Techniques pour l'analyse formelle de systèmes dynamiques non-linéaires

Romain Testylier

Université de Grenoble

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel :

Présentée par

Romain Testylier

Thèse dirigée par **Thao Dang**

préparée au sein **Verimag**
et de **Mathématiques, Sciences et Technologies de l'Information, Informatique**

Analyse d'accessibilité des systèmes dynamiques non-linéaires

Thèse soutenue publiquement le **7/12/2012**,
devant le jury composé de :

Erika Abraham
Professeur, Université d'Aix-la-Chapelle, Rapporteur
Eric Goubault
Directeur de Recherche, CEA, Rapporteur
Olivier Bournez
Professeur, Ecole Polytechnique, Examinateur
Bertrand Jeannet
Chargé de Recherche, INRIA, Examinateur
Carla Piazza
Professeur, Université d'Udine, Examinatrice
Thao Dang
Chargée de Recherche, Verimag/CNRS, Directeur de thèse

Remerciements

J'adresse ces premiers remerciements à l'ensemble des membres de mon jury de thèse, pour m'avoir fait l'honneur d'accepter d'évaluer mes travaux. Je les remercie d'avoir assisté à ma soutenance malgré une météo éxécrable ainsi qu'une visio-conférence peu audible.

Je remercie également ma directrice de Thèse, Thao Dang pour m'avoir donné l'opportunité de faire cette thèse et, sans qui, je n'aurai pu mener ces travaux à terme. Je lui suis particulièrement reconnaissant pour son soutien, ses conseils ainsi que pour sa patience et sa disponibilité.

Je remercie par la même occasion Oded Maler pour m'avoir ouvert les portes de son équipe au sein du laboratoire Verimag.

Je remercie de plus l'ensemble des membres du laboratoire pour l'ambiance chaleureuse dans laquelle j'ai pu travailler. Je voudrais spécialement remercier Nicolas, Valentin, Giovanni et Tayeb du bureau 39 pour leur aide ainsi que pour les discussions nombreuses et variées que nous avons eue.

Je remercie enfin mes parents pour leurs soutiens et leurs encouragements. Je les remercies de plus pour l'économie faites en divers suivies psychiatriques pendant ces derniers mois de rédaction ainsi que pour ce super pot de thèse.

Contents

Introduction (in French)

Cette thèse présente plusieurs contributions concernant la vérification formelle de systèmes dynamiques non-linéaires. Dans cette introduction nous présentons dans un premier temps, nos motivations et la problématique traitée dans cette thèse. Nous résumons ensuite les principales approches existantes pour la vérification formelle de systèmes dynamiques et nous finissons par présenter les grandes lignes de cette thèse.

Motivations

Les systèmes à dynamiques continues représentent un concept mathématique où un ensemble de règles fixées décrivent l'évolution de l'état d'un système en fonction du temps dans un ensemble appelé espace d'état.

Le nombre de variables de ce système correspond à la dimension de l'espace d'état; à tout moment le comportement du systèmes est donné par un vecteur de nombre réel représentant les valeurs des variables d'état. Les systèmes à temps continu sont largement utilisés pour modéliser des phénomènes physiques, des réactions chimiques et des systèmes biologiques dans de nombreux domaines d'ingénierie. De plus, pour refléter les changements de dynamique qui peuvent être induits par un contrôleur digital ou des interactions environnementales, le concept des systèmes hybrides, combinant des comportements continus et discrets, a été introduit. Depuis plus d'une décennie, l'analyse de ces systèmes hybrides et leur synthèse sont une part importante de la recherche dans la théorie des systèmes.

L'un des problèmes les plus étudiés concernant ces systèmes est l'analyses formelle de leur comportement. Parmi ces analyses, l'analyse d'accessibilité consiste à trouver les états qui peuvent être atteints par le système. Cette analyse est importante pour la vérification de sûreté et la synthèse de contrôleurs qui sont utilisées dans de nombreuses applications critiques telles que la planification de trajectoire sure pour les voitures autonomes ou l'évitement de collision en aéronautique, domaines où des erreurs de conception et des bugs peuvent mener à des résultats catastrophiques. L'un des facteurs contribuant

à la complexité de l'analyse formelle de ces systèmes vient de l'analyse exhaustive des comportements continus. De nombreux outils et algorithmes ont été développés pour analyser certaine dynamiques "simples", cependant l'analyse de dynamiques plus complexes tel que les dynamiques non-linéaires reste un challenge, en particulier en grandes dimensions.

Dans cette thèse, nous nous penchons sur le problème du calcul de l'ensemble atteignable de systèmes continus à dynamique non-linéaire en vue d'une extension vers l'analyse de systèmes hybrides. Nous nous concentrons en particulier sur les problèmes liés a la montée en dimension de nos méthodes.

Nous présentons dans la section suivante les approches existantes pour l'analyse des dynamiques continues des systèmes hybrides.

État de l'art

Comme mentionné précédemment, un important problème de vérification est celui de l'accessibilité. Étant donné un système dynamique et un ensemble d'états initiaux, ce problème de vérification consiste à trouver si une trajectoire peut atteindre certains états. De nombreuses méthodes ont été développées pour répondre à ce **problème d'accessibilité** utilisant différentes représentations d'ensemble, différents niveaux de complexité et différents niveaux de précision.

A cause de l'absence de solution analytique générale pour des équations différentielles, l'ensemble des états accessibles à un temps donné ne peut généralement qu'être approximé. Une première approche pour résoudre ce problème est le *calcul direct* basé sur la discrétisation du temps. Cela consiste à calculer l'ensemble contenant tous les états visités par les trajectoires possibles du système, dans un petit intervalle de temps et à recommencer la même opération pour l'intervalle suivant.

Des techniques d'approximation et d'abstraction de systèmes ont aussi été proposées pour réduire les problèmes d'accessibilité initiaux à des problèmes plus simples à analyser.

Une autre approche se concentre sur le calcul et la certification de barrières et d'invariants. Au lieu d'effectuer un calcul précis de l'ensemble accessible, leurs but sont d'obtenir des approximations suffisamment bonnes pour prouver des propriétés de sûreté données.

Nous présentons dans la suite de ce chapitre les méthodes utilisant ces approches.

Techniques de calcul direct

Les techniques dites de *calcul direct* de l'ensemble accessible nécessitent généralement d'établir un algorithme de calcul d'image basé sur des ensembles. Cette approche est basée sur les techniques d'intégration numérique qui permettent de résoudre des équations différentielles non-linéaires. Le but de cette intégration numérique est d'approximer une unique solution à chaque pas de temps basée sur une ou plusieurs solutions précédentes. Le calcul d'accessibilité requiert cependant de calculer l'ensemble de toutes les solutions possibles et donc d'être basé sur le calcul d'ensemble. Nous appellerons ce problème le *problème du calcul d'image*

Nous présentons dans ce qui suit, les techniques développées avec cette approche pour les systèmes dynamiques linéaires et non-linéaires.

Systèmes linéaires

Le calcul d'accessibilité pour les systèmes linéaires (éventuellement avec entrées) est un des problèmes les plus étudiés dans le cadre de la vérification des systèmes hybrides. De nombreux algorithmes ont été proposés.

Le calcul de l'ensemble accessible exact par calcul symbolique est proposé dans [65, 8] pour des systèmes linéaires avec certaines structures propres et des ensembles initiaux semi-algébriques.

Pour des systèmes non-autonomes linéaires, plus généraux, les entrées peuvent être traitées en utilisant la somme de Minkowski. La complexité de cette opération dépend principalement de la représentation des ensembles choisie pour décrire l'ensemble atteignable. Cette opération peut être effectuée efficacement sur les représentations des zonotopes et les fonctions de support. des méthodes de calcul d'accessibilité basées sur les zonotopes sont présentées dans [48, 1, 49]. Ces méthodes sont actuellement les plus à même de monter en dimension, cependant leur extension à l'analyse de systèmes hybrides est difficile à cause de l'opération d'intersection requise par la partie discrète de ces systèmes. Une généralisation récente de ces méthodes, aux représentations utilisant des fonctions de support a été présentée dans [66].

D'un autre coté, les entrées peuvent être traitées par optimisation en utilisant des polyhèdres convexes généraux, tel que dans les techniques présentées dans [96, 85, 28, 29]. Ces techniques sont basées sur le principe maximal du contrôle optimum [58]. D'autres techniques utilisent des classes particulières de polyhèdres tels que les hyper-rectangles [93] et les parallelotopes [60] pour approximer l'ensemble accessible. Ces techniques proposent un bon compromis entre précision et efficacité des opérations sur les ensembles.

De plus, une méthode basée sur les ellipsoides est présentée dans [61, 23]

pour les systèmes linéaires avec entrée et est étendue dans [62, 59] pour gérer les contraintes d'état.

Ces méthodes ont mené au développement de nombreux outils pour l'analyse d'accessibilité des systèmes hybrides avec des dynamiques linéaires tels que **d/dt** [10], **MPT** [64], **SpaceEx** [46], **Ellipsoidal Toolbox (ET)** [63].

Systèmes polynomiaux

Des techniques d'accessibilité basées sur le calcul d'image d'ensemble par des systèmes polynomiaux sont proposées dans [34, 39]. Elles sont inspirées par les techniques de modélisation issues du domaine de la création géométrique assistée par ordinateur et utilisent différentes représentations des polynômes tel que les simplexes de Bézier et les polynômes de Bernstein. Ces techniques peuvent être utilisées pour réduire un problème d'optimisation polynomial vers un problème d'optimisation linéaire plus simple à résoudre.

De plus, des méthodes d'analyse d'accessibilité pour les systèmes multi-affines sont proposées dans [19, 22]. Elles sont basées sur une propriété spéciale des fonctions multi-affines sur les sommets des hypercubes.

Systèmes non-linéaires

Une approche pour l'analyse des systèmes non-linéaires se base sur le suivi de l'évolution des faces de polyhèdres. La representation de ces faces pour les polyhèdre non-convexes est complexe et seul deux classes particulières de polyhèdre sont utilisées. L'une d'elle est la classe des polyhèdres orthogonaux [37] qui mena à la première version de l'outil **d/dt**; l'autre classe de polyhèdre est la projection en 2 dimensions de polyhèdre de plus grande dimension [50] qui est utilisée dans l'outil **Coho** [50]. Le désavantage de ces méthodes est leur limitation à des systèmes de faible dimension.

Une approche alternative est basée sur l'utilisation de courbes de niveaux pour représenter les états atteignables. Ceux-ci peuvent ainsi être déterminés par des solutions de viscosité pour des équations différentielles partielles de Hamilton-Jacobi-Isaacs ne dépendant pas du temps [72, 71, 70]. Cependant, la résolution numérique d'équations différentielles partielles est souvent basée sur la discrétisation de l'espace et est seulement efficace sur des systèmes de faible dimension.

Techniques basées sur l'abstraction de systèmes

L'abstraction de système est une approche différente et peut être employée pour analyser une large classe de systèmes non-linéaires. Ces techniques

reposent sur le calcul de systèmes approximés qui sont plus simples à analyser. Il est nécessaire que ces systèmes approximés assurent que la satisfaction de propriétés de sûreté par leur analyses implique celles des systèmes originaux.

Approximation de systèmes

L'approximation de systèmes a pour but de calculer un système de transition qui sur-approxime le système original et mène à l'utilisation de techniques de vérification qui ont fait leurs preuves.

La plupart de ces techniques sont basées sur l'utilisation de dynamiques pour lesquelles, il existe des algorithmes d'accessibilité efficaces pour chaque région de l'espace d'état. Ce type d'abstraction requiert de calculer une partition de l'espace d'état. Cette partition peut être calculée à priori, en utilisant des classes d'ensemble prédéfinis tel que les hypercubes ou les simplexes. Elle peut aussi être calculée à-la-volée pendant l'analyse. La technique présentée dans [12, 13], appelée *hybridization*, est basée sur le calcul de systèmes linéaires ou multi-affines par morceaux. Cette technique construit un automate hybride comme approximation d'une dynamique continue complexe. Sa variante, appelée *hybridization dynamique*, est présentée dans [36], elle utilise un partitionnement à-la-volée de l'espace en régions se chevauchant. Une autre méthode présentée dans [2] utilise des modèles linéarisés par morceau comme approximations.

Ces techniques ont été implémentées dans différents outils: **d/dt** [10] peut effectuer l'analyse d'accessibilité de systèmes non-linéaire en utilisant les techniques d'*hybridization*. **Hsolver** [82] est un outil permettant l'analyse de systèmes hybrides avec des dynamiques non-linéaires par l'utilisation de techniques de propagation des contraintes [82].

Interprétation abstraite

L'interprétation abstraite est une théorie de l'approximation discrète de la sémantique de systèmes sur ordinateur principalement appliquée à l'analyse statique et à la vérification de logiciel. Elle a été récemment étendue à l'analyse de systèmes hybrides en utilisant différents types de représentation pour les ensembles. L'outil **HybridFluctuat** [24] est le résultat de ces recherches. Il peut calculer l'ensemble atteignable d'un système non-linéaire en interaction avec un programme C. D'autres travaux basés sur l'interprétation abstraite utilisant des polyhèdres sont présenté dans [85, 35].

Abstraction de prédicats

L'abstraction de prédicats peut être utilisée pour améliorer l'analyse d'accessibilité. Elle consiste à analyser la valeur de vérité d'un ensemble de prédicats plutôt que de calculer des ensembles d'état continus [6, 5]. Des techniques de raffinement guidées par les contre-exemples [30] sont une autre application importante de cette approche. Ces techniques ont été utilisées dans de nombreux domaines tel que l'analyse de timing d'automates temporisés [7], la vérification de programmes C [17] et la vérification symbolique de modèles [31]. L'abstraction de prédicat est utilisée dans les outils **Bandera** [32], **SLAM** [18] et **Feaver** [52] pour l'analyse de programmes *C* et *Java*.

Calcul d'invariant

Le calcul d'invariant et les certificats de barrière sont des techniques alternatives à l'analyse d'accessibilité. L'idée derrière l'approche du certificat de barrière est de trouver un barrière séparant les bons et les mauvais états que nous pouvons certifier comme inviolables. Les barrières sont définies par des inégalités polynomiales et le signe de leurs dérivées directionnelles à leurs frontières est utilisé pour certifier son inviolabilité. Cette approche est proposée pour la vérification de sûreté dans [80, 95, 83] et utilisée dans [79, 81] pour le calcul d'invariants linéaires ou polynomiaux. Les certificats d'égalités polynomiaux [86, 88] sont une approche similaire qui correspond cependant plus à une certification de stabilité. Dans ce cas le certificat peut être établit par la résolution d'un système d'équation ou par le calcul de la base de Gröbner.

Les invariants différentiels sont une généralisation des barrières polynomiales. Ils peuvent être formés par des formules logiques contenant des équations et des inégalités polynomiales. Ils sont introduits dans [76] et utilisés dans des procédures automatisées de recherches d'invariants [77, 78]. Ces techniques ont été implémentées dans l'outil **KeYmaera** [75]. Cet outil de vérification peut calculer des invariants pour les systèmes hybrides avec des dynamiques non-linéaires.

Plan de la thèse

Le Chapitre 2 présente quelques notions basiques ainsi que les notations utilisées dans cette thèse. Nos contributions sont présentées dans les trois chapitres suivant. Le Chapitre 4 contient une section préliminaire qui ne concerne que le contenu de ce chapitre.

Le Chapitre 3 présente notre première contribution qui concerne la technique d'*hybridization dynamique*. Nous proposons une nouvelle méthode pour calculer les domaines d'approximations dans lesquels les dynamiques sont approximées. Nous utilisons la courbure du système dans la construction de domaine, pour obtenir un bon compromis entre précision et efficacité.

Dans le Chapitre 4 nous présentons nos contributions à l'analyse de systèmes polynomiaux. Nous proposons plusieurs améliorations d'une méthode utilisant l'expansion de Bernstein [34, 39].

Le Chapitre 5 présente une méthode de raffinement de la précision des analyses basées sur le calcul de polyhèdres pour les systèmes avec entrées. Cette méthode permet d'accroître la précision de l'analyse en ajoutant des contraintes redondantes aux polyhèdres. Nous montrons les possibilités d'utilisation de cette méthode sur des systèmes à grande dimension.

Pour chacun de ces chapitres, nous finissons avec des résultats expérimentaux obtenus en utilisant un nouvel outil qui est présenté dans le Chapitre 6.

Le Chapitre 3, le Chapitre 4 et le Chapitre 5 peuvent être lus séparément, cependant, nous conseillons au lecteur de lire le Chapitre 2 en premier, pour comprendre les notations utilisées dans cette thèse.

Chapter 1

Introduction

This thesis presents a number of contributions concerning the formal verification of nonlinear dynamical systems. In this introduction, we first present our motivations and the problems that we address. We then provide a survey of the existing approaches to the formal verification of dynamical systems and we finish by presenting the outlines of the thesis.

1.1 Motivations

Dynamical systems are a mathematical concept where a set of fixed rules describes the temporal evolution of the state of a system in a set called the state space. The number of variables of the system corresponds to the dimension of the state space; at any time the behavior of the system is given by the values of the variables. Continuous dynamical systems are widely used to model physical phenomena such as chemical reactions and biological phenomena in engineering applications. In addition, to capture the dynamics changes that can be induced by a digital controller or environmental interactions, the concept of hybrid systems, which combine continuous and discrete behaviors, has been introduced. Since a few decades, hybrid systems analysis and synthesis are an important part of the research in systems theory.

One of the most studied problems in hybrid systems is the formal analysis of their behaviors. One type of formal analysis, namely reachability analysis, involves finding the states that can be reached by the system. This analysis is important for safety verification and controller synthesis which are used in many critical applications, such as safe trajectory planification for autonomous cars or plane collision avoidance, where errors and bugs can lead to disastrous results. One factor contributing to the complexity of formal analysis of hybrid systems comes from the exhaustive analysis of their

continuous behaviors. While many efficient tools and algorithms have been developed to analyze constant-derivative and linear continuous dynamics, the analysis of non-linear dynamics remains a challenge.

In this thesis, we address the problem of computing reachable states of a nonlinear continuous system in view of an extension to hybrid systems analysis. We also focus on the problem of designing scalable methods to perform analysis on larger systems.

We continue in the next section with a review of the existing approaches to the analysis of continuous dynamics of hybrid systems.

1.2 State of the art

As mentioned earlier, an important verification problem is the reachability problem. Roughly speaking, it can be stated as follows. Given a dynamical system and a set of possible initial states, we want to know whether if there is a trajectory that reaches a given state. To address this *reachability problem*, a variety of methods have been developed to deal with different types of dynamics. They use different set representations and have different complexity and accuracy level.

Due to the lack of analytic solution of general differential equations, the reachable set at a given time is often approximated. A common approach to reachable set approximation is a *direct computation* based on a time discretization, that is computing the set containing all possible states visited by the trajectories for a small time interval, and then continuing for the next interval.

Systems approximation and abstraction techniques have also been proposed to reduce the reachability analysis problem to a problem on a simpler system to analyze.

Another approach focuses on the computation of *invariant and barrier certificate*. Instead of performing precise computation of the reachable sets, its goal is to obtain an approximation sufficiently good to prove a given safety property.

We now review the analysis methods developed using these approaches.

1.2.1 Direct reachability computation techniques

Direct reachability computation techniques usually require establishing a numerical scheme and realizing this scheme on sets. This approach is inspired by numerical integration which is a common method to solve nonlinear differential equations. The goal of traditional numerical integration

is to approximate a single solution at each time step based on the solution at one or several previous steps. Reachability computation however involves the set of all possible solutions and thus requires performing the scheme on sets, which often reduces to the problem of computing the image of a set by a function; from now on we call this problem the **image computation problem**.

We proceed with a discussion on the techniques developed along this line for linear and polynomial dynamical systems.

Linear systems

Reachable set computation for linear dynamical systems (possibly with input) has been one of the most studied problems of hybrid systems verification. Many efficient algorithms have been proposed.

Exact computation of reachable sets by symbolic computation is proposed in [65, 8] for linear systems with special eigenstructures and semi-algebraic initial sets.

For general non-autonomous linear systems, the input can be treated using the Minkowski sum. The complexity of this operation mainly depends on the set representation that is used to describe the reachable set. The Minkowski sum can be performed efficiently on the zonotope and support function representations. Zonotope-based reachability computation methods are presented in [48, 1, 49]. These techniques are currently the most scalable in dimension, however their extension to hybrid systems analysis is hard because of the difficulties in computing the intersection operation, needed to deal with discrete transitions of hybrid systems. A recent generalization of this method to the support functions has been presented in [66].

On the other hand, input can be treated by optimization using general convex polyhedra, such as in the techniques presented in [96, 85, 28, 29]. These techniques are based on the Maximum Principle in optimal control [58]. Other techniques use particular classes of polytopes like hyper-rectangles [93] and parallelotopes [60] to approximate the reachable sets. These techniques provide a good trade-off between precision and efficiency of set operations.

In addition, an ellipsoid-based method is presented in [61, 23] for linear systems with input and is extended in [62, 59] to treat state constraints.

The above results led to the development of a number of tools for reachability analysis of hybrid systems with linear dynamics such as **d/dt** [10], **MPT** [64], **SpaceEx** [46], **Ellipsoidal Toolbox (ET)** [63].

Polynomial systems

Reachability techniques based on image computation for polynomial systems are proposed in [34, 39]. These techniques are inspired by modeling techniques from Computer Aided Geometric Design (CAGD) and use different polynomial representations, such as the Bézier simplex, the box splines and the Bernstein polynomials, to reduce a polynomial optimization problem to a linear one which can be solved more efficiently.

In addition, reachability analysis methods for multi-affine systems, which are a particular class of polynomials, are proposed [19, 22]. They are based on a special property of the multi-affine function values on the rectangle vertices.

Nonlinear systems

One approach to reachability analysis of non-linear systems is based on using polyhedra and tracking the evolution of their faces under nonlinear dynamics. The facet representation of non-convex polyhedra is hard and thus two particular classes of non-convex polyhedra are used. One is the class of orthogonal polyhedra [37] which led to the first version of the tool **d/dt**; and the other is based on the two-dimensional projections of high-dimensional polyhedra [50] which led to the tool **Coho** [50]. The drawback of these methods is their high complexity in handling non-convex polyhedra and they are thus limited to low-dimensional systems.

An alternative approach is based on the use of level sets to represent the reachable sets which can be determined using the viscosity solution of a time dependent Hamilton-Jacobi-Isaacs partial differential equation [72, 71, 70]. However, numerical resolution of partial differential equations is often based on a spatial discretization and is efficient only for low dimensional systems.

1.2.2 Techniques based on systems approximation

Systems approximation is another approach to analyze a large class of nonlinear systems. These techniques rely on the computation of an approximate system which is easier to analyze. An important requirement for this approximation is conservativeness, which ensures that the satisfaction of a safety property by the approximate system implies that this property is also satisfied by the original system.

Systems approximation

Abstraction, aims to compute a transition system which is an over-approximation of the original systems and is amenable to verification by well-established verification techniques.

Most techniques for automatically constructing approximate systems are based on the use of simpler dynamics (for which there exist efficient reachability algorithms) in each region of the state space. This kind of abstraction requires computing a partition of the state space. This partition can be computed a-priori using predefined classes of sets such as boxes or simplices, or can be performed on-the-fly during the analysis. The technique proposed in [12, 13], called *hybridization*, is based on the computation of piecewise linear and multi-affine systems. This technique constructs a hybrid automaton as approximation of a complex continuous dynamical system. Its variant, called *dynamic hybridization*, where the partition is not fixed and created dynamically, is presented in [36]. Another method presented in [2] uses piecewise linearized models as approximations.

Systems approximation techniques have been implemented in different tools: the tool **d/dt** [10] can perform the reachability analysis of continuous non-linear dynamics using the *hybridization* techniques. **Hsolver** [82] is a software package for the safety verification of hybrid systems. It supports hybrid systems with nonlinear dynamics and nonlinear reset functions by using constraint propagation based abstraction refinement method [82].

Abstract interpretation

Abstract interpretation is a theory of discrete approximation of the semantics of computer systems mainly applied to the static analysis and verification of software. It has been recently extended to hybrid systems analysis using different set representations (such as polyhedra and zonotopes). The tool **HybridFluctuat** [24], the result of this research, is a static analyzer which can compute the reachable set of nonlinear continuous dynamics in interaction with a C program. Besides, other works based on abstract interpretation using template polyhedra are presented in [85, 35].

Predicates abstraction

Predicate abstraction can be used to enhance reachability analysis. It involves tracking the truth values of a set of predicates instead of tracking the continuous states [6, 5]. Counter-example guided abstraction refinement techniques [30] are another important result. These techniques have been

used in many fields such as timing analysis of timed automata [7], verification of C programs [17], and symbolic model checking [31]. The predicate abstraction approach is used in the tool **Bandera** [32], **SLAM** [18] and **Feaver** [52] for analysis of *C* and *Java* programs.

1.2.3 Invariant computation

Invariant and barrier certificate based verification techniques are an alternative to reachability analysis. The basic idea behind the barrier certificate approach is to find a barrier separating good and bad states that we can show to be impenetrable by the dynamics of the system. The barrier is defined using a polynomial inequality and the sign of a directional derivative at the barrier boundary is used to certify the impenetrability condition.

This approach is proposed for safety verification in [80, 95, 83] and used in [79, 81] for linear and polynomial invariant computation for linear and non-linear systems. Polynomial equality certificates [86, 88] are a similar approach which corresponds more to stability certificates than separation between good and bad states. In this case the certificate can be established by linear equation solving or Gröbner basis computation.

Differential invariants are a generalization of polynomial barriers. They can be formed by general logical formula containing polynomial equations and polynomial inequalities. They are introduced in [76] and later used in automatic procedures that search for differential invariants [77, 78]. These techniques have been implemented in the tool **KeYmaera** [75]. This hybrid verification tool can compute invariants for hybrid systems. This tool supports systems with nonlinear dynamics and non-deterministic discrete and continuous input.

1.3 Outline of the thesis

Chapter 2 is devoted to the basic notions and notations that are used in this thesis. Our contributions are presented in the three chapters that follows. Chapter 4 contains a preliminary section which concerns only the content presented therein.

Chapter 3 presents our first contribution which concerns the *dynamic hybridization* technique. We propose a new way to compute approximation domains in which the dynamics are approximated more accurately. In addition, we make use of curvature to develop a method for domain construction which provides a good trade-off between precision and domain size.

In Chapter 4 we describe our contributions to formal analysis of polynomial dynamical systems. We propose a number of enhancements for the reachability method which uses the Bernstein expansion [34, 39].

Chapter 5 presents a refinement method for polytope-based reachability analysis of systems with input. This method increases the accuracy of the reachable set computation by using redundant constraints. We show the applicability of this method for systems in high dimensions.

Each of these chapters ends with the experimental results obtained using a new prototype tool which is presented in Chapter 6.

Chapter 3, Chapter 4 and Chapter 5 can be read independently; however, we recommend the reader to read Chapter 2 first to understand the notations used in the thesis.

Chapter 2

Preliminaries

Résumé: *Dans ce chapitre, nous présentons quelques notions basiques sur l'analyse d'accessibilité, les représentations d'ensemble ainsi que les notations utilisées dans cette thèse.*

In this chapter, we present some preliminary notions that will be used in the rest of this thesis. In the following, we present some preliminaries on the continuous dynamical systems and hybrid systems representation. In a second section, we present some definitions about the set representations used in this thesis. At last, we present the notations which will be used in the following chapters.

2.1 Dynamical systems

In this section we present some generalities about the continuous and hybrid dynamical systems.

2.1.1 Continuous dynamical systems

Continuous dynamical systems are systems whose states evolve according to an independent variable often given as *time*. In this thesis we use the set of positive real numbers $\mathbb{R}^+$ including 0 as the *time domains*.

We can differentiate continuous autonomous systems and continuous systems with inputs. Continuous autonomous systems are systems which do not depend on an input. They can be defined as follows.

Definition 1 (Continuous autonomous system)**.** *Continuous autonomous systems are systems whose behaviors are governed by ordinary autonomous differential equations:*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ *represents the state of the system at time* t*,* $f : \mathbb{R}^n \to \mathbb{R}^n$ *is a function representing the system dynamics.*

The behavior of a continuous dynamical system is characterized by the solutions to the initial-value problem of its differential equation.

We assume that f is globally Lipschitz continuous in $\mathbf{x}$, which guarantees that there exists a unique solution to (2.1) for every initial condition in $\mathbb{R}^n$.

In many cases, to model physical phenomena, we need to add an input variable which can reflect, for example, perturbations, external controls or approximation errors.

Definition 2 (Continuous system with input)**.** *A continuous system with input is described by*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.2}$$

where $\mathbf{u}(\cdot) : \mathbb{R}^+ \to \mathcal{U}_\mu$ *represents the continuous input variables included in the set of admissible input values* $\mathcal{U}_\mu$*.*

In this thesis we assume that the sets $\mathcal{U}_\mu$ of all possible input values are compact and convex.

For simplicity we will sometimes use the following notations to represent continuous systems:

$$\dot{\mathbf{x}} = f(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^n.$$

and

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),\ \mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathcal{U} \subset \mathbb{R}^n.$$

Definition 3 (Trajectory)**.** *For a continuous autonomous system, a trajectory starting from a state in the state space* $\mathbf{p} \in \mathbb{R}^n$ *is a function* $\Phi^f(\mathbf{p}, t) : \mathbb{R}^+ \to \mathbb{R}^n$ *such that* $\Phi^f(\mathbf{p}, t)$ *is the solution of the system (2.1). For systems with input, the system trajectory starting from a state* $\mathbf{p} \in \mathbb{R}^n$ *under the input* $\mathbf{u}(\cdot) \in \mathcal{U}$ *is a function* $\Phi^f(\mathbf{p}, t, \mathbf{u}) : \mathbb{R}^+ \to \mathbb{R}^n$ *such that* $\Phi^f(\mathbf{p}, t, \mathbf{u})$ *is the solution of the system (2.2).*

Trajectories can be computed by solving analytically the differential equations or approximated by using numerical analysis methods, such as Euler or Runge-Kutta. If a state $\mathbf{q}$ belongs to a trajectory $\Phi^f(\mathbf{p}, t)$ starting from $\mathbf{p}$, we say that $\mathbf{q}$ is reachable from $\mathbf{p}$.

For deterministic systems, a single trajectory can represent all the reachable states from an initial state. However, in the presence of an input or

when the initial state is not known exactly and is given in a continuous set, the number of possible trajectories could be infinite. In this case, formal analysis often requires set-based computations.

In many modern engineering applications, the modeling process requires considering discrete events which change the continuous dynamics. For example, those discrete behaviors can be introduced in mechanical systems to model collisions. These models are called *hybrid systems* because they exhibit both continuous and discrete behaviors. In the following, we describe a well-established formalism used to model these systems.

2.1.2 Hybrid systems

Hybrid systems are systems which combine both continuous and discrete dynamics. Continuous behaviors usually represent the evolution of physical variables such as spatial position, temperature or proteinic concentration. The discrete behaviors represent changes in dynamics which can be induced among other things by switching control laws or environment responses.

A classical formalism to describe hybrid systems is given by the *hybrid automata* [68, 4]. The discrete behaviors are modeled by an automaton where in each discrete state (mode), the behaviors of continuous variables are specified using differential equations.

Definition 4 (Hybrid systems)**.** *A* hybrid automaton *is a tuple of the following components:*

- *A set of* **locations** *representing the discrete states.*
- *A set of* ***continuous variables****.*
- *A set of* **invariants** *(which are subsets of the state space) corresponding to the set of admissible values for the continuous variables in a given location.*
- *For each location, a* **continuous vector field** *specifying the evolution of the continuous variables (usually described by a differential equation).*
- *A set of discrete transitions between locations which specify the discrete behavior of the system. Each transition is associated with a* ***guard*** *represented by a subset of the state space. A* ***reset map*** *is associated with each transition to specify the change in the continuous variables after a transition.*

A classical example which illustrates hybrid systems is a bouncing ball, a physical system with impact. In this example, the ball is dropped from an initial height and bounces off the ground, dissipating its energy with each bounce. The ball exhibits continuous dynamics between each bounce; however, as the ball impacts the ground, its velocity undergoes a discrete change modeling an inelastic collision.

In the next section, we present the notion of reachable sets and discuss some set representations used in the next chapters.

2.2 Reachable sets

We consider a continuous dynamical system with input

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

where $\mathbf{x} \in \mathbb{R}^n$ is a state variable and $\mathbf{u} \in \mathcal{U}$ a possible input. The initial states are given by a compact set $\mathcal{P} \subset \mathbb{R}^n$. The reachable sets can be defined as follows:

Definition 5 (Reachable sets)**.** *A reachable set from an initial set $\mathcal{P} \subset \mathbb{R}^n$ under the autonomous dynamics f at time t, denoted $Reach(\mathcal{P}, t)$, is defined as follows:*

$$Reach(\mathcal{P}, t) = \{\mathbf{x} \in \mathbb{R}^n, \mid \exists \mathbf{y} \in \mathcal{P}, \mid \mathbf{x} = \Phi^f(\mathbf{y}, t)\}$$

In the case of system with input, the reachable set at time t from an initial set $\mathcal{P}$ under the dynamics f and the input $\mathbf{u}$, denoted $Reach(\mathcal{P}, t, \mathbf{u})$, is defined as follows:

$$Reach(\mathcal{P}, t, \mathbf{u}) = \{\mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathcal{U} \mid \exists \mathbf{y} \in \mathcal{P} \mid \mathbf{x} = \Phi^f(\mathbf{y}, t, \mathbf{u})\}$$

In a similar way, we denote by $Reach^{[0,t]}(\mathcal{P})$ the set containing all the reachable states starting from a state in $\mathcal{P}$ until a time t.

In the following, we present some preliminaries about set representations.

2.2.1 Set representations

In this thesis we base our work on convex polyhedra as a representation of sets. In addition, we consider only bounded sets. We begin by the notation for the box representation, which is frequently used in the thesis.

Boxes

Boxes (also called hyper-rectangles or intervals products) are one of the simplest sub-class of convex polyhedra. They can be represented by two vectors $\underline{\mathbf{b}}, \overline{\mathbf{b}} \in \mathbb{R}^n$ representing the lower and upper bounds of the intervals for each dimension. The resulting set is obtained by the product of these intervals and is symmetrical according to its centroid.

Definition 6 (Box)**.** *A box $\mathcal{B} \subset \mathbb{R}^n$ can be represented by its lower and upper bounds $\underline{\mathbf{b}} \in \mathbb{R}^n$ and $\overline{\mathbf{b}} \in \mathbb{R}^n$ such that*

$$\mathcal{B} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \underline{b}_i \leq \mathbf{x} \leq \overline{b}_i, i \in \{1 \cdots n\} \right\}$$

where $\underline{b}_i$ and $\overline{b}_i$ represent the i^{th} elements of $\underline{\mathbf{b}}$ and $\overline{\mathbf{b}}$. This representation is also known as interval product and can be expressed as

$$\mathcal{B} = [\underline{b}_1, \overline{b}_1] \times [\underline{b}_2, \overline{b}_2] \times \cdots \times [\underline{b}_n, \overline{b}_n].$$

Figure 2.1 gives an illustration of a 2-dimensional box in the interval $[2, 4] \times [1, 4]$, represented by two vectors $\underline{\mathbf{b}} = (2, 1)$ and $\overline{\mathbf{b}} = (4, 4)$.

For a bounded set $\mathcal{A}$ we give the definition of its *minimal bounding box* that we denote by $\square(\mathcal{A})$.

Definition 7 (Minimal bounding box)**.** *A box $\mathcal{B} \subset \mathbb{R}^n$ is the* minimum bounding box *of a set $\mathcal{A} \subset \mathbb{R}^n$, denoted by $\square(\mathcal{A})$, if*

$$\square(\mathcal{A}) = [\underline{b}_1, \overline{b}_1] \times [\underline{b}_2, \overline{b}_2] \times \cdots \times [\underline{b}_n, \overline{b}_n]$$

where $\underline{b}_i = inf\{x_i \mid \mathbf{x} \in \mathcal{A}\}$ and $\overline{b}_i = sup\{x_i \mid \mathbf{x} \in \mathcal{A}\}$.

In the remainder of the thesis, we will use the term *bounding box* as an abbreviation of the *minimal bounding box*.

Simplex

A simplex is a generalization of the notion of a triangle or tetrahedron to arbitrary dimension.

