generators in clockwise order
12: end if
13: while $[P; P+2g_j] \cap L_\gamma = \emptyset$ do
14:     $P \leftarrow P + 2g_j$
15:     $j \leftarrow j + 1$
16: end while
17: $(x,y) \leftarrow [P; P+2g_j] \cap L_\gamma$
18: return $y$
```

All the generators are taken pointing downward for simplicity. This does not change the zonogon since replacing a generator g by it opposite $-g$ does not modify the shape of a zonogon. Then, we compute the highest[3] vertex of $\mathcal{Z}$, and sort the generators according to the trigonometric or clockwise order. Scanning the generators in that order allows us to scan the vertices of $\mathcal{Z}$. While scanning these vertices, we check for the intersection with the line L_γ. This leads to an algorithm for the intersection between a line and a zonogon with r generators whose complexity is $\mathcal{O}(r \log r)$. The most time consuming part is to sort the generators.

[3]meaning the vertex with the highest y-coordinate, for simplicity of the explanations we assume it to be unique. Actually, it is the rightmost vertex amongst the highest vertices.

Remark. *The lowest vertex of the intersection can be found by stopping the* `while` *loop only after a second intersection is detected.*

In practice, the number of generators r can be very large (remember that the zonogon we want to intersect comes from the reachable set Ω_k computed by Algorithm 4.1; Ω_k has about kd generators). Further, each time a discrete transition occurs, this procedure is called several times by Algorithm 8.1 (one call for each direction of approximation). Thus, we need it to be as fast as possible. Hence, instead of scanning all the vertices of $\mathcal{Z}$, we look directly for the two edges that intersect the line L_γ with a dichotomic search.

8.2.2.2 Dichotomic Search of the Intersecting Edges.

We start again from the vertex of $\mathcal{Z}$ with the highest y-coordinate. At each step of the algorithm, P is a vertex of the zonogon representing the current position and G is a set of generators such that the segment $[P; P + \sum_{g\in G} 2g]$ intersects the line L_γ. We choose a pivot vector s and split the generators in G into two sets, G_1 and G_2, of generators respectively above and below s. Then, it is clear that L_γ intersects either $[P; P + \sum_{g\in G_1} 2g]$ or $[P + \sum_{g\in G_1} 2g; P + \sum_{g\in G} 2g]$. We continue either with P and G_1 or $P + \sum_{g\in G_1} 2g$ and G_2. Algorithm 8.3 implements this approach. Figures 8.3 and 8.4 illustrate the execution of the algorithm.

With a good pivot selection algorithm [BFP+73], the dichotomic search has a linear complexity. For our problem, we choose the sum of the remaining generators as the pivot. Even though this leads to a quadratic theoretical worst case complexity, it improves the practical behavior. Indeed, the sum of the remaining generators is already available and it has a nice geometric interpretation, as illustrated in Figure 8.4.

At each step, P and $P + \sum_{g\in G} 2g$ are both the closest computed vertex to the line $\{(x, y) : x = \gamma\}$, each on a different side of this line, thus defining the best computed under-approximation of $\rho_{\mathcal{Z}\cap L_\gamma}((0, 1)^\top)$ at this step. A pivot s defines a vertex $Q = P + \sum_{g\in G_1} 2g$ between P and $P + \sum_{g\in G} 2g$. The line of direction s going through Q is *tangent* to $\mathcal{Z}$ and its intersection with L_γ defines an over-approximation of $\rho_{\mathcal{Z}\cap L_\gamma}((0, 1)^\top)$. Choosing $s = \sum_{g\in G} g$ as the pivot ensures that the distance between the over-approximation and the under-approximation of the value we are looking for is not correlated with γ, the position of the intersecting line.

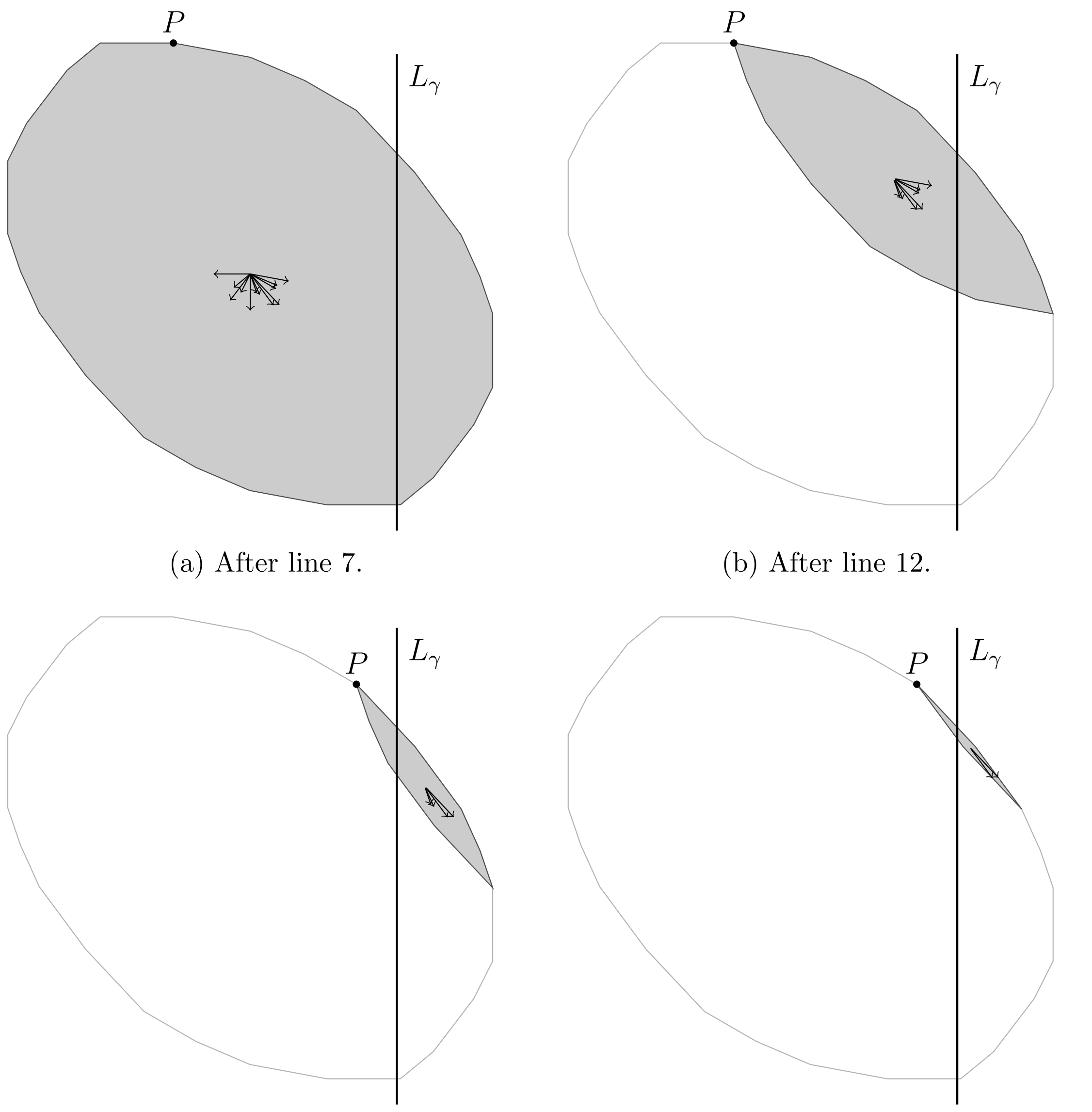
Remark. *Algorithms 8.2 and 8.3 require that the intersection between $\mathcal{Z}$ and L_γ is not empty. If one does not use an exact arithmetic it is possible that, even though the algorithm for the support function of $\mathcal{Z}$ returns two values indicating that $-\rho_{\mathcal{Z}}(-n) < \gamma < \rho_{\mathcal{Z}}(n)$, both algorithms fail to find an intersection. In that case one has to decide whether the intersection should be considered empty, or if $\rho_{\mathcal{Z}\cap L_\gamma}((0, 1)^\top)$ should be deduced from a support vector of $\mathcal{Z}$ in direction n or $-n$.*

Algorithm 8.3 BOUND_INTERSECT_2D

Input: A zonogon $\mathcal{Z} = \langle c|g_1, \ldots, g_r\rangle$ and a line $L_\gamma = \{(x, y) : x = \gamma\}$ such that $\mathcal{Z} \cap L_\gamma \neq \emptyset$.

Output: $\rho_{\mathcal{Z}\cap L_\gamma}((0, 1)^\top)$.