Definition 8 (Simplex)**.** *A simplex Δ is a n-dimensional polytope which is the convex hull of its n + 1 vertices $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}\}$.*

$$\Delta = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \sum_{\mathbf{v} \in V} \alpha^{\mathbf{v}} \mathbf{v}, \sum_{\mathbf{v} \in V} \alpha^{\mathbf{v}} = 1, \ \forall \mathbf{v} \in V \ \alpha^{\mathbf{v}} \geq 0 \right\}$$

Simplices can be used to compute affine interpolations of non-linear systems, this will be discussed in Chapter 3.

Figure 2.2 gives an illustration of a 2-dimensional simplex represented by 3 vertices.

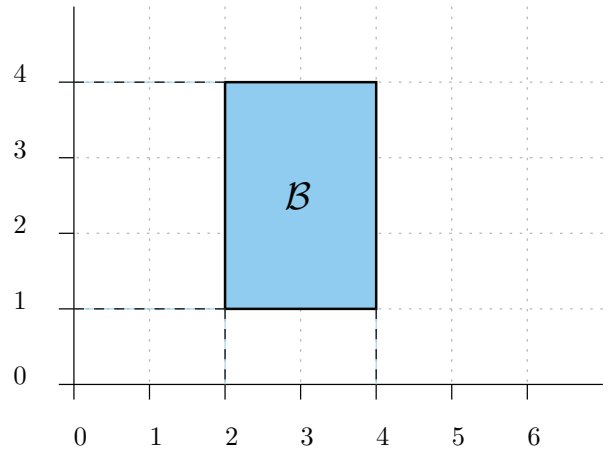

Figure 2.1: A box represented by two vectors $\underline{\mathbf{b}} = (2, 1)$ and $\overline{\mathbf{b}} = (4, 4)$.

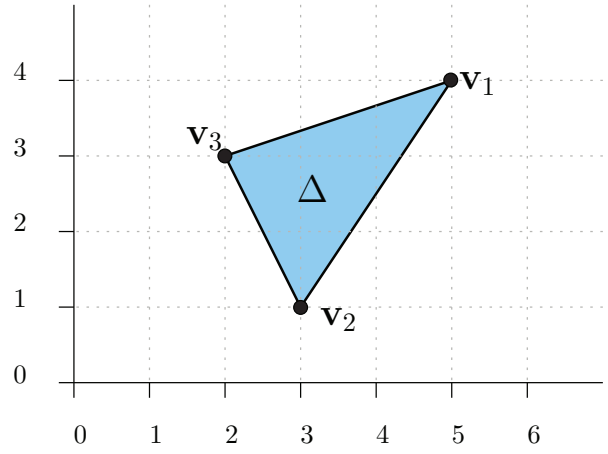

Figure 2.2: A 2-dimensional simplex represented by the vertices $\mathbf{v}_1 = (5, 4)$, $\mathbf{v}_2 = (3, 1)$ and $\mathbf{v}_3 = (2, 3)$.

Polytopes

Polytopes are a generalization of convex polyhedra. They can be defined by half-spaces intersection, or as the convex hull of vertices. [1]

We denote a **polytope represented by a set of half-spaces $\mathcal{H}$** by $\mathcal{H}$Polytope or constraint polytope and we define them as follows:

Definition 9 ($\mathcal{H}$Polytope)**.** *Half-space based polytopes, also called constraint-based polytopes, are convex polyhetopes represented by the intersection of a set H of half-spaces. Each half-space $\mathcal{H}$ of H corresponds to a linear inequality such that*

$$\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a} \cdot \mathbf{x} \leq b\}$$

where $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. We denote by normal vector *the vector $\mathbf{a}$ because it is always orthogonal to the boundary of the half-space $\mathcal{H}$. A $\mathcal{H}$Polytope $\mathcal{P}$ represented by a set H of half-spaces can be defined by*

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

Additionally we can use the following expression:

$$\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix whose lines correspond to normal vectors and $\mathbf{b} \in \mathbb{R}^m$ is called a polyhedral coefficient vector.

This representation allows to compute intersection and affine transformation easily. The intersection of a $\mathcal{H}$Polytope $\mathcal{P}$ represented by a list of constraints H with a half-space $\mathcal{H}$ can be expressed by adding $\mathcal{H}$ to the list H. The affine transformation by a matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ is given by

$$\mathbf{T}\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n \mid (\mathbf{T}^{-1})^T \mathbf{a}^{\mathcal{H}} \mathbf{x} \leq b^{\mathcal{H}},\ \forall \mathcal{H} \in H\}.$$

The convex union of two convex polyhedra can be efficiently computed in some cases with sparse matrix $\mathbf{A}$ [91], however in the general case it often requires computing or approximating vertex representations.

Another representation uses vertices in the state space. We call a **polytope represented by a list of vertices** V by $\mathcal{V}$Polytope, defined as follows:

Definition 10 ($\mathcal{V}$Polytope)**.** *Vertex-based polytopes can be represented by the convex hull of a list of vertices V. A $\mathcal{V}$Polytope $\mathcal{P}$ can be expressed as:*

$$\mathcal{V}\textit{Polytope} = \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} = \sum_{\mathbf{v} \in V} \alpha^{\mathbf{v}} \mathbf{v}, \sum_{\mathbf{v} \in V} \alpha^{\mathbf{v}} = 1,\ \forall \mathbf{v} \in V\ \alpha^{\mathbf{v}} \geq 0 \right\}.$$

[1] Vertices are points in the n-dimensional space which is usually used to represent a set boundary.

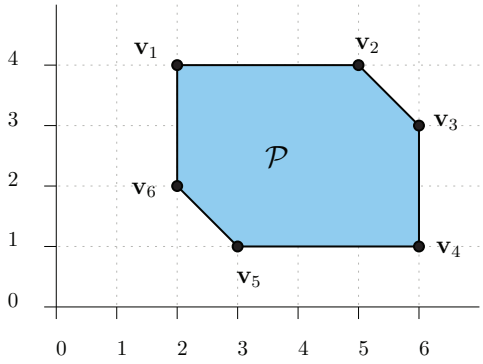

Figure 2.3: A polytope represented by the convex hull of the set of its vertices $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_6\}$.

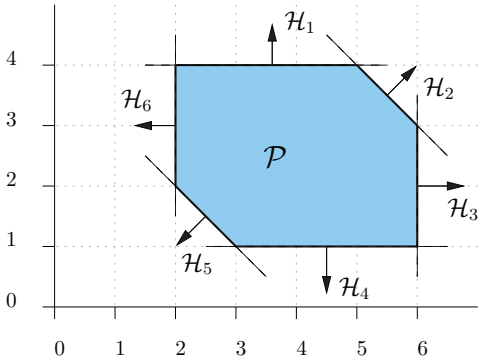

Figure 2.4: The polytope of Figure 2.3 represented by the intersection of its half-spaces $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_6$.

This representation is suitable for the affine transformation and the convex union. The intersection with half-spaces can be efficiently realized using Binary Partitioning Trees [94].

Figure 2.3 and Figure 2.4 illustrate these two representations for a 2-dimensional polytope with 6 facets. The dotted lines in Figure 2.4 represent the half-spaces and the arrows their normals.

Other representations such as zonotopes or template support functions can be used for special kinds of polytopes, however this thesis does not use them directly.

We end this chapter by a summary of the most common notations that we will use in this thesis.

2.3 General Notation

The following tables present the notation used in this document.

Variables	
$a, b, \ldots, z$	scalars
n	number of continuous variables, also called the system dimension
$x_1, \ldots, x_n$	components of the vector $\mathbf{x}$
$\mathbf{u}$	input to a dynamical system
Vectors and matrices	
$\mathbf{a}, \mathbf{b}, \ldots, \mathbf{z}$	vectors
$\mathbf{A}, \mathbf{B}, \ldots, \mathbf{Z}$	matrices
a_i	i^{th} element of the vector $\mathbf{a}$
$A_{i,j}$	j^{th} element of the i^{th} column of the matrix $\mathbf{A}$
$\|\|\mathbf{x}\|\|$	Euclidian norm of the vector $\mathbf{x}$
$\|\|\mathbf{x}\|\|_\infty$	infinity norm of the vector $\mathbf{x}$
$\mathbf{a} \cdot \mathbf{b}$	scalar product of the vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathbf{AB}$	multiplication of the matrices $\mathbf{A}$ and $\mathbf{B}$
Sets	
$\mathbb{R}$	set of real numbers
$[a, b]$	continuous interval between the scalar a and b
$\mathbb{R}^n$	state space
$\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$	sets
$\mathcal{U}$	input set
$\mathcal{A} \bigoplus \mathcal{B}$	Minkowski sum of the sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \ominus \mathcal{B}$	Minkowski difference of the sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \bigcap \mathcal{B}$	intersection between the sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{H}$Polytope	a polytope based on constraints
$\mathcal{V}$Polytope	a polytope based on vertices
Continuous system	
$f(\cdot), g(\cdot), l(\cdot)$	vector of functions
$f_i(\cdot)$	i^{th} component of the vector of function $f()$
$Reach(\mathcal{P}, t)$	reachable set from $\mathcal{P}$ at time t
$Reach(\mathcal{P}, [0, t])$	reachable set from $\mathcal{P}$ during the time interval $[0, t]$
$\Phi^f(t, \mathbf{p})$	the trajectory starting from the state $\mathbf{p}$ of the system f
$\Phi^f(t, \mathbf{p}, u(\cdot))$	the trajectory starting from the state $\mathbf{p}$ of the system f under the input $\mathbf{u}(\cdot)$

Chapter 3

Dynamic hybridization using curvature over simplicial domains

Résumé: *Nous présentons dans ce chapitre la première contribution de cette thèse qui concerne la technique d'hybridization dynamique qui peut être utilisée pour analyser un grand nombre de systèmes non-linéaires. Nous proposons une nouvelle formulation de l'erreur d'approximation basée sur la courbure du système. Nous présentons ensuite une méthode de construction de domaine d'approximation qui améliore l'efficacité de cette méthode. Nous prouvons, par la suite, l'optimalité de cette méthode de construction de domaine d'approximation pour certaines catégories de systèmes quadratiques. Nous finissons par présenter quelques résultats expérimentaux.*

The problem that we address in this chapter concerns the computation of approximate system for reachability analysis of non-linear dynamical systems, using the dynamic hybridization approach. We propose a new method for constructing linearization domains.

We consider the following domain construction problem:

- $\mathbb{R}^n \subseteq \mathbb{R}^n$ a state space.

- $\mathcal{P} \subseteq \mathbb{R}^n$ a set representing the initial states.
- An autonomous continuous system described by the following differential equations:
$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \tag{3.1}$$
where $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^n$, $f(\mathbf{x}(t)) : \mathbb{R}^n \to \mathbb{R}^n$ and $t \in \mathbb{R}^+$. We can write the equation (3.1) as follows
$$\begin{aligned} \dot{x}_1(t) &= f_1(\mathbf{x}(t)), \\ \dot{x}_2(t) &= f_2(\mathbf{x}(t)), \\ &\dots, \\ \dot{x}_n(t) &= f_n(\mathbf{x}(t)) \end{aligned}$$

We want to construct a domain $\Delta \subseteq \mathbb{R}^n$ containing the initial set where the dynamics f is approximated by a simpler one. To take into account the distance between the exact reachable set and the reachable set computed using this approximation, an input function included in a set $\mathcal{U}$ is added to the approximate system.

We are particularly interested in three criteria which are partially conflicting:

- The approximation error is smaller than a desired error bound.
- The approximate system can be efficiently computed.
- The system evolution remains in Δ as long as possible because when the system is outside the domain, the approximation error bound may be no longer valid.

We start by recalling the hybridizations techniques. We then describe a new method to create simplicial hybridization domains which provides a good trade-off between these criteria. This method can be used for the hybridization technique based on affine interpolation over simplices. We will also present an estimation of the interpolation error for a large class of nonlinear systems. The next sections explain how we integrate these new theoretical results in the reachability analysis and we present some experimental results.

3.1 Hybridization

These techniques are based on a partition of the state space which can be computed in a precomputation step or constructed dynamically during the analysis.

Static hybridization

This approach initially presented in [13] relies on the creation of a composite vector field with linear or multi-affine dynamics from a precomputed partitioning mesh. This composite vector field defines an abstract model for the original nonlinear system. This partitioning mesh can be defined as follows.

Definition 11 (mesh)**.** *A mesh of a set* $\mathbb{R}^n$ *is a collection* $\mathcal{M} = \{\mathcal{M}^l,\ l \in L\}$ *where* L *is a set of discrete location such that*

- $\bigcup_{l \in L} \mathcal{M}^l = \mathbb{R}^n$.
- *For all* $l \neq l' \in L$, $\mathcal{M}^l \cap \mathcal{M}^{l'} = bound(\mathcal{M}^l) \cap bound(\mathcal{M}^{l'})$ *where* $bound(\mathcal{M}^l)$ *denotes the boundary of the set* $\mathcal{M}^l$.

In the case where all the set $\mathcal{M}^l$ *are simplices, the mesh is called a simplicial mesh.*

The first condition in Definition 11 ensures the coverage of the entire state space; the second one indicates that the elements of the mesh have disjoint interiors.

The composite vector field can then be defined as follow.

Definition 12 (composite vector field)**.** *A composite vector field on a domain* $\mathbb{R}^n$ *is a collection*

$$\mathcal{F} = \{(\mathcal{M}^l, \mathcal{U}^l, f^l),\ l \in L\}$$

where $\mathcal{M} = \{\mathcal{M}^l,\ l \in L\}$ *is a mesh of* $\mathbb{R}^n$ *and for all* $l \in L$, $f^l : \mathbb{R}^n \to \mathbb{R}^n$ *is the dynamics of the element* $\mathcal{M}^l$ *and* $\mathcal{U}^l$ *is the set of admissible inputs. This set consists of continuous functions of the form* $u :\ \mathbb{R}^+ \to \mathbb{R}^n$.

A composite vector field which conservatively approximates a nonlinear system is defined as follow.

Definition 13 (conservative approximation using composite vector field)**.** *A composite vector field* $\mathcal{F} = \{(\mathcal{M}^l, \mathcal{U}^l, f^l),\ l \in L\}$ *approximates conservatively a nonlinear vector field* $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$,

$$\forall l \in L,\ \forall \mathbf{x} \in \mathcal{M}^l,\ \exists \mathbf{u} \in \mathcal{U}^l \ s.t.\ f(\mathbf{x}) = f^l(\mathbf{x}) + \mathbf{u}$$

This composite vector field corresponds to a hybrid system.

Definition 14 (hybrid system from composite vector field)**.** *The hybrid system corresponding to the composite vector field* $\mathcal{F} = \{(\mathcal{M}^l, \mathcal{U}^l, f^l),\ l \in L\}$ *is* $H(\mathcal{F}) = (L, \mathbf{x}, I, F, E)$ *where*

- *L is the set of discrete locations,*
- *$\mathbf{x} = (x_1, \ldots, x_n)$ are the continuous variables,*
- *I is the set of location invariants each of which is the domain of an element of $\mathcal{M}$.*
- *F a set of vector fields such that for each l the vector field inside $\mathcal{M}^l$ is defined by $F^l = f_l(\mathbf{x}, \mathbf{u})$ with $\mathbf{u}(\cdot) \in \mathcal{U}^l$.*
- *E is the set of discrete transitions such that*

$$\forall e = (l, l') \in EG^e = bound(\mathcal{M}^l) \cap bound(\mathcal{M}^{l'})$$

and the reset function R^e is the identity functions.

Figures 3.1 illustrates this definition. The Figure 3.1(a) represents the initial set of the reachability problem, $f(\mathbf{x})$ is a nonlinear system. In Figure 3.1(b) the state space is partitioned with a mesh over the state space where the dynamics f is conservatively approximated. Figure 3.1(c) shows the reachable set computed inside the first element of the mesh; the intersection with its boundary is then computed. Starting from the intersection the reachable sets are computed in the other elements, as illustrated by Figure 3.1(d).

Dynamic hybridization

This technique presented in [36] is based on a dynamically constructed mesh.

Initially a hybridization domain $\mathcal{M}^0$ (where the dynamics is approximated) is built around the initial set $\mathcal{P}^0$. An approximate system is then computed and the reachability analysis is performed on this system until the reachable set $\mathcal{P}^k$ is not included in $\mathcal{M}^0$.

Once the reachable set intersects with the neighboring elements of $\mathcal{M}^0$, the set $\mathcal{P}^k$ is removed and a new approximation domain is created around the previously computed set $\mathcal{P}^{k-1}$ and the analysis continues from $\mathcal{P}^{k-1}$ using the new approximate system, as illustrated in Figure 3.2(d).

Note that according to a desired precision, the size of the domains is bounded. In some cases, the reachable set $\mathcal{P}^k$ can be too large and its inclusion in a domain can be impossible. A simple method to deal with this problem is to split the set and create two approximation domains. Figures 3.2 illustrates this solution. The Figure 3.2(a) shows the initial set $\mathcal{P}^0$ of the reachability problem, $f(\mathbf{x})$ is a nonlinear dynamics. In Figure 3.2(b) a domain is created around the initial set where the dynamics is conservatively approximated. The reachability computation can then be carried out

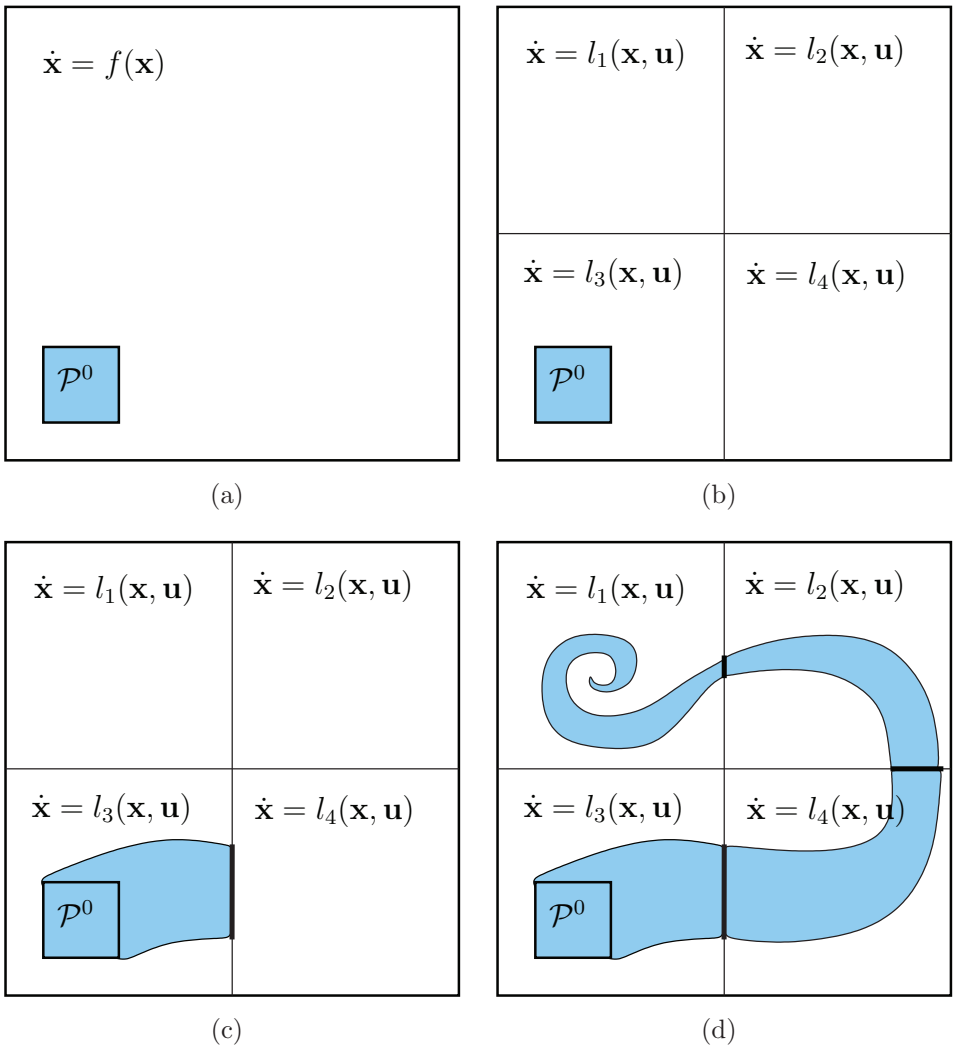

Figure 3.1: Illustration of the hybridization technique.

until intersection with the domain boundary, as shown in Figure 3.2(c). A new domain is then computed around the last reachable set which does not intersect the domain boundary in Figure 3.2(d).

The advantage of this method is the avoidance of intersection computation with the domains, only inclusion tests are required.

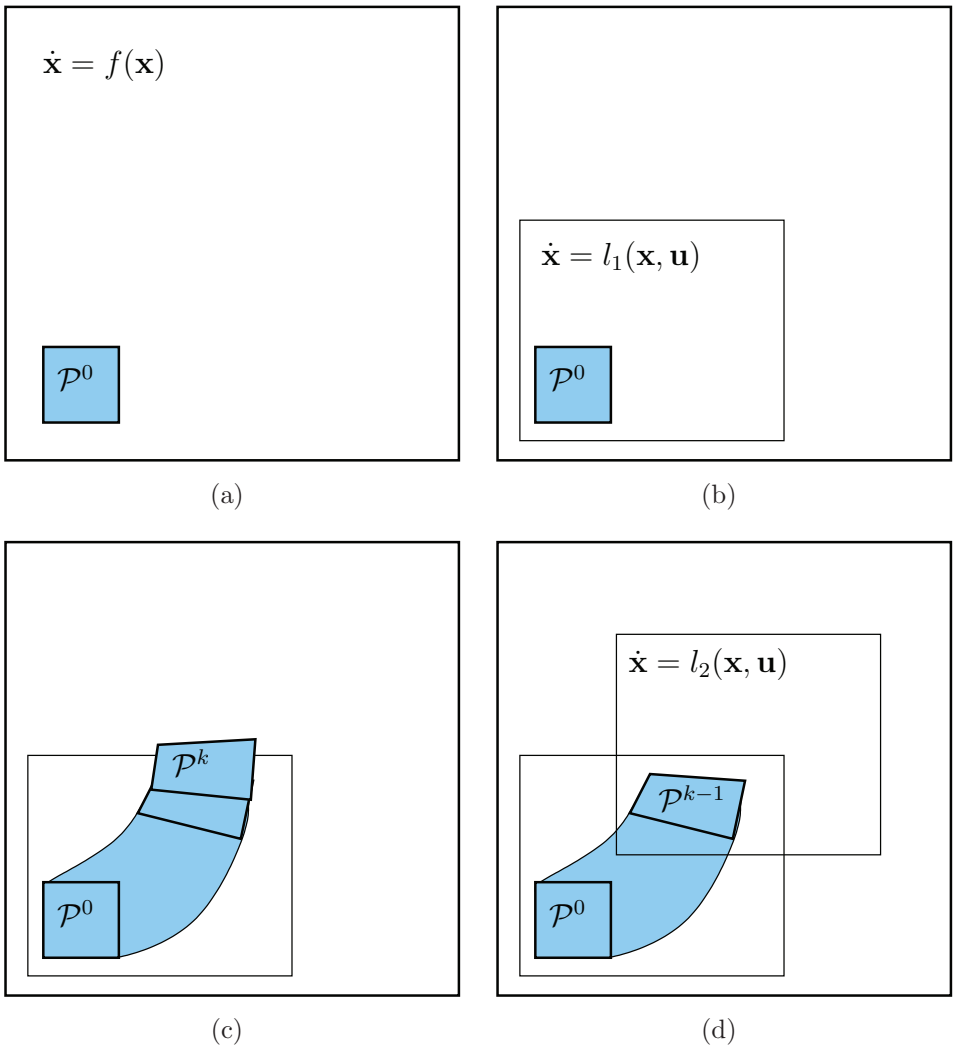

Figure 3.2: Illustration of the dynamic hybridization technique.

3.2 Affine interpolation

We want to approximate the vector of nonlinear functions $f(\mathbf{x})$ by a simpler one in a hybridization domain Δ. In this work, we choose affine systems with input as the approximate system class. This choice is motivated by the great number of available reachability techniques for this class of systems (see Section 1.2.1 page 15)

We denote an affine system by

$$\dot{\mathbf{x}(t)} = l(\mathbf{x}(t)), l : \mathbb{R}^n \to \mathbb{R}^n, \mathbf{x} \in \mathbb{R}^n$$

The choice of the class of approximation domains, which are simplices, is driven by the above choice of the class of approximate systems. Indeed, simplices are the most used class of polyhedra for computing affine approximation of nonlinear dynamics. Many numerical methods like finite-element methods use simplicial meshes [90].

More importantly over a non-degenerated simplex we can compute a linear interpolation of a nonlinear function by solving a set of linear equations for which the solution is unique. We recall that in a n dimensional space, a simplex can be represented by the convex hull of a set of $(n+1)$ vertices:

$$\Delta = conv\{\mathbf{v}^1, \ldots, \mathbf{v}^{n+1}\} \subset \mathbb{R}^n$$

The affine approximation is formed from a $n \times n$ matrix that we denote $\mathbf{A}$ and a column vector $\mathbf{b}$ of size n:

$$l(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{b} \in \mathbb{R}^n$$

To compute this affine function we interpolate the nonlinear function f at the vertices of the simplex Δ. To this end, we have to solve a set of $(n+1)n$ equations defined by

$$f(\mathbf{v}^i) = \mathbf{A}\mathbf{v}^i + \mathbf{b}, i \in \{1, \ldots, n+1\}$$

where $\{\mathbf{v}^i,\ i \in \{1, \ldots, n+1\}\}$ is the set of vertices of Δ.

The number of equations is equal to the number of unknown variables forming $\mathbf{A}$ and $\mathbf{b}$, which guarantees the uniqueness of the solution if the simplex Δ is full-dimensional.

In addition, to be conservative, it is important to take into consideration the distance between the original and the approximating functions for all points of the domain Δ. We define this metric by the infinite norm of the difference between the original and the approximate systems:

$$\|f(\mathbf{x}) - l(\mathbf{x})\|_\infty, \mathbf{x} \in \Delta$$

Since computing this distance for each point in Δ is impossible, we use a bound of this distance that we call error bound and denote it by μ. This bound can be defined as follows:

$$\mu = \max_{\mathbf{x} \in \Delta} \|f(\mathbf{x}) - l(\mathbf{x})\|_\infty$$

This bound is then used to define the input set $\mathcal{U}_\mu \subset \mathbb{R}^n$

$$\mathcal{U}_\mu : \{\mathbf{u} \mid \mathbf{u} \in \mathbb{R}^n \wedge \|\mathbf{u}\|_\infty \leq \mu\}.$$

Let $\mathcal{U}$ the set of input function in the form $\mathbb{R}^+ \to \mathcal{U}_\mu$. Finally the approximate system for the domain Δ is

$$\dot{\mathbf{x}}(t) = l(\mathbf{x}(t)) + \mathbf{u}(t), \mathbf{u}(t) \in \mathcal{U}_\mu,\ \mathbf{x}(t) \in \Delta. \tag{3.2}$$

The error bound μ is the key to the quality of the approximation; a large error can lead to large over-approximation error in the reachable set computation and to wrapping effects. Its effects on the distance between the trajectories of the original system and the approximate system is given in the following lemma

Lemma 1. *For all $\mathbf{u}(\cdot) \in \mathcal{U}$, for all $t \geq 0$, for all $\mathbf{x} \in \Delta$*

$$\|\Phi_f(\mathbf{x}, t) - \Phi_l(\mathbf{x}, t, \mathbf{u}(\cdot))\|_\infty \leq \frac{\mu}{2}(e^{Lt} - 1)$$

where L is the Lipschitz constant for the function f.

This result is a consequence of the Fundamental Inequality theorem [92] from the theory of dynamical systems. A proof of this lemma can be found in [12].

In the following, we consider the error bound μ as a given parameter for the reachability analysis, and we want to construct a simplicial domain Δ with the following property

1. The initial set $\mathcal{P}$ is included in Δ.

2. The distance between the interpolating and the original system does not exceed the error bound given as the parameter, that is

$$\forall \mathbf{x} \in \Delta,\ \|f(\mathbf{x}) - l(\mathbf{x})\|_\infty \leq \mu.$$

3. To save computation time we want to maximize the volume of this domain, which induces less new domain constructions.

It is important to consider that, in addition to the inclusion property, the simplex must be positioned around $\mathcal{P}$ according to the evolution direction. A good positioning strategy can increase the time during which the computed reachable set are included in the domain, and then it tends to minimize the occurrence of new domain computations. This will be discussed in Section 3.5.

In the next subsection, we establish the relation between the error bound and the shape of the simplicial domain.

3.2.1 Error bound

Estimating the errors between the original and the approximate systems is a key for controlling the analysis precision. We present in the following a new formulation of the error bound which is the base of our contribution concerning hybridization domains construction. We start by recalling the error bound used in previous works [13, 36].

Previous works

The following error bounds μ have been used in previous works for two cases: the vector field f is Lipschitz and f is a C^2 function.

- If f is Lipschitz and L is its Lipschitz constant, then

$$\mu \leq \varrho_{max} \frac{2n\,L}{n+1} = \overline{\mu}(\varrho_{max}).$$

  where ϱ_{max} is the maximal edge length of the simplex.

- If f is C^2 on Δ with bounded second order derivatives then

$$\mu \leq \frac{Kn^2}{2(n+1)^2} \varrho_{max}^2 = \overline{\mu}(\varrho_{max}) \tag{3.3}$$

  where K is a bound on the second derivatives of f

$$K = \max_{i \in \{1,\ldots,n\}} \sup_{\mathbf{x} \in \Delta} \sum_{j=1}^{j=n} \sum_{k=1}^{k=n} \left| \frac{\partial^2 f_i(\mathbf{x})}{\partial x_j \partial x_k} \right|.$$

We write the above error bounds as a function of ϱ_{max} to emphasize that it depends on the maximal simplex edge length ϱ_{max}.

Tighter error bound

We have shown in [38, 40] that one can obtain a better error bound which depends on.

- The radius of the *smallest containment ball* that we denote as r_{min}.
- A smoothness measure that we call *maximal curvature* denoted by γ.

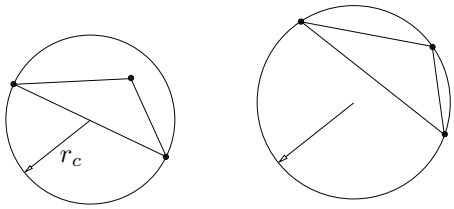

Figure 3.3: The smallest containment circle of the same triangle (shown on the left), which should not be confused with its circumcirle (shown on the right).

Given a hybridization domain $\Delta \subset \mathbb{R}^n$, The smallest containment ball is the smallest ball in $\mathbb{R}^n$ that contains the simplex. Figure 3.3 illustrates this notion.

To express the *maximal curvature* of $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$ in a domain Δ we first introduce the notion of directional curvature for one function $f_i(\mathbf{x}), i \in \{1, \ldots, n\}$. To compute the directional curvature of a function $f_i(\mathbf{x})$ we use the Hessian matrix defined as:

$$\mathbf{H}^i(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f_i}{\partial x_1^2} & \frac{\partial^2 f_i}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f_i}{\partial x_1 x_n} \\ \frac{\partial^2 f_i}{\partial x_1 x_2} & \frac{\partial^2 f_i}{\partial x_2^2} & \cdots & \frac{\partial^2 f_i}{\partial x_2 x_n} \\ & & \cdots & \\ \frac{\partial^2 f_i}{\partial x_1 x_n} & \frac{\partial^2 f_i}{\partial x_2 x_n} & \cdots & \frac{\partial^2 f_i}{\partial x_n^2} \end{pmatrix}. \tag{3.4}$$

The curvature along a direction given by a unit vector $\mathbf{d} \in \mathbb{R}^n$ is defined as:

$$\partial f_i(\mathbf{x}, \mathbf{d}) = \mathbf{d}^T \mathbf{H}^i(x) \mathbf{d}$$

The *maximal curvature* is given by the minimal real number that satisfies the following condition:

$$\forall i \in \{1, \ldots, n\} : \max_{\mathbf{x} \in \Delta,\, ||\mathbf{d}||=1} |\partial f_i(\mathbf{x}, \mathbf{d})| \leq \gamma. \tag{3.5}$$

The following theorem gives the bound of the interpolation error.

Theorem 1 (Interpolation error bound)**.** *Let $l(\mathbf{x})$ be the affine function that interpolates the functions $f(\mathbf{x})$ over a simplex Δ. Then, for all $\mathbf{x} \in \Delta$*

$$||f(\mathbf{x}) - l(\mathbf{x})|| \leq \gamma \frac{r_{min}^2}{2}.$$

A proof of this theorem can be found in [90].

Now the domain Δ is constructed based on the condition on its maximal radius of its smallest containment ball. We can see that within a ball of radius r_c, if the curvature is constant in every direction, the simplices with the largest volume that guarantee the interpolation error bound of Theorem 1 are equilateral (i.e. all the edges have the same length). However, this error bound is appropriate only when the directional curvatures are not much different in every direction. There are functions where the largest curvature in one direction greatly exceeds the largest curvature in another, and in these cases, it is possible to achieve the same accuracy with non-equilateral simplices. Intuitively, we can stretch an equilateral simplex along a direction in which the curvature is small in order to obtain a new simplex with larger size.

In order to show the convergence of this method, we consider the approximation error in terms of the Hausdorff distance. Let $Reach_f(\mathcal{P}, t)$ be the reachable set of the original nonlinear system (3.1) and $Reach_l(\mathcal{P}, t)$ be the set computed using the approximate system (3.2).

Theorem 2.

$$d_H(Reach_f(\mathcal{P}, t), Reach_l(\mathcal{P}, t)) \leq \mu(\frac{e^{Lt} - 1}{L} + 2re^{Lt})$$

where $t \in \mathbb{R}^+$. L is the Lipschitz constant for the function f and r is the time step of the integration scheme used to study the approximate systems.

A proof of this theorem can be established using the Lemma 1 and can be found in [12].

In the following section we introduce a way to exploit this fact in order to achieve better accuracy.

3.2.2 Isotropic map

A better way to judge the approximation quality of a simplex is to map it to an "isotropic" space where the curvature bounds are isotropic (that is identical in each direction). Indeed, it is possible to derive an error bound similar to the one in Theorem 1 but with the radius of the smallest containment ball in this "isotropic" space [90]. To explain this, we define a curvature tensor matrix. We assume the boundedness of directional curvature of f.

Definition 15. *Given a subset* Δ *of* $\mathbb{R}^n$ *and a symmetric positive-definite matrix* $\mathbf{C}$*, if for any unit vector* $\mathbf{d} \in \mathbb{R}^n$*,*

$$\forall i\{1,\ldots,n\}\ \forall \mathbf{x} \in \Delta:\ \ \max|\mathbf{d}^T\mathbf{H}^i(\mathbf{x})\mathbf{d}| \leq \mathbf{d}^T\mathbf{C}\mathbf{d},$$

we say that in the set Δ *the directional curvature of* f *is bounded by* $\mathbf{C}$ *and we call* $\mathbf{C}$ *a curvature tensor matrix of* f *in* Δ*.*

$$\mathbf{C} = \mathbf{\Omega}\mathbf{\Xi}\mathbf{\Omega}^T$$

where $\mathbf{\Omega} = [\boldsymbol{\omega}_1\boldsymbol{\omega}_2\ldots\boldsymbol{\omega}_n]$ and

$$\mathbf{\Xi} = \begin{pmatrix} \xi_1 & 0 & \ldots & 0 \\ 0 & \xi_2 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \ldots & \xi_n \end{pmatrix}.$$

The vectors $\boldsymbol{\omega}_i$ and values ξ_i are the eigenvectors and eigenvalues of a symmetric positive-definite matrix $\mathbf{C}$.

Let ξ_{max} and ξ_{min} be the largest and smallest eigenvalues of $\mathbf{C}$. The curvature matrix $\mathbf{C}$ can be specified using an estimate of the Hessian matrices $\mathbf{H}^i$. This will be discussed in more detail in Section 3.3.

We now define a matrix $\mathbf{T}$ which maps a point in the original space (that is, the domain over which the functions f are defined) to an isotropic space:

$$\mathbf{T} = \mathbf{\Omega}\begin{pmatrix} \sqrt{\xi_1/\xi_{max}} & 0 & \ldots & 0 \\ 0 & \sqrt{\xi_2/\xi_{max}} & \ldots & 0 \\ & & \ldots & \\ 0 & & \ldots & \sqrt{\xi_n/\xi_{max}} \end{pmatrix}\mathbf{\Omega}^T. \tag{3.6}$$

Given a set $\Delta \subset \mathbb{R}^n$, let $\widehat{\Delta}$ denote the set resulting from applying the linear transformation specified by the matrix $\mathbf{T}$ to Δ, that is,

$$\widehat{\Delta} = \{\mathbf{T}\mathbf{x} \mid \mathbf{x} \in \Delta\}.$$

Geometrically, the transformation $\mathbf{T}$ "shortens" a set along the directions in which f has high curvatures. An illustration of this transformation is depicted in Figure 3.4, where the application of the transformation $\mathbf{T}$ to an ellipsoid representing the level set of the directional curvature at simplex center produces a circle. When applying $\mathbf{T}$ to the triangle inscribed in the ellipsoid shown on the left, the result tends to be an equilateral triangle if the initial one is well shaped and has a maximal volume according to an error bound.

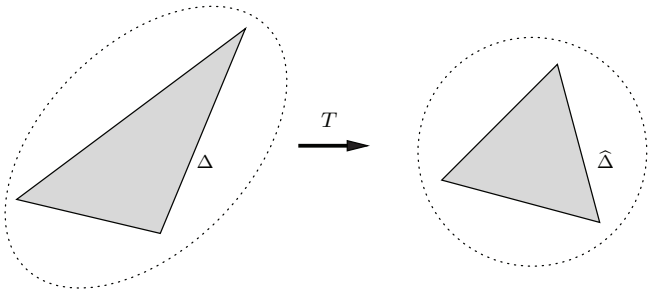

Figure 3.4: Illustration of the transformation to the isotropic space.

Theorem 3. *Let l be the affine function that interpolates the functions f over the simplex Δ. Then, for all $\mathbf{x} \in \Delta$*

$$||f(\mathbf{x}) - l(\mathbf{x})|| \leq \mathbf{C}^{\Delta} \frac{r_c^2(\widehat{\Delta})}{2} = \epsilon(r_c).$$

where $\mathbf{C}^{\Delta}$ is the maximal curvature in Δ and $r_c(\widehat{\Delta})$ is the radius of the smallest containement of the transformed simplex $\widehat{\Delta}$.

The idea of the proof is as follows.

Proof. Let

$$\phi(\mathbf{x}) = f(\mathbf{T}^{-1}\mathbf{x})$$

be the function defined over the isotropic space. Similarly, for the linear interpolating function l, we define

$$\lambda(\mathbf{x}) = l(\mathbf{T}^{-1}\mathbf{x}).$$

Note that $\phi(\widehat{\mathbf{x}}) = f(\mathbf{x})$. So the range of ϕ over the domain $\widehat{\Delta}$ is the same as the range of f over the domain Δ. The curvature of ϕ has a bound that is independent of direction. Let $G^i(\mathbf{x})$ denote the Hessian matrix of $\phi(\mathbf{x})$. Indeed,

$$\begin{aligned} \partial\phi_i(\mathbf{x}, \mathbf{d}) &= \mathbf{d}^T G^i(\mathbf{x}) \mathbf{d} \\ &= (\mathbf{T}^{-1}\mathbf{d})^T \mathbf{H}^i(x) (\mathbf{T}^{-1}\mathbf{d}) \end{aligned}$$

It then follows from the definition of the maximal curvature (Definition 15, page 41) that we have for all $i \in \{1, \ldots, n\}$

$$max_{\mathbf{x}\in\Delta, ||\mathbf{d}||\leq 1} |\partial\phi_i(\mathbf{x}, \mathbf{d})| \leq \gamma_{\Delta}.$$

Using Theorem 3,

$$\max_{\mathbf{x}\in\Delta} ||\phi(\mathbf{x}) - \lambda(\mathbf{x})|| \leq \gamma_\Delta \frac{r_c^2(\widehat{\Delta})}{2}.$$

By the above definitions of the functions ϕ and λ, we have $f(\mathbf{x}) = \phi(\widehat{\mathbf{x}})$ and $l(\mathbf{x}) = \lambda(\widehat{\mathbf{x}})$ we have

$$\max_{\mathbf{x}\in\Delta} ||f(\mathbf{x}) - l(\mathbf{x})|| = \max_{\mathbf{x}\in\widehat{\Delta}} ||\phi(\mathbf{x}) - \lambda(\mathbf{x})||.$$

It then follows that for all $\mathbf{x} \in \Delta$

$$||f(\mathbf{x}) - l(\mathbf{x})|| \leq \gamma_\Delta \frac{r_c^2(\widehat{\Delta})}{2}.$$

□

To show the interest of this error bound, we first show that using transformation $\mathbf{T}$ the smallest containment ball radius is reduced or at worst unchanged; hence we can use larger simplices for the same error bound.

Lemma 2. *Given a simplex $\Delta \subset \mathbb{R}^n$, the radius of the smallest containment ball of $\widehat{\Delta}$ is not larger than the radius of the smallest containment ball of Δ, that is $r_c(\widehat{\Delta}) \leq r_c(\Delta)$.*

The proof can be directly established from the construction of the transformation matrix $\mathbf{T}$. The error bound of Theorem 3 is at least as good as that of Theorem 1. For a "thin" simplex whose longer edges are along the directions of the eigenvectors associated with smaller eigenvalues, the ratio $\dfrac{r_c(\widehat{\Delta})}{r_c(\Delta)}$ can be as small as $\sqrt{\xi_{min}/\xi_{max}}$. In the worst case, when the simplex is "parallel" to an eigenvector associated with the largest eigenvalue, this ratio is 1. □

Furthermore, we compare the new error bounds with the ones shown on page 38 in (3.3) which were used in the previous work. We first notice that the bound K in (3.3) must be larger than γ_Δ. To see this, we notice that any matrix norm is always larger than the maximum of the absolute values of the eigenvalues. It is, however, not easy to relate the smallest containment ball with the simplex size. For comparison purposes, we can use the following result.

Lemma 3. *Let Δ be a simplex in $\mathbb{R}^n$ with the maximal edge length ϱ_{max}. Then, the radius $r_c(\Delta)$ of its smallest containment sphere satisfies*

$$r_c(\Delta) \leq \varrho_{max}\sqrt{\frac{n}{2(n+1)}}$$

where n is the dimension of the system.

Proof. We begin with some intermediate results.

We first prove that the center $\mathbf{p}$ of the smallest containment ball B is inside the simplex Δ. We suppose that the contrary is true. Hence, the hyperplane h of a face of the simplex separates the center and the vertex opposite this hyperplane. In other words, the vertex and the center are on the opposite side of the hyperplane. Let $\mathbf{p}^h$ be the intersection of h with $\mathbf{p}^h$. It is not hard to see that the smallest containment ball of $\mathbf{p}^h$, on one hand, has a radius smaller than that of B and, on the other hand, contains Δ. Thus, B is not the smallest containment ball.

We can also prove that if a vertex of Δ is not in the boundary of B then the center $\mathbf{p}$ lies in the face of the simplex opposite to this vertex. The proof of this can be found in, for example, [42].

We now proceed with the proof of the lemma. Since $\mathbf{p}$ is in Δ, we can write it as a linear combination of the vertices $\{\mathbf{v}^1, \ldots, \mathbf{v}^{n+1}\}$ of Δ:

$$\mathbf{p} = \lambda_1\mathbf{v}^1 + \lambda_2\mathbf{v}^2 + \ldots + \lambda_{n+1}\mathbf{v}^{n+1} \tag{3.7}$$

such that $\sum_{i=1}^{n+1}\lambda_i = 1$ and $\forall i \in \{1, \ldots, n+1\}\lambda_i \geq 0$.

Let λ_{n+1} be a positive coefficient among $\{\lambda_1, \ldots, \lambda_{n+1}\}$ such that $\lambda_{n+1} \geq \lambda_i$ for all $i \in \{1, \ldots, n+1\}$. Since $\lambda_{n+1} > 0$, the center $\mathbf{p}$ does not lie in the face opposite $\mathbf{v}^{k+1}$. Hence, using the above fact, $\mathbf{v}^{n+1}$ must lie in the boundary of B. Without loss of generality, we can assume that $\mathbf{v}^{n+1}$ is the origin. Similarly, for any positive coefficient λ_i, $\mathbf{v}^i$ is in the boundary of B, which means that the vector $2\mathbf{p} - \mathbf{v}^i$ is perpendicular to the vector $\mathbf{v}^i$, that is the scalar product

$$\mathbf{v}^i \cdot (2\mathbf{p} - \mathbf{v}^i) = 0.$$

It then follows that

$$2\mathbf{v}^i \cdot \mathbf{p} = \mathbf{v}^i \cdot \mathbf{v}^i.$$

Hence, for any coefficient $\lambda_i \geq 0$,

$$2\lambda_i(\mathbf{v}^i \cdot \mathbf{p}) = \lambda_i(\mathbf{v}^i \cdot \mathbf{v}^i).$$

Then,

$$\sum_{i=1}^{n}\lambda_i(\mathbf{v}^i \cdot \mathbf{p}) = \frac{1}{2}\sum_{i=1}^{n}\lambda_i(\mathbf{v}^i \cdot \mathbf{v}^i).$$

Using (3.7) and the fact that for every $i \in \{1, \ldots, n\}$ $\mathbf{v}^i \cdot \mathbf{v}^i$ is smaller than the maximal edge length of the simplex,

$$\mathbf{p} \cdot \mathbf{p} \leq \frac{1}{2} \varrho_{max}^2 \sum_{i=1}^{n} \lambda_i.$$

Since $\sum_{i=1}^{n} \lambda_i = 1 - \lambda_{n+1}$ and $\mathbf{p} \cdot \mathbf{p} = r_c(\Delta)$, we have

$$r_c(\Delta) \leq \frac{1}{2} \varrho_{max}^2 \sum_{i=1}^{n} \lambda_i.$$

In addition, since $\lambda_{n+1} \geq \lambda_i$ for all i, we have

$$r_c(\Delta) \leq \varrho_{max}^2 \frac{n}{2(n+1)}.$$

□

A direct consequence of this result is the following ratio between the old and new error bounds for any simplex.

Theorem 4. *For any simplex Δ with the maximal edge length ϱ_{max}, the ratio between the new error bound $\bar{\mu}_{new}$ of Theorem 3 and the old error bound $\bar{\mu}$ in (3.3) satisfies the following inequality:*

$$\frac{\bar{\mu}_{new}(r_c(\widehat{\Delta}))}{\bar{\mu}(\varrho_{max})} \leq \frac{n+1}{2n}.$$

In two dimensions, compared to the old error bound, the new error bound is reduced at least by the factor 4/3. The reduction factor $\dfrac{2n}{n+1}$ grows when the dimension n increases and approaches 2 when n tends to infinity.

This reduction is very useful especially in high dimensions because when dividing a simplex in order to satisfy some edge length bound, the number of resulting subsets grows exponentially with the dimension. Moreover, as in the above discussion of Lemma 2, by choosing an appropriate orientation, we can reduce this ratio further by $\sqrt{\xi_{min}/\xi_{max}}$.

To compute the transformation matrix $\mathbf{T}$ we need to estimate a curvature tensor for the system, the next section gives a method to compute this matrix for a large class of nonlinear functions.

3.3 Curvature tensor estimation

We first consider the case where the Hessian matrices are constant, as is the case with quadratic functions. To compute a curvature tensor matrix, we first define a matrix $\mathbf{C}^i$ as the matrix with the same eigenvectors and eigenvalues as $\mathbf{H}^i$, except that each negative eigenvalue ξ of $\mathbf{H}^i$ is replaced with the positive eigenvalue $-\xi$. Note that we can, in this case, omit the simplex in the notation of the curvature tensor matrix. Hence, $\mathbf{C}^i$ is guaranteed to be positive definite. If any eigenvalue of $\mathbf{H}^i$ is zero, we substitute it with some small positive value. That is, for each matrix $\mathbf{H}^i$, we define

$$\mathbf{C}^i(\Delta) = [\boldsymbol{\omega}_1^i \dots \boldsymbol{\omega}_n^i] \begin{pmatrix} |\xi_1^i| & 0 & \dots & 0 \\ 0 & |\xi_2^i| & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & |\xi_n^i| \end{pmatrix} [\boldsymbol{\omega}_1^i \dots \boldsymbol{\omega}_n^i]^T$$

where $\boldsymbol{\omega}_j^i$ (with $j \in \{1, \dots, n\}$) are the eigenvectors of $\mathbf{H}^i$. We denote by ξ_{max}^i the eigenvalue with the largest absolute magnitude of $\mathbf{C}^i$. Among the matrices $\mathbf{C}^i$ we can choose the one with the largest absolute eigenvalue to be a curvature tensor matrix.

For more general classes of functions where the Hessian matrices are not constant, we can estimate the curvature tensor matrix using optimization. We observe that, given a simplex Δ, by definition, for each $i \in \{1, \dots, n\}$ the eigenvalues of the Hessian matrix $\mathbf{H}^i$ are inside the interval $[-\gamma_{\mathcal{Q}}, \gamma_{\mathcal{Q}}]$ where $\gamma_{\mathcal{Q}}$ is the maximal curvature of f inside $\mathcal{Q}$, a subset of $\mathbb{R}^n$. Hence, the error bound is determined by the maximal eigenvalue $\xi^{max}(\mathbf{C})$ of the matrix $\mathbf{C}$. Note additionally that $r_c(\widehat{\Delta})$ depends on $|det(\mathbf{T})|^{1/n}$ where $det(\mathbf{T})$ is the determinant of the transformation matrix defined in (3.6).

Therefore, we want to find a positive-definite matrix $\mathbf{C}$ that satisfies the condition of Definition 15 of curvature tensor and, in addition, makes $|\xi^{max}(\mathbf{C})||det(\mathbf{T})|^{2/n}$ as small as possible. To do so, we formulate this problem as solving the following constrained optimization problem:

$$\begin{gathered} \min |\xi^{max}(\mathbf{C})||det(\mathbf{T})|^{2/n} \\ \text{s.t. } \forall i \in \{1, \dots, n\}\ \forall \mathbf{x} \in \mathcal{Q}\ \forall \mathbf{d} \in \mathbb{R}^n : \\ ||\mathbf{d}|| = 1\ \wedge\ |\partial^2 f_i(\mathbf{x}, \mathbf{d})| \leq |\mathbf{d}^T \mathbf{C}^i \mathbf{d}|. \end{gathered}$$

Again, we express $\mathbf{C}$ in its eigen-decomposition form. Let $\xi^1, \xi^2, \dots, \xi^n$ be the eigenvalues in increasing order of $\mathbf{C}$, that is $0 < \xi^1 \leq \xi^2 \leq \dots \leq \xi^n$, and $\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \dots, \boldsymbol{\omega}^n$ be the corresponding eigenvectors. From now on we use

superscripts to denote eigenvectors since subscripts will be used to denote their coordinates. Thus,

$$\mathbf{C} = \mathbf{S}\Xi\mathbf{S}^T$$

where

$$\Xi = diag(\xi^1, \xi^2, \ldots, \xi^n)$$

Therefore, minimizing over all possible matrices $\mathbf{C}$ satisfying the definition 15 page 41 is equivalent to minimizing over all possible ξ^i and all possible orthogonal matrices $\mathbf{S}$.

Notice that, by the definition of the matrix $\mathbf{T}$,

$$|det(\mathbf{T})| = |det(\mathbf{C})|^{1/2} = |(\Pi_{j=1}^n \xi^j)|^{1/2}$$

The objective function can therefore be written as:

$$|\xi^{max}(\mathbf{C})||det(\mathbf{T})|^{2/n} = |\xi^n|\ |(\Pi_{j=1}^n \xi^j)|^{1/n}.$$

On the other hand, the constraint from Definition 15 can be written as:

$$\begin{aligned}&\forall i \in \{1, \ldots, n\}\ \forall \mathbf{x} \in \mathcal{Q}\ \forall \mathbf{d} \in \mathbb{R}^n : \\ &\quad ||\mathbf{d}|| = 1\ \wedge\ |\partial^2 f_i(\mathbf{x}, \mathbf{d})| \leq |\mathbf{d}^T \mathbf{S}\, \Xi\, \mathbf{S}^T \mathbf{d}|.\end{aligned}$$

This problem might not have a solution or it might have a solution with some eigenvalue equal to 0, which makes $\mathbf{C}$ singular. In the following, we consider another approach, which involves approximating $\mathbf{C}$ by making the error bound as small as possible while respecting the constraint from Definition 15.

Since the error bound depends on the maximal eigenvalue of $\mathbf{C}$ and the product of the eigenvalues of $\mathbf{C}$, we estimate $\mathbf{C}$ by determining successively its eigenvalues ξ^j and eigenvectors $\boldsymbol{\omega}^j$ such that each ξ^j is made as small as possible while satisfying the condition of Definition (15).

More precisely, in the first step we determine ξ^n such that

$$\forall \mathbf{x} \in \mathcal{Q}\ \forall i \in \{1, \ldots, n\}\ \mathbf{d} \in \mathbb{R}^n :\ ||\mathbf{d}|| = 1\ \wedge\ \xi^n \geq |\partial^2 f_i(\mathbf{x}, \mathbf{d})|.$$

We can find ξ^n by solving the following n optimization problems

$$\xi^{n,i} = max_{\mathbf{x}\in\mathcal{Q}\ \wedge\ ||\mathbf{d}||=1} |\partial^2 f_i(\mathbf{x}, \mathbf{d})|, i \in \{1, \ldots, n\}$$

Then we take the largest among the computed maximal values:

$$\xi^n = max_{i\in\{1,\ldots,n\}} \xi^{n,i}.$$

In other words, by (3.5), ξ^n is exactly the largest curvature of f in $\mathcal{Q}$. The corresponding eigenvector $\boldsymbol{\omega}^n$ is chosen as the unit vector $\mathbf{d}$ along which the second-order directional derivative of the corresponding f_i is the largest.

To determine the remaining $(n-1)$ eigenvalues, let $(n-1)$ vectors

$$\boldsymbol{\omega}^1, \ldots, \boldsymbol{\omega}^{n-1}$$

in $\mathbb{R}^n$ form an orthonormal basis for the orthogonal complement of $span\{\boldsymbol{\omega}^n\}$ in $\mathbb{R}^n$. Then, any vector $\mathbf{d} \in \mathbb{R}^n$ can be expressed as:

$$\mathbf{d} = \sum_{j=1}^{n-1} \alpha_j \boldsymbol{\omega}^j + \beta \boldsymbol{\omega}^n \tag{3.8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{R}^{n-1}$ and $\beta \in \mathbb{R}$.

Let

$$\mathbf{S}^{n-1} = [\boldsymbol{\omega}^1 \boldsymbol{\omega}^2 \ldots \boldsymbol{\omega}^{n-1}]$$

and Ξ^{n-1} be the diagonal matrix with $\xi^1, \xi^2, \ldots, \xi^{n-1}$ as its diagonal elements. Note that the superscripts here indicate that these matrices are used to compute the eigenvalue ξ^{n-1}. We define then

$$\mathbf{C}^{n-1} = \mathbf{S}^{n-1} \Xi^{n-1} (\mathbf{S}^{n-1})^T.$$

Then, it is not hard to see that

$$\mathbf{d}^T \mathbf{C} \mathbf{d} = \boldsymbol{\alpha}^T \mathbf{C}^{n-1} \boldsymbol{\alpha} + \xi^n \beta^2.$$

The condition of Definition 15 becomes

$$\begin{gathered} \forall i \in \{1, \ldots, n\},\ \forall \boldsymbol{\alpha} \in \mathbb{R}^{n-1},\ \forall \beta \in \mathbb{R},\ \forall \mathbf{x} \in \mathcal{Q} : \\ \boldsymbol{\alpha}^T \mathbf{C}^{n-1} \boldsymbol{\alpha} \geq |\partial^2 f_i(\mathbf{x}, \mathbf{d})| - \xi^n \beta^2. \end{gathered} \tag{3.9}$$

We denote the right hand side as a function of $\boldsymbol{\alpha}$ and β:

$$w^i(\boldsymbol{\alpha}, \beta) = |\partial^2 f_i(\mathbf{x}, \mathbf{d})| - \xi^n \beta^2,$$

and

$$\eta^i(\boldsymbol{\alpha}) = max_{\beta \in \mathbb{R}} w^i(\boldsymbol{\alpha}, \beta). \tag{3.10}$$

The condition (3.9) is then equivalent to

$$\begin{gathered} \forall i \in \{1, \ldots, n\},\ \forall \boldsymbol{\alpha} \in \mathbb{R}^{n-1},\ \forall \mathbf{x} \in \mathcal{Q} : \\ \boldsymbol{\alpha}^T \mathbf{C}^{n-1} \boldsymbol{\alpha} \geq \eta^i(\boldsymbol{\alpha}) \end{gathered} \tag{3.11}$$

Lemma 4. *If β maximizes $w^i(\boldsymbol{\alpha}, \beta)$ in (3.10) then β satisfies*

$$\frac{\partial^2 f_i(\mathbf{x}, \mathbf{d})}{|\partial^2 f_i(\mathbf{x}, \mathbf{d})|} \partial g_i(\mathbf{x}, \mathbf{d}) = \xi^n \beta. \tag{3.12}$$

where

$$g_i(\mathbf{x}) = \partial f_i(\mathbf{x}, \boldsymbol{\omega}^n)$$

Intuitively, $\partial g_i(\mathbf{x}, \mathbf{d})$ is the first-order directional derivative of g_i with respect to the vector $\mathbf{d}$, and $g_i(\mathbf{x}) = \partial f_i(\mathbf{x}, \boldsymbol{\omega}^n)$ is the first-order directional derivative of f_i with respect to the vector $\boldsymbol{\omega}^n$.

Proof. To solve (3.10), we consider the critical points of $w^i(\boldsymbol{\alpha}, \beta)$ with respect to β. To express the partial derivative of $w^i(\boldsymbol{\alpha}, \beta)$, we first rewrite the second-order directional derivative of f_i as the following sum:

$$\partial^2 f_i(\mathbf{x}, \mathbf{d}) = \sum_{j,k} H^i_{j,k}(\mathbf{x}) d_j d_k$$

where $H^i_{j,k}$ is the element of the j^{th} line and the k^{th} column of the Hessian matrix of the function f_i.

Then,

$$\begin{aligned}
\frac{\partial}{\partial \beta}(\partial^2 f_i(\mathbf{x}, \mathbf{d})) &= \sum_{j,k} H^i_{j,k}(\mathbf{x})(d_j \frac{\partial d_k}{\partial \beta} + d_k \frac{\partial d_j}{\partial \beta}) \\
&= \sum_{j,k} H^i_{j,k}(\mathbf{x})(d_j \omega^n_k + d_k \omega^n_j) \quad \text{(from (3.8))} \\
&= 2 \sum_{j,k} H^i_{j,k}(\mathbf{x}) d_j \omega^n_k \\
&= 2\partial(\partial f_i(\mathbf{x}, \boldsymbol{\omega}^n), \mathbf{d}) \\
&= 2\partial g_i(\mathbf{x}, \mathbf{d}) \quad \text{(from (3.12))}
\end{aligned}$$

In addition,

$$\frac{\partial}{\partial \beta}(\xi^n \beta^2) = 2\xi^n \beta.$$

Note that the critical points that satisfy $\partial^2 f_i(\mathbf{x}, \mathbf{d}) = 0$ are not among the maxima of $w^i(\boldsymbol{\alpha}, \beta)$. It then follows from the above that the critical points that are candidates to be among the maxima satisfy:

$$\frac{\partial^2 f_i(\mathbf{x}, \mathbf{d})}{|\partial^2 f_i(\mathbf{x}, \mathbf{d})|} \partial g_i(\mathbf{x}, \mathbf{d}) = \xi^n \beta.$$

This establishes the proof of the lemma. □

To determine β, we consider two cases:

- If $\partial^2 f_i(\mathbf{x}, \mathbf{d}) > 0$, the equation (3.12) can be rewritten as:

$$\sum_{j,k} H^i_{j,k} \omega^n_j \mathbf{d}_k = \xi^n \beta.$$

  It then follows from (3.8) that

$$\sum_{j,k} H^i_{j,k} \omega^n_j (\sum_{1}^{n-1} \alpha_j \omega^j_k + \beta \omega^n_k) = \xi^n \beta.$$

  Note that ξ^n and $\boldsymbol{\omega}^n$ are now known, from the above we can determine β as a function of $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \ldots, \boldsymbol{\omega}^{n-1}$. From now on, for clarity, we denote by β^n the value of β satisfying the above, since this value corresponds to the eigenvalue ξ^n and the eigenvector $\boldsymbol{\omega}^n$ computed in the first step. Hence, if $\partial^2 f_i(\mathbf{x}, \mathbf{d}) > 0$

$$\beta^n = \frac{(\boldsymbol{\omega}^n)^T \mathbf{H}^i (\sum_{j=1}^{n-1} \alpha_j \boldsymbol{\omega}^j)}{\xi^n - (\boldsymbol{\omega}^n)^T \mathbf{H}^i \boldsymbol{\omega}^n}.$$

- If $\partial^2 f_i(\mathbf{x}, \mathbf{d}) < 0$, similarly we obtain

$$\beta^n = \frac{-(\boldsymbol{\omega}^n)^T \mathbf{H}^i (\sum_{j=1}^{n-1} \alpha_j \boldsymbol{\omega}^j)}{\xi^n + (\boldsymbol{\omega}^n)^T \mathbf{H}^i \boldsymbol{\omega}^n}.$$

With the above β^n, $\eta^i(\boldsymbol{\alpha})$ in (3.10) can be determined. Hence, the condition (3.9) becomes

$$\boldsymbol{\alpha}^T \mathbf{C}^{n-1} \boldsymbol{\alpha} \geq u(\boldsymbol{\alpha}) = |\partial^2 f_i(\mathbf{x}, \mathbf{d})| - \xi^n (\beta^n)^2 \tag{3.13}$$

where $d = \sum_1^{n-1} \alpha_j \boldsymbol{\omega}^j + \beta^n \boldsymbol{\omega}^n$.

We now come to the same problem as the initial but with only $(n-1)$ eigenvalues to determine. We can repeat this procedure to determine all the eigenvalues. More precisely, we determine ξ^{n-1} by solving the following optimization problem over the variables $z = \sum_{j=1}^{n-1} \alpha_j \boldsymbol{\omega}^j$ and $\mathbf{x}$.

$$\begin{gathered} \xi^{n-1,i} = \max(|\partial^2 f_i(\mathbf{x}, z + \beta^n \boldsymbol{\omega}^n)| - \xi^n (\beta^n)^2) \\ \text{s.t. } \mathbf{x} \in \mathcal{Q} \ \wedge \\ z \in \mathbb{R}^n \ \wedge \\ ||z + \beta^n \boldsymbol{\omega}^n|| = 1. \end{gathered}$$

Then, ξ^{n-1} is determined as the largest of all $\xi^{n-1,i}$.

To reduce further to the problem with $(n-2)$ eigenvalues, we write:

$$\mathbf{d}^T\mathbf{C}\mathbf{d} = \boldsymbol{\alpha}^T\mathbf{C}^{n-2}\boldsymbol{\alpha} + \xi^n(\beta^n)^2 + \xi^{n-1}(\beta^{n-1})^2.$$

where $\boldsymbol{\alpha}$ is now a vector in $\mathbb{R}^{n-2}$.

Then, the sequence of optimization problems can be formulated as follows. For $k = n-1, n-2, \ldots, 1$

$$\begin{gathered}\xi^{n-k,i} = \max |\partial^2 f_i(\mathbf{x}, z + \sum_{j=n-k+1}^{n} \beta^j \boldsymbol{\omega}^j)| - \sum_{j=k+1}^{n} \xi^j(\beta^j)^2 \\ \text{s.t. } \mathbf{x} \in \mathcal{Q} \ \wedge \\ z \in \mathbb{R}^n \ \wedge \\ ||z + \textstyle\sum_{j=k+1}^{n} \beta^j \boldsymbol{\omega}^j|| = 1\end{gathered} \tag{3.14}$$

Then, ξ^{n-k} is the largest value of all $\xi^{n-k,i}$.

In the above procedure, in order to proceed from the computation of the eigenvector ξ^{n-k} to that of ξ^{n-k-1} we need to compute the corresponding eigenvector $\boldsymbol{\omega}^{n-k}$. As an example, in the step $k = n-1$, we obtain the solution $\mathbf{d} = z + \beta^n\boldsymbol{\omega}^n$ of the optimization problem. Let denote this $\mathbf{d}$ by $\mathbf{d}^{n-1}$ and we want to compute the corresponding eigenvector $\boldsymbol{\omega}^{n-1}$. To this end, we use the following scheme. Indeed, at each step k, $\boldsymbol{\omega}^k$ is made orthogonal to the previous $\boldsymbol{\omega}^{k+1}, \ldots, \boldsymbol{\omega}^n$ by substracting the projection of $\mathbf{d}^k$ in the directions of $\boldsymbol{\omega}^{k+1}, \ldots, \boldsymbol{\omega}^n$.

$$u^k = \mathbf{d}^k - \sum_{j=k+1}^{n} \frac{(\boldsymbol{\omega}^j)^T\mathbf{d}^k}{(\boldsymbol{\omega}^j)^T\mathbf{v}^j}\boldsymbol{\omega}^j$$

Then we determine the eigenvector $\boldsymbol{\omega}^k$ as:

$$\boldsymbol{\omega}^k = \frac{u^k}{||u^k||}.$$

Such n vectors $\boldsymbol{\omega}^k$ span the same subspace as n vectors $\mathbf{d}^k$.

Finally, we construct the matrix $\mathbf{C}$ as follows:

$$\mathbf{C} = \mathbf{S}\begin{pmatrix} \xi^1 & 0 & \ldots & 0 \\ 0 & \xi^2 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \ldots & \xi^n \end{pmatrix}\mathbf{S}^T$$

where ξ^j with $j \in \{1, \ldots, n\}$ are the computed eigenvalues, and $\mathbf{S} = (\boldsymbol{\omega}^1 \boldsymbol{\omega}^2 \ldots \boldsymbol{\omega}^n)$ is an orthonormal matrix containing the computed eigenvectors.

In the following, we presents how this curvature tensor can be used in the hybridization domain construction.

3.4 Reachability analysis using hybridization

Once the curvature tensor matrix is estimated, we can compute from it an isotropic transformation $\mathbf{T}$. This can then be used to create hybridization domains for reachability computation. The reachability computation accuracy depends on the precision of the curvature tensor approximation, since the latter is directly related to the error bound that is used to define the input set $\mathcal{U}_\mu$. In the curvature estimation described in the previous section, the optimization problems are solved over all $\mathbf{x} \in \mathcal{Q}$, that is the computed estimate is valid for the whole set $\mathcal{Q}$. The estimation precision can be improved by using a dynamical curvature estimation that is invoked each time a new hybridization domain needs to be created. In this case, the optimization domains can be chosen as a neighborhood of the current states of the system.

The reachability computation algorithm using hybridization is summarized by Algorithm 1 where $\mathcal{P}$ is a convex polyhedron containing all the initial states of the system. In each iteration, we first estimate the curvature tensor within a zone containing the current set $\mathcal{P}^k$. This matrix is then used to construct the simplicial domain Δ. We perform the reachability computation from $\mathcal{P}^k$ under the approximated linear interpolating dynamics defined within Δ. This generates the convex polyhedron $\mathcal{P}^{k+1}$. If this polyhedron contains points outside the current domain Δ we retrieve the previous polyhedron $\mathcal{P}^k$ and construct around $\mathcal{P}^k$ a new hybridization domain. Note that when the polyhedron $\mathcal{P}^k$ becomes too large to be included in Δ, splitting is required.

It is worth mentioning that in this algorithm, we use convex polyhedra to represent reachable sets. However, the proposed hybridization and domain construction methods can be combined with the algorithms using other set representations (such as [61, 49]) however the linear interpolation requires the simplex vertices computation.

The next section presents an algorithm for computing the simplex vertices and propose some strategies for positioning and rotating the simplex in the state space.

Algorithm 1 Reachability computation using hybridization

Input: Initial convex polyhedron $\mathcal{P}$, interpolation error tolerance ε

$\mathcal{P}^0 = \mathcal{P}$, $k = 0$
$\mathbf{C} = CurvatureEstimation(\mathcal{P}^k)$
$\Delta = DomainConstruct(\mathcal{P}^k, \mathbf{C}, \varepsilon)$
while $k \leq k_{max}$ **do**
  $\mathcal{P}^{k+1} = Reach(\mathcal{P}^k, \Delta)$
  if $\mathcal{P}^{k+1} \cap \bar{\Delta} \neq \emptyset$ **then**
    /* $\bar{\Delta}$ *is the complement of* Δ */
    $\Delta = DomainConstruct(\mathcal{P}^k, \varepsilon)$
  else
    $k = k + 1$
  end if
end while

3.5 Domain construction algorithm

We consider the problem of constructing a simplex around a polyhedron $\mathcal{P}$ with the objective of achieving a good approximation quality when performing analysis on the approximate system to yield the result for the original system.

3.5.1 Shape and size

We first consider the accuracy criterion. More precisely, we want to guarantee that the linear function that interpolates f satisfies the given desired error bound, μ. Let γ be the maximal curvature within a region of interest around the initial set, and for short we write it without specifying the simplex.

Theorem 3 indicates that the interpolation error variation depends on the radius $r_c(\widehat{\Delta})$. In order to exploit this result, we first transform the set $\mathcal{P}$ to $\widehat{\mathcal{P}} = \mathbf{T}\mathcal{P}$ in the isotropic space. Let $\mathcal{C}$ be the ball of radius $\sqrt{2\mu/\gamma}$ the centroid of which coincides with that of the set $\widehat{P}$. We assume that $\widehat{\mathcal{P}}$ is entirely included in $\mathcal{C}$. If this is not the case, the polyhedron $\mathcal{P}$ have to be split, this will be discussed later.

Let $\varepsilon = \mathbf{T}^{-1}(\mathcal{C})$ be the ellipsoid resulting from applying the inverse transformation $\mathbf{T}^{-1}$ to the ball $\mathcal{C}$. Then, according to Theorem 3 the interpolation error associated with any simplex inside the ellipsoid ε is guaranteed to be smaller than or equal to μ.

Since there are many simplices that can be fit inside a ball, we proceed with the problem of choosing a simplex that is good with respect to other optimization criteria, namely the simplex volume and the time of evolution within the simplex.

Lemma 5. *Let Δ^r be an equilateral simplex that is circumscribed by the ball $\mathcal{C}$. Then, $\mathbf{T}^{-1}(\Delta^r)$ is a largest volume simplex inscribed in the ellipsoid $\varepsilon = \mathbf{T}^{-1}\mathcal{C}$.*

Proof. The proof of this result relies on two standard results. First, the linear transformation preserves the volume ratio between two measurable bodies. Second, the simplices inside a ball with the largest volume are equilateral. □

It follows from the lemma that we only need to consider the simplices resulting from applying $\mathbf{T}^{-1}$ to the largest equilateral simplices inscribing in the ball $\mathcal{C}$. Any such simplex is guaranteed to be inscribed in the ellipsoid ε and to have the largest volume.

3.5.2 Orientation and placement

It remains to select one of the above simplices to meet the staying time requirement. To this end, we use the following heuristics. We sample trajectories starting at a number of points inside and around the polyhedron $\mathcal{P}$ and then determine an *average evolution direction* $\mathbf{d}$ for a given time interval. We then want the simplex to be "aligned" with this direction $\mathbf{d}$, as shown in Figure 3.5.

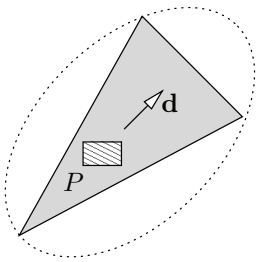

Figure 3.5: Illustration of the average evolution direction $\mathbf{d}$.

Note that we are considering only the equilateral simplices inscribed in $\mathcal{C}$. We now first pick an equilateral simplex Δ^r aligned with an axis, say x_1, as shown in Figure 3.6. This equilateral simplex can be efficiently constructed

since, due to its alignment, the construction can be done by recursively reducing to lower dimensions. Without loss of generality, we can assume that the simplex has a vertex $\mathbf{p}$ on this axis x_1. We now want to compute the linear transformation $\mathbf{M}$ which rotates it to align with $-\mathbf{d}$. To do so, we compute its inverse transformation as follows. Choosing a simplex vertex $\mathbf{p}$ as a "pivot" vertex, we define its associated median axis as the line passing through $\mathbf{p}$ and perpendicular to the hyperplane containing the corresponding base. Let $\mathbf{q}$ be the vector representing this median axis, as shown in Figure 3.6.

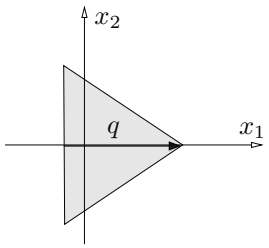

Figure 3.6: Illustration of a simplex median axis.