```
1: $P \leftarrow c$                                        ▷ current position $P = (x_P, y_P)$
2: for $i$ from 1 to $r$ do
3:     if $y_{g_i} > 0$ or ($y_{g_i} = 0$ and $x_{g_i} > 0$) then          ▷ $g_i = (x_{g_i}, y_{g_i})$
4:         $g_i \leftarrow -g_i$                ▷ Ensure all generators are pointing downward
5:     end if
6:     $P \leftarrow P - g_i$                    ▷ Drives $P$ toward the highest vertex of $\mathcal{Z}$
7: end for
8: if $x_p < \gamma$ then
9:     $G \leftarrow \{g_1, \ldots, g_r\} \cap (\mathbb{R}^+ \times \mathbb{R})$                    ▷ We should look right
10: else
11:     $G \leftarrow \{g_1, \ldots, g_r\} \cap (\mathbb{R}^- \times \mathbb{R})$                              ▷ or left
12: end if
13: $s \leftarrow \sum_{g\in G} 2g$
14: while $|G| > 1$ do
15:     $(G_1, G_2) \leftarrow$ SPLIT_PIVOT$(G, s)$
16:     $s_1 \leftarrow \sum_{g\in G_1} 2g$
17:     if $[P; P + s_1]$ intersects $L_\gamma$ then
18:         $G \leftarrow G_1$
19:         $s \leftarrow s_1$
20:     else
21:         $G \leftarrow G_2$
22:         $s \leftarrow s - s_1$
23:         $P \leftarrow P + s_1$
24:     end if
25: end while                                    ▷ Only one generator remains
26: $(x, y) \leftarrow [P; P + s] \cap L_\gamma$
27: return $y$
```



(a) After line 7.

(b) After line 12.

(c) After one iteration of the `while` loop.

(d) After two iterations of the `while` loop.

Figure 8.3: Dichotomic search of the intersecting edges. Line indications are given with respect to Algorithm 8.3. The light grey set is the zonogon generated by G.

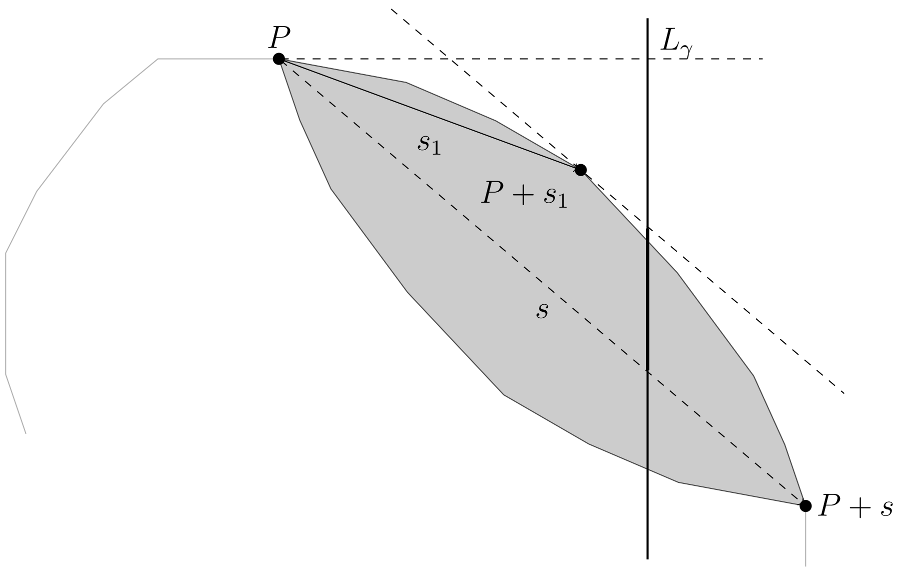


Figure 8.4: A good choice for the pivot allows a smart enclosure of the intersection point.

8.2.3 Intersection of the Reachable Set and a Line

Now that we know how to intersect a zonogon with a line, we can approximate the intersection of a zonotope with a hyperplane, using Algorithm 8.1. Approximating all the $\Omega_{i_{\min}} \cap \mathcal{G}$, ..., $\Omega_{i_{\max}} \cap \mathcal{G}$ allows us to improve the approximation $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$ of $\mathrm{CH}\left(\bigcup_{i=i_{\min}}^{i_{\max}} (\aleph_i \cap \mathcal{G})\right)$.

If the intersections $\Omega_i \cap \mathcal{G}$ are computed independently, we do not exploit the fact that the reachable sets Ω_i have a special structure. They actually share a lot of generators. Indeed, with the notations of Algorithm 4.1, the zonotopes intersecting the guards are:

$$\begin{aligned}
\Omega_{i_{\min}} &= \mathcal{A}_{i_{\min}} \oplus \mathcal{S}_{i_{\min}} \\
\Omega_{i_{\min}+1} &= \mathcal{A}_{i_{\min}+1} \oplus \mathcal{V}_{i_{\min}} \oplus \mathcal{S}_{i_{\min}} \\
&\vdots \\
\Omega_{i_{\max}} &= \mathcal{A}_{i_{\max}} \oplus \mathcal{V}_{i_{\max}-1} \oplus \ldots \oplus \mathcal{V}_{i_{\min}} \oplus \mathcal{S}_{i_{\min}}
\end{aligned}$$

They all share the generators in $\mathcal{S}_{i_{\min}}$. Actually each zonotope Ω_i shares all its generators but the ones in $\mathcal{A}_i$ with the zonotopes of greater index. Consequently, when approximating the intersection $\Omega_i \cap \mathcal{G}$, it is possible to reuse most of the computations already done for smaller indices. Not only the projections of most of the generators of Ω_i have already been computed, but they are also partially sorted.

Moreover, at each step of Algorithm 8.3, one can easily compute an under-approximation, $\underline{\rho}_{\ell,i}$, and an over-approximation, $\overline{\rho}_{\ell,i}$, of $\rho_{\Omega_i \cap \mathcal{G}}(\ell)$ as explained at the end of the previous subsection and on Figure 8.4. It is then possible to modify Algorithm 8.3 in order to compute all the intersection concurrently. Since we are not interested in each individual intersection, but rather in their union, we can add a variable $\underline{\rho}_\ell$ containing the maximum of all $\underline{\rho}_{\ell,i}$, and at each step:

- Determine the index j such that for all i in the same concurrent computation, $\overline{\rho}_{\ell,i}$ is smaller than $\overline{\rho}_{\ell,j}$.
- Choose the pivot according to Ω_j.
- Update all $\underline{\rho}_{\ell,i}$ and $\overline{\rho}_{\ell,i}$.
- Update $\underline{\rho}_\ell$.
- Drop the computation for all i such that $\overline{\rho}_{\ell,i}$ is smaller than $\underline{\rho}_\ell$. Indeed, there can be no support vector of $\mathcal{Y}_\mathcal{G}^\Omega$ in direction ℓ in the set $\Omega_i \cap \mathcal{G}$.
- Start two new concurrent computation base on the branching on line 17 in Algorithm 8.3.

The resulting algorithm might not be efficient for a small number of generators, because one has to maintain several concurrent computations, and each time an index i is dropped its shared generators must be moved to the right index. That is why, instead of stopping when $|G| = 1$, one should stop when $|G|$ reaches some value, and switch to Algorithm 8.2.

8.2.4 From Polytope to Zonotope

In the previous sections we described efficient methods to sample the support function of $\mathcal{Y}_\mathcal{G}^\Omega$ in several directions, when the Ω_i are obtained thanks to Algorithm 4.1 using zonotopes. In other word, we computed a polytope $\mathcal{P}$ represented by its faces that tightly over-approximate $\mathcal{Y}_\mathcal{G}^\Omega$. Then $\mathcal{P} \cap \mathcal{Y}_\mathcal{G}^{\overline{\aleph}}$ will be used as an initial set of points in a new location.

In Algorithm 8.1, we have the choice of the normal vectors to the facets of the approximating polytope $\mathcal{P}$. It might be interesting to choose these direction of approximation according to the shape of $\mathcal{Y}_\mathcal{G}^{\overline{\aleph}}$. But a stronger constraint is the production of a zonotope. Indeed, in order to continue the reachability analysis of the original hybrid system, we need to express $\mathcal{P} \cap \mathcal{Y}_\mathcal{G}^{\overline{\aleph}}$ as a zonotope. Unfortunately this set might not be a zonotope or even a polytope. Moreover, even if it was a polytope, to the best of the author's knowledge, there is no known efficient algorithm for the approximation of a general polytope by a zonotope (except in small dimension [GNZ03]).

We can choose the directions of approximation such that the resulting polytope can be easily approximated by a zonotope. Even better, we can choose these vectors such that the approximating polytope is a zonotope. Indeed, some polytopes are easily expressed as zonotopes; this is the case for the class of parallelotopes and particularly for hyper-rectangles. Hence, we choose the normal vectors to the facets such that the over-approximation $\mathcal{P}$ of the intersection $\mathcal{Y}_{\mathcal{G}}^{\Omega}$ of the reachable set with the the guard is a hyper-rectangle.

Then $\mathcal{P}$ can be used directly as an over-approximation of $\mathcal{P} \cap \mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$, or one can over-approximate this set by a hyper-rectangle by sampling the support function of $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$ in the directions given by the normals to the facets of $\mathcal{P}$.

Expressing the initial set in the next location as a zonotope is not our only constraint, we also want the approximation to be not too coarse. Thus we should not choose *any* set of directions generating a hyper-rectangle. $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$ might help. If not, one can first generate at random (in a way similar to [Mul59]) a set of directions, and only keep the direction ℓ_0 that induces the thinner approximation. Then, we randomly generate a set of directions orthogonal to the directions already chosen, and again we only keep the one for which $\rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}(\ell) + \rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}(-\ell)$ is minimal, until we get a hyper-rectangle after $d-2$ steps.

But even carefully chosen, a hyper-rectangular over-approximation might be too coarse. One way to improve it is to consider a partition of its generators into pairs. Without loss of generality, and in order to simplify the notations, we will consider that $\mathcal{P}$ is a box: $[\underline{x_1}; \overline{x_1}] \times \ldots \times [\underline{x_{d-1}}; \overline{x_{d-1}}] \times [\gamma; \gamma]$.

Grouping the axis by pairs we can express $\mathcal{P}$ as $\mathcal{P}_{1,2} \times \mathcal{P}_{3,4} \times \ldots \times \mathcal{P}_{d-2,d-1} \times [\gamma; \gamma]$ if d is odd or $\mathcal{P}_{1,2} \times \mathcal{P}_{3,4} \times \ldots \times \mathcal{P}_{d-1,d}$ if d is even, where, for all odd i, $\mathcal{P}_{i,i+1}$ is $[\underline{x_i}; \overline{x_i}] \times [\underline{x_{i+1}}; \overline{x_{i+1}}]$. Then, in each plane generated by axis i and $i+1$ we choose new directions of approximation in order to improve over $\mathcal{P}_{i,i+1}$; intersecting the resulting set with the projection of $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$ on the plane generated by axis i and $i+1$, we get a set $\mathcal{Q}_{i,i+1}$. Since $\mathcal{Q}_{i,i+1}$ is two dimensional we can approximate it by a zonotope $\mathcal{Z}_{i,i+1}$ efficiently [GNZ03].

Using the relation between Cartesian products and Minkowski sums already exposed in Section 2.2.4, we can easily express the product of the $\mathcal{Z}_{i,i+1}$ as a zonotope.

This improvement can be significant but is not the prefect cure against the curse of dimensionality (see Section 3.4.3). As an example, the Hausdorff distance between the unit sphere and the unit cube is $\sqrt{d}-1$. Using this technique we can not get a distance smaller than $\sqrt{\lceil d/2 \rceil}-1$, the distance between the unit sphere in dimension d and its over-approximation by a product of unit spheres in dimension 2 (and a segment if d is odd). Moreover it does not allow to take advantage of the combinatorial structure of zonotopes, because all the faces of the resulting set have to be actually computed.

In the next section we try to address this problem.

8.2.5 Playing with the Generators

In the previous sections we showed how to tightly approximate the intersection between a zonotope and a hyperplane. But as already stated in Section 3.4.2.2, tightness is an interesting property but does not always guarantee a good approximation. Moreover, tightness is relative to the sequence of Ω_i, and the resulting set is not tight with respect to the set of points reachable from the initial set in the current location, which is itself already an over-approximation.

Performing an initial over-approximation might reduce the Hausdorff distance while loosing tightness.

Proposition 8.3. *Let $\mathcal{G}$ be a hyperplane, $\mathcal{S}_\mathcal{G}$ a subset of the hyperplane parallel to $\mathcal{G}$ containing 0, and $\mathcal{S}$ a set. Then:*

$$(\mathcal{S}_\mathcal{G} \oplus \mathcal{S}) \cap \mathcal{G} = \mathcal{S}_\mathcal{G} \oplus (\mathcal{S} \cap \mathcal{G})$$

.

Proof. If $\mathcal{G} = \{x : x \cdot n = \gamma\}$ then for all x in $\mathcal{S}_\mathcal{G}$, $x \cdot n = 0$ by definition of $\mathcal{S}_\mathcal{G}$. Then:

$$\begin{aligned}(\mathcal{S}_\mathcal{G} \oplus \mathcal{S}) \cap \mathcal{G} &= \{x_1 + x_2 : x_1 \in \mathcal{S}_\mathcal{G} \text{ and } x_2 \in \mathcal{S} \text{ and } (x_1 + x_2) \cdot n = \gamma\} \\ &= \{x_1 + x_2 : x_1 \in \mathcal{S}_\mathcal{G} \text{ and } x_2 \in \mathcal{S} \text{ and } x_2 \cdot n = \gamma\} \\ (\mathcal{S}_\mathcal{G} \oplus \mathcal{S}) \cap \mathcal{G} &= \mathcal{S}_\mathcal{G} \oplus (\mathcal{S} \cap \mathcal{G})\end{aligned}$$ □

Thanks to this proposition we can approximate the intersection between a zonotope and a hyperplane without computing explicitly all the faces of the resulting approximation. Only $\mathcal{S} \cap \mathcal{G}$ needs to be approximated.

Unfortunately, most of the time, the Ω_i will not be expressible as a sum $\mathcal{S}_\mathcal{G} \oplus \mathcal{S}$. That is why it might be interesting to over-approximate the Ω_i by zonotopes of this form. This is where tightness is lost. Notice that the aim here is not to reduce the number of generators but to have some generators (generating a not too small zonotope) parallel to $\mathcal{G}$. The approximation may have more generators than the initial zonotope.

8.3 Support Function/Hyperplane Intersection

We now consider the approximation of $\mathcal{Y}_\mathcal{G}^\Omega$ when the Ω_i are represented by Algorithm 5.1. All we have is a function f that to a vector ℓ associates the tuple $\left(\rho_{\Omega_{i_{\min}}}(\ell), \ldots, \rho_{\Omega_{i_{\max}}}(\ell)\right)$:

$$f : \ell \mapsto \left(\rho_{\Omega_{i_{\min}}}(\ell), \ldots, \rho_{\Omega_{i_{\max}}}(\ell)\right)$$

Similarly to the previous case we will first only consider the intersection of one set $\mathcal{S}$, represented by its support function, with a hyperplane. Again we will use Proposition 8.2 to reduce the dimension to 2. Finally we will consider the sequence of sets represented by f.

Here we want to express $\mathcal{Y}_\mathcal{G}^\Omega$ by its support function. Thus we will describe an algorithm computing the value of the support function of $\mathcal{Y}_\mathcal{G}^\Omega$ in one direction ℓ. $\mathcal{Y}_\mathcal{G}^\Omega$ will the be represented by this algorithm.

8.3.1 Intersection of a 2-Dimensional Convex Set and a Line

As explained in Section 2.3.5, evaluating the support function of the intersection between a set $\mathcal{S}$ and a hyperplane $\mathcal{G}$, equal to $\{x : x \cdot n = \gamma\}$, can be reduced to the minimization of a unimodal function in one variable:

$$\rho_{\mathcal{S}\cap\mathcal{G}}(\ell) = \inf_{\lambda\in\mathbb{R}} \left(\rho_{\mathcal{S}}(\ell - \lambda n) + \lambda\gamma\right)$$

Instead, we use Proposition 8.2 and apply $\Pi_{n,\ell}$ to $\mathcal{S}$, then we compute the support function of the intersection between $\Pi_{n,\ell}\mathcal{S}$ and the line L_γ in direction $(0,1)^\top$. In order to do so we will minimize the function h:

$$\begin{array}{rccc} h : &]0;\pi[& \to & \mathbb{R} \\ & \theta & \mapsto & \frac{\rho_{\Pi_{n,\ell}\mathcal{S}}(v_\theta) - \gamma\cos\theta}{\sin\theta} \end{array}$$

where:

$$v_\theta = \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix} \qquad \rho_{\Pi_{n,\ell}\mathcal{S}}(v_\theta) = \rho_{\mathcal{S}}\left(\Pi_{n,\ell}^\top v_\theta\right) = \rho_{\mathcal{S}}\left(n\cos\theta + \ell\sin\theta\right)$$

Figure 8.5 gives a geometric interpretation of h. To an angle θ in $]0;\pi[$ we associate a vector $v_\theta = (\cos\theta \;\; \sin\theta)^\top$ in $\mathbb{R}^2$, the codomain of $\Pi_{n,\ell}$. Computing the value of the support function of $\Pi_{n,\ell}\mathcal{S}$ in direction v_θ gives us a supporting line of equation $x\cos\theta + y\sin\theta = \rho_{\Pi_{n,\ell}\mathcal{S}}(v_\theta)$. The value of $h(\theta)$ is the y-coordinate of the intersection between this line and L_γ.

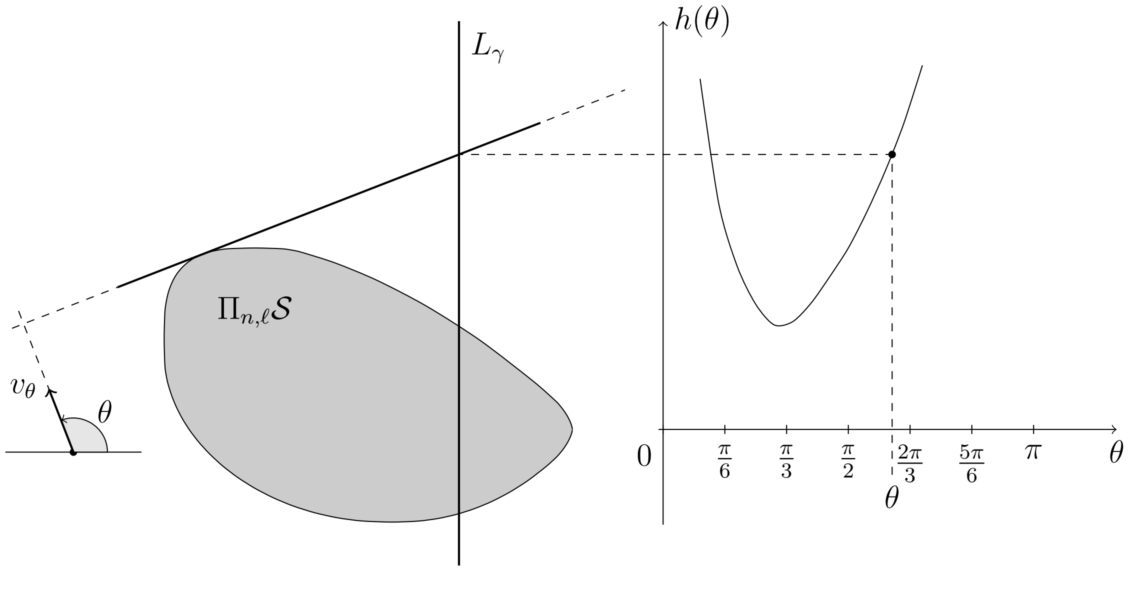


Figure 8.5: Definition of the function h.

In the following, we drop $\Pi_{n,\ell}$ and consider a 2-dimensional set $\mathcal{S}$ for simplicity of notation.

Lemma 8.1. *Let $\mathcal{S}$ be a 2-dimensional set and θ_0, θ_1 be in $]0;\pi[$ such that $\theta_0 < \theta_1$. Let (x_0, y_0) and (x_1, y_1) be support vectors of $\mathcal{S}$ associated to directions v_{θ_0} and v_{θ_1} respectively, we denote by (x, y) the intersection between the two supporting lines. Then:*

$$x_0 \geq x \geq x_1$$

Proof. (see A.5.1 on page 134) □

A geometric interpretation of this lemma is that by scanning the support vectors of $\mathcal{S}$ for angles from 0 to π, we will walk along the boundary of $\mathcal{S}$ from right to left. This will be useful to prove the next theorem.

Theorem 8.1. *For any compact convex set $\mathcal{S}$ and any real γ, the function h defined by:*

$$\begin{array}{rccc} h: &]0;\pi[& \rightarrow & \mathbb{R} \\ & \theta & \mapsto & \frac{\rho_{\mathcal{S}}\left(\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}\right) - \gamma\cos\theta}{\sin\theta} \end{array}$$

is monotonic or unimodal and:

$$\inf_{\theta \in]0;\pi[} h(\theta) = \rho_{\mathcal{S}\cap L_\gamma}\left(\begin{pmatrix}0 \\ 1\end{pmatrix}\right)$$

Proof. (see A.5.2 on page 135) □

We will use this theorem to over-approximate $\rho_{\mathcal{S}\cap L_\gamma}$. A first algorithm minimizes h using a dichotomic search and support vectors, a second one only uses support function with a golden section search [Kie53].

8.3.1.1 Dichotomic Search

Here we present a method very similar to Algorithm 8.3. In this algorithm, computing the vertex with the highest y-coordinate is actually computing a support vector of $\mathcal{Z}$ in direction $(0, 1)^\top$. Then choosing a vector v as a pivot leads to computing a support vector of $\mathcal{Z}$ in a direction orthogonal to v.

Algorithm 8.4 is an adaptation of Algorithm 8.3 to sets represented by their support function. We want to minimize h on $]0;\pi[$, keeping in mind its geometric interpretation. We start with a range $[\theta_0;\theta_1] = [0;\pi]$ and two support vectors P and Q corresponding to the directions defined by angles θ_1 and θ_0 respectively. Since $\mathcal{S}$ is convex, the intersection $(\gamma, y_{\min})$ between the line (PQ) and L_γ is inside $\mathcal{S}\cap L_\gamma$, thus $y_{\min}$ is an under-approximation of $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top = \inf_{\theta\in]0;\pi[} h(\theta)$.

At each step we choose an angle θ in $]\theta_0;\theta_1[$, and compute $h(\theta)$ and a support vector (x_θ, y_θ) of $\mathcal{S}$ in direction $v_\theta = (\cos\theta, \sin\theta)$. If, as an example, x_θ is smaller than γ, then for all θ' in $]\theta;\theta_1[$, $h(\theta') \geq h(\theta)$. Thus we update our range of angles to $[\theta_0;\theta]$, and the point P is replaced by (x_θ, y_θ). The new line (PQ) defines a new lower bound $y_{\min}$ for the minimum of h over $]0;\pi[$. In order to find the exact

Algorithm 8.4 BOUND_INTERSECT_2D

Input: The support function $\rho_{\mathcal{S}}$ of a set $\mathcal{S}$, a function $\nu_{\mathcal{S}}$ returning a support vector of $\mathcal{S}$ in the direction defined by the given angle, and a line $L_\gamma = \{(x,y) : x = \gamma\}$ such that $\mathcal{S} \cap L_\gamma \neq \emptyset$.

Output: $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$.

```
1: $\theta_0 \leftarrow 0$, $\theta_1 \leftarrow \pi$
2: $Q \leftarrow \nu_{\mathcal{S}}(0)$, $P \leftarrow \nu_{\mathcal{S}}(\pi)$
3: $y_{\min} \leftarrow -\infty$, $y_{\max} \leftarrow \infty$
4: while $y_{\max} - y_{\min} > 0$ do                      ▷ or $y_{\max} - y_{\min} > \epsilon$
5:     $\theta \leftarrow$ CHOOSE_$\theta(\theta_0, \theta_1, Q, P)$
6:     $(x_\theta, y_\theta) \leftarrow \nu_{\mathcal{S}}(\theta)$        ▷ one support vector in direction $v_\theta$
7:     if $x_\theta < \gamma$ then                           ▷ We should look right
8:         $P \leftarrow (x_\theta, y_\theta)$
9:         $\theta_1 \leftarrow \theta$
10:    else                                                  ▷ or left
11:        $Q \leftarrow (x_\theta, y_\theta)$
12:        $\theta_0 \leftarrow \theta$
13:    end if
14:    $(\gamma, y_{\min}) \leftarrow (PQ) \cap L_\gamma$
15:    $y_{\max} \leftarrow \min(h(\theta), y_{\max})$
16: end while
17: return $y_{\max}$
```

value of this minimum, we should stop when $x_\theta = \gamma$. Since this is not guaranteed to happen in finite time, we instead stop when the difference between the upper bound $y_{\max} = h(\theta)$ and the lower bound $y_{\min}$ is smaller than some ϵ.

There are several ways to choose an angle θ. Taking $(\theta_0 + \theta_1)/2$, the middle of the current range of angles, guarantees that at each step the width of $[\theta_0; \theta_1]$ is reduced by a factor 2. An other interesting choice is the angle defined by the normal to the line (PQ). In fact, this is similar to choosing s as a pivot in Algorithm 8.3, making it an interesting choice for the same reasons. Moreover, this choice of θ guarantees that the exact value $\inf_{\theta\in]0;\pi[} h(\theta)$ is eventually found if $\mathcal{S}$ is a polygon, or more generally if the intersection occurs on an edge delimited by two points of $\mathcal{S}$ that are corresponding to support vectors in several directions.

Figure 8.6 illustrates the execution of Algorithm 8.4. At first we choose θ equal to $\pi/2$, then we take the angle defined by the normal to the line (PQ), thus mimicking the behavior of Algorithm 8.3. The light grey area on the right corresponds to $[\theta_0; \theta_1] \times [y_{\min}; y_{\max}]$, the current search space for $(\theta_{\min}, h(\theta_{\min}))$, where $\theta_{\min}$ is the angle that minimizes h. After three iterations this area is smaller than $[0.87; 1.06] \times [2.20; 2.31]$, and we know that by bounding $\inf_{\theta\in]0;\pi[} h(\theta)$, in other words $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$, by 2.31 we are doing an over-approximation smaller than 0.11. In order to evaluate this error in terms of percentage of the size of $\mathcal{S} \cap L_\gamma$, we can apply the same method to the evaluation of $\rho_{\mathcal{S}\cap L_\gamma}((0,-1)^\top)$.

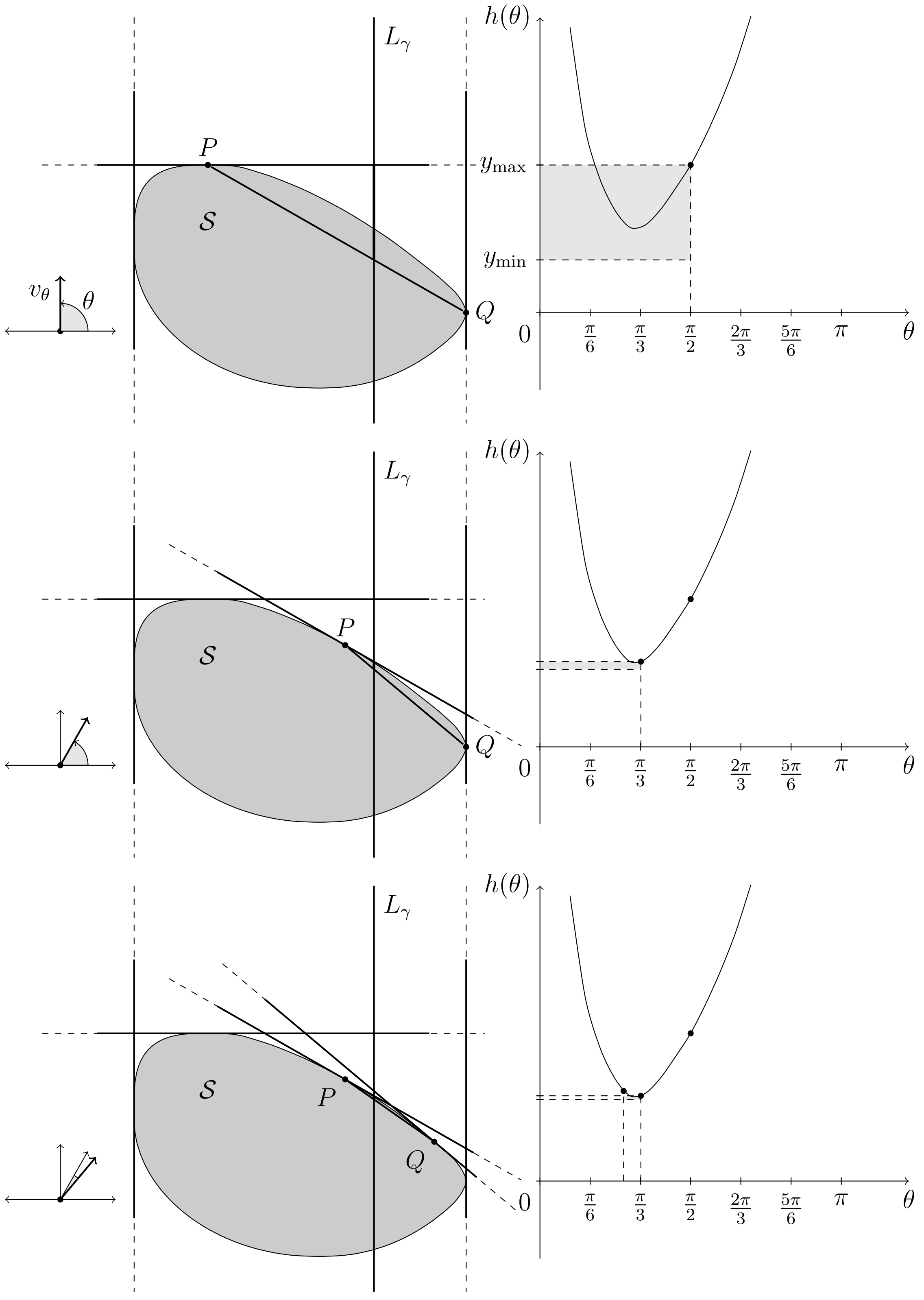


Figure 8.6: Algorithm 8.4: after 1, 2, and 3 iterations of the `while` loop.

8.3.1.2 Golden Section Search

Theorem 8.1 allows us to reduce the problem of evaluating $\rho_{S \cap L_\gamma}((0,1)^\top)$ to the minimization of a unimodal function h. In the previous section, the search for this minimum was guided by support vectors. But support vectors are not always easy to get. Using the method presented in Section 5.1.3, computing support vectors together with values of support functions costs twice as much as just computing the values of the considered support functions. Moreover, since reachability analysis of linear systems can be done with support functions only, it would be interesting to extend this work to hybrid systems without requiring support vectors.

Algorithm 8.5 Golden Section Search

Input: A unimodal function h.
Output: An upper-bound of $\inf_{\theta \in]0;\pi[} h(\theta)$.