We want to compute a transformation $\mathbf{R}$ which aligns $\mathbf{q}$ with $-\mathbf{d}$. This transformation is decomposed into $(n-1)$ successive rotations, each of which is on a two-dimensional plane defined by two coordinate axes.

These rotations are illustrated with a 3-dimensional example in Figure 3.7. The median axis $\mathbf{q}$ of the simplex lies on the axis x_1. The bold line segment represents the vector $-\mathbf{d}$ to rotate. After the first rotation by the angle θ_1 around the axis x_1, the new vector is on the plane (x_1, x_2). The second rotation by the angle θ_2 is around the axis x_3 to finally align the vector with $\mathbf{q}$. The required transformation $\mathbf{M}$ is then obtained by computing the inverse of $\mathbf{R}$, that is $\mathbf{M} = \mathbf{R}^{-1}$.

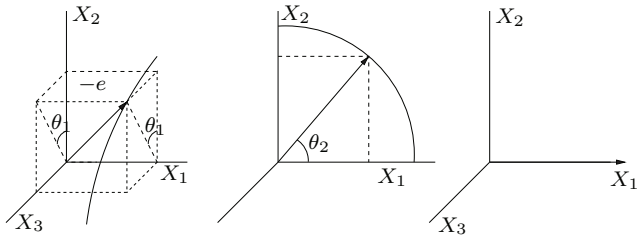

Figure 3.7: Successive rotations needed to align a vector with the axis x_1.

Once the simplex is correctly oriented, we want to position it in the state space such that we tend to maximize the staying time criteria. Intuitively, we can see this operation as translating the simplex along the *average evolution direction* $\mathbf{d}$ while satifying the inclusion constraint as shown in Figure3.8.

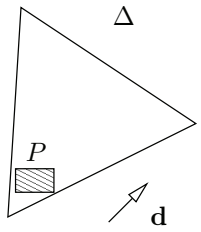

Figure 3.8: Placement of the simplex around the set P with respect to $\mathbf{d}$

This translation can be computed by solving the following optimization problem:

$$max\{\mathbf{d} \cdot \mathbf{t} \mid P \subset translation(\Delta, \mathbf{t})\}$$

Where $\mathbf{t} \in \mathbb{R}^n$ and $translation(\Delta, \mathbf{t})$ represents the simplex obtained by translating Δ by $\mathbf{t}$.

Note that this optimization failed if $\mathcal{P}$ cannot be contained in the simplex Δ. In the following subsection, we present some methods to deal with this problem.

3.5.3 Set splitting

In some cases, the set $\mathcal{P}$ can be elongated or just too big to be contained in a simplex satisfying the error bound. Then, to preserve a good precision, we can split the set $\mathcal{P}$ into smaller sets and continue the analysis with each set separately. We present now some heuristics to efficiently split the set $\mathcal{P}$.

An intuitive criterion for splitting a set is to detect the edge with the highest length and use an orthogonal slicing hyperplane to create two new sets. The costs of the edge length detection and the slicing operation depend strongly on the class of set representations.

An intuitive method uses bounding box approximation. We first consider axis aligned boxes which can be represented by two vectors $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ (see the set representation section page 24). Computing the bounding boxes of $\mathcal{P}$ is in general a trivial task.

We can then identify the axis along which the set is the most elongated by computing $max_{i \in \{1,\ldots,n\}} u_i - l_i$ then split our initial polytope using the hyperplane $x_i = (u_i + l_i)/2$. Figure 3.9-A illustrates this method; we can see

that the splitting hyperplane is an an axis with no consideration of the shape of the set.

A more sophisticated method involves using oriented boxes that keep a global shape information as shown in Figure 3.9-B. An oriented box can also be represented by two vectors $\mathbf{l}$ and $\mathbf{u}$ in conjunction with an n by n orthogonal matrix which represents the oriented bounding box axes. These axes can be determined by using advanced mathematical procedures like Principal Component analysis (PCA) [56].

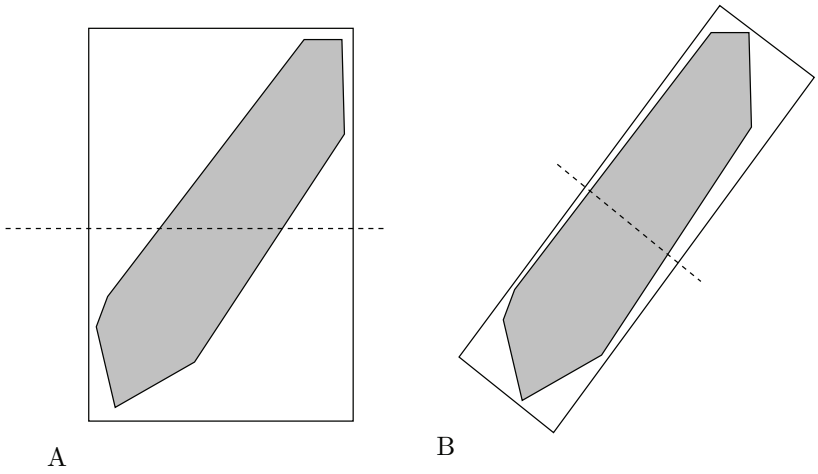

Figure 3.9: Set slicing using bounding box

The following section is concerned with a proof of the optimality of this construction method for the class of quadratic systems.

3.6 Optimal domain for quadratic functions

We show a class of quadratic functions f for which the domain construction based on equilateral simplices in an isotropic space is optimal. This optimality property is stated as follows: given an error tolerance ε, the computed simplex Δ has the largest volume and, in addition, the error between f and its linear interpolation over Δ is not greater than ε.

Let each quadratic function f_i be written as

$$f_i(\mathbf{x}) = \mathbf{x}^T \mathbf{H}^i \mathbf{x} + (\mathbf{m}^i)^T \mathbf{x} + p^i$$

where $\mathbf{H}^i$ is a real-valued matrix of size $n \times n$, $\mathbf{m}^i \in \mathbb{R}^n$ and $p^i \in \mathbb{R}$. Note that we use the same notation $\mathbf{H}^i$ here because the Hessian matrix of f_i is exactly

$\mathbf{H}^i$. For every $i \in \{1, \ldots, n\}$, we define the interpolation error function as

$$e^i(\mathbf{x}) = f_i(\mathbf{x}) - l^i(\mathbf{x})$$

which is also a quadratic function. We now study this error function and seek its maxima.

The error function can be expressed as:

$$e^i(\mathbf{x}) = (\mathbf{w}^i)^T\mathbf{x} + q^i + \mathbf{x}^T\mathbf{H}^i\mathbf{x}$$

where $\mathbf{w}^i \in \mathbb{R}^n$ and $q^i \in \mathbb{R}$. Note that the level sets of this function form a family of conics with a common center, denoted by $\mathbf{c}^i$. Indeed, they are ellipsoids if $det(\mathbf{H}^i) > 0$ and hyperboloids if $det(\mathbf{H}^i) < 0$. We now derive the error in a neighborhood of this common center. Let $\boldsymbol{\delta} \in \mathbb{R}^n$ be a deviation from the common center $\mathbf{c}^i$, then for every $i \in \{1, \ldots, n\}$

$$\begin{aligned} e^i(\mathbf{c}^i + \boldsymbol{\delta}) &= (\mathbf{w}^i)^T(\mathbf{c}^i + \boldsymbol{\delta}) + q^i - (\mathbf{c}^i + \boldsymbol{\delta})^T\mathbf{H}^i(\mathbf{c}^i + \boldsymbol{\delta}) \\ &= [(\mathbf{w}^i)^T\mathbf{c} + q^i - (\mathbf{c}^i)^T\mathbf{H}^i\mathbf{c}^i] + \\ &\quad (\mathbf{w}^i)^T\boldsymbol{\delta} - 2\boldsymbol{\delta}^T\mathbf{H}^i\mathbf{c}^i - \boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta} \\ &= e^i(\mathbf{c}^i) + (\mathbf{w}^i)^T\boldsymbol{\delta} - 2(\mathbf{c}^i)^T\mathbf{H}^i\boldsymbol{\delta} - \boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta}. \end{aligned}$$

Since $\mathbf{c}$ is the common center of the family of conics corresponding to the error function, $\mathbf{c}$ satisfies the following

$$\mathbf{w}^i - 2\mathbf{H}^i\mathbf{c}^i = 0.$$

Then,

$$\begin{aligned} (\mathbf{w}^j)^T\boldsymbol{\delta} &= 2(\mathbf{H}^i\mathbf{c}^i)^T\boldsymbol{\delta} \\ &= 2(\mathbf{c}^i)^T\mathbf{H}^i\boldsymbol{\delta}. \end{aligned}$$

It then follows from the above that

$$e^i(\mathbf{c}^i + \boldsymbol{\delta}) = e^i(\mathbf{c}^i) - \boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta}.$$

We also observe that, the symmetric matrix $\mathbf{H}^i$ can be decomposed as

$$\mathbf{H}^i = \mathbf{S}\mathbf{W}^T\mathbf{D}\mathbf{W}\mathbf{S}^T$$

where $\mathbf{D}$ is a diagonal matrix with entries $\sigma_j \in \{-1, 0, +1\}$; $\mathbf{W}$ is a diagonal matrix whose entries on the diagonal are the square roots of the absolute values of the eigenvalues ξ^j of $\mathbf{H}^i$; $\mathbf{S}$ is an orthonormal matrix containing the eigenvectors of $\mathbf{H}^i$. We define a linear transformation

$$\mathbf{T}^i = \mathbf{W}^T\mathbf{S}^T.$$

Lemma 6. *Using the transformation* $\widehat{\boldsymbol{\delta}} = \mathbf{T}^i\boldsymbol{\delta}$*, the term* $\boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta}$ *in the error* $e^i(\mathbf{c}^i + \boldsymbol{\delta})$ *can be transformed into a quadratic form*

$$\boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta} = \sum_{j=1}^{n} \sigma_1^i \widehat{\delta}_j^2$$

where for all $j \in \{1, \ldots, n\}$ $\sigma_j^i \in \{-1, 0, +1\}$.

Proof. Using $\boldsymbol{\delta} = \mathbf{T}^{-1}\widehat{\boldsymbol{\delta}}$ and $\mathbf{H}^i = \mathbf{S}\mathbf{W}^T\mathbf{D}\mathbf{W}\mathbf{S}^T$, we obtain after some straightforward calculation:

$$\begin{aligned} \boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta} &= (\mathbf{T}^{-1}\widehat{\boldsymbol{\delta}})^T\,\mathbf{S}\mathbf{W}\,\mathbf{D}\,\mathbf{W}^T\mathbf{S}^T\mathbf{T}^{-1}\widehat{\boldsymbol{\delta}} \\ &= (\widehat{\boldsymbol{\delta}})^T\mathbf{D}\widehat{\boldsymbol{\delta}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \boldsymbol{\delta}^T\mathbf{H}^i\boldsymbol{\delta} &= \widehat{\boldsymbol{\delta}}^T \begin{pmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 \\ & & \ldots & \\ 0 & 0 & \ldots & \sigma_n \end{pmatrix} \widehat{\boldsymbol{\delta}} \\ &= (\sigma_1\widehat{\delta}_1^2 + \sigma_2\widehat{\delta}_2^2 + \ldots + \sigma_n\widehat{\delta}_n^2). \end{aligned}$$

where $\forall j \in \{1, \ldots, n\} : \sigma_j \in \{-1, 0, +1\}$. In other words, using the linear transformation $\mathbf{T}^i$ we transform the matrix $\mathbf{H}^i$ into a diagonal matrix $\mathbf{D}$ which has only entries 0, +1 and −1 on the diagonal. □

Again, we can see that the interpolation error in the new space (resulting from the transformation $\mathbf{T}$) is isotropic, that is it does not depend on the direction of $\widehat{\boldsymbol{\delta}}$.

We identify a class of quadratic systems such that for every function f_i, σ_j^i are all equal to either +1 or −1. In the isotropic space the level sets of the error are spheres with a common center. The circumsphere of $\widehat{\Delta}$ is the level set of value zero (due to interpolation over the vertices). Hence, the maximal value of $|e^i(\mathbf{x})|$ is achieved at $\mathbf{c}^i$ and is directly related to the square of the radius of the circumsphere of $\widehat{\Delta}$.

Using the above reasoning, we can determine the maximal value of every error function $|e^i(\mathbf{x})|$ $(i \in \{1, \ldots, n\})$. Let f_i be the function that corresponds to the largest value. We then take the associated matrix $\mathbf{T}^i$ to be the isotropic transformation $\mathbf{T}$ for domain construction purposes. Note that in this case, the circumsphere radius is also the radius of the smallest containment ball of

Δ in the Theorem 3. For a given fixed circumsphere radius (corresponding to an error tolerance), an equilateral simplex has the optimal shape because it has the largest volume.

The number of entries $+1$ of $\mathbf{D}$ is called the positive index of inertia of $\mathbf{H}^i$, and the number of entry -1 is called the negative index of inertia. According to Sylvester's law of inertia, the positive and negative indices of $\mathbf{H}^i$ are also the number of positive and negative eigenvalues of $\mathbf{H}^i$.

Theorem 5. *For the class of quadratic functions f such that all the Hessian matrices have either only positive eigenvalues or only negative eigenvalues, the domain construction method based on equilateral simplices in the isotropic space is optimal.*

When this condition on the eigenvalues is not satisfied, the theorem no longer holds, that is starting from equilateral simplices in the isotropic space does not yield an optimal construction. For example, in 2 dimensions, in the case where the number of σ_j equal to $+1$ is equal to that of σ_j equal to -1 (which implies that the error is a harmonic function of $\widehat{\boldsymbol{\delta}}$), the maximal error is not achieved at the common center but on the boundary of the simplex. Investigating the optimality conditions for the remaining cases is part of our undergoing work.

In the next section, we present some experimental results.

3.7 Experimental results

We implemented the domain construction algorithm and tested in on various examples. For nonlinear optimization we use the publicly available NLopt library [55] which provides a common interface for a number of different optimization algorithms. For the computation of reachable sets of the approximate piecewise-affine systems, we used the algorithms implemented in the tool d/dt [10].

We first illustrate the interest of the algorithm by a number of experiments on a 2-dimensional system, the dynamics of which is described as follows:

$$\dot{x}_1 = x_2 - x_1^3 + x_1 x_2^2 \tag{3.15}$$

$$\dot{x}_2 = x_1^3 \tag{3.16}$$

This example is adapted from the one used in the study of stabilization of systems with uncontrollable linearization in [89] (page 346). The initial set is a small box $[0.5, 0.51] \times [0.5, 0.51]$. The error tolerance is equal to 0.5.

Figure 3.10 shows the reachable set computed using a hybridization without isotropic transformation. The hybridization domains are chosen to be equilateral and oriented along the local evolution direction of the dynamics. We can see that without using an isotropic transformation, the domains are small and thus a lot of domains were created.

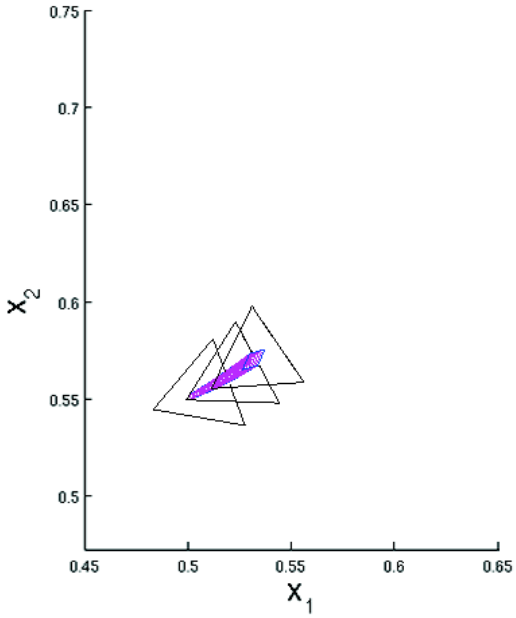

Figure 3.10: Domains constructed without isotropic transformation.

Figure 3.11 shows the reachable set computed using a hybridization with an isotropic transformation. In this experiment, the curvature estimation was done dynamically when each domain is created. We can see that the domains are larger for the same accuracy and less domains are needed.

The reachability computation can be further improved by using smaller optimization domains around the reachable sets. This in general requires some rough approximation of the reachable set within a number of next iterations. This is illustrated by the reachable set shown in Figure 3.12.

In order to illustrate the effect of error bounds, we fix the radius of the smallest containment ball in the isotropic space and perform two experiments: the first one with the curvature tensor estimated over a large zone and the second over a smaller zone. We observe that the hybridization domains in the two experiments are the same. However, a high curvature bound was computed in the first experiment and thus the corresponding error bound is

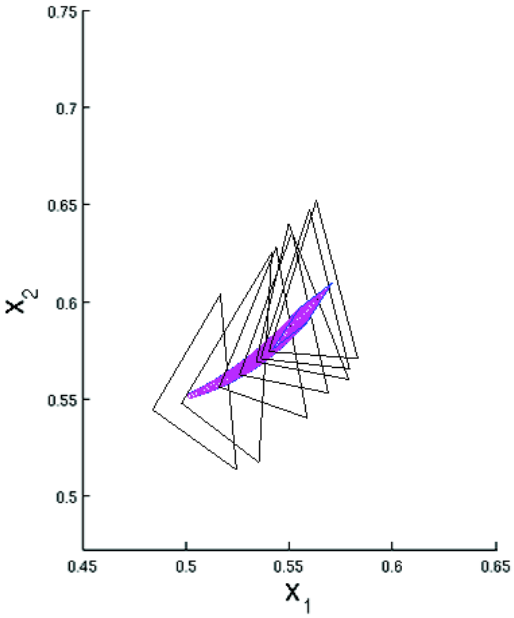

Figure 3.11: Domains constructed with the curvature tensor dynamically estimated over large zones.

large, which results in a large input set. This causes the system to expand fast, as one can see in Figure 3.7. On the other hand, with a better curvature estimate in the second experiment, the error bound is smaller and the reachable set computation is more accurate (see Figure 3.7).

In order to evaluate the performance of the algorithm, we performed a set of experiments on some polynomial systems (of degree 4) which are randomly generated. We report in the following the average computation times of 100 iterations for systems up to 7 dimensions. For each dimension, we tested 4 systems. In these experiments, for each system the curvature tensor matrix was estimated only once. The reason we did not go beyond 7 dimensions is that the optimization took a lot of computation time (while the computation time for treating approximate piecewise affine systems is much less), as shown in Figure 3.15. Indeed, for a n-dimensional system, to estimate the curvature tensor matrix, we need to compute n eigenvalues, each of which requires solving n constrained optimization problems with $2n$ variables. Indeed, the curvature tensor estimation can be done a-priori over a large analysis zone and such a global estimate can be used for the whole reachability computation process. This alone can significantly improve the accuracy of the reachable set approximation, compared to the domain construction without isotropic

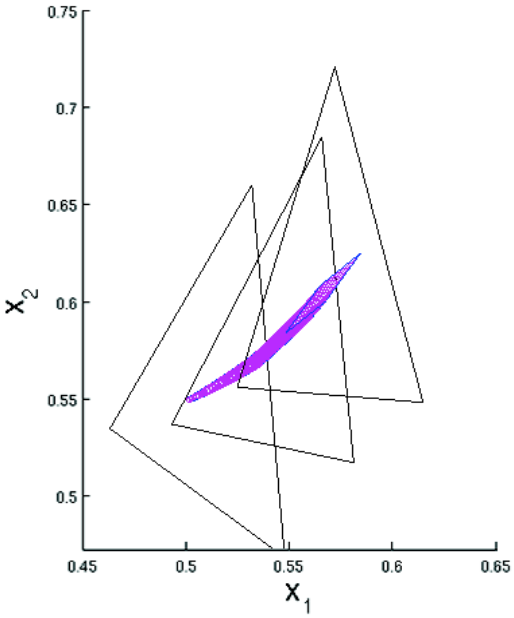

Figure 3.12: Domains constructed with the curvature tensor dynamically estimated over small zones.

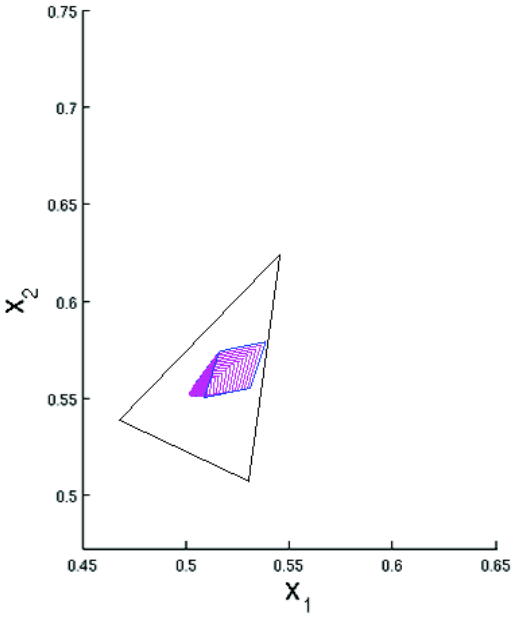

Figure 3.13: Reachability computation with a large error bound.

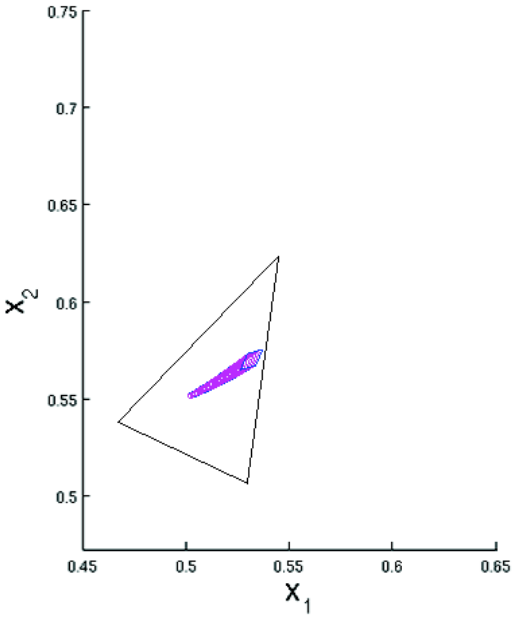

Figure 3.14: Reachability computation with a smaller error bound.

transformation, as shown in the above 2-dimensional example. In order to include dynamics curvature estimation, we need more performant optimization tools, such as those for specific classes of systems.

Dimension	Total time (s)	Optimisation time (s)
2	0.53	0.05
3	0.96	0.63
4	7.87	7.01
5	57.05	48.22
6	90.77	80.78
7	302.5	269.22

Figure 3.15: Computation times for polynomial systems.

To sum up, our preliminary experiments demonstrated the interest of the proposed domain construction algorithm in terms of accuracy improvements. The practical efficiency of the algorithm is still limited by the required nonlinear optimization.

Chapter 4

Reachability analysis for polynomial dynamical systems

Résumé: *Nous présentons dans ce chapitre, nos contribution concernant l'analyse d'accessibilité des systèmes polynomiaux. Nous présentons une approche basée sur la formulation du problème de calcul d'image en un problème d'optimisation polynomial. Nous présentons ensuite certains mécanismes permettant de réduire ce problème à des problèmes d'optimisation linéaires, en utilisant des fonctions affines de borne. Nous présentons ensuite des méthodes de calcul pour ces fonctions basées sur l'expansion de Bernstein. Nous présentons ensuite de nouvelles stratégies pour choisir des patrons utilisés pour fixer préalablement la forme des polyhèdres, de façon à refléter l'effet de la dynamique sur l'orientation des faces du polyhèdre approximant l'image. Nous finissons par présenter certaines études de cas sur plusieurs systèmes biologiques.*

While the hybridization technique can be used on a large class of nonlinear dynamics, other techniques can be developed to deal more efficiently with particular class of nonlinear dynamics, such as polynomial dynamics.

In this chapter, we focus on the following computation problem: given a set of initial states in $\mathbb{R}^n$, compute the trajectory set of a discrete-time dynamical system described by the following difference equation:

$$\mathbf{x}(k+1) = \pi(\mathbf{x}(k)) \tag{4.1}$$

where $\pi : \mathbb{R}^n \to \mathbb{R}^n$ is a multivariate polynomial.

Our interest in polynomial systems is motivated by their applicability in modeling a variety of phenomena in bio-chemical networks and economy. To extend to continuous-time models, a time discretization is needed and conservativeness of approximation needs to be guaranteed. It is however important to note that discrete-time systems can also be directly used to model many biological systems, since experimental data are often obtained by sampling biochemical reaction outputs, and in addition, such models are ready to be used for computer aided analysis and numerical simulation.

The problem (4.1) for polynomial systems was previously considered in the work [34, 39], which was inspired by the techniques from Computer Aided Geometric Design (CADG). In this chapter, we pursue the direction by exploiting further the special properties of a technique from CADG, namely the Bernstein expansion, we only need to solve linear programming (LP) problems instead of polynomial optimization problems.

In the first section, we present some preliminary notions concerning a special kind of convex polyhedra and the Bernstein expansion of polynomial functions. We then address the reachability computation by formulating an optimization problem and we show how to use bound functions to get a linear relaxation of this problem. The next section presents different algorithms to compute these bound functions over the unit box domain. The next section propose some methods for extending these techniques to polyhedral domains. Finally we present an implementation of these algorithms and some experimental results.

4.1 Preliminaries

4.1.1 Template polyhedra

When the system starts from an initial set $\mathcal{P}^0$, we have to deal with set of solutions. To charaterize this set of solutions we use a special kind of $\mathcal{H}Polytopes$ with fixed geometric form, called *template polyhedra* [87, 27]. Template polyhedra are commonly used in the static analysis of programs for computing invariants. Ranges [33] and the octagon domains [69] are special template polyhedra. General template polyhedra are used as an abstract domain to represent sets of states in [87, 27]. A *template* is a set of normal direction given by vectors in $\mathbb{R}^n$. We denote a template by an $m \times n$ matrix $\mathbf{T}$ which correspond to the matrix $\mathbf{A}$ of the polyhedron representation in Section 2.2.1 page 27. Given such a template $\mathbf{T}$ and a *polyhedral coefficient vector* $\mathbf{c} \in \mathbb{R}^m$, a template polyhedron is defined by considering the

conjunction of the linear inequalities of the form

$$\bigwedge_{i=1,\ldots,m} \mathbf{T}_i\mathbf{x} \leq c_i.$$

We denote this polytope by $\langle \mathbf{T}, \mathbf{c} \rangle$. By varying the values of the elements of $\mathbf{c}$, one can create a family of template polyhedra corresponding to the template $\mathbf{T}$.

The advantage of template polyhedra over general convex polyhedra is that the Boolean operations (union, intersection) and common geometric operations can be performed more efficiently [87].

4.1.2 Bernstein expansion

To discuss the Bernstein expansion of polynomials, we use multi-indices of the form $\mathbf{i} = (i_1, i_2, \ldots, i_n)$ where each i_j is a non-negative integer. Given two multi-indices $\mathbf{i}$ and $\mathbf{d}$, we write $\mathbf{i} \leq \mathbf{d}$ if for all $j \in \{1, \ldots, n\}$, $i_j \leq d_j$. Also, we write $\frac{\mathbf{i}}{\mathbf{d}}$ for $(i_1/d_1, i_2/d_2, \ldots, i_n/d_n)$ and $\binom{\mathbf{i}}{\mathbf{d}}$ for the product of binomial coefficients $\binom{i_1}{d_1}\binom{i_2}{d_2}\ldots\binom{i_n}{d_n}$. In addition we use the *unit box* that we denote by $\mathcal{B}_u$ which corresponds to the interval product $[0, 1]^n$

A n-variate polynomial $\pi : \mathbb{R}^n \to \mathbb{R}^n$ can be represented using the power base as follows:

$$\pi(\mathbf{x}) = \sum_{\mathbf{i} \in \mathsf{I}_\mathbf{d}} \mathbf{a}^\mathbf{i}\mathbf{x}^\mathbf{i}$$

where $\mathbf{a}^\mathbf{i}$ is a vector in $\mathbb{R}^n$; $\mathbf{x}^\mathbf{i}$ corresponds to the monomial term $\prod_{k=1}^{n} \mathbf{x}_k^{i_k}$; $\mathbf{i}$ and $\mathbf{d}$ are two multi-indices of size n such that $\mathbf{i} \leq \mathbf{d}$; $\mathsf{I}_\mathbf{d}$ is the set of *all* multi-indices $\mathbf{i} \leq \mathbf{d}$, that is $\mathsf{I}_\mathbf{d} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{d}\}$. The multi-index $\mathbf{d}$ is called the *degree* of π.

The polynomial π can also be represented using the Bernstein expansion. In order to explain this, we first introduce Bernstein polynomials. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the $\mathbf{i}^{th}$ Bernstein polynomial of degree $\mathbf{d}$ is defined as follows:

$$B^{\mathbf{d},\mathbf{i}}(\mathbf{x}) = \beta^{d_1,i_1}(x_1)\ldots\beta^{d_n,i_n}(x_n) \tag{4.2}$$

where for a real number y, $\beta^{d_j,i_j}(y) = \binom{d_j}{i_j} y^{i_j}(1 - y^{d_j - i_j})$.

Then, for all $\mathbf{x} \in \mathcal{B}_u$, the polynomial π can be written using the Bernstein expansion as follows:

$$\pi(\mathbf{x}) = \sum_{\mathbf{i} \in \mathsf{I}_\mathbf{d}} \mathbf{b}^\mathbf{i} B^{\mathbf{d},\mathbf{i}}(\mathbf{x})$$

where for each $\mathbf{i} \in \mathsf{I}_\mathsf{d}$ the Bernstein coefficient $\mathbf{b}^\mathbf{i}$ is defined as:

$$\mathbf{b}^\mathbf{i} = \sum_{\mathbf{j} \leq \mathbf{i}} \frac{\binom{\mathbf{i}}{\mathbf{j}}}{\binom{\mathbf{d}}{\mathbf{j}}} \mathbf{a}^\mathbf{j}. \tag{4.3}$$

The following lemma presents some important properties on the Bernstein coefficients.

Lemma 7. *Bernstein coefficient properties:*

1. *Convex-hull property:*

$$Conv\{(\mathbf{x}, \pi(\mathbf{x})) \ : \ \mathbf{x} \in \mathcal{B}_u\} \subseteq Conv\{(\mathbf{i}/\mathbf{d}, \mathbf{b}^\mathbf{i}) \mid \mathbf{i} \in \mathsf{I}_\mathsf{d}\}.$$

   The points $((\mathbf{i}/\mathbf{d}, \mathbf{b}^\mathbf{i})$ *are called the control points of* π*.*

2. *The above enclosure yields:*

$$\forall \mathbf{x} \in \mathcal{B}_u \ : \ \pi(\mathbf{x}) \in \square(\{\mathbf{b}^\mathbf{i} \mid \mathbf{i} \in \mathsf{I}_\mathsf{d}\})$$

   where $\square$ *denotes the bounding box of a point set.*

3. *Sharpness of some special coefficients:*

$$\forall \mathbf{i} \in \mathsf{I}_\mathsf{d}^0 : \ \mathbf{b}^\mathbf{i} = \pi(\mathbf{i}/\mathbf{d}),$$

   where I_d^0 *is the set of all the vertices of the box described by the interval product* $[0, d_1] \times [0, d_2] \ldots \times [0, d_n]$*.*

The first two properties can be visually illustrated by plotting a one-dimensional system with its control points as shown in the following example.

Example Figure 4.1 represents the control points of an arbitrary polynomial of degree 5 given by:

$$\pi(x) = 1 - x + 3x^2 - x^3 + 2x^4 - 2.5x^5.$$

The set of all multi-indices for this example is $\mathsf{I}_\mathsf{d} = \{0, 1, 2, 3, 4, 5\}$. Using the formula (4.3) we compute for each $\mathbf{i} \in \mathsf{I}_\mathsf{d}$ the corresponding Bernstein coefficient which gives us the vector ordered by the increasing order of the multi-indices $\mathbf{i}$:

$$\mathbf{B}_{\text{coef}} = (1, 0.8, 0.9, 1.2, 2, 1.5).$$

We observe the inclusion of the polynomial in the convex hull of the control points represented by the dotted polygon.

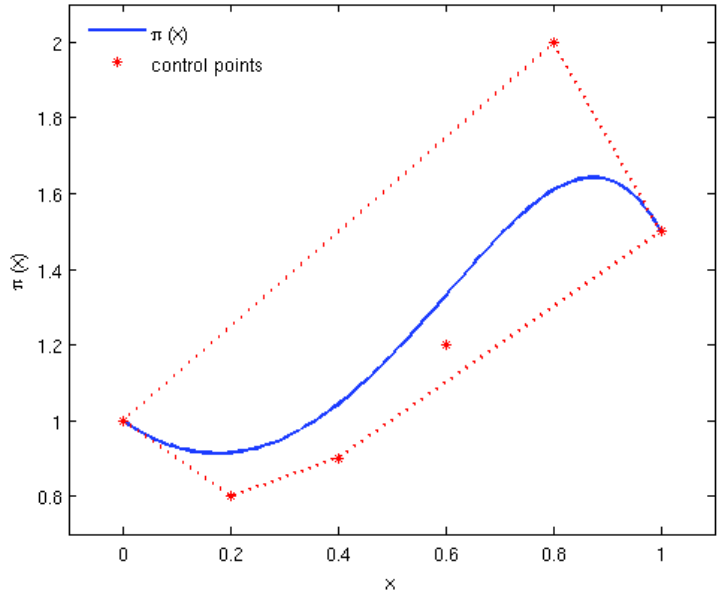

Figure 4.1: Control points of a one dimensional polynomial system of degree 5

Since we are concerned with dealing with n-dimensional polynomial dynamics, we will consider $\mathbf{B}_{\text{coef}}$ as a matrix whose columns are indexed by $\mathbf{i}/\mathbf{d}$ with $\mathbf{i} \leq \mathbf{d}$ and lines correspond to indexes of the polynomial $\pi_k,\ k \leq n$.

With respect to our reachability problem that requires computing the image of a set by a polynomial system, the Bernstein expansion is of particular interest, since they can be used to efficiently compute affine bounds for the system, as shown in Section 4.2.2.

4.2 Reachable set approximation using template polyhedra

In this section, we present two approaches for computing the reachable set, the first one concerns only multi-affine systems. The second one concerns more general polynomial dynamics and is based on the formulation of a polynomial optimization problem and then is reduced to a problem of linear optimization using bound functions.

4.2.1 Method for multi-affine functions

We propose a method specialized for multi-affine systems, a particular case of polynomial systems. Multi-affine systems are systems composed by polynomials which are affine in each of their variables, i.e. if $\mathbf{d}$ is the degree of an affine system π, $d_k \leq 1$ for all $k \in \{1, \ldots, n\}$. We will exploit the following property of such systems to obtain a time-efficient reachability algorithm.

Theorem 6. *Given a box $\mathcal{B} \subseteq \mathbb{R}^n$, let $V_{\mathcal{B}}$ be the set of its vertices. If π is a multi-affine function, then*

$$\pi(\mathcal{B}) \subseteq conv\{\pi(\mathbf{v}) \mid \mathbf{v} \in V_{\mathcal{B}}\}$$

A proof of this well-known property of multi-affine functions can be found in [20].

Note that the above theorem is only applicable to the sets which are axis-aligned hyper-rectangles. Hence, even if the initial set satisfies this condition, after the application of π, the resulting set is a general convex polyhedron and we need to approximate it by an axis-aligned hyper-rectangle. Using the theorem, to compute $\pi(\mathcal{B})$ it suffices to compute the images of the vertices of $\mathcal{B}$ and then take the convex hull of the resulting points.

Before continuing, we remark that a number of different methods for the multi-affine systems have also been developed in [19, 22]. These methods are however based on a rectangular partition of the state space, while we allow reachable sets to be represented by unions of polyhedra.

We now present the second method for more general polynomial dynamics.

4.2.2 Optimization-based method

Optimization problem formulation

To compute the reachable set from a template polyhedron, at each time step, we need to compute the image of a polyhedron $\mathcal{P}$ by the polynomial π. The template matrix $\mathbf{T}$, which is of size $m \times n$, is assumed to be given; the polyhedral coefficient vector $\mathbf{c} \in \mathbb{R}^m$ is, however, unknown. The problem we now focus on is thus to find $\mathbf{c}$ such that

$$\pi(\mathcal{P}) \subseteq \langle \mathbf{T}, \mathbf{c} \rangle. \tag{4.4}$$

For safety verification purposes, exact computation of reachable sets is often not possible (due to undecidability issues for example) and one thus needs to resort to over-approximations, and when an over-approximation does not allow proving a safety property, the approximation needs to be refined.

It is not hard to see that the following condition is sufficient for (4.4) to hold:

$$\forall \mathbf{x} \in \mathcal{P} \ : \ \mathbf{T}\pi(\mathbf{x}) \leq \mathbf{c}.$$

Therefore, $\forall i \in \{1, \ldots, m\}$ to determine c_i, one can formulate the following optimization problems:

$$\begin{array}{ll} \text{maximize} & \Sigma_{k=1}^{n} T_{i,k}\pi_k(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathcal{P}. \end{array} \tag{4.5}$$

where $\mathbf{T}_i$ is the i^{th} row of the matrix $\mathbf{T}$ and $T_{i,k}$ is its k^{th} element. Note that the above functions to optimize are polynomials. This problem is computationally difficult, despite recent progress in the development of methods and tools for polynomial programming (see for example [45]). An alternative solution is to find their affine bound functions as discussed in the next subsection.

Approximation using bound functions

Affine bound functions can be used in order to replace the polynomial optimization problem by a linear programming one, which can be solved more efficiently (in polynomial time) using well-developed techniques, such as Simplex and interior point techniques [84]. We first give a formal definition for these functions.

Definition 16 (Upper and lower bound functions)**.** *Given $f : \mathbb{R}^n \to \mathbb{R}$, the function $u : \mathbb{R}^n \to \mathbb{R}$ is called an upper bound function of f w.r.t. a set $\mathcal{P} \subset \mathbb{R}^n$ if $\forall \mathbf{x} \in \mathcal{P} : \ f(\mathbf{x}) \leq u(\mathbf{x})$. A lower bound function can be defined similarly.*

The following property of upper and lower bound functions is easy to prove.

Lemma 8. *Given $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$ s.t. $\mathcal{Y} \subseteq \mathcal{X}$, if v is an upper (lower) bound function of f w.r.t. $\mathcal{X}$, then v is an upper (lower) bound function of f w.r.t. $\mathcal{Y}$.*

For each $k \in \{1, \ldots, m\}$, let $u_k(\mathbf{x})$ and $l_k(\mathbf{x})$ respectively be an upper bound function and a lower bound function of $\pi_k(\mathbf{x})$ w.r.t. a bounded polyhedron $\mathcal{P} \subset \mathbb{R}^n$. Given a template matrix $\mathbf{T}$, we consider the following optimization problem:

$$\forall i \in \{1, \ldots, m\}, c_i = \Sigma_{k=1}^{n} T_{i,k}\omega_k. \tag{4.6}$$

where the term $T_{i,k}\omega_k$ is defined as follows:

- If the element $T_{i,k} > 0$,

$$T_{i,k}\omega_k = T_{i,k} \max u_k(\mathbf{x}) \text{ subject to } \mathbf{x} \in \mathcal{P}.$$

- If the element $T_{i,k} < 0$,

$$T_{i,k}\omega_k = T_{i,k} \min l_k(\mathbf{x}) \text{ subject to } \mathbf{x} \in \mathcal{P}.$$

The following lemma is a direct result of (4.6).

Lemma 9. *If $\mathbf{c} \in \mathbb{R}^m$ satisfies (4.6), then $\pi(\mathcal{P}) \subseteq \langle \mathbf{T}, \mathbf{c} \rangle$.*

Proof. It is indeed not hard to see that the solution c_i of the optimization problems (4.6) is greater than or equal to the solution of (4.5). Hence, if $\mathbf{c}$ satisfies (4.6), then $\forall i \in \{1, \ldots, m\}\ \forall \mathbf{x} \in \mathcal{P} :\ \Sigma_{k=1}^{n} T_{i,k}\pi_k(\mathbf{x}) \leq c_i$. This implies that $\forall \mathbf{x} \in \mathcal{P} :\ \mathbf{T}\pi(\mathbf{x}) \leq \mathbf{c}$, that is the image $\pi(\mathcal{P})$ is included in $\langle \mathbf{T}, \mathbf{c} \rangle$. □

We remark that if all the bound functions in (4.6) are affine and $\mathcal{P}$ is a bounded convex polyhedron, $\mathbf{c}$ can be computed by solving $2n$ linear programming problems. It remains now to find the affine bound functions $u_k(\mathbf{x})$ and $l_k(\mathbf{x})$ for π w.r.t. a polyhedron $\mathcal{P}$, which is the problem we tackle in the next section.

In the next section we present some methods to compute such bound functions.

4.3 Computing bound functions in the unit box domain

To compute bound functions, we use the methods based on the Bernstein expansion, published in [53]. Computing convex lower bound functions for polynomials is a problem of great interest, especially in global optimization. The reader is referred to [53, 54, 47] for more detailed descriptions of these methods.

It is important to note that the methods described in this section only work for the case where the variable domain is the unit box $\mathcal{B}_u$. The reason is that it employs the expression of the control points of the Bernstein expansion in (4.3) which is only valid for this unit box. Their extensions to arbitrary polyhedral domains are discussed in the next section. Therefore, in what follows, we assume that our initial polyhedron $\mathcal{P}$ is included in the unit box.

A simple lower bound function is a constant function, which can be directly deduced from the second property of the Bernstein expansion:

$$\pi_k(\mathbf{x}) \geq b_k^0$$

where $b_k^0 = \min\{\mathbf{b}_k^{\mathbf{i}} \mid \mathbf{i} \in \mathsf{I}_\mathsf{d}\}$. The lower bound function for the polynomial $\pi_k(\mathbf{x})$ is then:

$$l_k(\mathbf{x}) = b_k^{\mathbf{i}^0} = b_k^0.$$

Two methods for computing better bound functions are presented in the following sections, for each of them we illustrate their its algorithm with an example using a one-dimensional polynomial system presented on page 68.

4.3.1 Using a convex hull lower facet

The first step of this method [54] involves computing the affine lower bound function whose corresponding hyperplane passes through this control point $(\mathbf{i}^0\mathbf{b}^0)$. Then, additionally, $(n-1)$ hyperplanes passing through n other control points are determined. This allows constructing a sequence of n affine lower bound functions $l^0, l^1, \ldots l^n$. The method ends up with l^n, a function whose corresponding hyperplane passes through a lower facet of the convex hull spanned by these control points. Note that we can easily compute an upper bound function of π by computing a lower bound function for $-\pi$ using this method and then multiply each resulting function by -1.

We describe the algorithm for computing these functions published in [53]. Let us consider a polynomial $\pi_k(\mathbf{x})$, which is the k^{th} component of $\pi(\mathbf{x})$ and for simplicity, we denote it simply by $p(\mathbf{x})$. The Bernstein coefficient of p is denoted by the scalars $b_\mathbf{i}$. We shall compute an affine lower bound function denoted by $l(\mathbf{x})$.

- Iteration 1.
  - Define the direction $\mathbf{u}^1 = (1, 0, \ldots, 0)$.
  - Compute the slopes from each $b_\mathbf{i}$ to b^0 in the direction $\mathbf{u}^1$:

$$\forall \mathbf{i} \in \mathsf{I}_\mathsf{d} : i_1 \neq i_1^0,\ g_\mathbf{i}^1 = \frac{b_\mathbf{i} - b^0}{i_1/d_1 - i_1^0/d_1}$$

  - Let $\mathbf{i}^1$ be the multi-index with the smallest absolute value of $g_\mathbf{i}^1$. Define the lower bound function:

$$l^1(\mathbf{x}) = b^0 + g_{\mathbf{i}^1}^1 \mathbf{u}^1 \cdot (\mathbf{x} - \mathbf{i}^0/\mathbf{d}).$$

- Iteration $j = 2, \ldots, n$.
  - Compute the direction $\bar{\mathbf{u}}^j = (\beta_1, \ldots, \beta_{j-1}, 0, \ldots, 0)$ such that $\bar{\mathbf{u}}^j \cdot \left(\frac{\mathbf{i}^k - \mathbf{i}^0}{\mathbf{d}}\right) = 0$ for all $k = 1, \ldots, j-1$. This requires solving a system of $j-1$ linear equations with $j-1$ unknown variables. Then normalize $\mathbf{u}^j = \bar{\mathbf{u}}^j / ||\bar{\mathbf{u}}^j||$.
  - Compute the slopes from each $b_{\mathbf{i}}$ to b^0 in the direction $\mathbf{u}^j$:

$$\forall \mathbf{i} \in \mathsf{I}_\mathsf{d} : \frac{\mathbf{i} - \mathbf{i}^0}{\mathbf{d}} \cdot \mathbf{u}^j \neq 0, \; g^j_{\mathbf{i}} = \frac{b_{\mathbf{i}} - l^{j-1}(\mathbf{i}/\mathbf{d})}{(\mathbf{i}/\mathbf{d} - \mathbf{i}^0/\mathbf{d}) \cdot \mathbf{u}^j}$$

  - Let $\mathbf{i}^j$ be the multi-index with the smallest absolute value of $g^j_{\mathbf{i}}$. Define the lower bound function:

$$l^j(\mathbf{x}) = l^{j-1}(\mathbf{x}) + g^j_{\mathbf{i}^j} \mathbf{u}^j \cdot (\mathbf{x} - \mathbf{i}^0/\mathbf{d}).$$

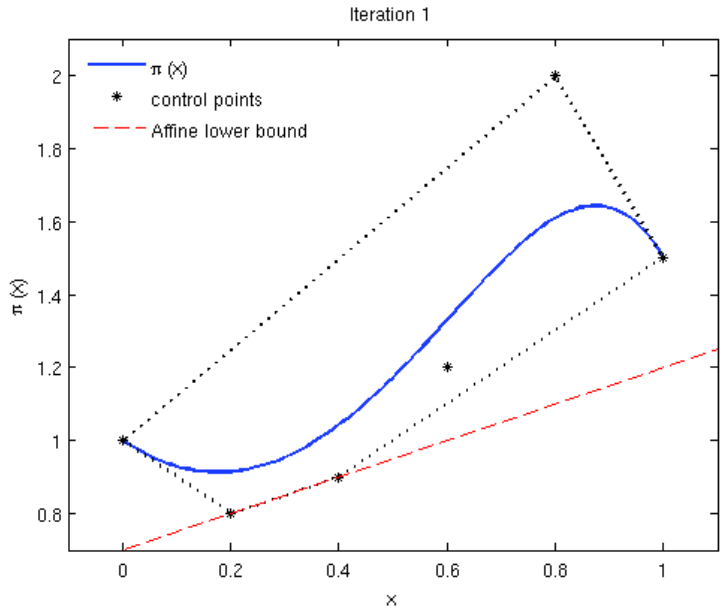

Figure 4.2: Affine lower bound functions passing by the lower facets of the control points convex hull

Example Figure 4.2 illustrates this algorithm for a one-dimensional case using the example presented in page 68. We recall that the Bernstein coefficients of this polynomial are

$$\mathbf{B}_{\text{coef}} = (1, 0.8, 0.9, 1.2, 2, 1.5)$$

We compute the lower bound in one iteration with the parameters $\mathbf{u}^1 = (1)$ and $b^0 = 0.8$. The smallest absolute value of the slope is computed with the $b_3 = 0.9$ with

$$g_3^1 = \frac{0.8 - 0.9}{0.4 - 0.6} = 0.5.$$

Then the computed lower bound is

$$l(x) = 0.8 + 0.5(x - 0.2) = 0.5x + 0.7$$

This method was previously proposed for the reachability analysis problem in [39], we now describe a new method to compute these affine bound functions which can produce more precise results.

4.3.2 Using linear least squares approximation

The essence of the second method [47] for computing bound functions is to find a hyperplane that is close to *all* the control points, using linear least squares approximation. This can lead to tighter bound functions since the general shape of the function graph is better captured. More concretely, we denote by $\{\mathbf{i}^j \mid 1 \leq j \leq n_b\}$ be the set of all the multi-indices, n_b is thus their number. The set of all control points is denoted similarly. Let A be a matrix of size $n_b \times (n+1)$ such that its elements are defined as follows. For all $1 \leq j \leq n_b$ and $1 \leq k \leq n$,

$$\mathbf{A}_k^j = \frac{i_k^j}{d_k}$$

and $\mathbf{A}_{n+1}^j = 1$. Let $\boldsymbol{\zeta}$ be the solution of the following linear least squares approximation problem:

$$\mathbf{A}^T\mathbf{A}\boldsymbol{\zeta} = \mathbf{A}^T\mathbf{b}. \tag{4.7}$$

Then, the affine function

$$\tilde{l}(\mathbf{x}) = \sum_{k=1}^{n} \zeta_k x_k + \zeta_{n+1}$$

corresponds to the "median" axis of the convex hull of all the control points. It thus suffices to shift it downward by the amount:

$$\delta = max\{\tilde{l}(\frac{\mathbf{i}^j}{\mathbf{d}}) - \mathbf{b}^j \mid 0 \leq j \leq n_b\}.$$

This results in a lower bound function illustrated at the right of Figure 4.3.

$$l(\mathbf{x}) = \tilde{l}(\mathbf{x}) - \delta, \text{ for all } \mathbf{x} \in \mathcal{B}_u.$$

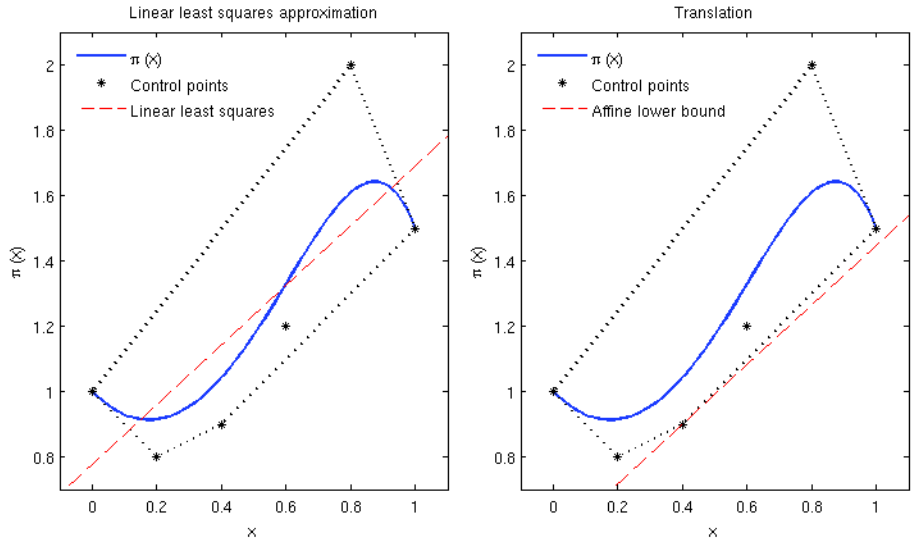

Figure 4.3: Computation of the affine lower bound functions using Linear Least Square approximation

Example Using the one dimensional system given page 68, we can compute the matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We recall that

$$\mathbf{b} = (1, 0.8, 0.9, 1.2, 2, 1.5).$$

We obtain from (4.7), the following problem

$$\begin{pmatrix} 2.2 & 3 \\ 3 & 6 \end{pmatrix} \begin{pmatrix} \zeta_1 \\ \zeta_2 \end{pmatrix} = \begin{pmatrix} 4.34 \\ 7.40 \end{pmatrix}.$$

By solving this linear equalities system we obtain the linear least square approximation

$$\tilde{l}(x) = \zeta_1 x + \zeta_2 = 0.9143x + 0.7762.$$

Then we compute $\delta = 0.2419$ with the coefficient b_3 and we get the affine lower bound

$$l(x) = \tilde{l}(x) - \delta = 0.9143x + 0.5343.$$

The left plot in Figure 4.3 shows the linear least squares approximation computed for the polynomial given in example. The second one represent the final affine function computed after shifting downward the previous affine function.

4.4 Extension to polyhedral domains

As mentioned earlier, the methods to compute affine bound functions for polynomials in Section 4.3 can be applied only when the set $\mathcal{P}$ is inside the unit box $\mathcal{B}_u$ anchored at the origin. To extend it to polyhedral domains, we have to perform a change of variable to map the initial polyhedron to the unit box. To this end, [39] presented a method based on box approximation. We propose another method on a change of variables. We also discuss a comparison between these two methods.

Throughout this section we use the following 2 dimensional system as an illustrative example for each method:

$$\begin{aligned} \pi_1(\mathbf{x}) &= x_1^2 + 2x_2 + 1, \\ \pi_2(\mathbf{x}) &= x_2^2 - x_2 + 2, \end{aligned}$$

with $x_1, x_2 \in \mathbb{R}$ and an initial set represented by a triangle which is not included in the unit box, with the following vertices: $(1,1), (3,3), (4,1)$.

4.4.1 Using a box approximation

If we over-approximate $\mathcal{P}$ with a box $\mathcal{B}$, it is then possible to derive a formula expressing the Bernstein coefficients of π over $\mathcal{B}$. However, this formula is complex and its representation and evaluation can become expensive.

We alternatively consider the composition of the polynomial π with an affine transformation τ that maps the unit box to $\mathcal{B}$. The functions resulting from this composition are still polynomials, for which we can compute their bound functions over the unit box, using the formula (4.3) of the Bernstein expansion. This is explained more formally in the following.

Let $\mathcal{B}$ be the bounding box of the polyhedron $\mathcal{P}$ represented by the vector $\underline{\mathbf{r}} \in \mathbb{R}^n$ and $\overline{\mathbf{r}} \in \mathbb{R}^n$. The affine function τ that maps the unit box $\mathcal{B}_u$ to $\mathcal{B}$ can be easily defined as: $\tau(\mathbf{x}) = diag(\lambda)\mathbf{x} + \mathbf{g}$ where $\mathbf{g} \in \mathbb{R}^n$ such that $g_i = \underline{r}_i$, and $diag(\lambda)$ is a $n \times n$ diagonal matrix with the elements on the diagonal defined as follows: for each $i \in \{1, \ldots, n\}$, $\lambda_i = \overline{r}_i - \underline{r}_i$.

The composition $\gamma = (\pi \; o \; \tau)$ is defined as $\gamma(\mathbf{x}) = \pi(\tau(\mathbf{x}))$. The functions τ and γ can be computed symbolically, which will be discussed later.

Lemma 10. *Let $\gamma = \pi \; o \; \tau$. Then, $\pi(\mathcal{P}) \subseteq \gamma(\mathcal{B}_u)$.*

Proof. By the definition of the composition γ, $\gamma(\mathcal{B}_u) = \{\pi(\tau(\mathbf{x})) \mid \mathbf{x} \in \mathcal{B}_u\}$. Additionally, $\tau(\mathcal{B}_u) = \mathcal{B}$. Therefore, $\gamma(\mathcal{B}_u) = \pi(\mathcal{B})$. Since the polyhedron $\mathcal{P}$ is included in its bounding box $\mathcal{B}$, we thus obtain $\pi(\mathcal{P}) \subseteq \pi(\mathcal{B}) = \gamma(\mathcal{B}_u)$. □

We remark that the above proof is still valid for any affine function τ. This means that instead of an axis-aligned bounding box, we can over-approximate $\mathcal{P}$ more precisely with an oriented (i.e. non-axis-aligned) bounding box. The directions of an oriented bounding box can be computed using Principal Component Analysis (PCA) [56]. A detailed description of the method can be found in [39].

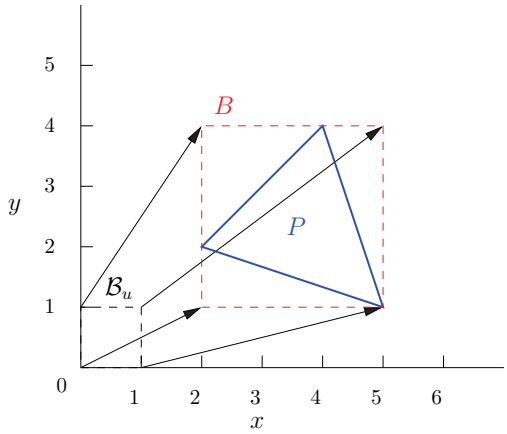

Figure 4.4: Box approximation of P and mapping function τ

Example Figure 4.4 shows the initial set of the section example and its box approximation oriented with axis in this case defined by the intervals $[2, 5] \times [1, 4]$. The transformation τ is illustrated by the arrows showing the mapping the unit box $\mathcal{B}_u$ vertices to the box $\mathcal{B}$, we denote the equations of

τ by τ_1 and τ_2 such that:

$$\begin{aligned}\tau_1(x_1) &= 3x_1 + 2,\\ \tau_2(x_2) &= 3x_2 + 1.\end{aligned}$$

Geometrically speaking, this transformation corresponds to a scale by a factor 3 followed by the translation by the vector $(2, 1)$. The final composition γ gives the following system:

$$\begin{aligned}\pi_1 \; o \; \tau(\mathbf{x}) &= 9x_1^2 + 12x_1 + 6x_2 + 7,\\ \pi_2 \; o \; \tau(\mathbf{x}) &= 9x_2^2 + 6x_2 - 3x_2 + 2,\end{aligned}$$

and the the initial set becomes $\mathcal{B}_u$. We can now use the methods presented in the previous section to compute a linear optimization problem approximating the initial problem 4.5 using the Bernstein expansion of this new system. As this method relies on the computation and the composition of mapping functions, we describe in the next subsection an innovative technique to map the polyhedron $\mathcal{P}$ to the unit box.

4.4.2 Using a change of variables

The polyhedron $\mathcal{P}$ can also be mapped to the unit box $\mathcal{B}_u$ by a change of variables as follows. We assume that the polyhedron $\mathcal{P}$ is bounded and let $V = \{\mathbf{v}^1, \ldots, \mathbf{v}^l\}$ be the set of its vertices. We first express the coordinates of a point $\mathbf{x}$ inside the polyhedron $\mathcal{P}$ as a linear combination of its the vertices, that is

$$\mathbf{x} = \sum_{j=1}^{l} \alpha_j \mathbf{v}^j = \nu(\alpha_1, \ldots, \alpha_l)$$

such that

$$\forall j \in \{1, \ldots, l\} \; \alpha_j \geq 0 \tag{4.8}$$

$$\sum_{j=1}^{l} \alpha_j = 1. \tag{4.9}$$

We then substitute $\mathbf{x}$ in π with $\nu(\alpha_1, \ldots, \alpha_l)$ to yield a new polynomial in $\alpha_1, \ldots, \alpha_l$.

We denote $\mu = \pi \; o \; \nu$, that is $\pi(\mathbf{x}) = \mu(\alpha_1, \ldots, \alpha_l)$. Furthermore, in order to retain the relation between α_j expressed in the constraint (4.9) we can use

$$\alpha_l = 1 - \sum_{j=1}^{l-1} \alpha_j$$

to substitute α_l in μ by the above sum, in order to obtain a polynomial with $(l-1)$ variables, denoted by $\xi(\beta)$ where $\tilde{\alpha} = (\alpha_1, \ldots, \alpha_{l-1})$.

Note that the constraints (4.8-4.9) indicate that γ is inside the unit box $\mathcal{B}'_u$ in $\mathbb{R}^{l-1}$. This implies that a bound function computed for the polynomial $\xi(\tilde{\alpha})$ on this unit box is also a bound function for the original polynomial π on the polyhedron P without additional error, unlike in the above-described case of box approximations. It then suffices to compute the bound functions for π over the polyhedron $\mathcal{P}$ using the Bernstein expansion of ξ over the $\mathcal{B}'_u$.

Example The initial set in the above example, is a triangle composed by the vertices (2,2), (5,1) and (4,4) which gives the followings change of variable:

$$\begin{aligned} x_1 &= 2\alpha_1 + 5\alpha_2 + 4\alpha_3, \\ x_2 &= 2\alpha_1 + \alpha_2 + 4\alpha_3. \end{aligned}$$

using the property 4.8 we can remove the last variable α using the equation

$$\alpha_3 = 1 - \alpha_2 - \alpha_1$$

that lead to the following equations:

$$\begin{aligned} x_1 &= -2\alpha_1 + \alpha_2 + 4, \\ x_2 &= -2\alpha_1 - 3\alpha_2 + 4. \end{aligned}$$

The polynomial system obtained using this change of variable is

$$\begin{aligned} \xi_1(\alpha) &= (-2\alpha_1 + \alpha_2 + 4)^2 + 2(-2\alpha_1 - 3\alpha_2 + 4) + 1, \\ \xi_2(\alpha) &= (-2\alpha_1 - 3\alpha_2 + 4)^2 - (-2\alpha_1 - 3\alpha_2 + 4) + 2, \end{aligned}$$

with $\alpha_1, \alpha_2, \alpha_3 \in [0 \ldots 1]$. We can now compute affine bounds for this polynomial system and use them for the reachability analysis. To do so, we have to transform each template direction in the $\tilde{\alpha}$ basis and use the previously presented algorithms to compute the next *polyhedral coefficient vector* $\mathbf{b}$.

We present now the main algorithm which performs the reachable set analysis, we will also present some strategy for computing appropriate template direction, bernstein coefficient by interpollation and accurately approximate a set by a box for unit box mapping.

4.5 Algorithms

4.5.1 Reachability analysis

Algorithm 2 summarizes the main steps of our approach with affine bounds for over-approximating the reachable set of the system $\mathbf{x}(k+1) = \pi(\mathbf{x}(k))$

where the initial set $\mathcal{P}^0$ is a bounded polyhedron in $\mathbb{R}^n$ represented by a template matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a polyhedra coefficient vector $\mathbf{b} \in R^m$ such that

$$\mathcal{P}^0 : \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}\}.$$

The function $ComputeTemplate(\mathbf{A}, \pi)$ computes a template matrix for the next reachable set, in the case of non variating template this function just return $\mathbf{A}$. To unify the two methods of mapping a polyhedron to the unit box in the same abstract algorithm, we use β to denote both of the transformations using either a box approximation or a change of variables. In the same way we use the function $BoundFunctions()$ to denote the methods for affine bounds computation presented in section 4.2.2.

Algorithm 2 Reachable set computation

/ Inputs: convex polyhedron $\mathcal{P}^0 = \langle \mathbf{A}, \mathbf{b} \rangle$, polynomial π, the number of iterations k_{max} */*

$k = 0$
repeat
  */*Compute the template for the next polyhedron*/*
  $\mathbf{T} = ComputeTemplate(\mathbf{A}, \pi)$
  */*Compute the function mapping the unit box $\mathcal{B}_u$ to the polyhedron $\mathcal{P}^k$ */*

  $\beta = UnitBoxMap(\mathcal{P}^k)$
  $\gamma = \pi \ o \ \beta$
  */*Compute the affine bound functions*/*
  $(u, l) = BoundFunctions(\gamma)$
  for all $\mathbf{T}_i$ in $\mathbf{T}$ **do**
    for all $j = 1$ to $j = n$ **do**
      if $T_{i,k} > 0$ **then**
        $\bar{b}_i = \bar{c}_i + T_{i,j} max(u_j(\mathbf{x}),\ \mathbf{x} \in \mathcal{B}_u)$
      else
        $\bar{b}_i = \bar{c}_i + T_{i,j} min(l_j(\mathbf{x}),\ \mathbf{x} \in \mathcal{B}_u)$
      end if
    end for
  end for
  $\mathcal{P}^{k+1} = \langle \mathbf{T}, \bar{\mathbf{b}} \rangle$ */* Construct the template polyhedron and return it */*
  $k++$
until $k = k_{max}$

In the following subsection, we present some strategy to efficiently choose template directions.

4.5.2 Choice of the template

In this subsection, we present a method for choosing a dynamical template that reflects as much as possible the changes in the shape of the reachable sets $\mathcal{P}^k$ induced by the dynamics.

We consider the template $\mathbf{T}^k$ for the polyhedron

$$\mathcal{P}^k = \left\{ \mathbf{x} \in \mathbb{R}^n | \ \mathbf{T}^k \cdot \mathbf{x} \leq b^k \right\}.$$

In the next iteration, we want to compute a new template $\mathbf{T}^{k+1}$ For that purpose, we use a local linear approximation of the dynamics of the polynomial dynamical system ($\mathbf{x}(k+1) = \pi(\mathbf{x}(k))$) given by the first order Taylor expansion around the centroid $\mathbf{c}^k \in \mathbb{R}^n$ of the last computed polyhedron $\mathcal{P}^k$:

$$\pi(\mathbf{x}) \ \approx L^k(\mathbf{x}) = f(\mathbf{c}^k) + \mathbf{J}(\mathbf{c}^k)(\mathbf{x} - \mathbf{c}^k)$$

where $\mathbf{J}$ is the Jacobian matrix of the function π. Let us denote $\mathbf{F}^k = \mathbf{J}(\mathbf{c}^k)$ and $\mathbf{h}^k = f(\mathbf{c}^k) - \mathbf{J}(\mathbf{c}^k)\mathbf{c}^k$, then in a neighborhood of $\mathbf{c}^k$ the nonlinear dynamics can be roughly approximated by $\mathbf{x}(k+1) = \mathbf{F}^k\mathbf{x}(k) + \mathbf{h}^k$. Assuming that $\mathbf{F}^k$ is invertible, this gives $\mathbf{x}(k) = {\mathbf{F}^k}^{-1}\mathbf{x}(k+1) - {\mathbf{F}^k}^{-1}\mathbf{h}^k$. Transposing the constraints on $\mathbf{x}(k)$ given by $\mathcal{P}^k$ to $\mathbf{x}(k+1)$, we obtain

$$\mathbf{T}^k{\mathbf{F}^k}^{-1}\mathbf{x}(k+1) \leq \mathbf{c}^k + \mathbf{T}^k{\mathbf{F}^k}^{-1}\mathbf{h}^k.$$

Then, it appears that a reasonable template for $\mathcal{P}^{k+1}$ can be $\mathbf{T}^{k+1} = \mathbf{T}^k{\mathbf{F}^k}^{-1}$. This new template $\mathbf{T}^{k+1}$ can then be used in the next iteration for the computation of the polyhedron $\mathcal{P}^{k+1}$ using the method described in the previous section.

Figure 4.5 illustrates the benefit of using dynamical templates computed using the above linear approximation. Indeed, the over-approximation error (drawn in grey) in the case of using a static template is larger than the error in the case of a dynamical template.

4.5.3 Dynamical templates for oriented boxes

We also present a low-cost method to compute an oriented box $\mathcal{B}$ used to over-approximate the reachable sets.

As presented in Section 4.2.2 we use a box approximation $\mathcal{B}^k$ to map a polyhedron $\mathcal{P}^k$ to the unit box $\mathcal{B}_u$. This box $\mathcal{B}^k$ can be oriented to map more precisely the set $\mathcal{P}^k$ however computing an oriented box by an analysis method such as Principal Component Analysis is a costfull operation. We

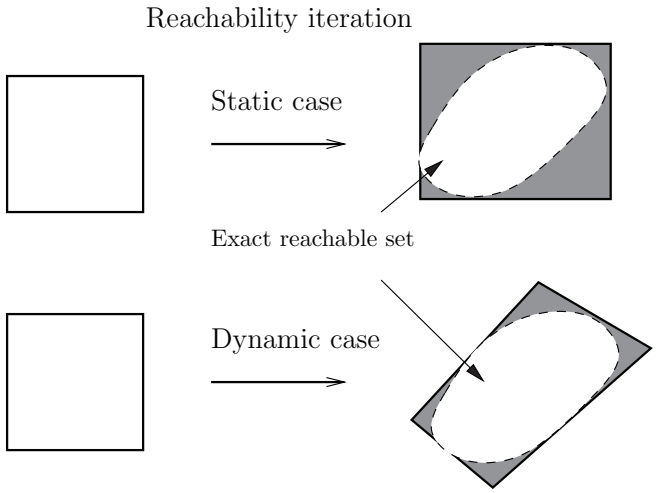

Figure 4.5: Benefits of using dynamical templates in place of static ones.

implement a simple method to approximate the oriented box $\mathcal{B}^k$ when the initial set is a box.

By considering that the initial set $\mathcal{P}^0$ is often given as a box, the bounding box $\mathcal{B}^0$ of this set is aligned with the axis. We can consider this box as a template polyhedron represented by

$$\mathcal{B}^k = \left\{ \mathbf{x} \in \mathbb{R}^n | \begin{pmatrix} \mathbf{T}^k \\ -\mathbf{T}^k \end{pmatrix} \mathbf{x} \leq \mathbf{c}^k \right\}.$$

where $\mathbf{T}^k \in \mathbb{R}^{n \times n}$ is a n by n orthogonal matrix and $\mathbf{c}^k \in \mathbb{R}^{2n}$ the polyhedral coefficient. In a similar way to the polyhedra $\mathcal{P}^k$, the accuracy will be better if we use dynamical templates which take into consideration the dynamics of the system (essentially the rotation effects).

We will actually choose $\mathbf{T}^k$ to obtain an oriented rectangular box. By

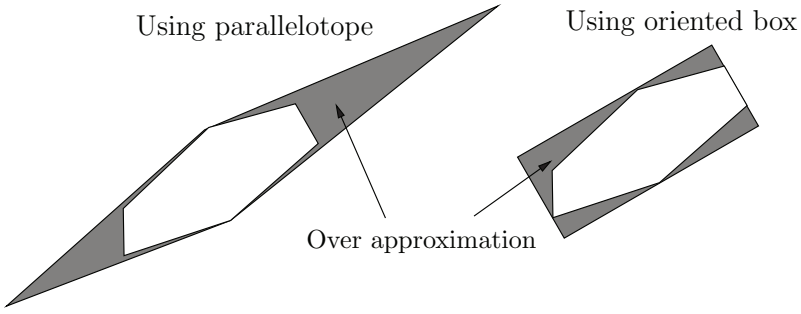

Figure 4.6: Benefits of using oriented rectangular boxes.

making this choice, we avoid some overapproximation that can occur when the parallelotope becomes elongated, (see Figure 4.6 for an illustration of this problem). We should mention that the image of an oriented rectangular box by a linear map is not necessarily an oriented rectangular box so we can not use directly the matrix $(\mathbf{F}^k)^{-1}$ computed previously. To solve this problem we will use an popular technique in interval analysis [74] based on the QR-Decomposition of matrices. Essentially, $\mathbf{F}^k$ will be written as the product of two matrices $\mathbf{F}^k = \mathbf{Q}^k\mathbf{R}^k$ where $\mathbf{Q}^k$ is an orthogonal matrix and $\mathbf{R}^k$ is an upper triangular one. Then, to choose the template C^{k+1} of the next oriented box $\mathcal{B}^{k+1}$, we apply our rotation transformation matrix $\mathbf{Q}^k$ to the given rectangular box $\mathbf{T}^k$ which is equivalent to choose the template

$$\mathbf{T}^{k+1} = \mathbf{T}^k{\mathsf{Q}^{\mathsf{k}}}^\top.$$

Of course, in that case, we will deal with non-aligned-axis boxes which can cause higher degrees for our polynomial but the approximation will be less conservative than using static templates for $\mathbf{T}^k$.

4.5.4 Bernstein coefficient interpolation

We present here an alternative and innovative approach to compute the Bernstein coefficient using interpolation at points $\mathbf{i}/\mathbf{d}$ with $\mathbf{i} \leq \mathbf{d}$.

Indeed, let $\mathbf{B}_{\text{inter}}$ the matrix whose lines are indexed by $\mathbf{i} \leq \mathbf{d}$ and columns by $\mathbf{j} \leq \mathbf{d}$ with the coefficients $B^{\mathbf{d},\mathbf{i}}(\mathbf{j}/\mathbf{d})$ and π the matrix whose lines are also indexed by $\mathbf{i} \leq \mathbf{d}$ are $\pi(\mathbf{i}/\mathbf{d})$.

Then the following equalities can be stated from the Bernstein expansion:

$$\pi(\mathbf{j}/\mathbf{d}) = \sum_{\mathbf{i}\in\mathsf{I}_\mathsf{d}} \mathbf{b}^{\mathbf{i}} B^{\mathbf{d},\mathbf{i}}(\mathbf{j}/\mathbf{d}),$$

$$\pi = \mathbf{B}_{\text{coef}}\mathbf{B}_{\text{inter}},$$

then

$$\mathbf{B}_{\text{coef}} = {\mathbf{B}_{\text{inter}}}^{-1}\pi.$$

where $\mathbf{B}_{\text{coef}}$ is a matrix whose lines indexed by $\mathbf{i} \leq \mathbf{d}$ are the polynomial system Bernstein coefficients $\mathbf{b}^{\mathbf{i}}$. The Bernstein coefficients can the be determined by computing the inverse of the matrix $\mathbf{B}_{\text{inter}}$. An interesting fact is that, if the polynomial system maximal dimension do not change during the unit box mapping step (by using axis aligned box approximation by example), the matrix ${\mathbf{B}_{\text{inter}}}^{-1}$ can be reused at each reachability iteration and can be computed before starting the analysis.

4.6 Discussion

In this section we discuss precision and complexity of the proposed methods.