```
 1: $\phi \leftarrow \frac{1+\sqrt{5}}{2}$                          ▷ The golden ratio.
 2: $\theta_1 \leftarrow 0$, $\theta_4 \leftarrow \pi$
 3: $\theta_3 \leftarrow \pi/\phi$
 4: $\theta_2 \leftarrow \theta_4 - \theta_3 + \theta_1$          ▷ Ensure that $\theta_4 - \theta_2 = \theta_3 - \theta_1$.
 5: $h_2 \leftarrow h(\theta_2)$, $h_3 \leftarrow h(\theta_3)$
 6: while $\theta_3 - \theta_2 > \epsilon$ do
 7:     if $h_2 < h_3$ then
 8:         $\theta_4 \leftarrow \theta_3$, $\theta_3 \leftarrow \theta_2$, $h_3 \leftarrow h_2$
 9:         $\theta_2 \leftarrow \theta_4 - \theta_3 + \theta_1$      ▷ Ensure that $\theta_4 - \theta_2 = \theta_3 - \theta_1$.
10:         $h_2 \leftarrow h(\theta_2)$
11:     else
12:         $\theta_1 \leftarrow \theta_2$, $\theta_2 \leftarrow \theta_3$, $h_2 \leftarrow h_3$
13:         $\theta_3 \leftarrow \theta_4 - \theta_2 + \theta_1$      ▷ Ensure that $\theta_4 - \theta_2 = \theta_3 - \theta_1$.
14:         $h_3 \leftarrow h(\theta_3)$
15:     end if
16: end while
17: return $\min(h_2, h_3)$
```

Since h is unimodal we can use a golden section search [Kie53] to find its minimum, and it will not involve computation of support vectors. The golden section search, implemented in Algorithm 8.5, works by successively narrowing the range of values inside which the minimum is known to exist. At each step, the value of h is known at four points $\theta_1 < \theta_2 < \theta_3 < \theta_4$, the minimum is known to lie between θ_1 and θ_4. If $h(\theta_2) < h(\theta_3)$ then, there is at least one point θ between θ_3 and θ_3 where h is increasing, and since h is unimodal, h is increasing between θ and θ_4: the minimum can not be reached in $[\theta_3; \theta_4]$. The new range of values is taken to be $[\theta_1; \theta_3]$. Conversely, if $h(\theta_2) > h(\theta_3)$, we continue with $\theta_2 < \theta_3 < \theta_4$.

In order to continue the algorithm we must choose a second point θ in $[\theta_1; \theta_3]$ and evaluate $h(\theta)$. This choice involves the golden ratio, $\phi = (\sqrt{5}+1)/2$, and guarantees that the width of the search interval is divided by a factor ϕ at each step. The algorithm stops when the width of the search interval becomes small enough.

Occasionally, we might have $h(\theta_2) = h(\theta_3)$. Then going back to the geometric interpretation of h, it means that the supporting lines indirection θ_2 and θ_3 intersect at the point $(\gamma, h(\theta_2))$. We can then deduce from Lemma 8.1 that the minimum of h is reached between θ_2 and θ_3.

This stopping criterion is not really satisfying, because we do not know if we are making a big over-approximation or not. In the previous section, the knowledge of some support vectors gives us an under-approximation of $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$. Here we do not have support vectors, but we still know they exists. Figure 8.7 illustrates how we can use this knowledge to deduce an under-approximation of $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$.

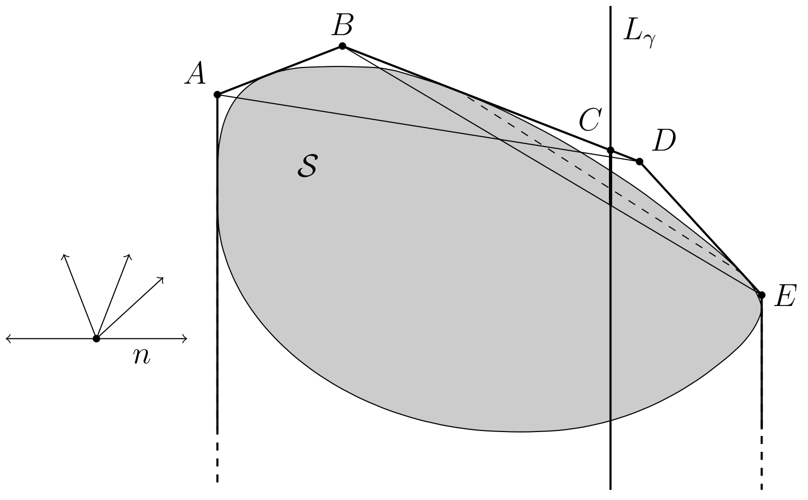


Figure 8.7: Deducing, from an over-approximation of $\mathcal{S}$, a lower bound for the maximum y-coordinate in the intersection between $\mathcal{S}$ and L_γ.

In this figure, the support function of $\mathcal{S}$ has been evaluated in 5 directions, leading to an unbounded polyhedral over-approximation with four vertices A, B, D, and E. We know that there is at least one point of $\mathcal{S}$ in each of the edges $[AB]$, $[BD]$, and $[DE]$, we will denote them respectively $P_{[AB]}$, $P_{[BD]}$, and $P_{[DE]}$. We also denote by C the intersection point between $[BD]$ and L_γ. If $P_{[BD]}$ is on the left of C, then the segment from $P_{[BD]}$ to $P_{[DE]}$ intersects L_γ. Since $\mathcal{S}$ is convex and $[P_{[BD]}P_{[DE]}]$ is *above* $[BE]$ then the intersection between $[BE]$ and L_γ defines an under-approximation of $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$. If on the contrary $P_{[BD]}$ is on the right of C, then, similarly, its the intersection between $[AD]$ and L_γ that defines an under-approximation of $\rho_{\mathcal{S}\cap L_\gamma}((0,1)^\top)$. Since we do not know where $P_{[BD]}$ is, we take the most conservative under-approximation.

We can now use this under-approximation to change the stopping criterion in Algorithm 8.5. Figure8.8 illustrates the execution of this algorithm. The light grey area on the right corresponds to $[\theta_1;\theta_4] \times [y_{\min}; y_{\max}]$, the current search space for $(\theta_{\min}, h(\theta_{\min}))$, where $\theta_{\min}$ is the angle that minimizes h. At first, the over-approximation of $\mathcal{S}$ induced by the computed supporting lines has only one unbounded edge on the right of L_γ. Thus, using the notations of Figure 8.7, we can not define the point E, and the point $P_{[DE]}$ can be anywhere on this unbounded half line. As a consequence we can not find a lower bound for the minimum of h. Fortunately, such a minimum can be defined, and improved, in the following iterations. This is not always the case, and we still have sometimes to rely on a bound on $\theta_3 - \theta_2$ if no lower bound for h can be found.

Remark. *If getting a support vector together with the value of the support function in one direction costs twice as much as just evaluating the support function. Then Algorithm 8.5 converges faster than Algorithm 8.4 with a mid point strategy. Indeed, in the former the width of the search interval is reduced by a factor ϕ^2 every two iterations, while in the latter it is reduced by a factor $2 < \phi^2$ every iteration.*

8.3.2 Intersection of the Reachable Set and a Line

We now have two algorithms for the approximation of the support function of the intersection between a set $\mathcal{S}$ and a hyperplane $\mathcal{G}$. The first one makes use of support vector and the second one proves useful when they are not readily available. Similarly to Section 8.2.3 we want to adapt these algorithms to the intersection between the reachable set and a hyperplane. We could compute the value of the support function of the intersection between each individual set and $\mathcal{G}$, but we want to take advantage of the redundancies of the system.

We have a sequence of sets $\Omega_{i_{\min}}$, …, $\Omega_{i_{\max}}$, each of them has a non-empty intersection with $\mathcal{G}$. This sequence is represented by a function f:

$$f : \ell \mapsto \left(\rho_{\Omega_{i_{\min}}}(\ell), \ldots, \rho_{\Omega_{i_{\max}}}(\ell)\right)$$

We want to over-approximate the support function of the set $\bigcup_{i=i_{\min}}^{i_{\max}} (\Omega_i \cap \mathcal{G})$, denoted as $\mathcal{Y}_{\mathcal{G}}^{\Omega}$:

$$\rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}(\ell) = \max_{i=i_{\min}}^{i_{\max}} \rho_{\Omega_i \cap \mathcal{G}}(\ell)$$

We will apply previous algorithms concurently.

Adaptation of Algorithm 8.4 For each index i in $[i_{\min}; i_{\max}]$ we have a range of angles $[\theta_0^i; \theta_1^i]$ and a range of values $[y_{\min}^i; y_{\max}^i]$. We define $y_{\min}$ and $y_{\max}$ to be the maximum of all $y_{\min}^i$ and $y_{\max}^i$ respectively. Then at each step:

- Determine the index j such that $y_{\max}^j = y_{\max}$.
- Choose an angle θ in $[\theta_0^j; \theta_1^j]$ according to Ω_j.
- Compute support vectors in the direction defined by θ.
- Update all the $[\theta_0^i; \theta_1^i]$ and $[y_{\min}^i; y_{\max}^i]$.

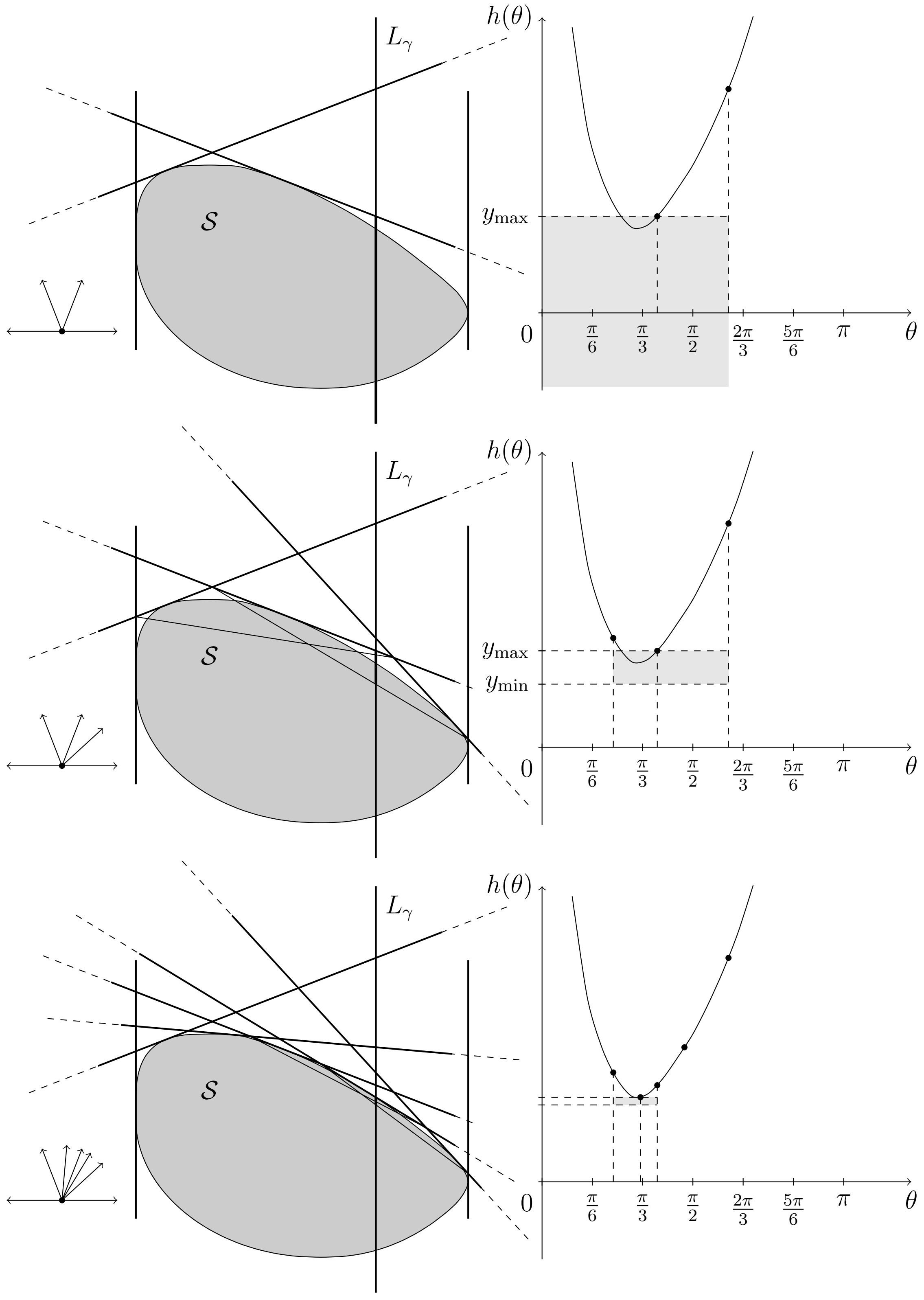


Figure 8.8: Algorithm 8.5: after the 1st, 2nd, and 4th execution of line 8 or 12.

- Drop the computation for all i such that $y^i_{\max}$ is smaller than $y_{\min}$. Indeed, there can be no support vector of $\mathcal{Y}_{\mathcal{G}}^{\Omega}$ in direction ℓ in the set $\Omega_i \cap \mathcal{G}$.

We stop when $y_{\max} - y_{\min}$ becomes smaller than some ϵ. As seen in Section 5.1.3 computing several support vectors can be very expensive compared to evaluating the support functions of the Ω_i. That is why the golden section search is preferable when evaluating $\rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}$.

Adaptation of Algorithm 8.5 For each index i, instead of having four angles θ_1, θ_2, θ_3, and θ_4, we only use three angles θ^i_1, θ^i, and θ^i_4. Similarly to the adaptation of Algorithm 8.4, we choose a new angle θ according to the Ω_i defining the biggest $y_{\max}$. Since this angle is not always chosen according to the same index i, the golden ratio might be lost. Thus, instead of taking θ equal to $\theta^i_4 - \theta^i + \theta^i_1$ we choose $\theta_1 + (\theta^i - \theta^i_1)/\phi$ if θ_i is bigger than the mean of θ_1 and θ_4, if not, we take $\theta_4 + (\theta^i - \theta^i_4)/\phi$.

8.3.3 Simplifying the Resulting Support Function

Contrary to the zonotope case, we do not need further transformations. Indeed, using Proposition 8.2 together with Algorithm 8.4 or 8.5, we can get a function $\overline{\rho}_{\mathcal{X}_{0,q'}}$, approximating the support function of the initial set in the next location q', such that for all ℓ we have:

$$\rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}(\ell) \le \overline{\rho}_{\mathcal{X}_{0,q'}}(\ell) \le \rho_{\mathcal{Y}_{\mathcal{G}}^{\Omega}}(\ell) + \epsilon$$

There are several problems with this function. First, it might not be convex – at the next guard intersection Algorithm 8.4 or 8.5 may fail if h is not unimodal. It is even worse if we want to take advantage of $\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}$, and use the following function to represent $\mathcal{X}_{0,q'}$:

$$\ell \mapsto \min\left(\overline{\rho}_{\mathcal{X}_{0,q'}}(\ell), \rho_{\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}}(\ell),\right)$$

Secondly, evaluating $\overline{\rho}_{\mathcal{X}_{0,q'}}$ in one direction ℓ requires several calls to Algorithm 5.1 in order to get the sequence of $\rho_{\Omega_i}(\ell')$ for some directions ℓ'. This, in turn, requires several calls to $\rho_{\mathcal{X}_{0,q}}$, the support function of the initial set in the previous location. If $\rho_{\mathcal{X}_{0,q}}$ has been defined using the same process, it will require even more calls to Algorithm 5.1. All this for evaluating $\overline{\rho}_{\mathcal{X}_{0,q'}}$ in only one direction. Thus, evaluating $\overline{\rho}_{\mathcal{X}_{0,q'}}$ can become prohibitively expensive after a few discrete locations have been visited.

In order to avoid this combinatorial explosion, and the non-convexity of $\overline{\rho}_{\mathcal{X}_{0,q'}}$, we will instead sample $\overline{\rho}_{\mathcal{X}_{0,q'}}$ in order to get a polyhedral over-approximation of $\mathcal{Y}_{\mathcal{G}}^{\Omega}$. The initial set in the next discrete location is taken to be:

$$\mathcal{X}_{0,q'} = \square_{\mathcal{L}_1}\left(\mathcal{Y}_{\mathcal{G}}^{\Omega}\right) \cap \square_{\mathcal{L}_2}\left(\mathcal{Y}_{\mathcal{G}}^{\overline{\aleph}}\right)$$

We can then represent this polytope by its support function.

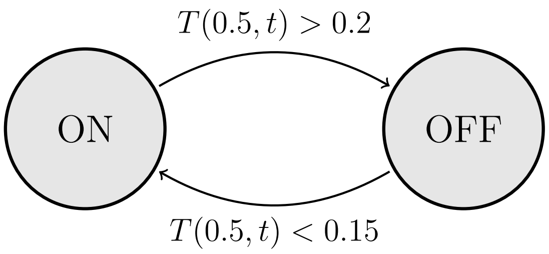


Figure 8.9: Hybrid model of a thermostat.

Remark. *In order to reduce the complexity of the evaluation of* $\rho_{\mathcal{X}_{0,q'}}$*, and thus the complexity of the overall reachability computation, it might be interesting to approximate* $\mathcal{X}_{0,q'}$ *by a box or a zonotope.*

In order to reduce the resulting over-approximation for the reachable set, $\mathcal{X}_{0,q'}$ *can be represented by several support functions of increasing precision and complexity of evaluation. If the interval hull of* $\mathcal{X}_{0,q'}$ *is sufficient to prove that some guard is not reached, there is no need to evaluate its support function more precisely.*

8.4 Applications

In this section we show the effectiveness of our approaches on some examples. The first one is treated using zonotopes, while in the second one we represent sets by their support function.

8.4.1 Thermostat

As a first example, we consider a high dimensional hybrid system with two discrete states. A heat source can be switched on or off at one end of a metal rod of length 1, the switching is driven by a sensor placed at the middle of the rod. The temperature at each point x of the rod, $T(x,t)$ is driven by the Heat equation:

$$\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2}.$$

When the heat source is *ON*, we have $T(0,t) = 1$, and when it is *OFF*, $T(0,t) = 0$. We approximate this partial differential equation by a linear ordinary differential equation using a finite difference scheme on a grid of size $\frac{1}{90}$.

The resulting hybrid system has 89 continuous variables and 2 discrete states (see Figure 8.9). We computed the reachable sets of this system for 1000 times step, during which 10 discrete transitions occured, in 71.6s using 406MB of memory. Figure 8.10 shows the reachable sets at three different time, each after a discrete transition.

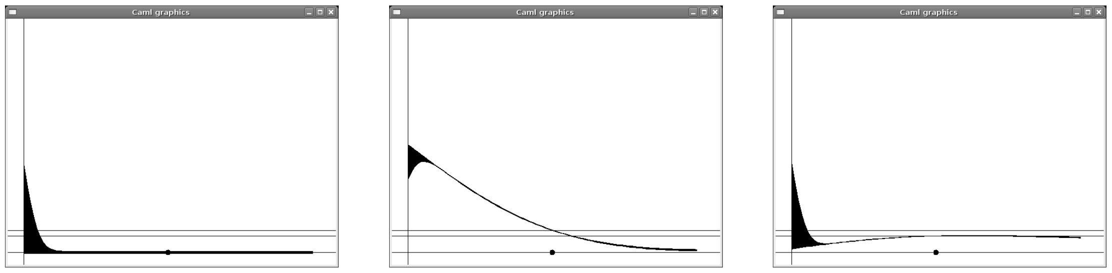


Figure 8.10: Reachable set at three different times. The x-axis represents the position in the metal rod, and the y-axis the temperature. The dot on the middle of the x-axis specify the position of the sensor, and the two horizontal line the switching temperatures. The heat source is on the left.

8.4.2 Navigation Benchmark

We now consider the navigation benchmark for hybrid systems verification proposed in [FI04]. It models an object moving in a plane, whose position and velocities are denoted $x(t)$ and $v(t)$. The plane is partitioned into cubes, each cube corresponds to one location of the hybrid system. At time t, a location q is active if $x(t)$ is in the associated cube; there, the object follows dynamically a desired velocity $v_q \in \mathbb{R}^2$. We use the instances NAV01 and NAV04 from [Fre08]. We render the problem slightly more challenging by including an additional continuous input $u(t)$ modelling disturbances. In the location q, the four-dimensional continuous dynamics is given by

$$\dot{x}(t) = v(t),\ \dot{v}(t) = A(v(t) - v_q - u(t)),\ \|u(t)\|_2 \leq 0.1 \text{ where } A = \left(\begin{smallmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{smallmatrix}\right).$$

In Figure 8.11, we represented the projection of the reachable sets on the position variables as well as the partition of the plane and, in each cube of the partition the set of velocities $v_q \oplus 0.1\mathcal{B}$ in the associated location.

One can check on Figure 8.11b that the intersection of the over-approximation $\overline{\aleph}_i$ of the reachable sets in one location with the guards does not coincide with the over-approximation $\overline{\mathcal{Y}_{\mathcal{G}}^{\aleph}}$ of the intersection of the reachable set in one location with the guards. The latter is more accurate because of the use of the exact Ω_i for the intersection.

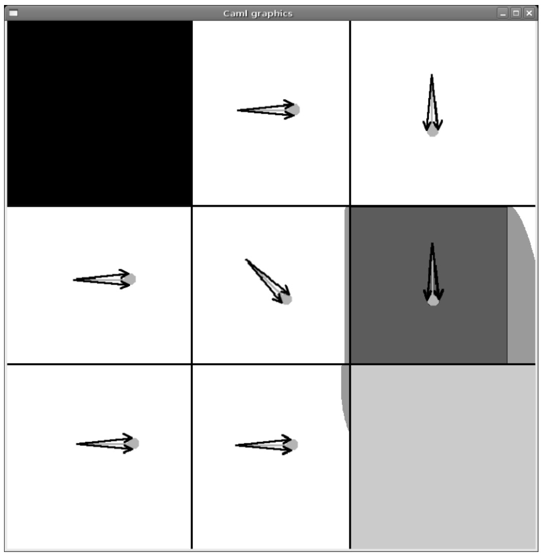
On Figures 8.11c and 8.11d the support function of this intersection is sampled in 6 directions defining its interval hull. We need to add 12 directions of approximation to get an octagon, used in Figures 8.11a and 8.11b. The benefit of using more directions of approximation can be clearly seen for NAV04. But the support function of an interval hull can be computed very efficiently using $\rho_{\mathcal{B}_\infty}$ as defined in Proposition 2.2. While the support function of an octagon is here computed with a LP solver. This explains the execution times in Table 8.1.

Table 8.1 also contains time and memory used by the optimized tool PHAVer on a similar computer as reported in [Fre08]. One should be careful when comparing this results. On one hand the navigation problem we consider here is more

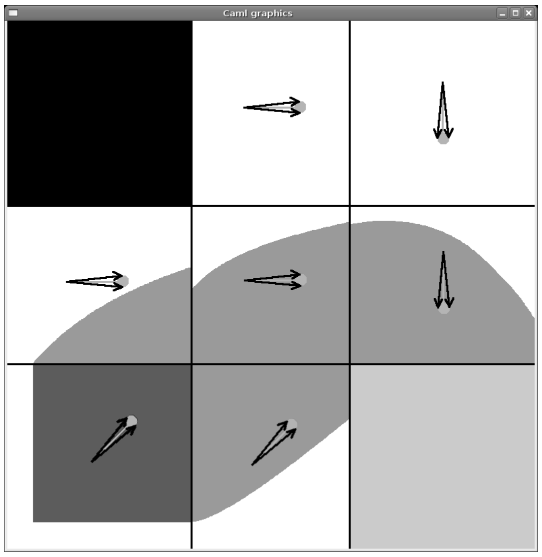
	NAV01		NAV04	
	time (s)	memory (MB)	time (s)	memory (MB)
octagon	10.28	0.24	54.77	0.47
cube	0.11	0.24	0.88	0.47
PHAVer	8.7	29.0	13.6	47.6

Table 8.1: Time and memory needed for reachability analysis of NAV01 and NAV04.

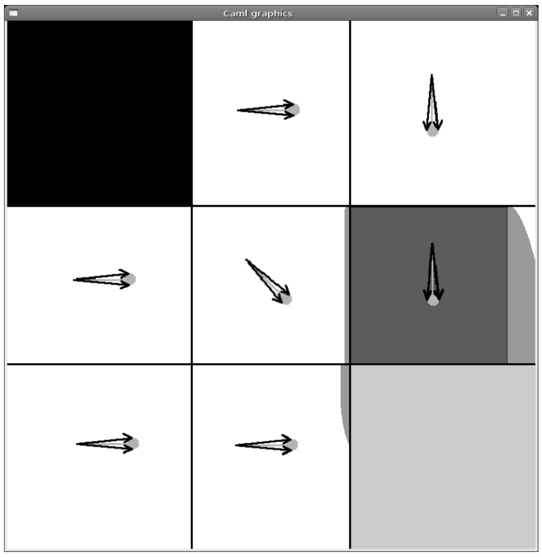