4.6.1 Approximation error

The approximation errors are caused by the bound functions and the use of template polyhedra. When a box approximation is used, this causes an additional error. The following lemma states an important property of the Bernstein expansion.

Lemma 11. *Let $C_{\pi,\mathcal{B}}(\mathbf{x})$ be the piecewise linear function defined by the Bernstein control points of π with respect to the box $\mathcal{B}$. Then, for all $\mathbf{x} \in \mathcal{B}$,*

$$||\pi(\mathbf{x}) - C_{\pi,\mathcal{B}}(\mathbf{x})||_\infty \leq K\rho^2(\mathcal{B})$$

where $||\cdot||_\infty$ is the infinity norm on $\mathbb{R}^n$, $\rho(\mathcal{B})$ is the box size (i.e. its largest side length), $K_k = max_{\mathbf{x}\in\mathcal{B};i,j\in\{1,\ldots,n\}}|\partial_i\partial_j\pi_k(x)|$, $K = max_{k\in\{1,\ldots,n\}}K_k$.

From this result it can be proven that in one-dimensional case, the error between the bound functions computed using the Bernstein expansion and the original polynomial is quadratic in the length of box domains. This quadratic convergence seems to hold for higher dimensional cases in practice, as shown in [53]. We conjecture that there exists a subdivision of the box $\mathcal{B}$ which allows a quadratic convergence of the error. This subdivision method is similar to the one used for finding roots of a polynomial with quadratic convergence [73].

Hence, when more accurate reachable set approximations are required, we can divide the unit box into non-overlapping sub-boxes. Then, for each sub-box, we compute a bounding function, with which we then compute a coefficient for each template. Finally, for each template, we take the largest coefficient to define the template polyhedron. Since the sub-boxes are smaller, the bound functions are more precise, we can thus improve the coefficients as much as desired. This division idea can also be similarly used to reduce the error caused by oriented box approximation. The error inherent to the approximation by template polyhedra can be controlled by fine-tuning the number of template constraints.

4.6.2 Computation cost

Concerning complexity, when a box approximation is used, the computation of bound functions and PCA only require manipulating matrices and linear equations. Linear programming can be solved in polynomial time. When

iterating these methods to compute the reachable set of a polynomial dynamical system, if the number of template constraints is constant, the complexity depends linearly on the number of iterations.

It is important to take into consideration that, when using box approximation for unit box mapping, the use of oriented boxes changes the degree of the composed polynomials. If $\mathbf{d} = (d_1 \dots d_n)$ is the degree of each variable of a polynomial and d_{max} is its maximal degree, the composition of the polynomial with the mapping functions can have in the worst case a degree $d_\gamma = (d_{max} \dots d_{max})$. This can heavily increase the number of Bernstein coefficients which would be equals to $(d_{max} + 1)^n$.

Regarding accuracy, the method using a change of variables is performant, since the polyhedral constraints are exactly captured. This is also confirmed by experimental results. However, the LP problems to solve are in higher dimension, which is $(l - 1)$ where l is the number of vertices of the polyhedra. In addition, this method requires computing the vertices of template polyhedra, which is expensive and our experimentation shows that this costs a large part of computation time. This can be improved by considering the coefficients of template polyhedra as parameters, and since the template is fixed, we can deduce a symbolic expression of the vertices of the parametric polyhedra, which can be used to derive the (parametric) change of variable to map the polyhedra to the unit box.

The next section presents some experimental results.

4.7 Experimental results

We developed a new tool for performing analysis of polynomial systems which is an implementation of the techniques presented in this chapter. This tool is presented in more detail in Chapter 6. In the following, we demonstrate the methods with four examples: a multi-affine system modeling a prey-predator dynamics, a model of insect nest-site choice, a biological system and a control system (modeled as a hybrid system). The time efficiency of the tool was also evaluated by considering a number of randomly generated polynomials.

4.7.1 Prey predator model and performance evaluation

In this first example we use the approximation with bound functions method to compute the reachable sets on generalized Lotka-Volterra equations. These systems model the dynamics of the population of n biological species known as the prey predator model. Its equations are given by $\dot{x_i} = x_i(r_i + A_i x)$

where $i \in \{1, 2, \ldots, n\}$, r_i is the i^{th} elements of a vector $r \in \mathbb{R}^n$ and A_i is the i^{th} line of a matrix $A \in \mathbb{R}^{n \times n}$.

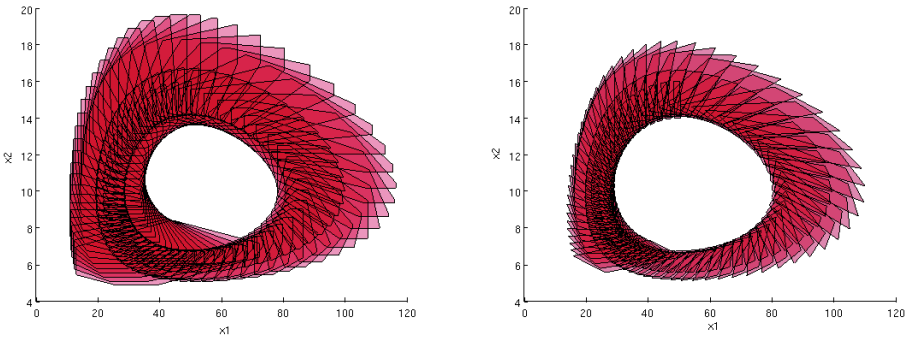

Figure 4.7: Reachability computation for a 2 dimensional predator-prey model using dynamical template with axis aligned (left) and oriented (right) bounding boxes.

We performed reachable set computation for an Euler discretized Lotka-Voltera system for the case $n = 2$:

$$\begin{cases} x_1(k+1) &= x_1(k) + h(0.1x_1 - 0.01x_1x_2) \\ x_2(k+1) &= x_2(k) + h(-0.05x_2 + 0.001x_1x_2) \end{cases}$$

Figure 4.7 shows the cyclic behavior of the reachable set analysis computed using a discretization time $h = 0.3$ with an initial box included in $[49, 51] \times [14, 16]$ during 700 iterations. The figure on the left was computed in 1.87 seconds using dynamical template polyhedra and bounding boxes aligned with axis. The other one was computed in 3.46 seconds using dynamical templates and oriented boxes. A significant gain of precision using the oriented box can be observed however the computation time is almost double.

We also evaluated the performance of our method using two ways of computing the Bernstein coefficients (explicitly and by interpolation) with recursively generated n-dimensional Lotka-Volterra equations given by:

$$\begin{cases} x_1(k+1) &= x_1(k) + h\left(x_1(k)(1 - x_2(k) + x_n(k))\right) \\ x_i(k+1) &= x_i(k) + h\left(x_i(k)(-1 - x_{i+1}(k) + x_{i-1}(k))\right) \\ x_n(k+1) &= x_n(k) + h\left(x_n(k)(-1 - x_0(k) + x_{n-1}(k))\right) \end{cases}$$

where $i \in \{2, \ldots, n-1\}$. We used axis aligned bounding boxes to make the change of variable. (see tables 4.1).

We observe that the interpolation method provides more effective results than the explicit computation of Bernstein coefficient but requires to compute the matrix ${\mathbf{B}_{\text{inter}}}^{-1}$ before starting the analysis.

dim	explicit	interpol	$\mathbf{B}_{\text{inter}}{}^{-1}$	dim	explicit	interpol	$\mathbf{B}_{\text{inter}}{}^{-1}$
2	0.00235	0.00221	0.00001	7	0.1905	0.1274	0.0099
3	0.00536	0.00484	0.00004	8	0.5682	0.3674	0.0494
4	0.01112	0.01008	0.00008	9	1.935	1.265	0.266
5	0.02612	0.02124	0.00052	10	6.392	4.441	1.623
6	0.068	0.0499	0.0016	11	21.98	16.03	10.36

Table 4.1: Computation time for one reachable set computation iteration for some generated Lotka-Voltera systems

4.7.2 A model of insect nest-site choice

We also study a model of a decision making mechanism used by a swarm of honeybees to select one among many different nest-sites [25]. This is built upon classical mathematical models of the spread of diseases, information and beliefs. The bee population is divided into 5 groups:

- X: neutral bees that have not chosen a site
- Y_1: evangelic bees dancing for the site 1
- Y_2: evangelic bees dancing for the site 2
- Z_1: non-evangelical bees which have been converted to the site 1
- Z_2: non-evangelical bees which have been converted to the site 2

The bees dance not only to advertise the quality of a site but also to transmit to the other bees information about the direction and distance to the site. Non-evangelical bees stay idle but may take up dancing when they are stimulated by a dancing bee. For simplicity, we also use the names of the groups to denote the respective numbers of individuals in each group.

The equations describing the evolutions of the variables are as follows:

$$\begin{aligned}
\dot{X}(t) &= -\beta_1 X(t)Y_1(t) - \beta_2 X(t)Y_2(t), \\
\dot{Y}_1(t) &= \beta_1 X(t)Y_1(t) - \gamma Y_1(t) + \delta\beta_1 Y_1(t)Z_1(t) + \alpha\beta_1 Y_1(t)Z_2(t), \\
\dot{Y}_2(t) &= \beta_2 X(t)Y_2(t) - \gamma Y_2(t) + \delta\beta_2 Y_2(t)Z_2(t) + \alpha\beta_2 Y_2(t)Z_1(t), \\
\dot{Z}_1(t) &= \gamma Y_1(t) - \delta\beta_1 Y_1(t)Z_1(t) - \alpha\beta_2 Y_2(t)Z_1(t), \\
\dot{Z}_2(t) &= \gamma Y_2(t) - \delta\beta_1 Y_2(t)Z_2(t) - \alpha\beta_2 Y_1(t)Z_2(t),
\end{aligned}$$

where β_1 and β_2 are scalar parameters representing a measure of how vigorously the bees dance for the sites 1 and 2 respectively; α is the parameter

representing the proportionality of switching back to the neutral state, and γ is the proportionality of conversion from the dancing state to the idle state. The proportionality of conversion from the neutral state to any site Y_i is 1 and from the idle state to Y_i is δ.

In this case study, we want to study the influence of the parameter β_2 on the possibility of achieving a consensus on nest-site choice.

Using the Euler discretization method we obtains the following discrete-time dynamics.

$$\begin{aligned}
X(k+1) &= X(k) + h(-\beta_1 X(k)Y_1(k) - \beta_2 X(k)Y_2(k)),\\
Y_1(k+1) &= Y_1(k) + h(\beta_1 X(k)Y_1(k) - \gamma Y_1(k) + \delta\beta_1 Y_1(k)Z_1(k) + \alpha\beta_1 Y_1(k)Z_2(k)),\\
Y_2(k+1) &= Y_2(k) + h(\beta_2 X(k)Y_2(k) - \gamma Y_2(k) + \delta\beta_2 Y_2(k)Z_2(k) + \alpha\beta_2 Y_2(k)Z_1(k)),\\
Z_1(k+1) &= Z_1(k) + h(\gamma Y_1(k) - \delta\beta_1 Y_1 Z_1 - \alpha\beta_2 Y_2 Z_1),\\
Z_2(k+1) &= Z_2(k) + h(\gamma Y_2(k) - \delta\beta_1 Y_2(k)Z_2(k) - \alpha\beta_2 Y_1(k)Z_2(k)),
\end{aligned}$$

We choose a discretization time $h = 0.01$. In the following, $\beta_1 N = 1.0$, the initial set is a small box centered at $(N, 0, 0, 0, 0)$, where N is the total number of honeybees arbitrary fixed to 1000.

Figure 4.8 and Figure 4.9 show the evolution of the proportions of converted honeybees $((Y_i + Z_i)/N)$ during 6000 iterations, with different interval values of the parameter β_2. The blue sets (in lighter color) correspond to the proportions of bees supporting the site 1 and the red sets (in darker color) correspond to those supporting the site 2. These results were obtained using the method for multi-affine systems which took about $7.5s$

First we consider that at the beginning no honeybees dance for the site 2, in other words $\beta_2 = 0$. After 300 iterations the site 2 is discovered and its propaganda begins. We can observe two main comportments: a consensus and the absence of consensus. We consider that there is a consensus if the distance between the proportions of honeybees supporting the site 1 or 2 is large and tends to increase.

Figure 4.8 shows the analysis using the parameters $\alpha = 0.7, \gamma = 0.3$ and $\beta_2 \in [1, 1.2]$. We observe that β_2 is marginally superior to β_1, however we can see that there is no consensus between the two evangelical groups.

Figure 4.9 shows that a consensus can be observed if the measure of how vigorously the bees dance for sites 2 is increased. In this analysis the parameters stay the same except for $\beta_2 \in [1.5, 2.0]$. We observe a clear consensus for the site 2 despite its lateness.

Another way to obtain a consensus without modifying β_2 is to reduce α which corresponds to the proportionality of honeybees reconversion to another site. When $\alpha = 0.2$, the late propaganda of the site 2 leads to a consensus to choose the site 1 as shown in Figure 4.10.

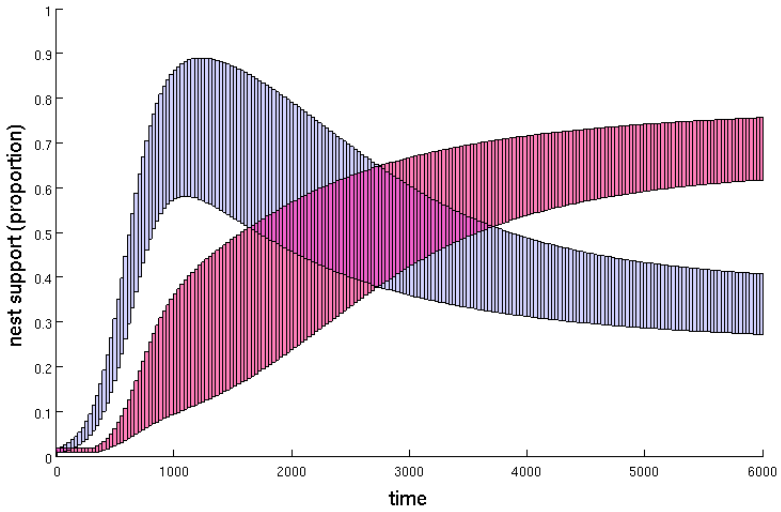

Figure 4.8: No consensus is observed with $\alpha = 0.7, \gamma = 0.3$ and $\beta_2 \in [1, 1.2]$.

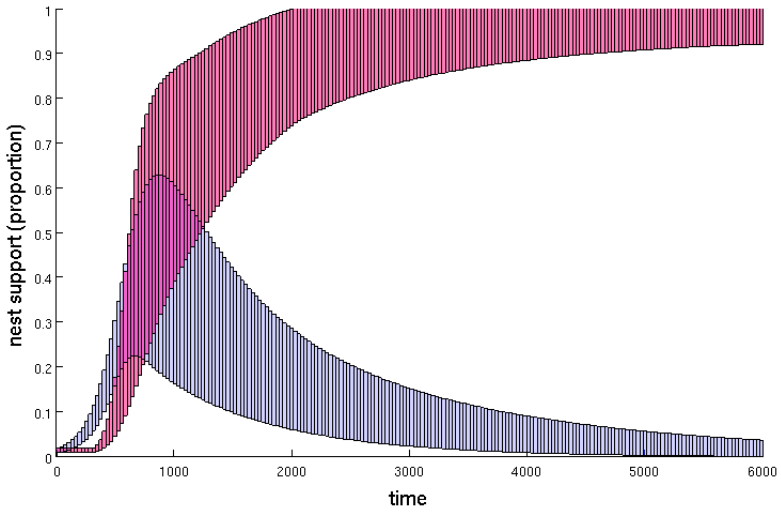

Figure 4.9: A consensus for the site 2 is observed with $\alpha = 0.7, \gamma = 0.3$ and $\beta_2 \in [1.5, 2]$.

Finally, to compare the two proposed methods, Figure 4.11 shows the projection of the reachable sets on the variables Y_1 and Y_2 during the first 150 iterations, with $\alpha = 0.7$, $\gamma = 0.3$ and $\beta_2 = 1.2$. We plot the results obtained by the two methods and we can see that the sets computed using the method for multi-affine systems are larger than those computed using

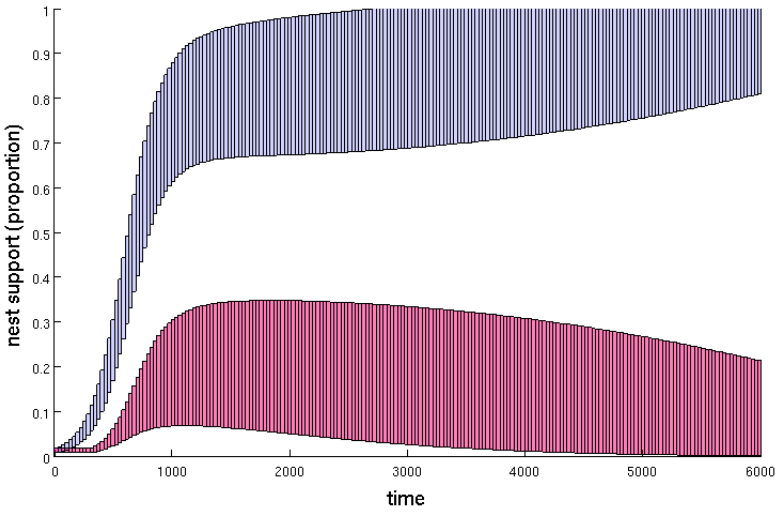

Figure 4.10: A consensus for the site 1 is observed with $\alpha = 0.2, \gamma = 0.3$ and $\beta_2 \in [1, 1.2]$.

the Bernstein expansion. We observe a gain of precision with the Bernstein method, however the computation time of this method is $25.06s$ and is more than 250 times superior than the computation time of the other method which is $0.09s$.

4.7.3 FitzHugh-Nagumo neuron model

The third biological example is the FitzHugh-Nagumo neuron model describing the electrical activity of a neuron [43]. It can be expressed by a polynomial dynamical system:

$$\dot{x} = x - x^3 - y + 7/8 \tag{4.10}$$

$$\dot{y} = 0.08(x + 0.7 - 0.8y) \tag{4.11}$$

We now study an Euler time discretization scheme of the above differential equation with the time step 0.2. The initial set is an octagon included in the bounding box $[0.9, 1.1] \times [2.4, 2.6]$.

Figure 4.12 shows two reachable set evolutions where the initial set is a regular octagon included in the bounding box $[0.9, 1.1] \times [2.4, 2.6]$. In both cases, we use box approximation method for unit box mapping using only axis aligned rectangles. The computation time is 8.16 seconds using a static template and 8.22 seconds using the dynamical templates each one with 8

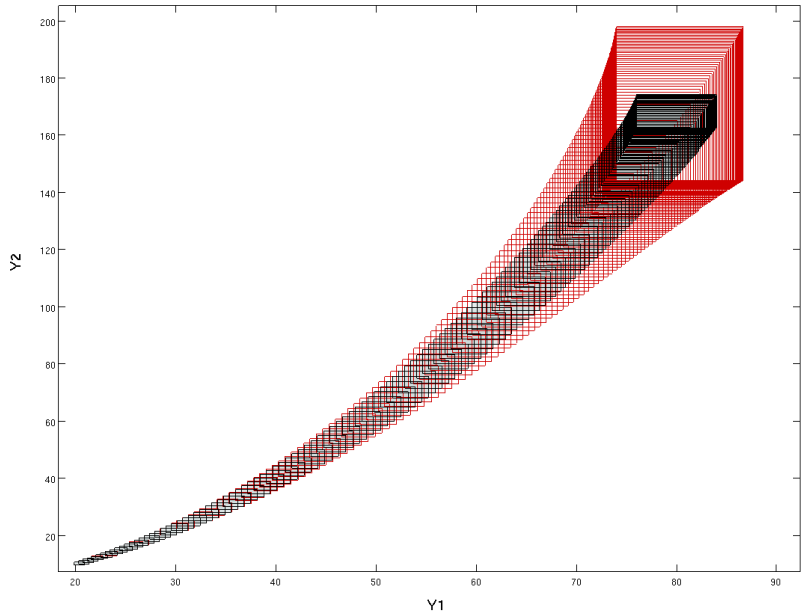

Figure 4.11: Reachability analysis during 150 iterations using the two reachability methods, the black outlines set are computed with the Bernstein expansion method, the red outlines set correspond to the result using the method for multi-affine systems.

directions. We can see from the figure a huge precision improvement obtained by using dynamical templates, at little additional cost.

Figure 4.13 shows two reachable sets computed using the same template. The one computed by the method using a change of variables is much more precise, which allowed observing a limit cycle. To visually demonstrate an accuracy improvement, Figure 4.13 only shows the results of a few first computation steps. The computation time of the method using a box approximation after 500 steps is 8.98 and that of the method using a change of variables is 8.8.

4.7.4 A control system

The fourth example is the Duffing oscillator taken from [57, 44]. This is a nonlinear oscillator of second order, the continuous-time dynamics is described by

$$\ddot{y}(t) + 2\zeta\dot{y}(t) + y(t) + y(t)^3 = u(t)$$

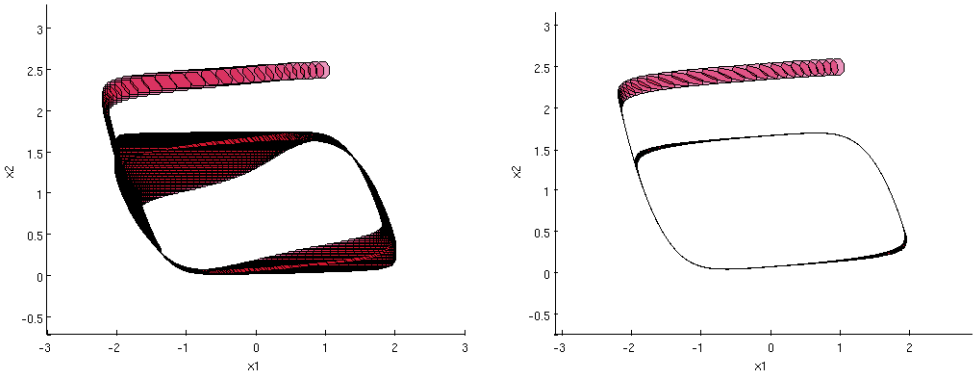

Figure 4.12: Reachability computation for the FitzHugh-Nagumo neuron model using static (left) and dynamical (right) template polyhedra.

where $y \in \mathbb{R}$ is the state variable and $u \in \mathbb{R}$ is the control input. The damping coefficient $\zeta = 0.3$. In [44], using a forward difference approximation with a sampling period $h = 0.05$ time units, this system is approximated by the following discrete-time model:

$$\begin{aligned} x_1(k+1) &= x_1(k) + hx_2(k) \\ x_2(k+1) &= -hx_1(k) + (1 - 2\zeta h)x_2(k) + hu(k) - hx_1^3(k) \end{aligned}$$

In [44], an optimal predictive control law $u(k)$ was computed by solving a parametric polynomial optimization problem.

We model this control law by the following switching law with 3 modes:

$$\begin{aligned} u(k) = 0.5k \quad & \text{if } 0 \leq k \leq 10 \\ u(k) = 5 - 0.5(k-10)/3 \quad & \text{if } 10 < k \leq 40 \\ u(k) = 0 \quad & \text{if } k > 40 \end{aligned}$$

The controlled system is thus modeled as a hybrid automaton [3] with 3 discrete states. The initial set is a rectangle such that $2.49 \leq x_1 \leq 2.51$ and $1.49 \leq x_2 \leq 1.51$.

The results obtained using the two methods are shown in Figure 4.14 which are coherent with the phase portrait in [44]. We can see that the method using a change of variables achieved better precision since the reachable set it computed is include in the set computed by the other method. However, the method using a change of variables is less time-efficient. For 70 steps, the computation time of the method using a box approximation is $1.25s$ while that of the method using a change of variables is $1.18s$. We also used this example to compare the two methods of computing bound functions and observed that they produced equally accurate results.

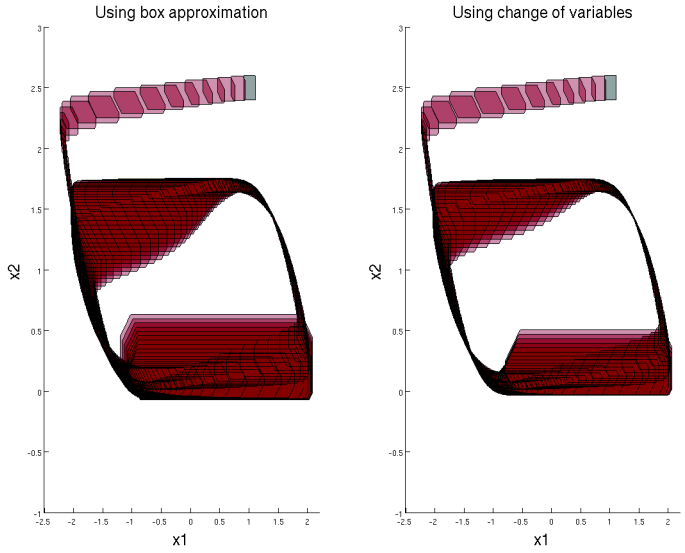

Figure 4.13: FitzHugh-Nagumo neuron model. The evolution of the reachable set computed using the two methods: using a box approximation and using a change of variables.

4.7.5 Randomly generated systems

In order to evaluate the performance of our methods, we tested them on a number of randomly generated polynomials in various dimensions and maximal degrees (the maximal degree is the largest degree for all variables). For a fixed dimension and degree, we generated different examples to estimate an average computation time. In the current implementation, polynomial composition is done symbolically, and we do not yet exploit the possibility of sparsity of polynomials (in terms of the number of monomials). The computation times in seconds for the method using a box approximation (abbreviated to BA) and the method using a change of variables (abbreviated to CV) are shown in the table in Figure 4.15.

As expected, the computation time grows linearly w.r.t. the number of steps. This can be explained by the use of template polyhedra where the number of constraints can be chosen according to required precisions and thus the complexity of the polyhedral operations can be better controlled, compared to general convex polyhedra. Indeed, when using general polyhedra, the operations, such as the convex hull, may increase their geometric complexity (roughly described by the number of vertices and constraints).

On the other hand, we also compared the two methods for computing

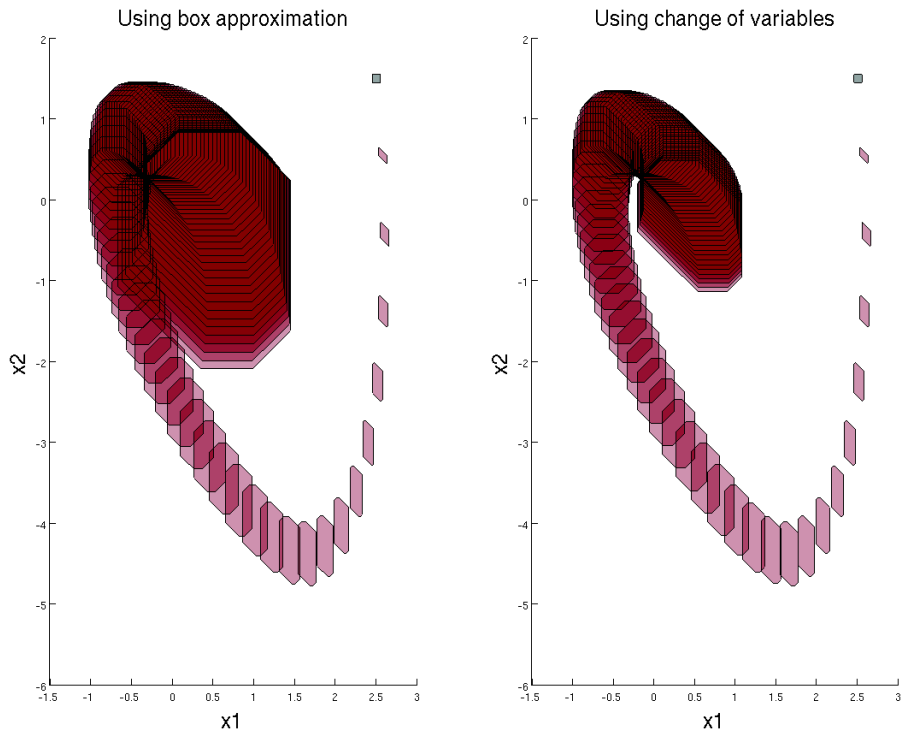

Figure 4.14: The Duffing oscillator: the reachable set computed using a change of variable is more accurate than the one computed using a box approximation.

dim	degree	nb monomials	template size	time (s) method BA	time (s) method CV
2	2	4	4	0.02	0.02
3	2	6	6	0.02	0.02
4	2	8	8	0.06	0.09
5	2	10	10	0.35	0.71
6	2	12	12	4.34	5.64
7	2	14	14	63.25	72.61

Figure 4.15: Computation time for randomly generated polynomial systems in various dimensions and degrees

dim	degree	time (s) method LSA	time (s) method CHF
1	2	0.003	0.002
2	2	0.005	0.005
3	2	0.012	0.008
4	2	0.036	0.039
5	2	0.138	0.139
6	2	0.725	0.692
7	2	3.176	2.621
8	2	17.461	10.868
9	2	116.383	43.664

Figure 4.16: Comparing efficiency of the two methods for computing bound functions on randomly generated polynomial systems

bound functions: using a lower convex hull facet (abbreviated to CHF) and using the least squares approximation (abbreviated to LSA). The average computation time for one step of reachability computation is shown in Table 4.16. In this experiment we used box templates. Moreover, the computation time for polynomial composition is not included, since the computation of bound functions is not a dominant part of the total computation time. We were not able to test systems of dimensions higher than 9 because polynomial composition becomes prohibitively costly. This issue can be handled by computing the Bernstein coefficients without explicit polynomial composition, which is indeed a topic of our current research. We have observed that the method using the least squares approximation would be more performant than the one using a lower convex hull facet for systems of dimension beyond 9. The latter requires solving n systems of linear equations in dimensions increasing from 1 to n. The former requires solving only one linear system in dimension $(n + 1)$. Using Gaussian elimination to solve a system of n equations for n unknowns has complexity of $O(n^3)$. Thus, the complexity of the method using a lower convex hull facet is roughly $O((n-1)^2 n^2/4)$ while the complexity of the other is $O((n+1)^2)$.

Chapter 5

Redundant constraints for refinement

Résumé: *Nous présentons dans ce chapitre, la dernière contribution qui concerne l'analyse d'accessibilité de systèmes linéaires avec entrées. Ces systèmes ont un grand intérêt malgré leur linéarité car ils peuvent être les résultats de techniques d'approximation de systèmes comme celle présentée dans le Chapitre 3. Nous présentons ici, une technique pour raffiner l'approximation de l'ensemble accessible durant l'analyse d'accessibilité. Nous présentons comment l'addition de contraintes redondantes dans la description des ensembles accessibles peut contribuer, par la suite, à réduire l'erreur d'approximation. Nous proposons ensuite deux critères afin de trouver des directions optimales pour l'ajout de contraintes redondantes. Nous finissons par présenter nos résultats expérimentaux sur des systèmes de dimension 2 à 100.*

In this chapter, we revisit the problem of approximating reachable sets of linear continuous systems with input, which was investigated in [11, 41]. Linear systems with input are of particular interest in the approximation of non-linear systems as in the hybridization technique.

As discussed in Chapter 3, the approximation accuracy is important especially when the computed over-approximations do not allow proving a property. We propose a method for refining a reachable set approximation by

adding redundant constraints to decrease the over-approximation in some critical directions. We also introduce the notion of directional distance which is appropriate for measuring approximation effectiveness with respect to verifying a safety property.

The refinement techniques presented in the following sections concern the reachability computation methods based on convex polyhedra which possess a constraints based representation. This representation facilitates the add of redundant constraints and will be discussed in Section 5.2.

The reachability problem is given by:

- $\mathcal{P}^0 \subset \mathbb{R}^n$ a convex polyhedron which represents the initial states,
- $\mathcal{B} \subset \mathbb{R}^n$ a convex polyhedron in a state space which represents a set of unsafe states,
- An affine transition system which can represent the integration scheme of a continuous time dynamical system.

$$\mathbf{x}(k+1) = l(\mathbf{x}(k)) + \mathbf{u}(k) = \mathbf{A}\mathbf{x}(k) + \mathbf{b} + \mathbf{u}(k) \tag{5.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{x}(k) : \mathbb{N}^* \to \mathbb{R}^n$ and $\mathbf{u}(k) : \mathbb{N}^* \to \mathcal{U}$ represents the input included in a set $\mathcal{U} \subset \mathbb{R}^n$.

In the following, we present a reachability algorithm for linear systems with input. In the next sections we will present a novel way to reduce the approximation error that we call refinement. We present two refinement criteria refinement, the first is based on the angle between adjacent faces and the second on finding some critical directions. We will end by presenting some experimental results.

5.1 Reachability algorithm

We consider that $\mathcal{P}^0$ is a convex polyhedron which admits the following representation:

$$\mathcal{P}^0 = \{\mathbf{x} \mid \mathbf{N}^0\mathbf{x} \leq \mathbf{c}^0\} = \langle \mathbf{N}^0, \mathbf{c}^0 \rangle$$

where $\mathbf{N}^0 \in \mathbb{R}^{m\times n}$ is a matrix whose lines represent the normal vectors of the half-spaces of the initial set and $\mathbf{c}^0$ is its associated polyhedral coefficient vector.

For simplicity we denote each normal by $\mathbf{n}_i^k$ with $i \in \{1, \ldots, m\}$ such that

$$\mathbf{N}^0 = (\mathbf{n}_0^0, \ldots, \mathbf{n}_m^0)^T.$$

We will also denote by $Reach(\mathcal{H}, \mathbf{u})$ the image of an half-space $\mathcal{H}$ by the affine dynamics under a fixed input $\mathbf{u} \in \mathcal{U}$

5.1.1 Algorithm

To deal with such dynamics, we make use of the technique, which was first suggested in [96] and then applied in [11, 41, 14] for hybrid systems. This technique is based on the Maximum Principle in optimal control [58]. In the following, we recap the main idea of this technique.

It can be proven that for an autonomous linear system the evolution of the normal $\mathbf{n}_i^k$ of each half-space is governed by the adjoint system whose dynamics is described by the following equation:

$$\mathbf{n}_i^{k+1} = -\mathbf{A}^T \mathbf{n}_i^k. \tag{5.2}$$

By the Maximum Principle [58], for every half-space

$$\mathcal{H}^i = \mathcal{H}\text{space}(\mathbf{n}_i^k, c_i^k) = \{\mathbf{x} \mid \mathbf{n}_i^k \cdot \mathbf{x} \leq c_i^k\}$$

supporting $\mathcal{P}^k$ there exists an input $\tilde{\mathbf{u}} \in \mathcal{U}$ such that computing the successors of $\mathcal{H}^i$ under $\tilde{\mathbf{u}}$ is sufficient to derive a tight polyhedral approximation of the reachable set.

It is easy to see that the change of normal direction under the dynamics of this system is only induced by the matrix $\mathbf{A}$. Indeed, geometrically speaking, the term $(\mathbf{b}+\mathbf{u}(k))$ of the system (5.1) corresponds to a translation and does not change the normal vector direction. Thus the evolution of a half space orientation under a linear dynamics with inputs is also given by the equation 5.2.

The following proposition shows that one can compute the successor of a half-space of the convex polyhedron $\mathcal{P}^k$.