challenging than the original one since we add disturbances on the input. These disturbances add, in several locations, chattering effects that cannot occur without them, and produces bigger exact reachable sets. On the other hand PHAVer uses exact arithmetic.



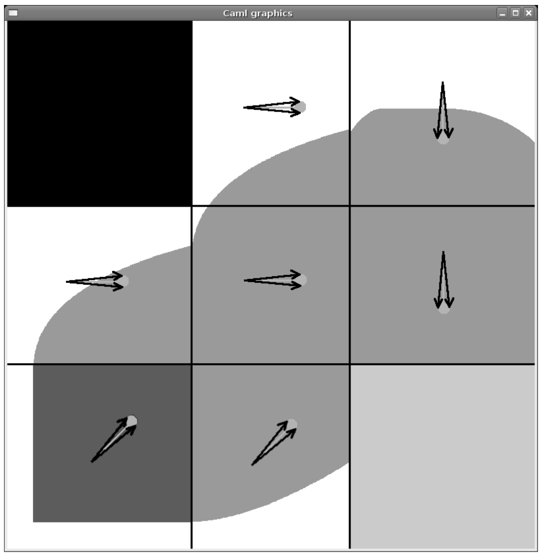
(a) NAV01 – octagon



(b) NAV04 – octagon



(c) NAV01 – cube



(d) NAV04 – cube

Figure 8.11: Two navigation benchmarks with perturbations. In a and b the intersections with the guards are over-approximated by octagons, whereas in c and d they are over-approximated by their interval hulls. Dark grey: initial set. Grey: reachable sets. Light grey: Target State. Black: Forbidden State.

Chapter 9

Conclusion and Future work

In this thesis we made several contributions to reachability analysis of hybrid automata with linear time-invariant flow, convex invariants, hyperplanar guards and affine reset maps.

The simplest and most powerful contributions of this thesis is the decomposition of the following set valued recurrence relation:

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V} \tag{9.1}$$

into three:

$$\begin{aligned} \mathcal{A}_0 &= \Omega_0 & \mathcal{A}_{i+1} &= \Phi\mathcal{A}_i \\ \mathcal{V}_0 &= \mathcal{V} & \mathcal{V}_{i+1} &= \Phi\mathcal{V}_i \\ \mathcal{S}_0 &= \{0\} & \mathcal{S}_{i+1} &= \mathcal{S}_i \oplus \mathcal{V}_i \end{aligned}$$

Ω_i can then be expressed as the Minkowski sum of $\mathcal{A}_i$ and $\mathcal{S}_i$.

Thanks to the disentanglement of linear transformation and Minkowski sum we were able to devise two efficient algorithms computing the N first sets of the sequence defined by equation (9.1), one using zonotopes, and one using support functions, as well as a whole family of approximation algorithms that are not subject to the wrapping effect. The main interest of these approximation algorithms is the ease of use of their output which consists of a list of sets in the chosen representation.

In the second part of this thesis, we extended this work to the case of hybrid automata. We showed how to use the invariant in order to improve the approximation of the reachable set in the continuous part. But the main contribution of Part II is the use of the exact sequence of Ω_i, expressed as high order zonotopes with a lot of common generators, or as a support function, to compute an approximation of the intersection between the reachable set in the continuous part, and hyperplanar guards. Not only does it improve substantially the approximation of this set, it also proves that the exact Ω_i are useful in spite of their cumbersome representation.

Most of the algorithms described in this thesis have been implemented in a prototype tool in OCaml. In the purely continuous case it outperforms all other tools in term of execution speed and quality of approximation. In the hybrid case it shows promising results and can compete with PHAVer, a highly optimized tool, on some non-trivial examples.

Naturally, a lot remains to be done and there are numerous opportunities for future work.

Periodic Linear Systems Our algorithms only work for linear time-invariant systems. A first extension would be to adapt them to periodic systems $\dot{x}(t) = A(t)x(t) + u(t)$ where either A, or the set of admissible input $\mathcal{U}$, or both, are periodic.

Time-Varying Linear Systems Our algorithms can not be modified to compute all the Ω_i for time-varying systems, but it might be possible to modify them in order to compute only one Ω_k for some k. Combining this with a standard algorithm for time-varying systems it could reduce the wrapping effect for some indexes k. Another way to reduce the wrapping effect is to adapt our discretization procedure to the time-varying case. By reducing the number of time steps required to achieve a certain precision it will indirectly reduce the wrapping effect.

Parameterized Linear Systems In this thesis we assumed that all matrices were known exactly, but it is not always the case and it would be particularly interesting to use our algorithms in the context of parameterized systems where the matrix A is only known to belong to a small subset of $M_d(\mathbb{R})$.

Ellipsoids As discussed in Section 3.4.2.2, tight ellipsoidal enclosures of the Ω_i tend to become flat, and releasing tightness may improve the approximation in terms of Hausdorff distance. It would be interesting to analyze this phenomenon and maybe derive a new approximation for the Minkowski sum of two ellipsoids that is *almost* tight in one direction without resulting into flat over-approximations.

Zonotopes This compact representation appears in numerous fields, and as already discussed in Section 3.4.2.4. To the best of the author knowledge, there are no satisfying order-reduction algorithms. Improving the existing one would already be interesting.

Hybrid Systems A first step towards improvement in term of speed of the analysis would be to investigate a set representation by support functions with increasing complexity and precision as suggested in Section 8.3.3. Moreover, among the aspects of hybrid systems that have not been addressed in this thesis, Zeno executions, chattering, and sliding modes are of particular interest; they are poorly handled by our algorithms.

Non-Linear Systems In order to analyze more complex systems one can approximate the continuous flow by a hybrid system with simpler continuous dynamics. This technique is called hybridization [ADG07]. Variants of this technique, using hysteretic transitions or dynamic hybridization, show promising results [DLGM09]. This kind of dynamic abstraction is particularly suited to multi-affine system [KB06, MB08], because here the value of the flow inside a box is equal to the convex hull of the values of the flow at the vertices of this box. It would be interesting to characterize and extend this work to a multivariate function whose value inside a box is equal to the interval hull of its values at the vertices of this box.

Tool Last but not least, some efforts should be made to bring our tool from the prototype stage to a more usable stage.

Appendix A

Proofs

A.1 Proofs of Chapter 2

A.1.1 Proof of Proposition 2.2

Proposition. *For any $k > 1$, the unit ball for the k-norm in dimension d is:*

$$\mathcal{B}_k^d = \left\{ x \in \mathbb{R}^d : \ \|x\|_k = \left(\sum_{i=0}^{d-1} |x_i|^k \right)^{1/k} \leq 1 \right\}$$

its support function is given by:

$$\rho_{\mathcal{B}_k^d}(\ell) = \|\ell\|_{\frac{k}{k-1}}$$

For $\ell \neq 0$, the associated support vector is given by:

$$\nu_{\mathcal{B}_k^d}(\ell) = \frac{v}{\|v\|_k}$$

where v is such that $v_i \ell_i = |\ell_i|^{\frac{k}{k-1}}$ for all i.

Proof. If we pose $p = k$ and $q = \frac{k}{k-1}$, then p and q are Hölder conjugates. Then using Hölder's inequality we know that for any x:

$$x \cdot \ell \leq \|x\|_p \|\ell\|_q \tag{A.1}$$

And thus for any x in $\mathcal{B}_k^d$ we have $x \cdot \ell \leq \|\ell\|_q$ and:

$$\rho_{\mathcal{B}_k^d}(\ell) \leq \|\ell\|_{\frac{k}{k-1}}$$

Let us now take v such that $v_i = 0$ if $\ell_i = 0$ or $\frac{|\ell_i|^q}{\ell_i}$ otherwise. If ℓ is different from 0, so is v, and $v/\|v\|_p$ belongs to $\mathcal{B}_k^d$.

$$\frac{v}{\|v\|_p} \cdot \ell = \frac{\sum_{i=0}^{d-1} |\ell_i|^q}{\left(\sum_{i=0}^{d-1} |\ell_i|^{p(q-1)} \right)^{1/p}}$$

Since p and q are Hölder conjugates, we have $p(q-1) = q$, and:

$$\begin{aligned}\frac{v}{\|v\|_p} \cdot \ell &= \left(\sum_{i=0}^{d-1} |\ell_i|^q\right)^{1-\frac{1}{p}} \\ &= \|\ell\|_q\end{aligned}$$

which ends the proof of the proposition. □

A.2 Proofs of Chapter 4

A.2.1 Proof of Proposition 4.4

Proposition. *The Hausdorff distance between $\mathcal{R}_{[i\delta;(i+1)\delta]}$ and its convex hull can be proportional to δ.*

Proof. Take:

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{ and } \mathcal{U} = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\}$$

It is easy to show that starting from $\left\{(x,0)^\top : -1 \le x \le 1\right\}$ the set of points reachable at time t is:

$$\mathcal{R}_t = \left\{ \begin{pmatrix} x\cos t \\ x \sin t \end{pmatrix} : -1 \le x \le 1 \right\}$$

Since CH $\left(\mathcal{R}_{[0;\delta]}\right)$ is convex and contains $\mathcal{X}_0$ and $\mathcal{R}_\delta$, it also contains:

$$z = \frac{1}{2}\begin{pmatrix} -1 \\ 0 \end{pmatrix} + (1 - \frac{1}{2})\begin{pmatrix} \cos\delta \\ \sin\delta \end{pmatrix}$$

as illustrated on Figure A.1.

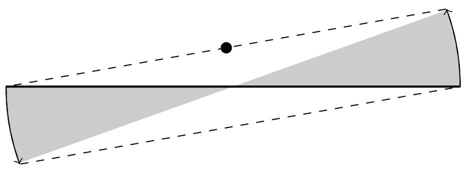


Figure A.1: $\mathcal{R}_{[0;\delta]}$ and its convex hull.

Thus the Hausdorff distance between $\mathcal{R}_{[0;\delta]}$ and its convex hull is at least:

$$\inf \left\{ \left\| \begin{pmatrix} x\cos t \\ x\sin t \end{pmatrix} - \frac{1}{2}\begin{pmatrix} \cos\delta - 1 \\ \sin\delta \end{pmatrix} \right\| : -1 \le x \le 1 \text{ and } 0 \le t \le \delta \right\}$$

And since

$$\begin{aligned} x\cos t - \tfrac{1}{2}(\cos\delta - 1) &= x + o(\delta) \\ x \sin t - \tfrac{1}{2}\sin\delta &= xt - \tfrac{\delta}{2} + o(\delta) \end{aligned}$$

we do not have: $d_H\left(\text{CH}\left(\mathcal{R}_{[0;\delta]}\right), \mathcal{R}_{[0;\delta]}\right) = o(\delta)$ □

A.2.2 Proof of Lemma 4.1

Lemma. *Let $\Omega \subseteq \mathbb{R}^d$, let Ω' be the set defined by :*

$$\Omega' = e^{\delta A}\Omega \oplus \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}} \tag{4.3}$$

where $\mathcal{E}_{\mathcal{U}} = \square\left(|A|^{-2}\left(e^{\delta|A|} - I - \delta|A|\right) \boxdot (A\mathcal{U})\right)$

Then, $\mathcal{R}_\delta(\Omega) \subseteq \Omega'$ and, if $\|\cdot\|$ is the infinity norm:

$$d_H\left(\Omega', \mathcal{R}_\delta\left(\Omega\right)\right) \leq 2R_{\mathcal{E}_{\mathcal{U}}} \leq 2\left(e^{\delta\|A\|} - 1 - \delta\|A\|\right)\frac{R_{\mathcal{U}}}{\|A\|} \tag{4.4}$$

Proof. Let us consider $x(.)$ a trajectory of the system, then there exists an initial state $x_0 \in \mathcal{X}_0$ and an input $u(.)$ such that for all t, $u(t) \in \mathcal{U}$ and

$$x(\delta) = e^{\delta A}x_0 + \int_0^\delta e^{(\delta-s)A}u(s)ds$$

Thus $\mathcal{R}_\delta\left(\mathcal{X}_0\right) = e^{\delta A}\mathcal{X}_0 \oplus \mathcal{V}$ with

$$\mathcal{V} = \left\{\int_0^\delta e^{(\delta-s)A}u(s)ds : \forall s \in [0;\delta]\, u(s) \in \mathcal{U}\right\}$$

We want to over-approximate this set by a *simpler* one. $e^{\delta A}\mathcal{X}_0$ is already simple enough, we will focus on $\mathcal{V}$.

In order to do so, we over-approximate the support function of $\mathcal{V}$ by the support function of another set. For any $v \in \mathcal{V}$ there exist $u(.)$ such that for any ℓ:

$$\begin{aligned}
v \cdot \ell &= \int_0^\delta e^{(\delta-s)A}u(s)ds \cdot \ell \\
&= \sum_{i=0}^{\infty}\int_0^\delta \frac{(\delta-s)^i}{i!}A^i u(s)\cdot \ell ds \\
&\leq \sum_{i=0}^{k-1}\int_0^\delta \frac{(\delta-s)^i}{i!}\sup_{u\in\mathcal{U}}\left(A^i u \cdot \ell\right) ds + \sum_{i=k}^{\infty}\int_0^\delta \frac{(\delta-s)^i}{i!}A^i u(s)\cdot \ell ds \\
v \cdot \ell &\leq \sum_{i=0}^{k-1}\frac{\delta^{i+1}}{(i+1)!}\rho_{A^i\mathcal{U}}(\ell) + \sum_{i=k}^{\infty}\left|\int_0^\delta \frac{(\delta-s)^i}{i!}A^k u(s)ds \cdot (A^{i-k})^\top \ell\right|
\end{aligned}$$

By taking $\rho_{A^i\mathcal{U}}(\ell)$ we lose the correlation between all the u, but it allows us to apply a coarse over-approximation on a smaller set. For sake of simplicity we will limit ourselves to $k = 1$.

$$v \cdot \ell \leq \delta\rho_{\mathcal{U}}(\ell) + \sum_{i=1}^{\infty}\left|\int_0^\delta \frac{(\delta-s)^i}{i!}Au(s)ds \cdot (A^{i-1})^\top \ell\right|$$

Before going further, let us remark a few properties of $|.|$. For any two vectors x and y, it is easy to show that: $|x \cdot y| \leq |x| \cdot |y|$. It is also clear that for any two matrices A and B, $|ABx| \leq |A||B||x|$ component wise.

We now focus on: $\left|\int_0^\delta \frac{(\delta-s)^i}{i!} Au(s)ds \cdot (A^{i-1})^\top \ell\right|$

$$\left|\int_0^\delta \frac{(\delta-s)^i}{i!} Au(s)ds \cdot (A^{i-1})^\top \ell\right| \leq \int_0^\delta \frac{(\delta-s)^i}{i!} |Au(s)|\, ds \cdot |A^\top|^{i-1}|\ell|$$

We now take $u_1^{\max} \in \boxdot(A\mathcal{U})$ such that for all $u \in \mathcal{U}$: $|Au| \leq u_1^{\max}$ component wise.

$$\begin{aligned}\left|\int_0^\delta \frac{(\delta-s)^i}{i!} Au(s)ds \cdot (A^{i-1})^\top \ell\right| &\leq \int_0^\delta \frac{(\delta-s)^i}{i!} u_1^{\max} ds \cdot |A^\top|^{i-1}|\ell| \\ &\leq \frac{\delta^{i+1}}{(i+1)!} u_1^{\max} \cdot |A^\top|^{i-1}|\ell| \\ &\leq \frac{\delta^{i+1}}{(i+1)!} |A|^{i-1} u_1^{\max} \cdot |\ell|\end{aligned}$$

Going back to $v \cdot \ell$ we get:

$$v \cdot \ell \leq \delta\rho_{\mathcal{U}}(\ell) + \sum_{i=1}^{\infty} \frac{\delta^{i+1}}{(i+1)!} |A|^{i-1} u_1^{\max} \cdot |\ell|$$

If we call e_1 the point:

$$e_1 = \sum_{i=1}^{\infty} \frac{\delta^{i+1}}{(i+1)!} |A|^{i-1} u_1^{\max}$$

we have:

$$\rho_{\mathcal{V}}(\ell) \leq \delta\rho_{\mathcal{U}}(\ell) + |e_1| \cdot |\ell|$$

Let $\mathcal{E}_{\mathcal{U}} = \boxdot(\{e_1\})$, it is easy to show that $\rho_{\mathcal{E}_{\mathcal{U}}}(\ell) = |e_1| \cdot |\ell|$, and that $\mathcal{E}_{\mathcal{U}}$ is also equal to $\square\left(\sum_{i=1}^{\infty} \frac{\delta^{i+1}}{(i+1)!} |A|^{i-1} \boxdot(A\mathcal{U})\right)$

For all ℓ, we have:

$$\rho_{\mathcal{V}}(\ell) \leq \delta\rho_{\mathcal{U}}(\ell) + \rho_{\mathcal{E}_{\mathcal{U}}}(\ell)$$

Thus,

$$\mathcal{V} \subset \delta\mathcal{U} \oplus \mathcal{E}_{\mathcal{U}}$$

which proves the first part of the lemma.

We now have to prove that

$$d_H(\Omega', \mathcal{R}_\delta(\Omega)) \leq 2\left(e^{\delta\|A\|} - 1 - \delta\|A\|\right) \frac{R_{\mathcal{U}}}{\|A\|}$$

Since $\mathcal{R}_\delta(\Omega) \subset \Omega'$,

$$d_H(\Omega', \mathcal{R}_\delta(\Omega)) = \sup_{x \in \Omega'} \inf_{y \in \mathcal{R}_\delta(\Omega)} \|x - y\|$$

Let $x \in \Omega'$ there exists $x_0 \in \Omega$, $u_x \in \mathcal{U}$ and $e \in \mathcal{E}_{\mathcal{U}}$, such that $x = e^{\delta A}x_0 + \delta u_x + e$. Starting from x_0 and with $u(s) = u_x$ we get $y \in \mathcal{R}_\delta(\Omega)$, and:

$$\begin{aligned}\|x-y\| = \left\|\delta u_x + e - \int_0^\delta e^{(\delta-s)A}u_x ds\right\| &= \left\|e - \sum_{i=1}^{\infty}\frac{\delta^{i+1}}{(i+1)!}A^i u_x\right\| \\ &\le \|e\| + \left\|\sum_{i=1}^{\infty}\frac{\delta^{i+1}}{(i+1)!}A^i u_x\right\| \\ &\le 2R_{\mathcal{E}_{\mathcal{U}}} \le 2\frac{e^{\delta\|A\|}-1-\delta\|A\|}{\|A\|}R_{\mathcal{U}}\end{aligned}$$

which ends the proof of lemma 4.1. □

A.3 Proofs of Chapter 5

A.3.1 Proof of Lemma 5.1

Lemma. *Let $\lambda \in [0;1]$, and $\Omega_{0,\lambda}$ be the convex set defined by :*

$$\begin{aligned}\Omega_{0,\lambda} = (1-\lambda)\mathcal{X}_0 \oplus \lambda e^{\delta A}\mathcal{X}_0 \oplus \lambda(1-\lambda)\mathcal{E}_{\mathcal{X}_0} \\ \oplus \lambda\delta\mathcal{U} \quad \oplus \lambda^2\mathcal{E}_{\mathcal{U}}\end{aligned} \tag{5.3}$$

where

$$\begin{aligned}\mathcal{E}_{\mathcal{X}_0} =& \square\left(|A|^{-1}\left(e^{\delta|A|}-I\right)\boxdot\left(A(I-e^{\delta A})\mathcal{X}_0\right)\right) \\ &\oplus \square\left(|A|^{-2}\left(e^{\delta|A|}-I-\delta|A|\right)\boxdot\left(A^2e^{\delta A}\mathcal{X}_0\right)\right) \\ \textit{and } \mathcal{E}_{\mathcal{U}} =& \square\left(|A|^{-2}\left(e^{\delta|A|}-I-\delta|A|\right)\boxdot\left(A\mathcal{U}\right)\right)\end{aligned}$$

If we define Ω_0 as:

$$\Omega_0 = CH\left(\bigcup_{\lambda\in[0;1]}\Omega_{0,\lambda}\right) \tag{5.4}$$

then, $\mathcal{R}_{[0;\delta]}(\mathcal{X}_0) \subseteq \Omega_0$ and

$$\begin{aligned}d_H\left(\Omega_0, \mathcal{R}_{[0;\delta]}(\mathcal{X}_0)\right) &\le \tfrac{1}{4}\left(e^{\delta\|A\|}-1\right)D_{\mathcal{X}_0} + \tfrac{1}{2}R_{\mathcal{E}_{\mathcal{X}_0}} + 2R_{\mathcal{E}_{\mathcal{U}}} \\ &\le \tfrac{1}{4}\left(e^{\delta\|A\|}-1\right)D_{\mathcal{X}_0} + 2\left(e^{\delta\|A\|}-1\right)^2\left(\tfrac{R_{\mathcal{X}_0}}{2} + \tfrac{R_{\mathcal{U}}}{\|A\|}\right)\end{aligned} \tag{5.5}$$

Proof. We already now that for any t:

$$\mathcal{R}_t(\mathcal{X}_0) = e^{tA}\mathcal{X}_0 \oplus \mathcal{V}_t$$

with $\mathcal{V}_t = \{\int_0^t e^{(t-s)A}u(s)ds : \forall s \in [0;t]\, u(s) \in \mathcal{U}\}$.

We want to over-approximate $\mathcal{R}_t(\mathcal{X}_0)$ by a *simpler* set. A set that can be expressed as a sum of sets constructed from $\mathcal{X}_0$, $e^{\delta A}\mathcal{X}_0$ and $\mathcal{U}$.

Let us first focus on $e^{tA}\mathcal{X}_0$. To prove the lemma, we need to introduce the following sets for $\lambda \in [0;1]$:

$$\mathcal{P}_\lambda = \left\{(1-\lambda)x + \lambda e^{\delta A}x : \ x \in \mathcal{X}_0\right\} \text{ and } \mathcal{Q}_\lambda = (1-\lambda)\mathcal{X}_0 \oplus \lambda e^{\delta A}\mathcal{X}_0$$

Let us remark that $\mathcal{P}_\lambda \subseteq \mathcal{Q}_\lambda$, which implies that

$$d_H\left(\mathcal{P}_\lambda, \mathcal{Q}_\lambda\right) = \sup_{z \in \mathcal{Q}_\lambda} \inf_{z' \in \mathcal{P}_\lambda} \|z - z'\|.$$

Let $z \in \mathcal{Q}_\lambda$, then there exists x and y in $\mathcal{X}_0$ such that

$$z = (1-\lambda)x + \lambda e^{\delta A}y.$$

Since $\mathcal{X}_0$ is convex, $(1-\lambda)x + \lambda y \in \mathcal{X}_0$. Then, let us consider the point $z' \in \mathcal{P}_\lambda$ defined by

$$z' = (1-\lambda)((1-\lambda)x + \lambda y) + \lambda e^{\delta A}((1-\lambda)x + \lambda y).$$

Then, we have

$$z - z' = \lambda(1-\lambda)(e^{\delta A} - I)(y - x).$$

We showed that for all $z \in \mathcal{Q}_\lambda$, there exists $z' \in \mathcal{P}_\lambda$ such that

$$\|z - z'\| \leq \frac{1}{4}(e^{\delta\|A\|} - 1)D_{\mathcal{X}_0}.$$

It follows that

$$d_H\left(\mathcal{P}_\lambda, \mathcal{Q}_\lambda\right) \leq \frac{1}{4}(e^{\delta\|A\|} - 1)D_{\mathcal{X}_0} \tag{A.2}$$

Let $x_0 \in \mathcal{X}_0$ it is clear that $e^{tA}x_0$ is equal to $e^{(t-\delta)A}e^{\delta A}\mathcal{X}_0$. Taking a convex combination of these two expressions we get:

$$\begin{aligned}
e^{tA}x_0 &= \left(1 - \frac{t}{\delta}\right) e^{tA}x_0 + \frac{t}{\delta}e^{(t-\delta)A}(e^{\delta A}x_0) \\
&= \sum_{i=0}^{\infty}\left(1 - \frac{t}{\delta}\right)\frac{t^i}{i!}A^i x_0 + \sum_{i=0}^{\infty}\frac{t}{\delta}\frac{(t-\delta)^i}{i!}A^i e^{\delta A}x_0 \\
&= \left(1 - \frac{t}{\delta}\right)x_0 + \frac{t}{\delta}e^{\delta A}x_0 \\
&\quad + \frac{t}{\delta}\left(1 - \frac{t}{\delta}\right)\left(\sum_{i=1}^{\infty}\frac{\delta t^{i-1}}{i!}A^i x_0 - \sum_{i=1}^{\infty}\frac{\delta(t-\delta)^{i-1}}{i!}A^i e^{\delta A}x_0\right) \\
&= \left(1 - \frac{t}{\delta}\right)x_0 + \frac{t}{\delta}e^{\delta A}x_0 \\
&\quad + \frac{t}{\delta}\left(1 - \frac{t}{\delta}\right)\sum_{i=1}^{\infty}\left(\frac{t}{\delta}\right)^{i-1}\frac{\delta^i}{i!}A^i(I - e^{\delta A})x_0 \\
&\quad + \frac{t}{\delta}\left(1 - \frac{t}{\delta}\right)\sum_{i=2}^{\infty}\left(\left(\frac{t}{\delta}\right)^{i-1} - \left(\frac{t}{\delta} - 1\right)^{i-1}\right)\frac{\delta^i}{i!}A^i e^{\delta A}x_0
\end{aligned}$$

Similarly to the proof of lemma 4.1, and using the fact that for any $x \in [0;1]$ and any $k > 0$:

$$-1 \leq x^k - (x-1)^k \leq 1$$

we have:

$$e^{tA}x_0 - \left(\left(1 - \frac{t}{\delta}\right)x_0 + \frac{t}{\delta}e^{\delta A}x_0\right) \in \frac{t}{\delta}\left(1 - \frac{t}{\delta}\right)\mathcal{E}_{\mathcal{X}_0}$$

Thus

$$e^{tA}\mathcal{X}_0 \subset \mathcal{P}_{\frac{t}{\delta}} \oplus \frac{t}{\delta}\left(1 - \frac{t}{\delta}\right)\mathcal{E}_{\mathcal{X}_0} \qquad \text{(A.3)}$$

and

$$d_H\left(e^{tA}\mathcal{X}_0, \mathcal{P}_{\frac{t}{\delta}}\right) \leq \frac{R_{\mathcal{E}_{\mathcal{X}_0}}}{4} \qquad \text{(A.4)}$$

Applying the same method to $\mathcal{V}_t$ we get:

$$\mathcal{V}_t \subset \frac{t}{\delta}\delta\mathcal{U} \oplus \frac{t^2}{\delta^2}\mathcal{E}_{\mathcal{U}} \qquad \text{(A.5)}$$

and

$$d_H\left(\mathcal{V}_t, \frac{t}{\delta}\delta\mathcal{U}\right) \leq R_{\mathcal{E}_{\mathcal{U}}} \qquad \text{(A.6)}$$

Using equations (A.3) and (A.5) we can deduce that:

$$\mathcal{R}_t(\mathcal{X}_0) \subset \Omega_{0,\frac{t}{\delta}}$$

and

$$\mathcal{R}_t([0;\delta]) = \bigcup_{t\in[0;\delta]} \mathcal{R}_t(\mathcal{X}_0) \subset \bigcup_{t\in[0;\delta]} \Omega_{0,\frac{t}{\delta}} \subset \Omega_0$$

For the second part of the lemma, we use equations (A.2), (A.4) and (A.6), and the fact that for any $x \geq 0$:

$$e^x - 1 - x \leq e^x(e^x - 1 - x) \leq (e^x - 1)^2$$

□

A.3.2 Proof of Theorem 5.1

Theorem. *Let us consider the sequence of sets Ω_i defined by equations* (5.4) *and* (4.3)*. Then, for all $i \in \mathbb{N}$, we have $\mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{X}_0) \subseteq \Omega_i$ and*

$$d_H\left(\Omega_i, \mathcal{R}_{[i\delta;(i+1)\delta]}(\mathcal{X}_0)\right) \leq \delta e^{T\|A\|}\left(\frac{\|A\|}{4}D_{\mathcal{X}_0} + \frac{3}{4}\delta\|A\|^2 e^{\delta\|A\|}R_{\mathcal{X}_0} + e^{\delta\|A\|}R_{\mathcal{U}}\right)$$

Proof. From lemma 5.1, $\mathcal{R}_{[0,\delta]}(\mathcal{X}_0) \subseteq \Omega_0$. Assume $\mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0) \subseteq \Omega_i$, then

$$\mathcal{R}_{[(i+1)\delta,(i+2)\delta]}(\mathcal{X}_0) = \mathcal{R}_\delta\left(\mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right) \subseteq \mathcal{R}_\delta\left(\Omega_i\right) \subseteq e^{\delta A}\Omega_i \oplus \delta\mathcal{U} \oplus \mathcal{E}_\mathcal{U}.$$

Therefore, $\mathcal{R}_{[(i+1)\delta,(i+2)\delta]}(\mathcal{X}_0) \subseteq \Omega_{i+1}$ and the first part of the theorem holds. Let us note $\Delta_i = d_H\left(\Omega_i, \mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right)$, then

$$\begin{aligned}\Delta_{i+1} &= d_H\left(\Omega_{i+1}, \mathcal{R}_\delta\left(\mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right)\right)\\ &\leq d_H\left(\Omega_{i+1}, \mathcal{R}_\delta\left(\Omega_i\right)\right) + d_H\left(\mathcal{R}_\delta\left(\Omega_i\right), \mathcal{R}_\delta\left(\mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right)\right)\end{aligned}$$

From lemma 4.1, $d_H\left(\Omega_{i+1}, \mathcal{R}_\delta\left(\Omega_i\right)\right) \leq 2R_{\mathcal{E}_\mathcal{U}}$ and it is easy to show that

$$d_H\left(\mathcal{R}_\delta\left(\Omega_i\right), \mathcal{R}_\delta\left(\mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right)\right) \leq e^{\delta\|A\|} d_H\left(\Omega_i, \mathcal{R}_{[i\delta,(i+1)\delta]}(\mathcal{X}_0)\right).$$

Thus, we have that $\Delta_{i+1} \leq e^{\delta\|A\|}\Delta_i + 2R_{\mathcal{E}_\mathcal{U}}$. Therefore, for all $i = 0, \ldots, N-1$

$$\Delta_i \leq e^{i\delta\|A\|}\Delta_0 + 2R_{\mathcal{E}_\mathcal{U}} \sum_{k=0}^{i-1} e^{k\delta\|A\|}.$$

Then, from lemma 5.1,

$$\begin{aligned}\Delta_i &\leq e^{i\delta\|A\|}\left(\frac{e^{\delta\|A\|}-1}{4}D_{\mathcal{X}_0} + 2R_{\mathcal{E}_\mathcal{U}} + \frac{1}{2}R_{\mathcal{E}_{\mathcal{X}_0}}\right) + 2R_{\mathcal{E}_\mathcal{U}}\sum_{k=0}^{i-1} e^{kr\|A\|}\\ &\leq \delta e^{(i+1)\delta\|A\|}\frac{\|A\|}{4}D_{\mathcal{X}_0} + 2(e^{\delta\|A\|} - 1 - \delta\|A\|)\frac{R_\mathcal{U}}{\|A\|}\sum_{k=0}^{i} e^{k\delta\|A\|}\\ &\quad + \frac{e^{i\delta\|A\|}}{2}\left(e^{\delta\|A\|}(e^{\delta\|A\|} - 1 - \delta\|A\|) + (e^{\delta\|A\|}-1)^2\right)R_{\mathcal{X}_0}\\ &\leq \delta e^{(i+1)\delta\|A\|}\left(\frac{\|A\|}{4}D_{\mathcal{X}_0} + \frac{3}{4}\delta\|A\|^2 e^{\delta\|A\|}R_{\mathcal{X}_0}\right) + \delta^2\|A\|^2 e^{\delta\|A\|}\frac{R_\mathcal{U}}{\|A\|}\frac{e^{(i+1)\delta\|A\|}-1}{e^{\delta\|A\|}-1}\\ &\leq \delta e^{(i+1)\delta\|A\|}\left(\frac{\|A\|}{4}D_{\mathcal{X}_0} + \frac{3}{4}\delta\|A\|^2 e^{\delta\|A\|}R_{\mathcal{X}_0}\right) + \delta^2\|A\|^2 e^{\delta\|A\|}\frac{R_\mathcal{U}}{\|A\|}\frac{e^{(i+1)\delta\|A\|}}{\delta\|A\|}\\ &\leq \delta e^{(i+1)\delta\|A\|}\left(\frac{\|A\|}{4}D_{\mathcal{X}_0} + \frac{3}{4}\delta\|A\|^2 e^{\delta\|A\|}R_{\mathcal{X}_0} + e^{\delta\|A\|}R_\mathcal{U}\right).\end{aligned}$$

This leads to the estimate of the theorem since $(i+1)\delta \leq T$. □

A.3.3 Proof of proposition 5.4

Proposition. *For all $t \in \mathbb{R}^+$,*

$$\rho_{\mathcal{R}_t}(\ell) = \rho_{\mathcal{X}_0}\left(e^{tA^\top}\ell\right) + \int_0^t \rho_\mathcal{U}\left(e^{\tau A^\top}\ell\right)d\tau. \qquad (5.6)$$

Proof. One trajectory of system (3.2) is given by:

$$x(t) = e^{tA}x_0 + \int_0^t e^{(t-s)A}u(s)\ ds$$

Then for all ℓ,

$$\begin{aligned}\ell \cdot x(t) &= \ell \cdot e^{tA}x(0) + \ell \cdot \int_0^t e^{(t-\tau)A}u(\tau)d\tau \\ &= \ell \cdot e^{tA}x(0) + \ell \cdot \int_0^t e^{\tau A}u(t-\tau)d\tau \\ &= x(0) \cdot e^{tA^\top}\ell + \int_0^t u(t-\tau) \cdot e^{\tau A^\top}\ell d\tau.\end{aligned}$$

Then $\rho_{\mathcal{R}_t}(\ell)$ is obtained by maximizing $\ell \cdot x(t)$ over the initial condition $x(0) \in \mathcal{X}_0$ and input function $u : [0,t] \to \mathcal{U}$. Then,

$$\begin{aligned}\rho_{\mathcal{R}_t}(\ell) &= \max_{x(0)} x(0) \cdot e^{tA^\top}\ell + \max_u \int_0^t u(t-\tau) \cdot e^{\tau A^\top}\ell d\tau \\ &= \rho_{\mathcal{X}_0}\left(e^{tA^\top}\ell\right) + \max_u \int_0^t u(t-\tau) \cdot e^{\tau A^\top}\ell d\tau\end{aligned}$$

For all $u : [0,t] \to \mathcal{U}$, for all $t \in \mathbb{R}^+$ and $\tau \in [0,t]$, we have that

$$u(t-\tau) \cdot e^{\tau A^\top}\ell \le \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right)$$

Therefore,

$$\max_u \int_0^t u(t-\tau) \cdot e^{\tau A^\top}\ell d\tau \le \int_0^t \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right) d\tau$$

Let us consider the input function $u^* : [0,t] \to \mathcal{U}$ that associate to $\tau \in [0,t]$, a support vector of $\mathcal{U}$ in the direction $e^{(t-\tau)A^\top}\ell$. Then, for all τ in $[0,t]$:

$$u^*(t-\tau) \cdot e^{\tau A^\top}\ell = \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right)$$

It follows that

$$\max_u \int_0^t u(t-\tau) \cdot e^{\tau A^\top}\ell d\tau \ge \int_0^t \rho_{\mathcal{U}}\left(e^{\tau A^\top}\ell\right) d\tau$$

and equation (5.6) holds. □

A.4 Proofs of chapter 7

A.4.1 Proof of corollary 7.1

Corollary. *Let $\aleph_i$ be the i^{th} term of the sequence defined by equation (7.2), then:*

$$\aleph_i \subseteq \Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$$

where Ω_i and the $\mathcal{I}_j$ are defined by:

$$\begin{aligned} \Omega_{k+1} &= \Phi\Omega_k \oplus \mathcal{V} & \qquad \Omega_0 &= \aleph_0 \\ \mathcal{I}_{k+1} &= \Phi\mathcal{I}_k \oplus \mathcal{V} & \qquad \mathcal{I}_0 &= \mathcal{I} \end{aligned}$$

Proof. By induction on i. By definition, $\aleph_0$ is equal to Ω_0. Let us now suppose that $\aleph_i$ is a subset of $\Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j$. Then:

$$\Phi\aleph_i \oplus \mathcal{V} \subseteq \Phi\left(\Omega_i \cap \bigcap_{j=0}^{i-1} \mathcal{I}_j\right) \oplus \mathcal{V}$$

Then, from proposition 7.1:

$$\Phi\aleph_i \oplus \mathcal{V} \subseteq (\Phi\Omega_i \oplus \mathcal{V}) \cap \bigcap_{j=0}^{i-1} (\Phi\mathcal{I}_j \oplus \mathcal{V})$$

$$(\Phi\aleph_i \oplus \mathcal{V}) \cap \mathcal{I} \subseteq \left(\Omega_{i+1} \cap \bigcap_{j=1}^{i} \mathcal{I}_j\right) \cap \mathcal{I}$$

$$\aleph_{i+1} \subseteq \Omega_{i+1} \cap \bigcap_{j=0}^{i} \mathcal{I}_j$$

□

A.5 Proofs of chapter 8

A.5.1 Proof of lemma 8.1

Lemma A.1. *Let $\mathcal{S}$ be a 2-dimensional set and θ_0, θ_1 be in $]0;\pi[$ such that $\theta_0 < \theta_1$. Let (x_0, y_0) and (x_1, y_1) be support vectors of $\mathcal{S}$ associated to directions v_{θ_0} and v_{θ_1} respectively, we denote by (x, y) the intersection between the two supporting lines. Then:*

$$x_0 \geq x \geq x_1$$

Proof. By definition $\rho_{\mathcal{S}}(v_{\theta_1})$ is greater than $x_0 \cos\theta_1 + y_0 \sin\theta_1$, thus:

$$\begin{aligned} x_1 \cos\theta_1 + y_1 \sin\theta_1 &\geq x_0 \cos\theta_1 + y_0 \sin\theta_1 \\ (x_1 - x_0)\cos\theta_1 &\geq (y_0 - y_1)\sin\theta_1 \end{aligned}$$

Since θ_1 is in $]0;\pi[$, we have $(x_1 - x_0)\cot\theta_1 \geq (y_0 - y_1)$. Symmetrically, we have $(x_0 - x_1)\cot\theta_0 \geq (y_1 - y_0)$ and thus:

$$(x_1 - x_0)\cot\theta_1 \geq (x_1 - x_0)\cot\theta_0$$

Since cotangent is decreasing on $]0;\pi[$ and $\theta_0 < \theta_1$, this implies that $x_0 \geq x_1$.

If we now consider the set delimited by the two supporting lines associated to θ_0 and θ_1, then (x, y) is the support vector associated to angle $(\theta_0 + \theta_1)/2$ and we can apply what we just proved to finish the proof of the lemma. □

A.5.2 Proof of theorem 8.1

Theorem. *For any compact convex set $\mathcal{S}$ and any real γ, the function h defined by:*

$$\begin{array}{rccc} h: &]0;\pi[& \to & \mathbb{R} \\ & \theta & \mapsto & \dfrac{\rho_{\mathcal{S}}\left(\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}\right) - \gamma\cos\theta}{\sin\theta} \end{array}$$

is monotonous or unimodal and:

$$\inf_{\theta\in]0;\pi[} h(\theta) = \rho_{\mathcal{S}\cap L_\gamma}\left(\begin{pmatrix}0 \\ 1\end{pmatrix}\right)$$

Proof. We will consider three different cases:

- $-\rho_{\mathcal{S}}((-1,0)) < \gamma < \rho_{\mathcal{S}}((1,0))$
- $-\rho_{\mathcal{S}}((-1,0)) = \gamma$ or $\gamma = \rho_{\mathcal{S}}((1,0))$
- $-\rho_{\mathcal{S}}((-1,0)) > \gamma$ or $\gamma > \rho_{\mathcal{S}}((1,0))$

In the first case, $\mathcal{S} \cap L_\gamma$ is not empty and, being unidimensional, has a unique support vector P_0 in direction $(0,1)^\top$. P_0 is on the boundary of $\mathcal{S}\cap L_\gamma$, thus it is also on the boundary of $\mathcal{S}$. Since this set is compact and convex, and $-\rho_{\mathcal{S}}((-1,0)) < \gamma < \rho_{\mathcal{S}}((1\ 0))$, there is an angle θ_0, not necessarily unique, in $]0;\pi[$, such that P_0 is a support vector of $\mathcal{S}$ in direction $v_{\theta_0} = (\cos\theta_0, \sin\theta_0)^\top$.

The supporting line of $\mathcal{S}$ in direction v_{θ_0} crosses L_γ in P_0 thus:

$$h(\theta_0) = \rho_{\mathcal{S}\cap L_\gamma}\left(\begin{pmatrix}0 \\ 1\end{pmatrix}\right)$$

Using lemma 8.1, one can show that for any θ_1, θ_2 in $]\theta_0;\pi[$ such that $\theta_1 < \theta_2$, we have $h(\theta_1) < h(\theta_2)$, the supporting line associated to θ_2 is above the one associated to θ_1 when they intersect L_γ. Thus h is increasing on $]\theta_0;\pi[$, similarly h is decreasing on $]0;\theta_0[$. h is unimodal and $\inf_{\theta\in]0;\pi[} h(\theta) = \rho_{\mathcal{S}\cap L_\gamma}((0,1))$.

In the other cases, we can similarly show that h is either increasing or decreasing on $]0;\pi[$, whether $\gamma \geq \rho_{\mathcal{S}}((1,0))$ or $-\rho_{\mathcal{S}}((-1,0)) \geq \gamma$ respectively.

If $\gamma > \rho_{\mathcal{S}}((1,0))$ or $-\rho_{\mathcal{S}}((-1,0)) > \gamma$, then its is clear that $\inf_{\theta \in]0;\pi[} h(\theta) = \rho_{\mathcal{S} \cap L_\gamma}((0\ 1)) = -\infty$.

Let us now suppose that $-\rho_{\mathcal{S}}((-1,0)) = \gamma$, then h is decreasing, since $\mathcal{S}$ is closed, $\mathcal{S} \cap L_\gamma$ is not empty and h converges towards a value y greater than $\rho_{\mathcal{S} \cap L_\gamma}((0,1))$. Let us suppose that (γ, y) does not belong to $\mathcal{S}$, since $\mathcal{S}$ is closed and convex there is a line, separating $\mathcal{S}$ and (γ, y), defined by an angle θ in $]0;\pi[$, and we have

$$\begin{aligned} \rho_{\mathcal{S}}(\theta) &< \gamma \cos\theta + y \sin\theta \\ h(\theta) &< y \end{aligned}$$

Which is in contradiction with the fact that h is decreasing toward y. Thus (γ, y) does belong to $\mathcal{S}$ and $\inf_{\theta \in]0;\pi[} h(\theta) = \rho_{\mathcal{S} \cap L_\gamma}((0,1))$. Similarly, if $\rho_{\mathcal{S}}((1\ 0)) = \gamma$, the theorem is verified. □

Appendix B

Sources for Figure 3.2

We want to know if a *tight* approximation always implies a *good* approximation. We use the Ellipsoidal Toolbox [KV06] version 1.1.2 released on June 6th 2009, together with Matlab 7.5.0.338 (R2007b).

The system we will study is taken from [Gir05]:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

with $x(0)$ in $\mathcal{X}_0$ and, for all t, $u(t)$ in $\mathcal{U}$.

The matrix A is generated from a bloc diagonal matrix D, and a change of variables given by a matrix P:

```
D=[-1  -4   0   0   0;
    4  -1   0   0   0;
    0   0  -3   1   0;
    0   0  -1  -3   0;
    0   0   0   0  -2];
P=[ 0.6  -0.1   0.1   0.7  -0.2;
   -0.5   0.7  -0.1  -0.8   0  ;
    0.9  -0.5   0.3  -0.6   0.1;
    0.5  -0.7   0.5   0.6   0.3;
    0.8   0.7   0.6  -0.3   0.2];
A=P*D*inv(P)
```

The matrix B is the identity matrix, and $\mathcal{U}$ is the ball of radius μ centered at the origin:

```
B=eye(5)
mu=0.01
```

The initial set, $\mathcal{X}_0$ is the ball centered at x_0 of radius r_0.

```
x0=[1 0 0 0 0]'
r0=0.01
EX0=ellipsoid(x0,r0^2*eye(5))
```

We will now discretize this system with a time step δ. We first take the matrix exponential of A:

```
delta=0.005
Phi=expm(delta*A)
```

Then, we use Lemma 3.1 in order to get $\mathcal{V}$:

```
nA=norm(A,2)
beta=mu*(exp(delta*nA)-1)/nA
V = beta^2 * eye(5)
EV = ellipsoid(V)
```

And Lemma 3.2 in order to get Ω_0:

```
nu=norm(x0,2)+r0
alpha=(exp(delta*nA)-1-delta*nA)*nu
B0=ellipsoid((alpha+beta)^2*eye(5))

EX1=Phi*EX0
CH=ellunion_ea([EX0 EX1])

l=2*rand(5,1)-1
Omega0=minksum_ea([CH B0], l)
```

The function `minksum_ea` takes as input an array a of ellipsoids and one vector l, it returns an over-approximation, tight in direction l, of the Minkowski sum of all the ellipsoids in a. Here, we chose a random vector in the unit cube.

We can now define the discretized system $\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V}$:

```
sys = linsys(Phi, B, EV, [], [], [], [], 'd')
```

We then compute a tight over-approximation of the reachable set for the first 1000 time steps:

```
N=1000
l0=2*rand(5,1)-1
rs_tight = reach(sys,Omega0,l0,N)
```

Tightness is guaranteed by applying `minksum_ea` to $\Phi\overline{\Omega}_k$ and $\mathcal{V}$ with vector $\ell_k = (\Phi^{-1})^\top \ell_0$. Indeed if $\overline{\Omega}_k$ is a tight over-approximation of Ω_k in direction ℓ_k, then $\Phi\overline{\Omega}_k$ is a tight over-approximation of $\Phi\Omega_k$ in direction $(\Phi^{-1})^\top \ell_k$, and applying `minksum_ea` with vector $(\Phi^{-1})^\top \ell_k$ will maintain tightness.

The problem with tightness in one direction, is that it does not guarantee a *good* approximation in the other directions. In order to remove tightness, we will call `minksum_ea` with a random vector at each step. For that purpose, we change the line that update l in the source of the function `eesm_de` called by `reach`. On line 57 in file `eesm_de.m` we change `l  = Ai' * l;` to `l=2*rand(5,1)-1;`. Then we compute a non-tight over-approximation of the reachable set:

```
rs_not_tight = reach(sys,Omega0,l0,N)
```

The first graph in Figure 3.2 was obtained by projecting the reachable tubes on the first variable:

```
Lproj=[1;0;0;0;0];

Y_tight=[ rho(get_ea(rs_tight),Lproj);
         -rho(get_ea(rs_tight),-Lproj)];
Y_not_tight=[ rho(get_ea(rs_not_tight),Lproj);
             -rho(get_ea(rs_not_tight),-Lproj)];

line([0:N;0:N],Y_tight,'Color','b'); hold on;
line([0:N;0:N],Y_not_tight,'Color','r');
```

The second one by projecting the reachable sets on the first two variables:

```
BB = [1 0 0 0 0; 0 1 0 0 0]';
P_tight = projection(rs_tight, BB);
P_not_tight = projection(rs_not_tight, BB);
plot_ea(P_tight,'b'); hold on;
plot_ea(P_not_tight,'r');
```

Appendix C

Introduction (in French)

C.1 Motivations

Cette thèse consiste en quelques contributions au problème du calcul des états atteignables pour les systèmes continus et hybrides. Pour les systèmes continus, ce problème peut être formulé de la façon suivante :

Considérons un système dynamique avec entrées défini dans un espace d'états $\mathcal{X}$ par une équation différentielle de la forme :

$$\dot{x} = f(x, u) \tag{C.1}$$

où u appartient à un ensemble prédéfini de signaux d'entrée admissible. Étant donné un ensemble $\mathcal{X}_0 \subset \mathcal{X}$, calculer tout les états visités par au moins une trajectoire générée par ce système à partir d'un point $x_0 \in \mathcal{X}_0$.

Résoudre ce problème permet de décider si un système garde un comportement correct pour *toutes* les perturbations admissibles ; *correct* peut signifier l'évitement d'un *mauvais* sous-ensemble de l'espace d'états ou, éventuellement, la vérification de propriétés temporelles plus complexes. L'ensemble atteignable peut être utilisé pour vérifier la robustesse d'un système de contrôle ou d'un circuit analogique soumis à des perturbations ou pour étudier des modèles biologiques où les valeurs de certains paramètres ainsi que les conditions environnementales sont mal connues. Historiquement, cette approche trouve son origine dans l'étude des systèmes hybrides (discret/continu) et l'extension des méthodes de vérification de programmes et de circuits digitaux à des systèmes manipulant des variables réelles.

Une explication intuitive du calcul de l'ensemble atteignable peut être donnée en terme de *simulation numérique*, l'approche la plus répandue pour la validation des systèmes complexes. Pour chaque simulation, on choisit *une* condition initiale et *un* signal d'entrée (aléatoire, périodique, étagé, etc.), la trajectoire induite est obtenue par intégration numérique ; on peut ensuite vérifier si cette trajectoire à un comportement correct ou non. Idéalement, ce processus devrait être répété pour *toutes* les conditions initiales et *tous* les signaux d'entrée possibles. Malheureusement cela implique *beaucoup* de simulations, en général une infinité indénombrable.

L'analyse d'atteignabilité permet d'obtenir les mêmes résultats que des simulations *exhaustives* en effectuant la simulation *en largeur d'abord* : au lieu de simuler complètement chaque trajectoire avant d'en commencer une autre, l'ensemble des points atteignables pendant un court instant est calculé à chaque pas de temps. Cette simulation basée sur des ensembles est, bien sûr, plus coûteuse qu'une simulation d'une trajectoire individuelle, mais permet de garantir le bon comportement d'un système.

À la différence de la plupart des méthodes d'analyse pour les systèmes continus, le calcul d'atteignabilité donne une information sur le comportement *transitoire* du système étudié et pas uniquement sur son comportement asymptotique. Cette approche est donc particulièrement intéressante pour l'analyse des systèmes *hybrides* (discret/continu) que les méthodes d'analyse classiques abordent difficilement. Un modèle hybride peut exprimer, par exemple, certains comportements d'un système idéalement linéaire soumis à saturations, ou d'autres phénomènes de commutation.

Notons que le calcul d'atteignabilité présente des similitudes avec le domaine de l'*analyse par intervalle* qui permet des calculs numériques rigoureux malgré les erreurs d'arrondi. Ces deux techniques manipulent des ensembles, et garantissent que ces ensembles contiennent le résultat recherché, elles partagent donc des techniques de calcul ensembliste. Toutefois, la source d'incertitude dans l'analyse par intervalle est en général liée aux erreurs de calcul provoquées par l'utilisation d'une arithmétique à précision bornée, alors qu'ici elle est liée à l'environnement du système modélisé ou au fait que le modèles lui-même a certains paramètres qui ne sont pas connus précisément.

Les principales contributions de cette thèse sont :

1. *Systèmes Linéaires* : pour les systèmes définis pas une équation différentielle linéaires (invariante en temps), nous développons un nouveau schéma algorithmique qui présente une amélioration tant au niveau théorique que pratique. Ce schéma permet d'analyser des systèmes auparavant inaccessible sans souffrir de l'effet d'emballage ;
2. *Systèmes Hybrides* : nous étendons ce schéma au systèmes linéaires par morceaux[1] définis par un automate hybride avec des équations différentielle linéaires. La principale difficulté est de calculer efficacement l'*intersection* entre l'ensemble atteignable dans un mode avec les gardes de transition ;

Dans la suite de ce chapitre nous présentons plus précisément le problème de l'atteignabilité ainsi que les techniques associées avant de présenter le contenu de la thèse plus en détail.

C.2 Prouver la Sûreté

Nous ne nous intéressons pas ici au comportement asymptotique d'un système dynamique, pas plus qu'à son comportement au voisinage d'une trajectoire de réfé-

[1]ou affine par morceaux

rence, mais plutôt au comportement transitoire de toutes les trajectoires générées par ce système :

$$\dot{x} = f(x, u)$$

où u appartient à un ensemble prédéfini de signaux d'entrée admissible $\widetilde{\mathcal{U}} \subset \mathbb{R}^+ \mapsto \mathcal{U}$, et $x(0)$ est pris dans $\mathcal{X}_0$.

L'ensemble de toutes les trajectoires possibles est alors :

$$\Xi(\mathcal{X}_0) = \left\{ \xi : \xi(0) \in \mathcal{X}_0 \text{ and } \exists u \in \widetilde{\mathcal{U}},\, \forall t \geq 0,\, \dot{\xi}(t) = f(\xi(t), u(t)) \right\}$$

Étant donné une trajectoire, il est possible de vérifier qu'elle respecte certaines propriétés. Une propriété particulièrement intéressante est la propriété de sûreté. Une trajectoire ξ est *sûre* si elle n'entre pas dans un *mauvais* ensemble $\mathcal{F}$:

$$\forall t \geq 0,\, \xi(t) \notin \mathcal{F}$$

L'ensemble $\mathcal{F}$ est une région de l'espace d'états qui doit être évitée. Par exemple, un avion doit éviter de voler à une vitesse inférieure à sa vitesse de décrochage, la température du cœur d'une centrale nucléaire ne doit pas dépasser un certain seuil critique.

Qu'une trajectoire donnée soit sûre n'est pas suffisant, il est souhaitable que toutes les trajectoires du système soient sûres :

$$\forall \xi \in \Xi(\mathcal{X}_0),\, \forall t \geq 0,\, \xi(t) \notin \mathcal{F}$$

Si $\mathcal{F}$ est évité, on dit que le système est *sûr*.

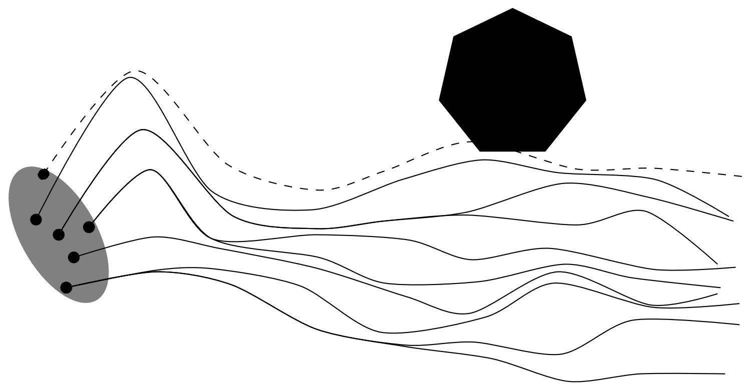


Fig. C.1 – Quelques trajectoires quittent l'ensemble initial (en gris). Seule une trajectoire (en pointillé) intersecte l'ensemble *mauvais* (en noir).

Nous allons maintenant présenter les trois principales méthodes utilisées pour vérifier la sûreté d'un système ; la dernière, l'analyse d'atteignabilité, est le sujet de cette thèse.

C.2.1 Simulations

Si l'ensemble des conditions initiales $\mathcal{X}_0$ ainsi que l'ensemble des entrées admissibles sont finis, alors il est possible d'analyser le système de façon exhaustive par simulations. Malheureusement $\Xi(\mathcal{X}_0)$ est souvent infini et rarement dénombrable, il est impossible d'étudier toutes les trajectoires individuellement. Par ailleurs, les trajectoires sont en général simulées pour un intervalle de temps donnés, on n'étudie donc souvent la sûreté d'un système que sur un temps borné :

$$\forall \xi \in \Xi(\mathcal{X}_0),\ \forall t \in [0;T],\ \xi(t) \notin \mathcal{F}$$

où T est la borne temporelle.

En appliquant des méthodes statistiques [CDL08], un ensemble fini de trajectoires générée aléatoirement peut donner une certaine confiance en la sûreté du système.

Une autre solution est de chercher un ensemble fini de trajectoires qui vont mettre en évidence tous les comportement du système [KKMS03, BF04, BCLM06, GP06, DM07]. La plupart de ces techniques sont basées sur les arbres RRT (Rapidly-exploring Random Trees) développés à l'origine dans le domaine de la planification de mouvement, l'analyse de sensibilité, ou les relations de bisimulation approximative. L'idée est de couvrir l'ensemble des états atteignable par un petit voisinage d'un ensemble fini de trajectoires. En d'autres termes, cet ensemble fini de trajectoires doit être tel que tout état atteignable par une trajectoire du système est à une petite distance d'au moins une des trajectoires choisies. En respectant certaines conditions il est parfois possible de borner cette distance.

En conclusion, en se basant sur des simulations, si une des trajectoires simulées n'est pas sûre alors le système n'est pas sûr, dans le cas contraire on ne peut pas conclure. Il est tout de même possible d'utiliser une analyse statistique ou de générer les trajectoires d'une façon qui apporte un certain degré de confiance en la sûreté du système. Sous certaines conditions il est même possible de garantir que le système est sûr.

C.2.2 Fonction Barrière

Une autre approche au problème de sûreté est de chercher une surface qui sépare l'ensemble des états initiaux $\mathcal{X}_0$ et l'ensemble interdit $\mathcal{F}$ et qui ne peut être traversée par aucune trajectoire. Formellement :

Définition C.1 ([PJ04]). *Une fonction différentiable $B : \mathcal{X} \to \mathbb{R}$ est une* fonction barrière *si et seulement si elle vérifie les conditions suivantes :*

$$\begin{aligned} B(x) &> 0 & \forall x \in \mathcal{F} \\ B(x) &\leq 0 & \forall x \in \mathcal{X}_0 \\ \frac{\partial B}{\partial x} \cdot f(x,u) &\leq 0 & \forall (x,u) \in \mathcal{X} \times \mathcal{U} \textit{ tels que } B(x) = 0 \end{aligned}$$

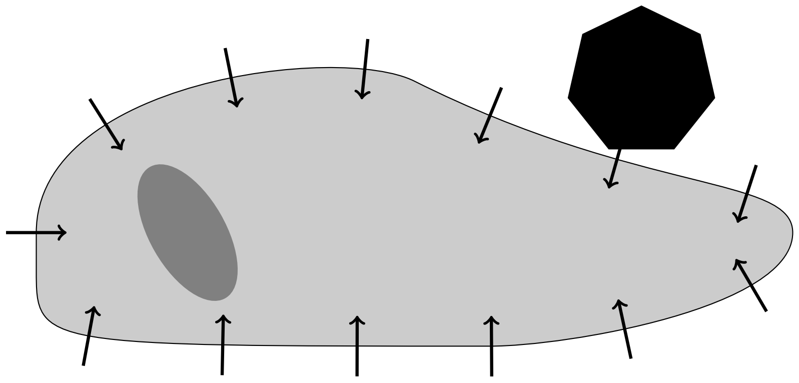


Fig. C.2 – Une barrière prouve la sûreté du système.

Le Figure C.2 illustre cette notion.

Cette notion de barrière est liée à celle de fonction de Lyapunov, développée à l'origine pour l'analyse de stabilité à la fin du 19$^{\text{ème}}$ siècle par Aleksandr Lyapunov et encore étudiée aujourd'hui. Intuitivement, une fonction de Lyapunov V est positive partout sauf en un point x_0 où $V(x_0) = 0$, et sa dérivée temporelle, égale à $\frac{\partial V}{\partial x} \cdot f(x, u)$ si elle est différentiable, n'est nulle part positive. Si une telle fonction existe, alors x_0 est un point d'équilibre stable.

Ici, nous ne sommes pas intéressés par la stabilité du système, mais par sa sûreté. Si il existe un k dans $\mathbb{R}$ tel que $V(x) > k$ pour tout x dans $\mathcal{F}$, et $V(x) \leq k$ pour tout x dans $\mathcal{X}_0$ alors $x \mapsto V(x) - k$ est une fonction barrière pour notre problème. Ainsi, une fonction de Lyapunov peut être utilisée pour construire une fonction barrière.

Malheureusement, même pour les système linéaire stable, pour lesquels on sait construire de façon systématique des fonctions de Lyapunov, il n'est pas toujours facile de trouver une fonction barrière. En effet, pour une fonction de Lyapunov donnée, il n'est pas garantie qu'un tel k existe. Pour résumer, les fonctions barrières peuvent être dures à trouver mais elles garantissent la sûreté du système pour un temps non borné.

C.2.3 Atteignabilité

L'approche envisagée dans cette thèse est de calculer l'ensemble atteignable[2]. L'ensemble atteignable est l'ensemble de tous les points que le système peut atteindre :

$$\mathcal{R}(\mathcal{X}_0) = \{x : \exists \xi \in \Xi(\mathcal{X}_0),\, \exists t \in \mathbb{R}^+,\, x = \xi(t)\}$$

[2]Un autre objet intéressant dans l'analyse d'atteignabilité est le tube d'atteignabilité :

$$\mathcal{R}_{\text{tube}}(\mathcal{X}_0) = \{(x, t) : \exists \xi \in \Xi(\mathcal{X}_0),\, x = \xi(t)\}$$

mais nous allons ici principalement nous concentrer sur l'ensemble atteignable.

Malheureusement cet ensemble est souvent impossible à calculer. Nous allons donc l'approcher. Pour pouvoir tout de même vérifier certaines propriétés, comme la sûreté, cette approximation doit être conservative, en d'autres termes, elle doit être une sur-approximation : $\mathcal{R} \subseteq \overline{\mathcal{R}}$.

Ainsi, si $\overline{\mathcal{R}} \cap \mathcal{F} = \emptyset$, alors $\mathcal{R} \cap \mathcal{F} = \emptyset$, et le système est sûr. Si $\overline{\mathcal{R}} \cap \mathcal{F}$ n'est pas vide, l'analyse n'est pas concluante et on peut essayé d'améliorer la sur-approximation. Mais si le système n'est pas sûr, toute sur-approximation intersecte l'ensemble interdit $\mathcal{F}$, et aucune sur-approximation ne permet de conclure. Il est donc parfois utile de calculer une sous-approximation $\underline{\mathcal{R}} \subseteq \mathcal{R}$. Si $\underline{\mathcal{R}}$ intersecte $\mathcal{F}$ alors le système n'est pas sûr.

Plusieurs approximations sont possibles, les plus évidentes, et plus inutiles, peuvent être obtenues en remarquant que $\emptyset \subseteq \mathcal{R} \subseteq \mathcal{X}$. L'algorithme le plus rapide renvoie donc tout simplement $\emptyset$ et $\mathcal{X}$ comme sous- et sur-approximations de l'ensemble atteignable, c'est pourquoi nous allons avoir besoin d'une notion de qualité d'une approximation.

D'autres méthodes bien plus intéressantes ont été présentées dans les précédentes sections. En effet, un ensemble fini de trajectoires donne une sous-approximation de $\mathcal{R}$. Il est même possible, sous certaines conditions, de garantir qu'en continuant à générer des trajectoires pendant un temps infini, l'ensemble des points visités sera dense dans $\mathcal{R}$. Par ailleurs, si on sait que tous les points de $\mathcal{R}$ sont dans un ϵ-voisinage de l'ensemble fini de trajectoires considérées, alors ce voisinage définit une sur-approximation de $\mathcal{R}$.

Une fonction barrière V définit également une sur-approximation de l'ensemble atteignable : $\mathcal{R} \subseteq \{x : V(x) \leq 0\}$. Réciproquement, la frontière de l'ensemble atteignable peut définir une fonction barrière, celle dont la partie négative est la plus petite pour l'inclusion.

Nous allons prendre une approche similaire à la simulation. Mais au lieu d'avoir un point qui évolue dans l'espace d'états, nous allons avoir un ensemble. Comme pour la simulation, il est difficile de considérer un horizon temporel non-borné. C'est pourquoi nous nous concentrerons sur l'atteignabilité avec une borne temporelle. L'ensemble que nous voulons sur-, ou sous-, approcher est :

$$\mathcal{R}_{[0;T]}(\mathcal{X}_0) = \{x : \exists \xi \in \Xi(\mathcal{X}_0),\, \exists t \in [0;T],\, x = \xi(t)\}$$

où T est notre borne temporelle.

Dans la suite, $\mathcal{R}_{[t_0;t_1]}(\mathcal{Y})$ désigne l'ensemble des points atteignables à partir de $\mathcal{Y}$ entre les temps t_0 et t_1 par le système (C.1), où, plus formellement :

$$\mathcal{R}_{[t_0;t_1]}(\mathcal{Y}) = \{x : \exists \xi \in \Xi(\mathcal{Y}),\ \text{ et } \exists t \in [t_0;t_1],\, x = \xi(t)\}$$

Nous définissons également $\mathcal{R}_t(\mathcal{Y})$ comme étant $\mathcal{R}_{[t;t]}(\mathcal{Y})$. Quand $\mathcal{Y}$ est omis, c'est qu'il s'agit de $\mathcal{X}_0$.

Remarquons que pour un horizon temporel borné en temps discret il existe une autre méthode pour vérifier qu'un système est sûr : l'existence d'une trajectoire non-sûre est exprimée par une formule obtenue en déroulant la relation de récurrence définissant le système k fois, avec un nouvel ensemble de variables pour chaque pas de temps. Cette approche sous-tend en fait la commande prédictive, son adaptation à la vérification des systèmes hybrides a été étudiée dans [BM99].

C.3 Plan de la Thèse

Cette annexe est une traduction de l'introduction de cette thèse, rédigée en anglais. Dans la suite nous suivons le plan de la version anglaise, chaque section correspond à un chapitre de la thèse. Elle est constituée de deux parties principales. Dans la première nous nous intéressons à l'atteignabilité des systèmes continus linéaires. La seconde adapte ces résultats aux systèmes hybrides. Mais d'abord, nous étudions dans le Chapitre 2 quelques représentations d'ensembles utilisées dans le contexte de l'atteignabilité.

Partie I La première partie de cette thèse est constituée de trois chapitres. Après avoir donné un aperçu de l'état de l'art dans le Chapitre 3 nous présentons, dans le Chapitre 4, la principale contributions de cette thèse : un nouveau schéma algorithmique pour l'analyse d'atteignabilité des systèmes linéaires invariants. Le seul inconvénient de son implémentation exacte est que sa sortie peut être difficile à manipuler. Nous nous intéressons donc également à une implémentation produisant une sur-approximation non soumise à l'effet d'emballage, une accumulation incontrôlée des erreurs d'approximation, sujet de nombreuses recherches depuis le début des années soixante. Dans le dernier chapitre de cette première partie, Chapitre 5, nous présentons une variante de cet algorithme utilisant les fonctions support, une représentation fonctionnelle des ensembles convexes.

Partie II La seconde partie de cette thèse débute par une courte introduction à l'analyse d'atteignabilité pour les systèmes hybrides au Chapitre 6. Nous nous concentrons ensuite sur deux aspects des systèmes hybrides : premièrement, nous étudions, Chapitre 7, les conséquences de l'introduction d'un invariant dans la partie continue, et pour finir nous expliquons comment intersecter la sortie des algorithmes décrits Partie I avec des gardes hyerplanaires dans le Chapitre 8.

Le dernier chapitre, traduit Annexe D, résume les contributions de cette thèse et suggère quelques pistes de recherche.

Appendix D

Conclusion (in French)

Dans cette thèse nous avons donné quelques contributions à l'analyse d'atteignabilité des systèmes hybrides à partie continue linéaire, invariants convexes, gardes hyperplanaires, et saut affine.

La plus simple, et la plus puissante, d'entre elles est la décomposition de la relation de récurrence :

$$\Omega_{i+1} = \Phi\Omega_i \oplus \mathcal{V} \tag{D.1}$$

en trois :

$$\begin{aligned} \mathcal{A}_0 &= \Omega_0 & \mathcal{A}_{i+1} &= \Phi\mathcal{A}_i \\ \mathcal{V}_0 &= \mathcal{V} & \mathcal{V}_{i+1} &= \Phi\mathcal{V}_i \\ \mathcal{S}_0 &= \{0\} & \mathcal{S}_{i+1} &= \mathcal{S}_i \oplus \mathcal{V}_i \end{aligned}$$

Ω_i peut alors être exprimé comme la somme de Minkowski de $\mathcal{A}_i$ et $\mathcal{S}_i$.

La séparation de la transformation linéaire et la somme de Minkowski nous a permis de développer deux algorithmes efficaces calculant les N premiers termes de la séquence définie par l'équation (D.1), l'une utilisant des zonotopes, l'autre des fonctions support, ainsi qu'une famille d'algorithme sur-approchants qui ne sont pas soumis à l'effet d'emballage. L'intérêt principale des algorithmes approchants est la simplicité de leur sortie, qui est constituée d'une liste d'ensembles dans la représentation choisie.