Let a half-space $\mathcal{H}^i = \mathcal{H}\text{space}(\mathbf{n}_i^k, c_i^k)$ which supports the convex polyhedron $\mathcal{P}^k$. We can compute with (5.2) the vector $\mathbf{n}_i^{k+1}$, the resulting normal under the system dynamics. Let $\tilde{\mathbf{u}}^i \in \mathcal{U}$ be the solution of the following linear optimization problem :

$$\tilde{\mathbf{u}}^i = \text{ argmax} \mathbf{n}_i^{k+1} \cdot \mathbf{u} \text{ over } \mathbf{u} \in \mathcal{U}$$

Using this input, we can compute the coefficient $\tilde{c}_i^{k+1}$ corresponding to the result of the following optimization problem:

$$\begin{array}{rlll} \tilde{c}_i^{k+1} = & \text{maximize} & \mathbf{n}_i^{k+1} \cdot (\mathbf{A}\mathbf{x}(k) + \mathbf{b} + \tilde{\mathbf{u}}^i) \\ & \text{over} & \mathbf{x}(k) \in \mathbb{R}^n \\ & \text{subject to} & \mathbf{n}_i^k \cdot \mathbf{x}(k) \leq c_i^k. \end{array}$$

Then, the half-space $\mathcal{H}\text{space}(\mathbf{n}_i^{k+1}, \tilde{c}_i^{k+1})$ corresponds to the image of $\mathcal{H}^i$ under the input $\tilde{\mathbf{u}}^i$,

$$\mathcal{H}\text{space}(\mathbf{n}_i^{k+1}, \tilde{c}_i^{k+1}) = \mathit{Reach}(\mathcal{H}^i, \tilde{\mathbf{u}}^i).$$

Proposition 12.

$$\forall \mathbf{u} \in \mathcal{U}, \mathit{Reach}(\mathcal{H}^i, \mathbf{u}) \subseteq \mathcal{H}\mathit{space}(\mathbf{n}_i^{k+1}, \tilde{c}_i^{k+1}). \tag{5.3}$$

Proof. We start by considering the problem of the successor of a half-space $\mathcal{H}^i = \mathcal{H}\text{space}(\mathbf{n}_i^k, c_i^k)$ under the dynamics (5.1) with a fixed input u. Given a normal $\mathbf{n}_i^{k+1}$ computed with using (5.2), the polyhedral coefficient c_i^{k+1} can be then obtained by the Maximum Principle [96] by solving the following optimization problem:

$$\begin{array}{rlll} c_i^{k+1} = & \text{maximize} & \mathbf{n}_i^{k+1} \cdot (\mathbf{A}\mathbf{x}(k) + \mathbf{b} + \mathbf{u}) \\ & \text{over} & \mathbf{x}(k) \in \mathbb{R}^n \\ & \text{subject to} & \mathbf{n}_i^k \cdot \mathbf{x}(k) \leq c_i^k. \end{array}$$

By considering the associativity of the dot product, the only term depending of the input $\mathbf{u}$ is $\mathbf{n}_i^{k+1} \cdot \mathbf{u}$. This term corresponds to the objective function of the maximization problem for computing $\tilde{\mathbf{u}}^i$ then

$$\forall \mathbf{u} \in \mathcal{U},\ \mathbf{n}_i^{k+1} \cdot \mathbf{u} \leq \mathbf{n}_i^{k+1} \cdot \tilde{\mathbf{u}}^i.$$

This implies that $c_i^{k+1} \leq \tilde{c}_i^{k+1}$ then

$$\mathcal{H}\text{space}(\mathbf{n}_i^{k+1}, c_i^{k+1}) \subseteq \mathcal{H}\text{space}(\mathbf{n}_i^{k+1}, \tilde{c}_i^{k+1}).$$

□

As the evolution of the normal $\mathbf{n}_i^{k+1}$ is independent of the input, for all $\mathbf{u} \in \mathcal{U}$ the boundaries of the half-spaces $\mathcal{H}\text{space}(\mathbf{n}_i^{k+1}, c_i^{k+1})$ are parallel to each other, as shown in Figure 5.1.

The following proposition [96, 41] shows that one can *conservatively* approximate the reachable set from the convex polyhedron $\mathcal{P}^k = \langle \mathbf{N}^k, \mathbf{c}^k \rangle$.

Proposition 13. *For every $i \in \{1, \ldots, m\}$, let $\tilde{c}_i^k$ the polyhedral coefficient obtained from Proposition 12. We obtain the following inclusion:*

$$\mathit{Reach}(\mathcal{P}^0, k+1) \subseteq \bigcap_{i=0}^{m} \mathcal{H}\mathit{space}(\mathbf{n}_i^{k+1}, \tilde{c}_i^{k+1}).$$

Algorithm 3 summarizes this method.

We denote by $\widehat{\mathit{Reach}^{k+1}}(\mathcal{P}^0)$ the approximation obtained. Note that if the input set $\mathcal{U}$ is a bounded convex polyhedron then $\tilde{\mathbf{u}}^i(k)$ can be selected at one of its vertices at every time point k.

In the following, we talk about the over-approximation induced by this algorithm and we present a way to refine the reachable sets.

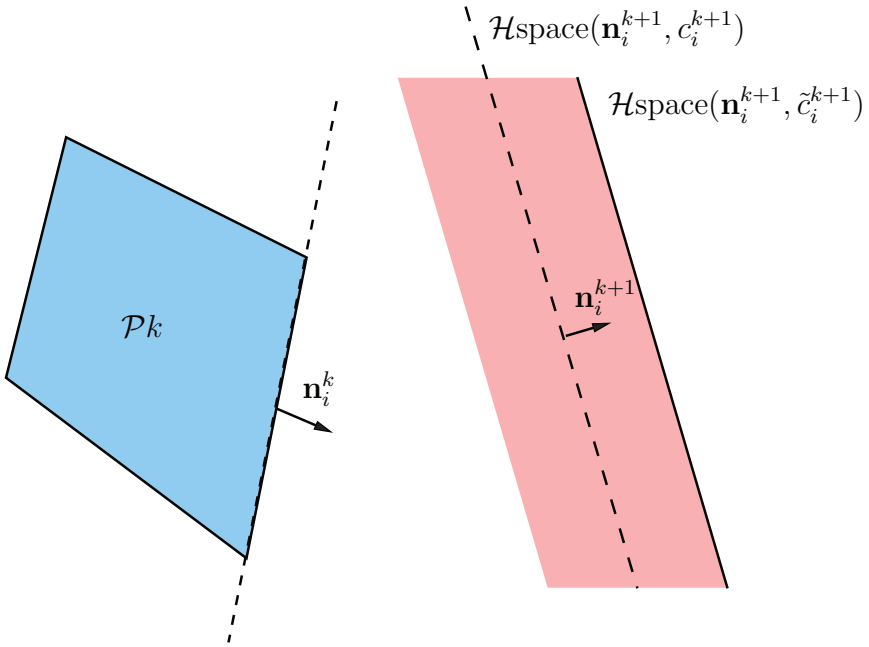

Figure 5.1: Illustration of the new half-space computation; the set on the right represents the interval where the half-space can be shifted according to $\mathbf{u} \in \mathcal{U}$. The dashed line is an example with an arbitrary $\mathbf{u}$, the plain line represents the half space computed with $\tilde{\mathbf{u}}$.

5.1.2 Approximation error

The following propositions concern the over-approximation error.

Proposition 14. *For a linear system* $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{b} + \mathbf{u}(k)$ *where the set of input function is a singleton, the equality in Proposition 13 is achieved, that is the set* $\widehat{Reach}^k(\mathcal{P}^0) = Reach(\mathcal{P}^0, k)$.

To prove this result, we notice that the image of the intersection of two sets $\mathcal{A}$ and $\mathcal{B}$ by a function g satisfies $g(\mathcal{A} \cap \mathcal{B}) \subseteq g(\mathcal{A}) \cap g(\mathcal{B})$. In particular, if g is injective, $g(\mathcal{A} \cap \mathcal{B}) = g(\mathcal{A}) \cap g(\mathcal{B})$. Indeed, the solutions of deterministic systems are injective since they do not cross one another. On the contrary, the solutions of non-deterministic systems are generally not injective (since from different initial states, different inputs may lead the system to the same state). This implies that for linear systems with uncertain input, Algorithm 3 produces only an over-approximation of this set.

Proposition 15. *If a half-space* $\mathcal{H}$ *supports the initial set* $\mathcal{P}^0$*, that is its boundary contains a point in* $\mathcal{P}^0$*, then for every* $k > 0$ *the half-space* $Reach(\mathcal{H}, k)$

Algorithm 3 Over-approximating $Reach(\mathcal{P}^0, k)$

```
/*Compute the half-space directions for the next polyhedron*/
/*Input: Pk = ⟨N^k, c^k⟩*/
N^{k+1} = ComputeTemplate(N^k)
for all n_i^{k+1} in N^{k+1} do
  /*Compute ũ^i(k)*/
  ũ^i(k) = maximize n_i^{k+1} · u(k)
           over u(k) ∈ R^n
           subject to u(k) ∈ U.
  /*Compute c_i^{k+1}*/
  c_i^{k+1} = maximize n_i^{k+1} · (A_j x(k) + b_j + ũ^i(k))
           over x(k) ∈ R^n
           subject to T(k) · x(k) ≤ c(k)
end for
return Pk + 1 = ⟨N^{k+1}, c^{k+1}⟩
```

computed as shown in the formula (5.3) supports the exact reachable set $Reach(\mathcal{P}^0, k)$.

The proof of this can be straightforwardly established from the fact that the supporting point $\mathbf{y}^*(k) = \Phi^l(\mathbf{y}, k, \tilde{\mathbf{u}})$ of each $Reach(\mathcal{H}, k)$ is indeed a point in the exact reachable set from $\mathcal{P}^0$ since it is computed as a successor from a point in $\mathcal{P}^0$ under an admissible input function.

It is important to consider that the shape of the exact image of a convex polyhedron by a discrete linear dynamics with inputs depends of the input set $\mathcal{U}$ and is not necessary a polyhedron. The following example illustrates this fact.

Example We consider the following 2-dimensional system:

$$\mathbf{x}(k+1) = \begin{pmatrix} 2.04 & -0.64 \\ 0.84 & 0.24 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 1.5 \\ -0.8 \end{pmatrix}.$$

The initial set $\mathcal{P}^0$ is a box represented by the following constraints:

$$\begin{cases} x_1 \leq 2 \\ x_2 \leq 2 \\ -x_1 \leq -1 \\ -x_2 \leq -1 \end{cases}$$

Figure 5.2 shows the exact image $Reach(\mathcal{P}^0, k)$ of a convex polyhedron $\mathcal{P}^0$ by the this system (5.1.2) and the approximation obtained using Algo-

rithm (3) $\widehat{Reach}^k(\mathcal{P}^0)$ with an input included in a ball centered at origin of radius 0.5. In this case the exact reachable set has curved faces and is not a polyhedra.

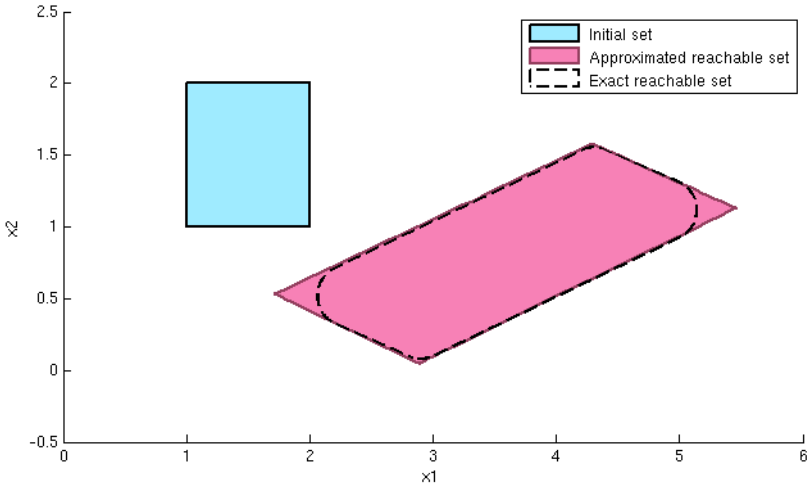

Figure 5.2: One reachable set for a linear system with an input included in a ball

When the input set is a polyhedron, the exact reachable set is a polyhedron with more facets that the initial set as illustrated in figure 5.3.

Refinement From the above results, to improve the approximation accuracy one can use additional half-spaces in the description of the initial set, such that with respect to the initial set their associated constraints are redundant, but under the system's dynamics they evolve into new half-spaces which can be useful for bounding the approximation in some directions. In order to significantly reduce the approximation error which is measured by the Hausdorff distance between the approximated and the exact sets, it is important to find the area when the difference between these sets is large. In the following, we propose two methods for refining the reachable set approximation by dynamically adding constraints.

5.2 Refinement using sharp angle

As mentioned earlier, the constraints to add should not change the polyhedron and thus must be redundant. Another requirement is that the corresponding half-spaces should be positioned where the approximation error is

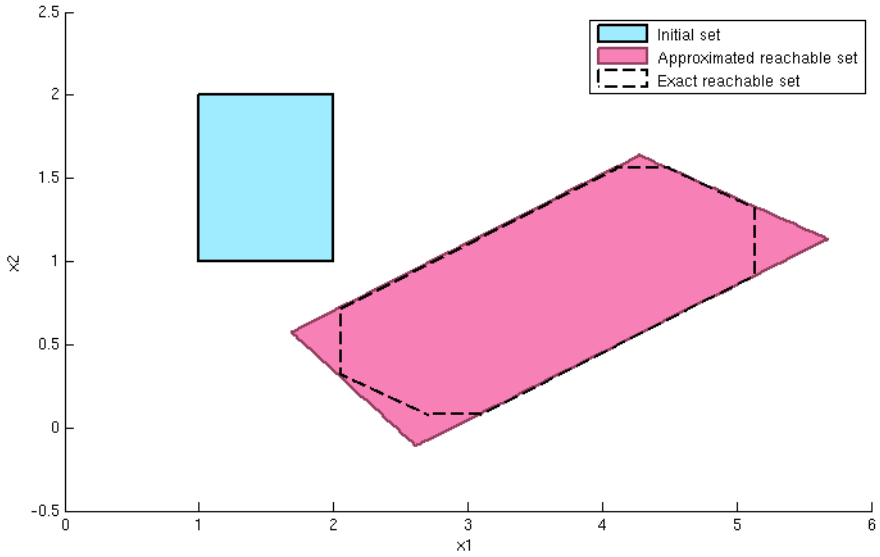

Figure 5.3: One reachable set for a linear system with input included in a cube

large. The over-approximation error indeed occurs mostly around the intersections of the half-spaces. In addition, this error is often large when the angle between two adjacent half-spaces, which can be determined from their normal vectors, is smaller than some threshold σ. We call them *adjacent half-spaces with sharp angle*.

Indeed, when two adjacent half-spaces form a sharp angle, the area near their intersection is elongated, which can causes a large difference between the polyhedral approximation and the actual reachable set. Hence, in order to better approximate the exact boundary, one needs to use more approximation directions.

A constraint to add can be determined as follows. We consider a convex polyhedron $\mathcal{P} = \langle \mathbf{N}, \mathbf{c} \rangle$ where $\mathbf{N} \in \mathbb{R}^{m \times n}$ and $\mathbf{c} \in \mathbb{R}^m$. Let $\mathcal{H}^i = \mathcal{H}\text{space}(\mathbf{n}^i, \mathbf{c}^i)$ and $\mathcal{H}^j = \mathcal{H}\text{space}(\mathbf{n}^j, \mathbf{c}^j)$ two half-space of $\mathcal{P}$ corresponding to *adjacent faces with sharp angle*. The normal vector $\mathbf{n}^{ij}$ of the new constraint can be defined by a linear combination of $\mathbf{n}^i$ and $\mathbf{n}^j$: $\mathbf{n}^{ij} = w_1\mathbf{n}^i + w_2\mathbf{n}^j$ where $w_1, w_2 \in \mathbb{R}^+$. We next determine a supporting point of the constraint. To this end, we find a point on the facet in the intersection between the half-spaces $\mathcal{H}^i$ and $\mathcal{H}^j$ by solving the following linear programming problem:

$$\begin{array}{lll} \text{minimize} & \mathbf{n}^{ij} \cdot \mathbf{x} & \\ \text{over} & \mathbf{x} \in \mathbb{R}^n & \\ \text{subject to} & \mathbf{n}^k \cdot \mathbf{x} \leq \mathbf{c}, \quad k \in \{1, \cdots, m\} \backslash \{i, j\} & (5.4) \\ & \mathbf{n}^i \cdot \mathbf{x} = \mathbf{c}^i & \\ & \mathbf{n}^j \cdot \mathbf{x} = \mathbf{c}^j. & \end{array}$$

The solution $\mathbf{x}^*$ of the above LP problem yields the new constraint $\mathbf{n}^{ij} \cdot \mathbf{x} \leq \mathbf{n}^{ij} \cdot \mathbf{x}^*$, which can be used for refinement purposes.

It is important to note that while sharp angles between half-spaces are useful to identify the areas where the approximation error might be large, sharp angles can also be formed when the system converges to some steady state. In this case, "curving" the approximated set does not significantly improve the accuracy, because their normal vectors under the system's dynamics become very close to each other, and we choose not to add constraints in this case.

Dynamical refinement

In the above we showed how to determine redundant constraints for refinement. The refinement process can be done dynamically as follows. During the computation, if at the k^{th} step, the sharp angle criterion alerts that the approximation error might be large, we move $r \leq k$ steps backwards and add constraints for each pair of adjacent half-spaces with sharp angles. The computation is then resumed from the $(k-r)^{th}$ step.

An important remark is that the larger r is, the more significant accuracy improvement is achieved, at the price of more computation effort. Indeed, when we backtrack until the first step, the half-spaces corresponding to the added constraints actually support the boundary of the initial set. Thus, by Proposition 15, their successors also support the exact reachable set from $\mathcal{P}^0$, which guarantees *approximation tightness*. On the contrary, if $r < k$, it follows from the above LP problem (5.4) that the added constraints only support the approximated reachable set and their boundaries therefore are not guaranteed to contain a truly reachable point.

Figure 5.4 illustrates the improvement in accuracy achieved by this method for a 2-dimensional linear system whose matrix is a Jordan block[1]. The colored parts correspond to the approximation error which is reduced by adding constraints using the sharp angle criterion.

[1] A Jordan block is a matrix whose main diagonal is filled with a fixed number and all the entries which are directly above and to the right of the main diagonal are 1.

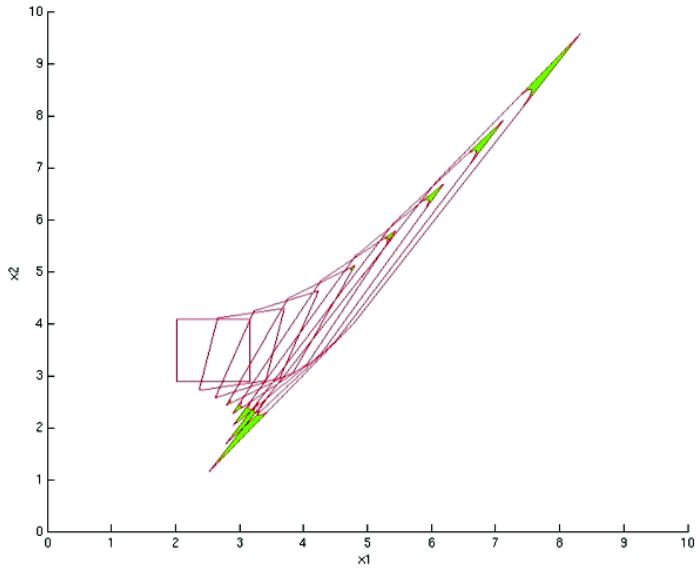

Figure 5.4: Adding constraints can reduce approximation error (which is the colored areas in the figure).

5.2.1 Refinement using critical direction

A good compromise between accuracy and computation time depends on the problems to solve and the available computation budget. In this section, we discuss a refinement procedure specific for safety verification problems. As we have seen earlier, in order to guarantee a desired approximation error bound, measured using the Hausdorff distance between the approximated and the exact sets, one needs to assure that their difference does not exceed the bound in all directions. Nevertheless, when reachability analysis is used to prove a safety property, this measure is not appropriate for characterizing the potential of reaching a unsafe zone. In this case, one is more interested in computing an approximation that may not be precise (with respect to the Hausdorff distance to the exact set) but enables deciding whether the exact set intersects with the unsafe set. To this end, we use a measure, called *directional distance*.

Directional distance

Given two convex sets $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^n$, a Boolean function $contact(\mathcal{A}, \mathcal{B}) = true$ if and only if the following two conditions are satisfied:

1. $Int(\mathcal{A}) \cap Int(\mathcal{B}) = \emptyset$;

2. $\delta\mathcal{A} \cap \delta\mathcal{B} \neq \emptyset$ where $Int(\mathcal{A})$ is the interior of $\mathcal{A}$ and $\delta\mathcal{A}$ is its boundary.

Intuitively, $contact(\mathcal{A}, \mathcal{B})$ is true if and only if $\mathcal{A}$ and $\mathcal{B}$ intersect with each other and, in addition, they intersect only on their boundaries.

If $\mathcal{A} \cap \mathcal{B} \neq \emptyset$, the directional distance between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$\rho(\mathcal{A}, \mathcal{B}) = \inf_{\mathbf{p} \in \mathbb{R}^n} \{||\mathbf{p}|| : contact(\mathcal{A}, \mathcal{B} + \mathbf{p})\}.$$

where $||\mathbf{p}||$ denotes the Euclidian length of the vector $\mathbf{p}$, $\mathcal{B}+\mathbf{p} = \{\mathbf{x}+\mathbf{p} \mid \mathbf{x} \in \mathcal{B}\}$ is the result of translating the set $\mathcal{B}$ by the vector $\mathbf{p}$.

If $\mathcal{A}$ and $\mathcal{B}$ are in contact, $\rho(\mathcal{A}, \mathcal{B}) = 0$. Note that if the sets $\mathcal{A}$ and $\mathcal{B}$ are overlapping, the above $\rho(\mathcal{A}, \mathcal{B}) > 0$ measures the "depth" of the intersection of the two sets. To generalize this definition to overlapping sets, we need to distinguish this case from the case where $\mathcal{A}$ and $\mathcal{B}$ do not intersect. To do so, we use a signed version of the directional distance that has positive values if the two sets are non-overlapping and negative values otherwise.

Definition 17. *Given two convex sets $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^n$, the signed directional distance between $\mathcal{A}$ and $\mathcal{B}$ is defined as*

$$\rho_s(\mathcal{A}, \mathcal{B}) = \begin{cases} \rho(\mathcal{A}, \mathcal{B}) & \text{if} \mathcal{A} \cap \mathcal{B} \neq \emptyset, \\ -\rho(\mathcal{A}, \mathcal{B}) & \text{otherwise.} \end{cases}$$

This directional distance in two and three dimensions has been used in robotics for collision detection [67]. The advantages of using the signed directional distance for the safety verification problem are the following:

- It measures the potential and the degree of collision between two moving objects, which, in the context of reachability computation, provides useful information on the necessity of refining reachable set approximations.

- For convex polyhedral sets, it can be estimated efficiently, as we will show in the sequel.

Signed directional distance estimation and refinement algorithm

The signed directional distance between two convex polyhedra can be computed using existing algorithms (such as in [26]). However, these algorithms are specific for two and three dimensions and require complete polyhedral boundary descriptions (such as their vertices and facets). In high dimensions

these descriptions are very expensive to compute, which will be discussed more in Section 5.3. We therefore focus on the problem of estimating the signed distance using only constraint descriptions of polyhedra. Our solution can be summarized as follows. The underlying idea is based on the relation between the signed directional distance $\rho_s(\mathcal{A}, \mathcal{B})$ and the Minkowski difference $\mathcal{A} \ominus \mathcal{B}$. The latter is defined as follows:

$$\mathcal{A} \ominus \mathcal{B} = \{\mathbf{b} - \mathbf{a} \mid \mathbf{a} \in \mathcal{A} \wedge \mathbf{b} \in \mathcal{B}\}.$$

Intuitively, the Minkowski difference contains the translation directions that can bring $\mathcal{A}$ into contact with $\mathcal{B}$. Its relation with the signed distance that we exploit is expressed by: $\rho_s(\mathcal{A}, \mathcal{B}) = \rho_s(\mathbf{0}, \mathcal{A} \ominus \mathcal{B})$ where $\mathbf{0}$ is the singleton set that contains the origin of $\mathbb{R}^n$. Again, the vertex description of the Minkowski difference set can be expensive to compute, we resort to a constraint description of this set that can be efficiently computed. To this end, we consider a particular set of translation vectors corresponding to moving the half-space of a face e corresponding to a half-space $\mathcal{H}^e$ of $\mathcal{A}$ in the direction of its normal $\mathbf{n}^e$ by a distance d_e so that the half-space touches $\mathcal{B}$. Then, it can be proved that the half-space $\mathcal{H}^{d_e} = \{\mathbf{x} \ : \ \mathbf{n}^e \cdot \mathbf{x} \leq d_e\}$ contains at least one face of the exact Minkowski difference $\mathcal{A} \ominus \mathcal{B}$. Let $\mathcal{M}$ be the convex polyhedron composed by the intersections of all such half-spaces $\mathcal{H}^{d_e}$. This implies that

$$\mathcal{A} \ominus \mathcal{B} \subseteq \mathcal{M} = \bigcap_{\forall e} \{\mathbf{x} \mid \mathbf{n}^e \cdot \mathbf{x} \leq d_e\}.$$

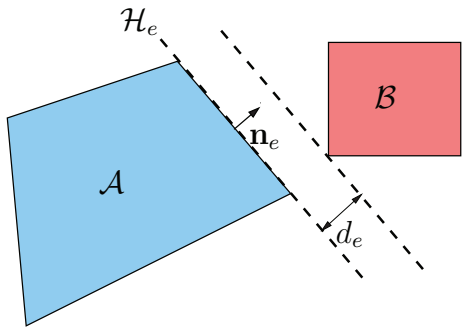

Figure 5.5: Estimation of the directional distance.

In two or three dimensions, it is possible to obtain the exact constraint description of $\mathcal{A} \ominus \mathcal{B}$ by considering other translation types (such as an edge of $\mathcal{A}$ moves along an edge of $\mathcal{B}$). Nevertheless, this computation requires the vertex descriptions of the polyhedra and we thus omit it.

Since $\mathcal{M}$ is an over-approximation of $\mathcal{A} \ominus \mathcal{B}$, it can be proved that

$$\begin{cases} \rho_s(\mathbf{0}, \mathcal{M}) \leq \rho_s(\mathcal{A}, \mathcal{B}) & \text{if} \mathcal{A} \cap \mathcal{B} = \emptyset, \\ \rho_s(\mathbf{0}, \mathcal{M}) \geq \rho_s(\mathcal{A}, \mathcal{B}) & \text{otherwise.} \end{cases}$$

The distance $\rho_s(\mathbf{0}, \mathcal{M})$ can then be easily determined, since it is exactly the largest value of all d_e. We use $\rho_s(\mathbf{0}, \mathcal{M})$ as an estimate of the signed directional distance between $\mathcal{A}$ and $\mathcal{B}$.

It is important to note that this estimate is conservative regarding its utility as a critical situation alert. Indeed, if the two sets do not overlap, the result is smaller than the exact separating distance; otherwise, its absolute value is larger than the exact penetration distance. It is important to note again that the above estimation, which does not involve expensive vertex computation, is time-efficient.

In the context of safety verification, the set $\mathcal{A}$ plays the role of the reachable set and $\mathcal{B}$ the unsafe set. The constraints of $\mathcal{A}$ corresponding to the largest values of d_e are called *critical* because they are closest to $\mathcal{B}$ with respect to the directional distance measure. Their identification is part of the above computation of the signed directional distance between $\mathcal{A}$ and $\mathcal{B}$.

Even if the angle between a critical constraint and one of its adjacent constraints does not satisfy the sharp angle criterion, we still refine around their intersection. The refinement can then be done using the same method for adjacent half-spaces with sharp angles, described in the previous section.

Refinement using constraints from the safety specification

Let $\Lambda_{\bar{\mathcal{B}}}$ be the set of normal vectors of the complement of each half-space of the unsafe set $\mathcal{B}$. In many cases, intersection of the reachable set and $\mathcal{B}$ can be tested more easily if the reachable set description contains a constraint whose normal vector coincides with a vector in $\Lambda_{\bar{\mathcal{B}}}$. Hence, for each direction $\mathbf{d} \in \Lambda_{\bar{\mathcal{B}}}$, the predecessors of $\mathbf{d}$ by the adjoint system can be used to define constraints to add, again by solving the LP problem (5.4). The refinement using the predecessors of such directions is needed when the reachable set is close to the unsafe set. The refinement procedure using critical directions is summarized in Algorithm 4.

In this algorithm, $\mathcal{P}^0$ is the initial polyhedron. The function $dir(\mathcal{P}^k)$ returns the set of normal vectors of the constraints which represent $\mathcal{P}^k$. The function $AddConstraints(\mathcal{P}^k, \Lambda)$ adds to the half-spaces of $\mathcal{P}^k$ the new half-spaces that support $\mathcal{P}$ with normal vectors in Λ. The operator $Pre_r(\Lambda)$ computes the orientation of Λ vectors r steps backward using the adjoint system presented in (5.2) page 99.

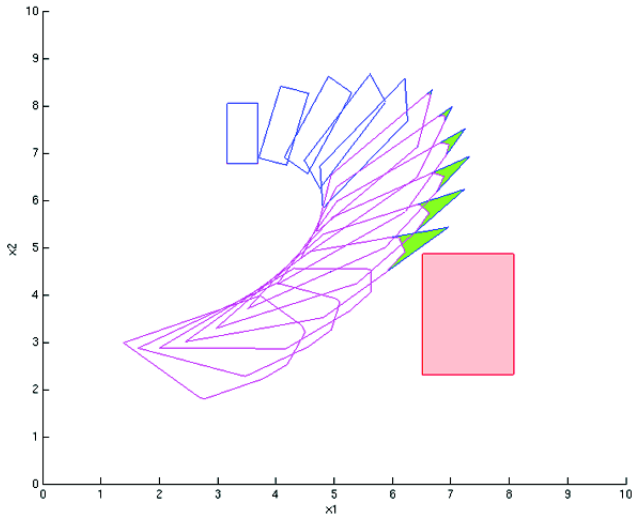

Figure 5.6: Refinement using critical directions on a 2-dimensional example.

Algorithm 4 Refinement Using Critical Directions

$\mathcal{P}^1 = \widehat{Reach}^1(\mathcal{P}^0)$ */* One step computation */*
$k = 1$
while $k \leq k_{max}$ **do**
  if $\rho_s(\mathcal{P}^k, \mathcal{B}) \leq \eta$ **then**
    / retrieve the set of critical constraints */*
    $H_c^k = criticalConstraints(\mathcal{P}^k, \mathcal{B})$
    / Retrieve r-step predecessors of the normal vectors of H_c^k */*
    $\Lambda_c^{k-r} = Pre_r(dir(H_c^k))$
    $\Lambda_b^{k-r} = Pre_r(\Lambda_{\bar{\mathcal{B}}})$ */* Predecessors of the normal vectors of $\mathcal{B}$ */*
    $\Lambda^{k-r} = \Lambda_b^{k-r} \cup \Lambda_c^{k-r}$
    $\mathcal{P}^{k-r} = AddConstraints(Poly(\mathcal{P}^{k-r}), \Lambda^{k-r})$
    $k = k - r$
  end if
  $\mathcal{P}^{k+1} = \widehat{Reach}^1(\mathcal{P}^k)$ */* One step computation */*
  $k = k + +$
end while

Figure 5.6 shows the result obtained for a 2-dimensional system in Jordan block form using the above algorithm. The rectangle on the right is the unsafe set. When the reachable set is close to the unsafe set, the algorithm backtracks a number of steps and adds new constraints. This refinement allows approximating more precisely the actual reachable set near the bad set and thus proving that the system does not enter the unsafe set. The colored zones are the parts of the over-approximation error eliminated by the added constraints. It can be seen from the figure that the aprroximation is refined only in the critical zones near the unsafe set.

5.3 Experimentation

We emphasize that in our development so far the algorithms use the constraint description and do not require the vertex description of polyhedra. Indeed, the transformation from a constraint description to a vertex description is known as vertex enumeration and the inverse transformation is known as facet enumeration. To show the computational complexity of these problems, we mention the algorithm published in [16] which finds m_v vertices of a polyhedron defined by a non-degenerate system of m inequalities in n dimensions (or, dually, the facets of the convex hull of m points in n dimensions, where each facet contains exactly n given points) in time $O(mnm_v)$ and $O(mn)$ space.

From our experience in using polyhedra for reachability computation for continuous and hybrid systems, we noticed that many operations (such as, the convex-hull) are extremely time-consuming, especially in high dimensions. Degeneracy of sets, such as flatness, which occurs frequently in reachable set computation, is also another important factor that limits the scalabibility of existing polyhedron-based algorithms. It is fair to say that they can handle relatively well systems of dimensions only up to 10. This therefore motivated a lot of research exploiting other set representations, as discussed in the state of the art.

On the other hand, when trying to solve a specific verification problem, it is not always necessary to maintain both the vertex and the constraint descriptions of polyhedra. Indeed, for many tasks in a verification process, vertex enumeration can be avoided, such as in the algorithms we presented so far. We have implemented the above described algorithms of reachability computation with refinement for linear continuous systems and this implementation enabled us to handle continuous systems of dimensions much higher than what can be treated by typical polyhedron-based reachability analysis algorithms, such as [14].

dim n	Final number of added constraints	Computation time in seconds
2	32	0.4
5	59	9.14
10	10	150.93
20	–	–
50	–	–
100	–	–

Table 5.1: Computation time for 100 steps on some linear systems in Jordan block form using the implementation with vertex computation.

In the following, we present some experimental results obtained using this implementation. To evaluate the performance of our methods, we generated a set of linear systems in Jordan block form in various dimensions up to 100 with the values in the diagonal are all equal to (-0.8). The input set $U = [-0.1, 0.1]^n$ and the initial set $\mathcal{P}^0 = [-1, 1]^n$ are boxes (whose number of constraints equal to $2n$). The threshold for the sharp angle criterion is $\sigma = 60$ degrees.

To show the advantage of the constraint-based implementation, we also tested an implementation using vertex computation on the same examples. In this implementation with vertex computation, the constraint adjacency information can be directly derived and the constraints to add are easier to compute. However, the cost for vertex computation is high, which limited the application of this implementation to the examples of dimensions only up to 10, as shown in Table 5.1. For the example in 10 dimensions, we had to fix a smaller maximal number of constraints to add, in order to produce the result in a feasible computation time.

The constraint-based implementation is more time-efficient and thus allows us to handle systems of higher dimensions, as shown in Table 5.2.

This method was presented and published in [15]. This refinement technique is similar to the well-known counter-example based refinement approaches in the idea of guiding the refinement process using the previously explored behaviors. However, to the best of our knowledge, the idea of using redundant constraints for refinement purposes is new. Another novelty in our results is the use of the directional distance to measure approximation effectiveness in proving safety properties and to guide the refinement process.

dim n	Final number of added constraints	Computation time in seconds
2	32	0.48
5	59	0.76
10	36	2.22
20	38	3.67
50	94	42.07
100	197	335.95

Table 5.2: Computation time for 100 steps on the same linear systems in Jordan block form using the constraint-based implementation.

Chapter 6

Implementation

Résumé: *Dans ce chapitre, nous présentons un outil prototype dans lequel nous avons implanté les méthodes présentées dans les chapitres précédents. Nous présentons dans un premier temps l'architecture globale du programme, nous décrivons ensuite ses fonctionnalité et nous finissons par présenter ses interfaces d'entrée et de sortie.*

In this chapter, we describe an implementation of the techniques presented in the previous chapters in a prototype tool named NLToolBox. The two major functionalities of the tool are

- reachability analysis of discrete-time polynomial systems,
- reachability analysis of continuous-time and discrete-time nonlinear systems using hybridization.

For both of these functionalities, reachable set refinement using redundant constraints can be used to obtain accuracy gain. For the analysis of approximated systems generated by the hybridization process, the tool can be connected to other existing tools for linear systems. Currently this part is implemented in the framework of the tool **d/dt** [10] and we are planning to connect it to the tool **SpaceEx** [46] of Verimag.

In the following we first present the global architecture of this tool, then we describe the implemented data structures and main algorithms. We end this chapter with a short description of the user interface.

6.1 Architecture

The architecture of this tool has been designed to be extendable in terms of set representations, methods for handling approximated systems, and dynamical system description formalisms. The architecture of the tool is summarized by the class diagram in Figure 6.1.