Dans la seconde partie nous avons adapté ces résultats à l'étude des systèmes hybrides. Nous avons montré comment utiliser l'invariant pour améliorer l'approximation de l'ensemble atteignable dans la partie continue. Mais la principale contribution de la Partie II est l'utilisation de la séquence exacte des Ω_i, représentés par des zonotopes d'ordre élevé ayant des générateurs en commun, ou par leur fonction support, pour calculer une approximation de l'intersection entre l'ensemble atteignable de la partie continue avec des gardes hyperplanaires. Non seulement cela permet d'améliorer considérablement l'approximation de cette ensemble, mais cela prouve aussi que les Ω_i exactes sont utiles malgré leur représentation particulière.

La plupart des algorithmes décrits dans cette thèse ont été implantés dans un prototype en OCaml. Pour des systèmes purement continus il est plus performant que les autres outils en terme de vitesse d'exécution ainsi que terme de qualité de l'approximation. Dans le cas hybride, il présente des résultats prometteurs et peut se mesurer à PHAVer, un outil hautement optimisé, sur des exemples non-triviaux.

Naturellement, beaucoup reste à faire, et il y a de nombreuses opportunités pour de futurs travaux.

Systèmes Linéaires Périodiques Nos algorithmes sont dédiés aux systèmes linéaires invariant en temps. Une première extension serait de les adapter aux systèmes périodiques $\dot{x}(t) = A(t)x(t) + u(t)$ où A, ou l'ensemble des signaux d'entrée admissibles $\mathcal{U}$, ou les deux, sont périodiques.

Systèmes Linéaires Pour les systèmes linéaires variants en temps, il est sans doute possible de modifier nos algorithmes pour qu'ils calculent *un* Ω_k pour un certain k. Cela pourrait permettre de réduire l'effet d'emballage produit par les algorithmes standards. Un autre moyen de réduire l'effet d'emballage est d'adapter notre procédure de discretization aux systèmes temps variant, en augmentant la taille du pas de temps pour obtenir une certain précision cela permet de réduire le nombre de pas temps nécessaire pour couvrir une certain durée et donc de réduire l'effet d'emballage indirectement.

Systèmes Linéaires Paramétrisés Nous considérons ici que toutes les matrices sont connues exactement, mais ce n'est pas toujours le cas et il serait intéressant d'utiliser notre schéma algorithmique dans le contexte des systèmes paramétrisés pour lesquels ont sait juste que la matrice A appartient à un petit sous ensemble de $M_d(\mathbb{R})$. Cela pourrait également permettre de prendre en compte l'utilisation d'une arithmétique à virgule flottante.

Ellipsoïdes Comme nous en avons discuté dans la Section 3.4.2.2, les enveloppes ellipsoïdales ajustées des Ω_i ont tendances à s'aplatir ; relâcher cette propriété de contact peut améliorer l'approximation en terme de distance de Hausdorff Il serait intéressant d'analyser ce phénomène pour peut-être en déduire une méthode offrant un bon compromis entre l'ajustement et l'aplatissement.

Zonotopes Cette représentation compacte est utilisée dans de nombreux domaines. Malheureusement il semble qu'il n'y ait pas d'algorithme permettant de réduire l'ordre d'un zonotope de façon satisfaisante. Améliorer les méthodes existantes serait déjà un premier pas.

Systèmes Hybrides Une première piste pour accélérer l'analyse est de considérer une représentation par plusieurs fonctions support de complexité, et précision, croissante comme nous l'avons suggéré dans la Section 8.3.3. Par ailleurs, parmi les aspects des systèmes hybrides qui n'ont pas été traités dans cette thèse, les exécutions Zeno, les changements de mode fréquents, et les modes glissant présentes un intérêt particulier ; Moreover, among the aspects of hybrid systems that have not been addressed in this thesis, Zeno executions, chattering, and sliding modes are of particular interest ; ils sont mal gérés par nos algorithmes.

Systèmes Non-Linéaires Des systèmes plus complexes peuvent être analysé en remplaçant la dynamique non-linéaire par un système hybride avec une dynamique continue plus simple, cette technique est connue sous le terme hybridisation [ADG07]. Des variantes de cette technique utilisant des transitions hystérèse ou une hybridisation dynamique semblent prometteuses [DLGM09]. Abstraire la dynamique continue peut être fait efficacement pour les systèmes multi-affine [KB06, MB08], en effet, la valeur du flux à l'intérieur d'une boîte est exactement l'enveloppe convexe des valeurs du flux aux sommets de cette boîte. La prochaine étape serait de caractérisé l'ensemble des fonctions pour lesquelles la valeur à l'intérieur d'une boîte est exactement l'enveloppe par intervalles des valeurs du flux aux sommets de cette boîte.

Outils Finalement, un effort important doit être fourni pour faire passer notre outil de l'état de prototype à l'état d'outil distribuable.

Bibliography

[ACH$^+$95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995. (Cited on page 82.)

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993. (Cited on page 80.)

[ADG07] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Inf.*, 43(7):451–476, 2007. (Cited on pages 123 and 151.)

[AF96] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. (Cited on page 96.)

[ASB09] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, In Press, Corrected Proof:–, 2009. (Cited on page 40.)

[BCLM06] Michael S. Branicky, Michael M. Curtiss, Joshua Levine, and Stuart Morgan. Sampling-based planning, control and verification of hybrid systems. *Control Theory and Applications, IEE Proceedings*, 153(5):575–590, Sept. 2006. (Cited on pages 4 and 144.)

[BE01] Marshall Wayne Bern and David Eppstein. Optimization over zonotopes and training support vector machines, 2001. Talk given at WADS. (Cited on page 99.)

[BF04] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *Hybrid*

Systems: Computation and Control, volume 2993 of *Lecture Notes in Computer Science*, pages 451–471. Springer, 2004. (Cited on pages 4 and 144.)

[BFP+73] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973. (Cited on page 101.)

[BM99] Alberto Bemporad and Manfred Morari. Verification of hybrid systems via mathematical programming. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 1999. (Cited on pages 6 and 147.)

[BNO03] Dimitri P. Bertsekas, Angelia Nedic, and Asuman E. Ozdaglar. *Convex analysis and optimization.* Athena Scientific, 2003. (Cited on page 19.)

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge University Press, 2004. (Cited on page 19.)

[BZ80] Egon Balas and Eitan Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28(5):1130–1154, Sep. - Oct. 1980. (Cited on page 99.)

[CC77] Patrick Cousot and Radhia Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *POPL'77: 4th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252. ACM, January 1977. (Cited on page 15.)

[CC04] Robert Clarisó and Jordi Cortadella. The octahedron abstract domain. In *SAS*, volume 3148 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2004. (Cited on page 15.)

[CDL08] Edmund M. Clarke, Alexandre Donzé, and Axel Legay. Statistical model checking of mixed-analog circuits with an application to a third-order delta-sigma modulator. In *Haifa Verification Conference*, volume 5394 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2008. (Cited on pages 4 and 144.)

[Chu99] Alongkrit Chutinan. *Hybrid System Verification Using Discrete Model Approximations.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1999. (Cited on pages 33 and 43.)

[CK98] Alongkrit Chutinan and Bruce H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. volume 2, pages 2089–2094 vol.2, Dec 1998. (Cited on pages 33 and 43.)

[CPPAV06] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. Languages and tools for hybrid systems design. *Foundations and Trends in Electronic Design Automation*, 1(1/2):1–193, 2006. (Cited on page 83.)

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. (Cited on pages 26 and 47.)

[Dan00] Thao Dang. *Vérification et Synthèse des Systèmes Hybrides.* PhD thesis, Institut National Polytecnique de Grenoble, 2000. (Cited on pages 33 and 43.)

[DFGLG08] Thao Dang, Goran Frehse, Antoine Girard, and Colas Le Guernic. *Approches formelles des systèmes embarqués communicants*, Chapitre Outils pour l'analyse des modèles hybrides. Traité IC2, série Informatique et systèmes d'information. Hermes Science, 2008. (Cited on page 83.)

[DLGM09] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, volume 5688 of *Lecture Notes in Bioinformatics*, pages 126–141. Springer, 2009. (Cited on pages 123 and 151.)

[DM07] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 174–189. Springer, 2007. (Cited on pages 4 and 144.)

[FI04] Ansgar Fehnker and Franjo Ivancic. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2004. (Cited on page 118.)

[Fre08] Goran Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. *International Journal on Software Tools for Technology Transfer (STTT)*, 10(3), June 2008. (Cited on pages 15, 40, and 118.)

[Gir04] Antoine Girard. *Analyse Algorithmique des Systèmes Hybrides.* PhD thesis, Institut National Polytecnique de Grenoble, 2004. (Cited on page 33.)

[Gir05] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2005. (Cited on pages 34, 35, 40, 42, 57, and 137.)

[GLG08a] Antoine Girard and Colas Le Guernic. Efficient reachability analysis for linear systems using support functions. In *IFAC World Congress 2008*, 2008. (Cited on page 65.)

[GLG08b] Antoine Girard and Colas Le Guernic. Zonotope-hyperplane intersection for hybrid systems reachability analysis. In *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2008. (Cited on page 85.)

[GLGM06] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2006. (Cited on page 45.)

[GM99] Mark R. Greenstreet and Ian Mitchell. Reachability analysis using polygonal projections. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 103–116. Springer, 1999. (Cited on page 24.)

[GNZ03] Leonidas J. Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 803–812, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. (Cited on pages 18, 105, and 106.)

[GP06] Antoine Girard and George J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2006. (Cited on pages 4 and 144.)

[Han05] Zhi Han. *Reachability Analysis of Continuous Dynamic Systems Using Dimension Reduction and Decomposition.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 2005. (Cited on pages 40 and 74.)

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998. (Cited on page 82.)

[Jar73] R. A Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973. (Cited on page 100.)

[KB06] Marius Kloetzer and Calin Belta. Reachability analysis of multi-affine systems. In *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 2006. (Cited on pages 123 and 151.)

[KGB04] Michal Kvasnica, Pascal Grieder, and Mato Baotić. Multi-Parametric Toolbox (MPT), 2004. (Cited on pages 35 and 39.)

[KGBM04] Michal Kvasnica, Pascal Grieder, Mato Baotić, and Manfred Morari. Multi-parametric toolbox (mpt). In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2004. (Cited on page 35.)

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming (in russian). *Doklady Akademiaa Nauk USSR.*, 244:1093–1096, 1979. English Translation: Soviet Mathematics Doklady, Volume 20, 191–194. (Cited on page 13.)

[Kie53] Jack Carl Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4:502–506, 1953. (Cited on pages 109 and 112.)

[KKMS03] James Kapinski, Bruce H. Krogh, Oded Maler, and Olaf Stursberg. On systematic simulation of open continuous systems. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2003. (Cited on pages 4 and 144.)

[Küh98] Wolfgang Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61(1):47–68, 1998. (Cited on page 40.)

[KV06] Alex A. Kurzhanskiy and Pravin Varaiya. Ellipsoidal toolbox. Technical Report UCB/EECS-2006-46, EECS Department, University of California, Berkeley, May 2006. (Cited on pages 13, 31, 38, 39, 54, and 137.)

[LGG09a] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer, 2009. (Cited on page 85.)

[LGG09b] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, In Press, Corrected Proof:–, 2009. (Cited on page 65.)

[Loh88] Rudolf Lohner. *Einschliessung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen.* PhD thesis, Universität Fridericiana Karlsruhe, 1988. (Cited on page 38.)

[MB08] Oded Maler and Grégory Batt. Approximating continuous systems by timed automata. In *Formal Methods in Systems Biology*, volume

5054 of *Lecture Notes in Computer Science*, pages 77–89. Springer, 2008. (Cited on pages 123 and 151.)

[Min01] Antoine Miné. The octagon abstract domain. In *Proc. of the Workshop on Analysis, Slicing, and Transformation (AST'01)*, IEEE, pages 310–319, Stuttgart, Gernamy, October 2001. IEEE CS Press. (Cited on page 15.)

[MKC09] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis.* SIAM, January 2009. (Cited on page 12.)

[MMP91] Oded Maler, Zohar Manna, and Amir Pnueli. From timed to hybrid systems. In *Real-Time: Theory in Practice, REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 447–484. Springer, 1991. (Cited on page 80.)

[Moo66] Ramon E. Moore. *Interval Analysis.* Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs N. J., January 1966. (Cited on page 2.)

[MT00] Ian Mitchell and Claire Tomlin. Level set methods for computation in hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 310–323. Springer, 2000. (Cited on page 24.)

[Mul59] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2(4):19–20, 1959. (Cited on page 106.)

[Neu03] Arnold Neumaier. Taylor forms–use and limits. *Reliable Computing*, 9(1):43–79, 2003. (Cited on page 22.)

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2004. (Cited on pages 4 and 144.)

[Rou96] Mireille Rousset. *Sommes de Minkowski de Triangles.* PhD thesis, Université Joseph Fourier, 1996. (Cited on page 18.)

[RST02] Lluís Ros, A. Sabater, and Federico Thomas. An ellipsoidal calculus based on propagation and fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(4):430–442, 2002. (Cited on pages 13 and 38.)

[RW98] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational analysis.* Springer, 1998. (Cited on pages 19, 27, and 96.)

[Set99] James Albert Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge University Press, 1999. (Cited on page 24.)

[SK03] Olaf Stursberg and Bruce H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 482–497. Springer, 2003. (Cited on page 38.)

[Tiw08] Hans Raj Tiwary. On the hardness of computing intersection, union and minkowski sum of polytopes. *Discrete Computational Geometry*, 40(3):469–479, 2008. (Cited on page 26.)

[Tod02] Michael J. Todd. The many facets of linear programming. *Mathematical Programming*, 91(3):417–436, February 2002. (Cited on page 13.)

[Var98] Pravin Varaiya. Reach set computation using optimal control. In *Proc. KIT Workshop on Verification of Hybrid Systems.* Verimag, Grenoble, 1998. (Cited on pages 43 and 66.)

[Wei07] Christophe Weibel. *Minkowski Sums of Polytopes.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2007. (Cited on page 26.)

[YN76] D. B. Yudin and A. S. Nemirovski. Informational complexity and efficient methods for the solution of convex extremal problems (in russian). *Ékonomika i Matematicheskie metody*, 12:357–369, 1976. English translation: Matekon 13(2), 3–25. (Cited on page 13.)

[Zas75] Thomas Zaslavsky. *Facing Up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes.* Number 154 in Memoirs of the American Mathematical Society. American Mathematical Society, 1975. (Cited on pages 26 and 96.)

[Zie95] Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer, 1995. (Cited on page 13.)

[Zon05] Chuanming Zong. What is known about unit cubes. *Bulletin (New Series) of the American Mathematical Society*, 42(2):181–211, January 2005. (Cited on page 17.)

Calcul d'Atteignabilité des Systèmes Hybrides à Partie Continue Linéaire

Mots clés : atteignabilité, système hybride, équation différentielle linéaire, approximation, zonotope, fonction support, hyperplan, intersection.

Résumé : Cette thèse est consacrée au calcul des états atteignables des systèmes linéaires et hybrides. La première partie est consacrée aux systèmes linéaires. Après avoir présenté les méthodes existantes, nous introduisons notre principale contribution : un nouveau schéma algorithmique pour l'analyse d'accessibilité des systèmes linéaires invariants qui surclasse nettement les algorithmes existants. Une implémentation exacte peut produire des ensembles difficiles à manipuler, nous proposons donc une version produisant une sur-approximation non soumise à l'effet d'emballage, une accumulation incontrôlée des erreurs d'approximation, ainsi qu'une variante dédiée aux fonctions support, une représentation fonctionnelle des ensembles convexes. La deuxième partie adapte ces résultats aux systèmes hybrides. Nous montrons d'abord comment gérer les invariants, avant de nous intéresser à l'approximation de l'intersection entre l'ensemble atteignable par la dynamique continue et des gardes hyperplanaires.

Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics

Keywords: reachability, hybrid system, linear differential equation, approximation, zonotope, support function, hyperplane, intersection.

Abstract: This thesis is devoted to the problem of computing reachable sets of linear and hybrid systems. In the first part, after exposing existing approaches for reachability analysis of linear systems, we present the main contribution of the thesis: a new algorithmic scheme for linear time-invariant systems that definitely outperforms existing algorithms. As the exact implementation furnishes a representation of the reachable sets that is sometimes hard to manipulate, we propose an approximate version that is not subject to the wrapping effect, an uncontrolled growth of the approximation errors. We also discuss a variant of this algorithm specialized to support functions, a functional representation of convex sets. In the second part, we extend this work to hybrid systems. We first show how to deal with the constraints on the continuous dynamics imposed by the invariants. Then, we present algorithms for approximating the intersection of the continuous reachable sets with hyperplanar guards.

VERIMAG
Centre Équation - 2, avenue de Vignate
38610 Gières, FRANCE