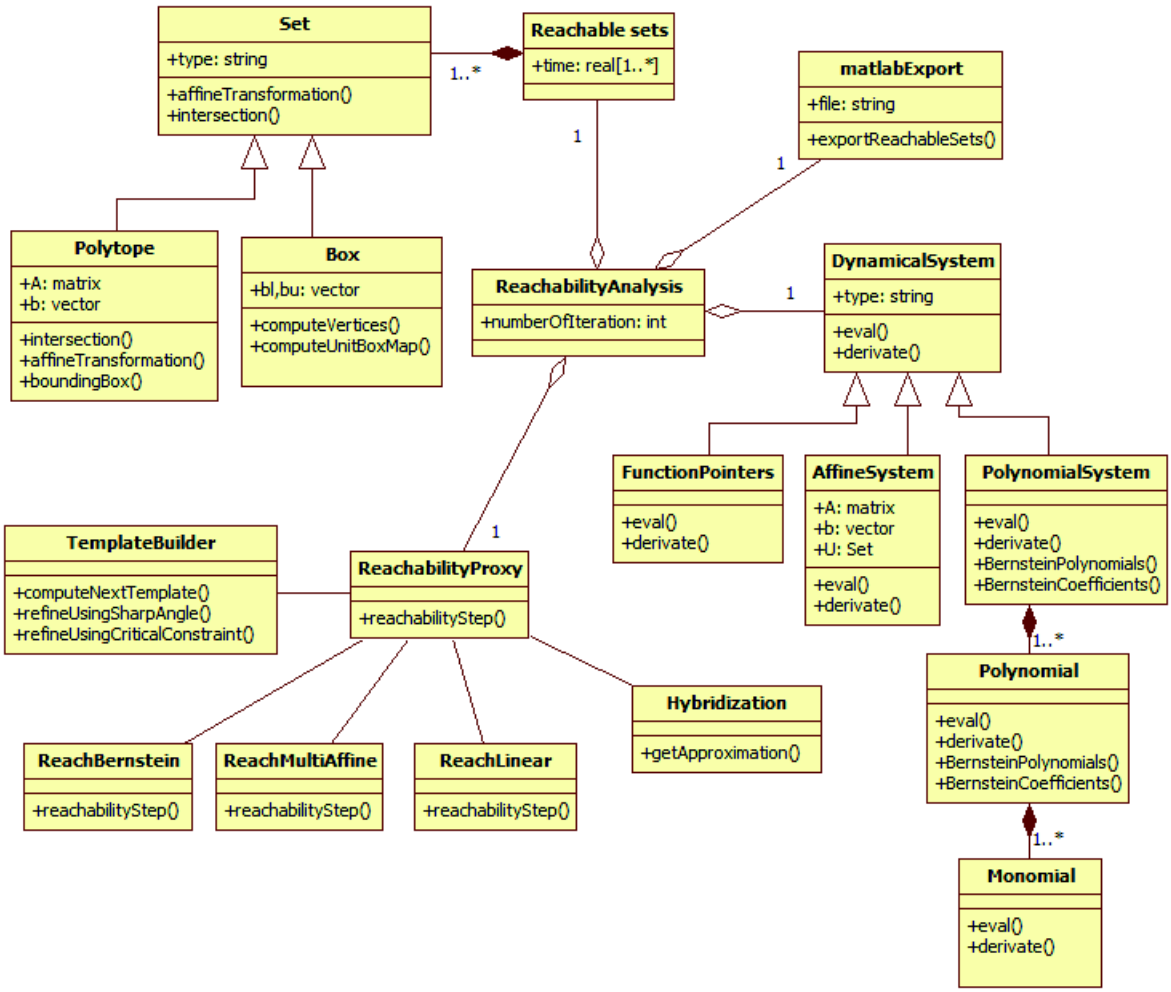

Figure 6.1: Class diagram of the program

The main components of this tool are the following:

- **ReachableSet**: an interface for the set representations which specifies the common operations needed by the reachability analysis methods.

- **DynamicalSystem**: an interface for the dynamical system description formalisms.

- **ReachProxy**: a proxy class that calls the image computation and hybridization from the chosen reachability method. Additionally, associated with this class is a class for computing polytopic templates and for performing refinement using the techniques presented in Chapter 5.

6.2 Data structures

6.2.1 Dynamical system description

We have implemented three classes for dynamical system descriptions, which are children of the interface **DynamicalSystem**:

- The class **NonlinearSystem** is a general proxy class to describe a large class of nonlinear dynamics. The systems are described in **C** functions.

- The class **AffineSystem** describes affine dynamics with input represented by a matrix $\mathbf{A}$, a vector $\mathbf{b}$ and a possible input set $\mathcal{U}$ which correspond to the following differential equation:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b} + \mathbf{u}(t), \mathbf{u}(\cdot) \in \mathcal{U},$$

  The current version of the tool supports polytopes and hypersphere for representing of the input sets.

- The class **PolynomialSystem** is composed of objects of the class **Polynomial** that are themselves composed of objects of the class **Monomial**. Each monomial is characterized by a multi-index vector representing the exponent of each variable and a coefficient. For example, the monomial of 3 variables x_1, x_2 and x_3 given by $-3x_1^2x_3$ can be associated with the multi-index vector $(2, 0, 1)$ and the coefficient -3. We implement a parser to construct monomials, polynomials and polynomial systems from strings. For example, the FitzHugh-Nagumo neuron model described in Cahpter 4 by the differential equations

$$\begin{aligned} \dot{x}_1 &= x_1 - 0.33x_1^3 - x_2 + 0.875 \\ \dot{x}_2 &= 0.08x_1 - 0.64x_2 + 0.056 \end{aligned}$$

  can be constructed using the following strings

```
x1' = x1 - 0.33*x1^3 - x2 + 0.875
x2' = 0.08*x1 - 0.64*x2 + 0.056
```

The common functions of each dynamical system class are: *eval*() which returns the evaluation of the dynamics at a given point, and *derivative*() which returns the derivation of the right part of the differential equations. The **PolynomialSystem** class contains additional methods to compute the Bernstein representation coefficients. Such a coefficient is stored in a structure of map type using its associated multi-index as a key of the map. To avoid iterating on the multi-indices the Bernstein coefficients of which are all 0, we only store multi-indices associated with at least one non-zero Bernstein coefficient.

6.2.2 Set representations

The current set representations are

- boxes, represented by two interval bound vectors (left and right);
- polytopes, represented by constraints (described by a matrix $\mathbf{A}$ and a vector $\mathbf{b}$ which corresponds to the constraints $\mathbf{Ax} \leq \mathbf{b}$). This representation contains additional constructors for template polyhedra, such as rectangles or octagons in 2 dimensions.

The methods needed by the reachability analysis are the inclusion test and the affine transformation. The polytope class contains an additional method to add redundant constraints used by the refinement techniques.

Inclusion test This operation is required for checking the inclusion of a set in an approximation domain when using the hybridization method. For boxes, the inclusion test is trivial, we just compare the bound vector coefficients. For polytopes the inclusion is tested by a partial vertex enumeration described in the following. To check if the polytope described by the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and the vector $\mathbf{b} \in \mathbb{R}^m$ is included in the polytope represesented by the matrix $\mathbf{A'} \in \mathbb{R}^{m' \times n}$ and the vector $\mathbf{b'} \in \mathbb{R}^{m'}$, we solve a set of linear optimization problems

$$\begin{array}{lll} \text{maximize} & \mathbf{A}'_i\mathbf{x} & i \in 1, 2, \ldots, m' \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b} & \end{array}$$

where $\mathbf{A}'_i$ corresponds to a line of the matrix $\mathbf{A'}$. The optimization result is then compared to the coefficient of $\mathbf{b}$, if it is larger than b'_i, the inclusion test returns false; otherwise it continues with other values of i. If for all $i \in 1, 2, \ldots, m'$ the optimization result is not larger than b'_i, the inclusion test returns true.

Bounding box computation This operation is needed by the computation of the unit box map for the Bernstein methods. It is implemented for polytopes by solving linear optimization problems.

6.3 Algorithms

We present in this section the implemented algorithms to perform set image computation and hybridization during the reachability analysis.

6.3.1 Image computation

According to the reachability parameters, the **ReachProxy** class creates an instance of the chosen methods and calls iteratively the image computation method *reach*(). The methods *reach*() takes as input arguments in parameters a set instance and a dynamical system. There are 3 reachability methods implemented as classes using the factory design pattern.

- the class **ReachBernstein** implements the techniques developed in Chapter 4 for the analysis of polynomial system using polytopes. For clarity we summarize the main steps of this image computation method as follows:

    1. A unit box map is computed for mapping the set in question to the box represented by the intervals $[0, 1]^n$.
    2. A template is computed using the class **TemplateBuilder**. This template will define the shape of resulting polytope of the image approximation. According to the analysis parameters, the template can be static (that is, all the instances of the class **Polytope** share the same constant matrix **A**) or dynamic. The dynamic templates computation uses a local approximation of the dynamics to compute a transformation matrix reflecting the rotation of the template induced by the dynamics.
    3. A new polynomial dynamical system is created by composing the initial system given in the argument of the problem and the unit box mapping function computed int Step 1. The Bernstein coefficients are then computed for this new polynomial system.
    4. Using the Bernstein coefficients, a set of affine bound functions are computed.
    5. For each template direction, we solve a linear optimization problem to compute a new constraint of the polytope.

6. Once all new constraints have been computed, the method returns the new polytope.

- An additional class **ReachMultiAffine** as been written to efficiently perform analysis on multi-affine systems using the bounding boxes vertices as presented in Chapter 4.

- The class **ReachLinear** performs the analysis of linear system with input with the method based on optimal control presented in Chapter 5.

6.3.2 Hybridization

This part involves the hybridization techniques presented in Chapter 3. If the reachability method for approximated systems has been chosen in the reachability parameters, the **ReachProxy** class will create an instance of the class **ReachHybridization** and will call the method *getApproximation*() to obtain a hybridization domain and an affine system with input associated with this domain. This approximated system is then used as parameter of the *reach*() method from the **ReachLinear** class.

The *getApproximation*() method takes as arguments a nonlinear dynamics, an initial set and a desired error bound (given in the analysis parameter). The hybridization computation is done in the following steps:

1. According to the dynamics, a curvature tensor is computed globally for quadratic systems or locally for general nonlinear dynamics (which requires solving nonlinear optimization problems). An isotropic transformation matrix is computed from the curvature tensor.

2. A set of vertices representing a simplex containing the initial set is constructed with respect to a size constraint given by the specified error bound.

3. An affine system is interpolated from the simplex vertices. An input is added to the system to take into account the approximation error.

4. The system and the simplex are returned as the current approximated system.

6.3.3 Reachability analysis

The main steps of the reachability analysis procedure are the following:

1. **Initialization**: in this step the user gives the analysis parameters and chooses the reachability methods to use. The methods proposed depends on the class of system dynamics. An instance of the class **ReachProxy** is created with the initialization parameters and an instance of the image computation classes (or the hybridization class) are created in **ReachProxy**. If the hybridization method is used, a first approximated system is computed.

2. **Iterative computation**: In this step, the reachable sets are iteratively computed, and each step returns a new reachable set which is stored in a list and then used as argument in the next reachability iteration. The number of iterations is fixed by the user.

   If the hybridization method is used, an inclusion test is performed on the current hybridization domain. If the result is negative, a new approximation domain is computed and the analysis restarts from the previous reachable set.

   If refinement is used, an additional test is performed after the calling the *reach*() method to check refinement conditions, and if necessary redundant constraints are added to the previously computed set and the analysis is restarted from this set.

3. **Result export**: The final step consists in saving the result in a file which can be executed in Matlab to visualize the result.

6.4 Interfaces

The main procedure of the program takes as input an initial set and a dynamical system. Additional user-defined parameters include a maximal number of iterations, the output file name and the reachability method.

6.4.1 Reachability parameters

Here is the list of reachability parameters athe user should provide:

- **Reachability method**: A choice between the available reachability methods. The available methods according to the type of dynamical systems are shown in the following table.

- **Number of iterations**: An integer corresponding to the maximal number of reachability iterations.

	Affine	Multi-affine	Polynomial	Nonlinear
ReachLinear	✓			
ReachMultiAffine		✓		
ReachBernstein	✓	✓	✓	
Hybridization + ReachLinear	✓	✓	✓	✓

- **Output file name**: A string.

According to the chosen reachability method, additional parameters are required:

- For image computation with the Bernstein and Linear methods:
  - Use dynamical refinement with sharp angle (y/n).
  - Use dynamical refinement with critical directions(y/n).
  - Maximal number of constraints for each set (integer).
- For the Bernstein method: use dynamical template (y/n).
- Hybridization: maximal error bound (real).

6.4.2 Visualization

To visualize the reachability analysis results, the user can use an auto-generated script for Matlab. Boxes and polytopes represented by vertices are plotted using the basic plotting tool of matlab. For plotting polytopes represented by constraints we use the freely available function *plotregion* [21] which plots closed convex regions in 2 and 3 dimensions.

In the case of systems with higher dimensions we plot its bounding projections on the axes chosen by the user. Figure 6.2 shows an example of the exported script that represents 3 octagonal reachable sets of an Euler discretization of the system (6.2.1). The result of the script execution in **Matlab** is shown in Figure 6.3.

6.5 External ressources

This tool was written in **C++** and contains more than 3000 lines of code. The linear optimization problems (needed by the approach using the Bernstein expansion or for adding redundant constraints) are solved using the

```
% AUTO-GENERATED SCRIPT
% This script requires the PLOTREGION function
% http://www.mathworks.com/matlabcentral/fileexchange/9261
% Be sure to have this function in the same folder of this script

pcolor = [0.8,0.3,0.3]; % polyhedra color
% ######################### iteration 1
A = [ 1 0; 0 1; -1 0; 0 -1 ;
    0.7 0.7; -0.7 0.7; -0.7 -0.7; 0.7 -0.7
    ];
b = [0.572568; 2.57669; -0.202627; -2.37616;
    2.16096; 1.67215; -1.88935; -1.28145;  ];

plotregion(-A,-b,[],[],pcolor);
% ######################### iteration 2
A = [ 1 0 ;0 1 ;-1 0 ;0 -1 ;
    0.7 0.7 ;-0.7 0.7 ;-0.7 -0.7 ;0.7 -0.7
    ];
b = [0.372606; 2.56258; 0.0958849; -2.36167;
    1.99753; 1.87028; -1.68001; -1.41524;  ];

plotregion(-A,-b,[],[],pcolor);
% ######################### iteration 3

A = [ 1 0 ;0 1 ;-1 0 ;0 -1 ;
    0.7 0.7 ;-0.7 0.7 ;-0.7 -0.7 ;0.7 -0.7
    ];
b = [0.143865; 2.54518; 0.449822; -2.34287;
    1.81466; 2.10437; -1.42514; -1.56737;  ];

plotregion(-A,-b,[],[],pcolor);

set(gca,'Xlim',[-0.6 0.7]);
set(gca,'Ylim',[2.2 2.7]);
xlabel('x1','fontsize',12);
ylabel('x2','fontsize',12);

%END
```

Figure 6.2: An example of auto-generated script which plot 3 sets

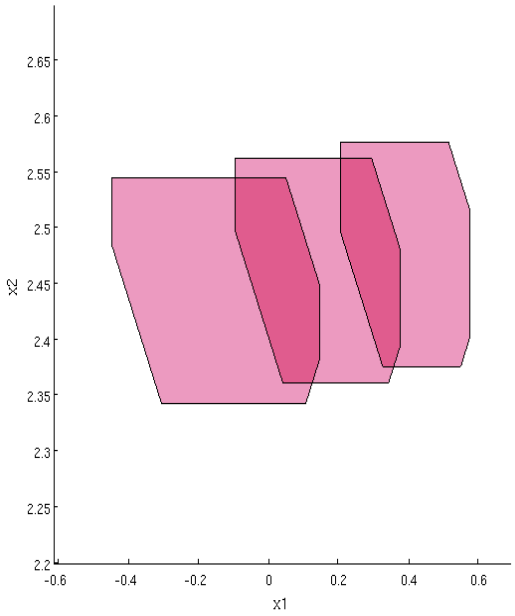

Figure 6.3: The plotting result of the execution of the script in Figure 6.2

open source **lpsolve API** (v 5.5.2.0)[1]. The nonlinear optimization problems (required to compute the curvature tensor in the hybridization technique) are solved using the **NLopt library** [55], a free/open-source library which provides a common interface for a number of different free optimization routines. A number of usual matrix operations such as inversion, eigenvalues an eigenvectors computation or QR-Decomposition are performed using the routines of the freely available software package **LAPACK** [9]. Some reachability function for continuous-time system analysis use the **d/dt** library.

[1] http://lpsolve.sourceforge.net/5.5/

Chapter 7

Conclusion

7.1 Contributions

In this thesis, we presented our contributions to the formal analysis of dynamical systems. These analysis methods can be used for safety verification of critical systems. More concretely, we focused on the problem of efficiently computing an accurate approximation of the reachable sets under nonlinear dynamics given by differential equations. Our aim was also to design scalable methods which can handle large systems.

Hybridization technique The first contribution of this thesis concerns the dynamic hybridization technique for a large class of nonlinear systems. We focused on the hybridization domain construction such that the linear interpolation realized in this domain ensures a desired error between the original system trajectories and those computed with the approximated system. In addition, we showed how one can use the curvature of the system to increase the domain volume. This aims to decrease the occurrences of new hybridization domain computations and thus save computational efforts. We also proved the optimality of this domain construction method for a class of quadratic systems. This construction method was implemented in a new tool and we have experimentally demonstrated the effectiveness of this method.

Bernstein expansion based techniques The second research direction that we followed concerns a subclass of nonlinear dynamical systems which are the polynomial systems. Our results for this problem are based on the Bernstein expansion properties to approximate an initial reachability computation (which requires solving polynomial optimization problems) with an accurate over-approximation (which requires solving linear optimization

problems). To this end, we proposed some novel methods to map a set to the unit box using a change of variables, to compute affine function using Linear Least Square approximation and to choose some template directions which approximate accurately the exact reachable sets. We applied our methods to many biological systems. Due to the optimization problem formulation, the current version is restrained to polytopes represented by constraints.

Refinement using redundant constraints The last theoretical contribution concerns the reachability analysis of linear systems with polyhedral input. We were interested in this problem because such systems often result from approximation of nonlinear systems (as the one presented in Chapter 3). We proposed a technique to refine the reachable sets during the analysis to gain accuracy. This technique is based on the addition of redundant constraints in the description of the reachable set which contributes to reduce the over-approximation error in the next reachable set. Criteria for adding new faces can be the angle between adjacent faces or a directional distance to the bad set. We have implemented and tested this technique on high dimensional systems.

7.2 Perspectives

These contributions have opened many possibilities for extension and research opportunities.

Hybridization technique We plan to investigate alternative methods for computing the isotropic transformation matrix without computing the curvature tensor (which requires solving non-linear optimization problems). Another possible amelioration of this technique concerns the set splitting method in which the dynamics could be taken into consideration.

Bernstein expansion based techniques We plan to extend this technique to continuous time polynomial systems and to nonlinear systems with input. The dynamic template computation can be improved by computing a system approximation for each facet of the polytope rather than at its centroid. An extension to another type of polytope representation could be developed with support functions, encoding the set as the the solution to linear optimization problems. This representation is efficient for the convex-hull and the Minkowsky sum operations, often required by the analysis of systems with input.

Refinement using redundant constraints A possible direction concerns the application of this technique to the hybrid systems analysis especially for the refinements using critical directions by involving guards and invariants. In addition, exploring the Minkowski difference between the reachable set and the unsafe set would allow a better measure of critical proximity of the reachable set under the dynamics of the system.

The integration of all these methods in a hybrid system verification framework also will be an important part of future works.

Conclusion (in French)

Contributions

Nous avons présenté dans cette thèse nos contributions à l'analyse formelle de systèmes. Les méthodes proposées peuvent être utilisées dans le cadre de l'analyse de sûreté de systèmes critiques. Plus concrètement, nous nous sommes concentrés sur le problème du calcul efficace d'approximation précise des ensembles accessibles sous des dynamiques non-linéaires données par des équations différentielles. Nous avons aussi travaillé à rendre ces méthodes applicables sur des systèmes à grande dimension.

Hybridization dynamique La première contribution de cette thèse concerne la technique d'hybridization dynamique qui peut être utilisée pour analyser un grand nombre de systèmes non-linéaires. Nous nous sommes penchés sur la construction de domaines d'approximation telle que l'approximation de système qui y est calculée, respecte une précision donnée impactant la distance entre ses trajectoires et celles du système original. De plus, nous avons proposé l'utilisation de la courbure du système afin d'augmenter le volume du domaine. Cette technique de construction de domaine d'approximation augmente l'efficacité de cette approche en diminuant le nombre de domaines d'approximation requis durant l'analyse. Nous avons, de plus, prouvé l'optimalité de cette méthode pour certaines catégories de systèmes quadratiques. Cette nouvelle méthode de construction a été implantée dans un outil prototype et nous avons expérimentalement démontré l'efficacité de cette méthode.

Calcul d'image pour les systèmes polynomiaux La seconde direction de recherche que nous avons prise, concerne une classe particulière de système non-linéaire: les système polynomiaux. Notre approche est basée sur la formulation du problème de calcul d'image en un problème d'optimisation polynomial puis sur sa relaxation linéaire en utilisant des fonctions affines de borne. Nous utilisons pour calculer ces fonctions, l'expansion de Bernstein. Nous avons proposé de nouvelles méthodes de calcul de ces bornes affines qui

améliorent la précision de l'approximation de l'image. Nous avons, de plus, proposé de nouvelles stratégies afin de choisir des patrons utilisés pour fixer préalablement la forme des polyhèdres, de façon à refléter l'effet de la dynamique sur l'orientation des faces du polyhèdre approximant l'image. Nous avons implanté cette méthode de calcul d'image et l'avons testé sur plusieurs systèmes biologiques.

Raffinement en utilisant des contraintes redondantes La dernière contribution concerne l'analyse d'accessibilité de systèmes linéaires avec entrées. Ces systèmes ont un grand intérêt malgré leur linéarité car ils peuvent être les résultats de techniques d'approximation de systèmes comme celle présentée dans le Chapitre 3. Nous proposons ici, une technique pour raffiner l'approximation de l'ensemble accessible durant l'analyse d'accessibilité. Cette technique est basée sur l'addition de contraintes redondantes dans la description des ensembles accessibles qui contribuent par la suite à réduire l'erreur d'approximation. Nous proposons deux critères afin de trouver des directions optimales pour l'ajout de contraintes redondantes. Nous avons implanté et testé cette technique sur des systèmes de dimension 2 à 100.

Perspectives

Ces contributions ouvrent de nouvelles opportunités de recherches. L'intégration de ces méthodes dans un environnement de vérification de systèmes hybrides est pour le moment, notre priorité.

Hybridization dynamique Nous comptons rechercher des solutions alternatives pour calculer la matrice de transformation isotropique sans passer par le tenseur de courbure qui demande de résoudre des problèmes d'optimisation non-linéaires. Une autre amélioration de cette technique concerne le découpage d'ensemble qui pourrait prendre en compte la dynamique afin de choisir de coupes plus efficaces.

Calcul d'image pour les systèmes polynomiaux L'objectif principal des travaux actuels sur cette méthodes est son extension aux systèmes polynomiaux à temps continu ainsi qu'aux systèmes avec entrées. La stratégie de choix de patron peut être améliorée en calculant une approximation du système sur chaque face plutôt qu'en son centre. Une extension à d'autres types de représentation d'ensemble pourrait être développée en utilisant, par exemple, les fonctions de support. Cette représentation permettrait une plus

grande facilité à effectuer les opérations de calcul de l'enveloppe convexe et des sommes de Minkowski souvent requises avec les systèmes non-autonomes.

Raffinement en utilisant des contraintes redondantes L'ajout de nouveaux critères pour calculer des contraintes redondantes représente une évolution possible de cette méthode. Son utilisation dans le cadre de systèmes hybrides permettrait d'ajouter les gardes dans le calcul des directions critiques. De plus, l'utilisation de cette technique pour l'analyse de systèmes non-linéaire est envisagée pour les systèmes polynomiaux.

Bibliography

[1] Matthias Althoff, Colas Le Guernic, Bruce H. Krogh, and Bruce H. Krogh. Reachable set computation for uncertain time-varying linear systems. In *HSCC*, pages 93–102, 2011.

[2] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. IEEE, 2008.

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[4] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, Pei-Hsin Ho, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

[5] Rajeev Alur, Thao Dang, Franjo Ivancic, and Franjo Ivancic. Progress on reachability analysis of hybrid systems using predicate abstraction. In *HSCC*, pages 4–19, 2003.

[6] Rajeev Alur, Thao Dang, Franjo Ivancic, and Franjo Ivancic. Predicate abstraction for reachability analysis of hybrid systems. pages 152–199, 2006.

[7] Rajeev Alur, Alon Itai, Robert P. Kurshan, Mihalis Yannakakis, and Mihalis Yannakakis. Timing verification by successive approximation. pages 142–157, 1995.

[8] Hirokazu Anai, Volker Weispfenning, and Volker Weispfenning. Reach set computations using real quantifier elimination. In *HSCC*, pages 63–76, 2001.

[9] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[10] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems, 2000.

[11] Eugene Asarin, Olivier Bournez, Thao Dang, Oded Maler, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. 2000.

[12] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In *HSCC*, pages 20–35, 2003.

[13] Eugene Asarin, Thao Dang, Antoine Girard, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. pages 451–476, 2007.

[14] Eugene Asarin, Thao Dang, Oded Maler, and Oded Maler. The d/dt tool for verification of hybrid systems. In *CAV*, pages 365–370, 2002.

[15] Eugene Asarin, Thao Dang, Oded Maler, Romain Testylier, and Romain Testylier. Using redundant constraints for refinement. 2011.

[16] David Avis, Komei Fukuda, and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. pages 295–313, 1992.

[17] Thomas Ball, Sriram K. Rajamani, and Sriram K. Rajamani. Bebop: A symbolic model checker for boolean programs. In *SPIN*, pages 113–130, 2000.

[18] Thomas Ball, Sriram K. Rajamani, and Sriram K. Rajamani. The slam project: debugging system software via static analysis. In *POPL*, pages 1–3, 2002.

[19] Grégory Batt, Boyan Yordanov, Ron Weiss, Calin Belta, and Calin Belta. Robustness analysis and tuning of synthetic gene networks. pages 2415–2422, 2007.

[20] Calin Belta and Luc C. G. J. M. Habets. Controlling a Class of Nonlinear Systems on Rectangles. *IEEE Transactions on Automatic Control*, 51:1749–1759, 2006.

[21] Per Bergström.

[22] Spring Berman, Ádám M. Halász, Vijay Kumar, and Vijay Kumar. Marco: A reachability algorithm for multi-affine systems with applications to biological systems. In *HSCC*, pages 76–89, 2007.

[23] Oleg Botchkarev, Stavros Tripakis, and Stavros Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *HSCC*, pages 73–88, 2000.

[24] Olivier Bouissou, Eric Goubault, Sylvie Putot, Karim Tekkal, Franck Védrine, and Franck Védrine. Hybridfluctuat: A static analyzer of numerical programs within a continuous environment. In *CAV*, pages 620–626, 2009.

[25] N.F. Britton, N. R. Franks, S. C. Pratt, and T. D. Seeley. Deciding on a new home: how do honeybees agree? In *Proceedings of the Royal Society of London Series B - Biological Sciences*, 269(1498), pages 1383–1388, 2002.

[26] S. A. Cameron and R. K. Culley. Determining the minimum translational distance between two convex polyhedra. *Proceedings of International Conference on Robotics and Automation,*, 48.

[27] Liqian Chen, Antoine Miné, Ji Wang, Patrick Cousot, and Patrick Cousot. Interval polyhedra: An abstract domain to infer interval linear relationships. In *SAS*, pages 309–325, 2009.

[28] Alongkrit Chutinan, Bruce H. Krogh, and Bruce H. Krogh. Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems*, pages 16–37, 1997.

[29] Alongkrit Chutinan, Bruce H. Krogh, and Bruce H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *HSCC*, pages 76–90, 1999.

[30] Edmund M. Clarke, Ansgar Fehnker, Zhi Han, Bruce H. Krogh, Olaf Stursberg, Michael Theobald, and Michael Theobald. Verification of hybrid systems based on counterexample-guided abstraction refinement. In *TACAS*, pages 192–207, 2003.

[31] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, Helmut Veith, and Helmut Veith. Counterexample-guided abstraction refinement. In *CAV*, pages 154–169, 2000.

[32] James C. Corbett, Matthew B. Dwyer, John Hatcliff, Shawn Laubach, Corina S. Pasareanu, Robby, Hongjun Zheng, and Hongjun Zheng. Bandera: extracting finite-state models from java source code. In *ICSE*, pages 439–448, 2000.

[33] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. of the Second Int. Symp. on Programming*, pages 106–130, 1976.

[34] Thao Dang. Approximate reachability computation for polynomial systems. In *HSCC*, pages 138–152, 2006.

[35] Thao Dang, Thomas Martin Gawlitza, and Thomas Martin Gawlitza. Template-based unbounded time verification of affine hybrid automata. In *APLAS*, pages 34–49, 2011.

[36] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology, 7th International Conference, CMSB 2009, Bologna, Italy, August 31-September 1, 2009. Proceedings*, volume 5688 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2009.

[37] Thao Dang and Oded Maler. Reachability analysis via face lifting. In *HSCC*, pages 96–109, 1998.

[38] Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *HSCC*, pages 11–20, 2010.

[39] Thao Dang, David Salinas, and David Salinas. Image computation for polynomial dynamical systems using the bernstein expansion. In *CAV*, pages 219–232, 2009.

[40] Thao Dang and Romain Testylier. Hybridization domain construction using curvature estimation. In *HSCC*, pages 123–132, 2011.

[41] Thi Xuan Thao Dang, Docteur De L'inpg, Specialite Automatique, Dang Thi, Xuan Thao, Verification Et, Synthese Des, Systemes Hybrides, Oded Maler, M. Jean, Della Dora, M. Bruce Krogh, M. Marcel Staroswiecki, M. Oded Maler, Directeur De These, Directeur De These, M. Eugene Asarin, M. Pravin Varaiya, and M. Pravin Varaiya. Vérification et synthèse des systèmes hybrides. 2000.

[42] Ding-Zhu Du and Frank Hwang, editors. *Computing in Euclidean geometry*. 1992.

[43] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical J.*, 1:445–466, 1961.

[44] I.A. Fotiou, P. Rostalski, P.A. Parrilo, and M. Morari. Parametric optimization and optimal control using algebraic geometriy methods. *International Journal of Control*, 79(11):1340–1358, 2006.

[45] Ioannis A. Fotiou, Pablo A. Parrilo, Manfred Morari, and Manfred Morari. Parametric optimization and optimal control using algebraic geometry. 2006.

[46] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[47] J. Garloff and A.P. Smith. Rigorous affine lower bound functions for multivariate polynomials and their use in global optimisation. In *Proceedings of the 1st International Conference on Applied Operational Research, Tadbir Institute for Operational Research, Systems Design and Financial Services*, volume 1 of *Lecture Notes in Management Science*, pages 199–211, 2008.

[48] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC*, pages 291–305, 2005.

[49] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In João P. Hespanha and Ashish Tiwari, editors, *HSCC*, volume 3927. Springer, 2006.

[50] Mark R. Greenstreet, Ian Mitchell, and Ian Mitchell. Reachability analysis using polygonal projections. In *HSCC*, pages 103–116, 1999.

[51] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, Pravin Varaiya, and Pravin Varaiya. What's decidable about hybrid automata? pages 94–124, 1998.

[52] Gerard J. Holzmann, Margaret H. Smith, and Margaret H. Smith. Automating software feature verification. pages 72–87, 2000.

[53] J. Garloff and A.P. Smith. An improved method for the computation of affine lower bound functions for polynomials. In C. A. Floudas and P. M. Pardalos, editor, *Frontiers in Global Optimization, Series Nonconvex Optimization and Its Applications*, pages 135–144. Kluwer Academic Publ.,Boston, Dordrecht, New York, London, 2004.

[54] J. Garloff and A.P. Smith. A comparison of methods for the computation of affine lower bound functions for polynomials. In C. Jermann, A. Neumaier, and D. Sam, editors, *Global Optimization and Constraint Satisfaction*, LNCS, pages 71–85. Springer, 2005.

[55] S.G. Johnson. The NLopt nonlinear optimization package http://ab-initio.mit.edu/nlopt.

[56] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[57] D.W. Jordan and P. Smith. *Nonlinear Ordinary Differential Equations*. Oxford Applied Mathematics and Computer Science. Oxford University Press, 1987.

[58] Rudolf Emil Kalman. Topics in mathematical system theory. 1969.

[59] Shahab Kaynama, John Maidens, Meeko Oishi, Ian M. Mitchell, Guy A. Dumont, and Guy A. Dumont. Computing the viability kernel using maximal reachable sets. In *HSCC*, pages 55–64, 2012.

[60] E.K. Kostoukova. State estimation for dynamic systems via parallelotopes: Optimization and parallel computations. In *Optimization Methods and Software*, pages 269–306, 1999.

[61] Alexander B. Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In Nancy A. Lynch and Bruce H. Krogh, editors, *HSCC*, volume 1790. Springer, 2000.

[62] Alexander B. Kurzhanski, Pravin Varaiya, and Pravin Varaiya. Ellipsoidal techniques for reachability under state constraints. pages 1369–1394, 2006.

[63] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal toolbox. Technical report, EECS Department, University of California, Berkeley, May 2006.

[64] Michal Kvasnica, Pascal Grieder, Mato Baotic, Manfred Morari, and Manfred Morari. Multi-parametric toolbox (mpt). In *HSCC*, pages 448–462, 2004.

[65] Gerardo Lafferriere, George J. Pappas, Sergio Yovine, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. pages 231–253, 2001.

[66] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In Ahmed Bouajjani and Oded Maler, editors, *21st International Conference on Computer Aided Verification, CAV 2009, June, 2009*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554, Grenoble, France, 2009. Springer.

[67] M.C. Lin and D. Manocha. Collision and proximity queries. In *Handbook of Discrete and Computational Geometry*, 2003.

[68] Oded Maler, Zohar Manna, Amir Pnueli, and Amir Pnueli. From timed to hybrid systems. In *REX Workshop*, pages 447–484, 1991.

[69] Antoine Miné. A new numerical abstract domain based on difference-bound matrices. 2007.

[70] Ian M. Mitchell. Scalable calculation of reach sets and tubes for nonlinear systems with terminal integrators: a mixed implicit explicit formulation. In *HSCC*, pages 103–112, 2011.

[71] Ian M. Mitchell, Re M. Bayen, and Claire J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50:947–957, 2005.

[72] Ian M. Mitchell, Claire Tomlin, and Claire Tomlin. Overapproximating reachable sets by hamilton-jacobi projections. pages 323–346, 2003.

[73] Bernard Mourrain, Jean Pascal Pavone, and Jean Pascal Pavone. Subdivision methods for solving polynomial equations. pages 292–306, 2009.

[74] N.S. Nedialkov, K.R. Jackson, and G.F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, (105):21–68, 1999.

[75] André Platzer and Jan-David Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *IJCAR*, pages 171–178, 2008.

[76] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput*, 2008.

[77] André Platzer, Edmund M. Clarke, and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In *CAV*, pages 176–189, 2008.

[78] André Platzer, Edmund M. Clarke, and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. pages 98–120, 2009.

[79] Stephen Prajna. Barrier certificates for nonlinear model validation. pages 117–126, 2006.

[80] Stephen Prajna, Ali Jadbabaie, and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, pages 477–492, 2004.

[81] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1429, 2007.

[82] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Trans. Embed. Comput. Syst.*, 6(1), February 2007.

[83] Enric Rodríguez-Carbonell, Ashish Tiwari, and Ashish Tiwari. Generating polynomial invariants for hybrid systems. In *HSCC*, pages 590–605, 2005.

[84] S. Boyd and S. Vandenberghe. *Convex optimization*. Cambridge Uni. Press, 2004.

[85] Sriram Sankaranarayanan, Thao Dang, Franjo Ivancic, and Franjo Ivancic. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS*, pages 188–202, 2008.

[86] Sriram Sankaranarayanan, Henny Sipma, Zohar Manna, and Zohar Manna. Constructing invariants for hybrid systems. In *HSCC*, pages 539–554, 2004.

[87] Sriram Sankaranarayanan, Henny B. Sipma, Zohar Manna, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. pages 25–41, 2005.

[88] Sriram Sankaranarayanan, Henny B. Sipma, Zohar Manna, and Zohar Manna. Constructing invariants for hybrid systems. pages 25–55, 2008.

[89] S. Sastry. *Nonlinear Systems: Analysis, Stability and Control*.

[90] Jonathan Richard Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. 2002.

[91] Axel Simon and Andy King. Exploiting sparsity in polyhedral analysis. In *SAS*, pages 336–351, 2005.

[92] A. Stuart and A.R. Humphries. *Dynamical Systems and Numerical Analysis*. Number vol. 8.

[93] Olaf Stursberg, Bruce H. Krogh, and Bruce H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *HSCC*, pages 482–497, 2003.

[94] William C. Thibault and Bruce F. Naylor. Set operations on polyhedra using binary space partitioning trees. *SIGGRAPH Comput. Graph.*, 21(4), August 1987.

[95] Ashish Tiwari, Gaurav Khanna, and Gaurav Khanna. Nonlinear systems: Approximating reach sets. In *HSCC*, pages 600–614, 2004.

[96] Pravin Variaya. Reach set computation using optimal control. In *KIT workshop on verification of hybrid systems, Grenbole, France*, 1998.