Reachability Analysis for Polynomial Dynamical Systems Using the Bernstein Expansion*

Thao Dang and Romain Testylier †
Laboratory VERIMAG, CNRS, Grenoble, France
Thao.Dang@imag.fr;Romain.Testylier@imag.fr

Abstract

This paper is concerned with the reachability computation problem for polynomial discrete-time dynamical systems. Such computations constitute a crucial component in algorithmic verification tools for hybrid systems and embedded software with polynomial dynamics, which have found applications in many engineering domains. We describe two methods for over-approximating the reachable sets of such systems; these methods are based on a combination of the Bernstein expansion of polynomial functions and a representation of reachable sets by template polyhedra. Using a prototype implementation, the performance of the methods was demonstrated on a number of examples of control systems and biological systems.

*Submitted: October 7, 2011; Revised: May 15, 2012 Accepted: November 16, 2012
†This work is supported by the ANR project VEDECY.

1 Introduction

Hybrid systems have become a common mathematical model for engineering systems exhibiting both continuous and discrete dynamics. Recently they have proved appropriate for modeling phenomena in molecular biology. Generally, a hybrid system can be thought of as a combination of a discrete and a continuous process. It consists of a collection of continuous modes; each continuous mode is associated with a vector field governing the evolution of n continuous variables within a subset of the state space $\mathcal{X} \subseteq \mathbb{R}^n$, and this set is called the staying set of the mode. The discrete dynamics is described by a set of transitions between the continuous modes, which can be triggered when the continuous variables satisfy their associated guards. Between two transitions, the system evolves according to the continuous vector field of the active mode. Various hybrid systems models have been proposed, and modelling remains an active research area [2, 5, 22].

The problem of safety verification of hybrid systems can be roughly stated as proving that a hybrid system never enters a dangerous (i.e. unsafe) state. It is thus important to note the following major types of non-determinism in the behavior of a hybrid system. First, the continuous dynamics can be subject to uncertain input modelling external disturbances or under-specified control. Secondly, non-determinism in discrete dynamics manifests when multiple discrete transitions are simultaneously activated, or when the system simultaneously satisfies the staying condition of the current mode (that is, it can continue with the current continuous dynamics) and the guard condition of a transition (that is, it can take the transition to switch to a different continuous mode). In addition, the initial conditions may not be exactly known and are often described by a set of initial states, called the initial set. Even starting at a single initial state, the system may generate a possibly infinite set of trajectories. Therefore, to prove that the system satisfies a property, one often needs to consider a set of solutions instead of single solutions.

A major component of a safety verification algorithm for a hybrid system is an efficient method to compute its reachable set, which is the set of all the states visited by all the possible trajectories. The computation of reachable sets by discrete dynamics mainly requires Boolean operations over sets in $\mathbb{R}^n$ (such as intersection of the reachable set with the guard sets of the transitions to determine the trajectories that can switch to a different mode). Nevertheless, computing the set of states reachable by continuous dynamics requires handling sets of solutions of differential or difference equations, and this problem is difficult. For general non-linear equations, their closed form solutions are not known; and even for linear systems the solutions of which can be written in a closed form, their manipulation is difficult because they can contain exponential functions. Therefore, the reachability problem for continuous systems has been a major obstacle towards applying hybrid systems formal verification to real-life problems. This has motivated much research in hybrid systems verification to focus on this particular problem. Using well-established results on linear dynamical systems, numerous methods and tools for such systems have

been developed [1]. Nevertheless, non-linear systems still remain a challenge.

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

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

where $\pi : \mathbb{R}^n \to \mathbb{R}^n$ is a multivariate polynomial. Such equations can arise in embedded control systems, such as when a physical system is controlled by a computer program which is the implementation of some continuous (or possibly hybrid) controller using appropriate discretization. In addition, our interest in discrete-time polynomial systems is motivated by their applicability in the analysis of a variety of phenomena in biochemical networks. In this area, discrete-time models are useful since experimental data are often measured by sampling continuous biochemical reaction outputs, and computer based analysis and simulation depend on discrete-time data.

The results presented in this paper can be extended to continuous-time dynamical systems described by differential equations, provided these equations can be approximated by an appropriate time discretization scheme. As in solving initial value problems for continuous-time ordinary differential equations, it is crucial to obtain a time discretization method that can guarantee conservativeness of the resulting reachable set approximations, and the enclosure methods (such as [27, 23]) can be applied.

Roughly speaking, the goal of reachability analysis is to study sets of all possible trajectories. Many existing reachability computation methods can be seen as an extension of numerical integration. That is, one has to solve the equation (1) with sets, that is $\mathbf{x}[k]$ and $\mathbf{x}[k+1]$ in this equation are subsets of $\mathbb{R}^n$ (while they are points if we only need a single solution, as in numerical integration).

This problem was previously considered in the work [34, 10], which was inspired by modeling techniques from Computer Aided Geometric Design (CADG) and tried to exploit special geometric properties of polynomials. The drawback of the Bézier simplex based method proposed in this work is that it requires expensive mesh computation, which restricts its application to systems of dimensions not higher than $3, 4$. In this paper, we pursue the direction initiated in [34] and make use of a special class of polyhedra. These polyhedra can be thought of as local meshes of fixed form. This enables a significant reduction of complexity. The manipulation of such polyhedra is handled by optimization techniques. In addition, by exploiting a technique from CADG, namely the Bernstein expansion, we only need to solve linear programming (LP) problems instead of polynomial optimization problems. In this paper, we describe our results achieved along this direction, in particular, a significant accuracy improvement compared to [10], thanks to a more precise representation of the Bernstein expansion over polyhedra.

[1] The reader is referred to the recent proceedings of the conference Hybrid Systems: Computation and Control HSCC.

The paper is organized as follows. In Section 2 we introduce basic definitions of reachable sets, template polyhedra and the Bernstein expansion of polynomials. We then formally state our reachability problem and describe an optimization-based solution. In order to transform the polynomial optimization problem to a linear programming (LP) problem, two methods for computing affine bound functions for polynomials over polyhedral sets are presented. Section 6.2 describes an algorithm summarizing the main steps of our reachability analysis approach. Some experimental results, in particular the analysis of a control system and two biological systems, are reported in Section 8.

2 Preliminaries

Let $\mathbb{R}$ denote the set of reals. Throughout the paper, vectors are often written using bold letters. Exceptionally, scalar elements of multi-indices, introduced later, are written using bold letters. Given a vector $\mathbf{x}$, x_i denotes its i^{th} component. Capital letters, such as A, B, X, Y, denote matrices or sets. If A is a matrix, A^i denotes the i^{th} row of A.

We use $\mathcal{B}$ to denote the unit box anchored at the origin, that is $\mathcal{B} = [0,1]^n$. We use π to denote a vector of n functions such that for all $i \in \{1,\ldots,n\}$, π_i is an n-variate polynomial of the form $\pi_i : \mathbb{R}^n \to \mathbb{R}$. In the remainder of the paper, we sometimes refer to π simply as "a polynomial".

To discuss the Bernstein expansion of polynomials, we use multi-indices of the form $\mathbf{i} = (\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_n)$ where each $\mathbf{i}_j$ is a non-negative integer. Given two multi-indices $\mathbf{i}$ and $\mathbf{d}$, we write $\mathbf{i} \leq \mathbf{d}$ if for all $j \in \{1,\ldots,n\}$, $\mathbf{i}_j \leq \mathbf{d}_j$. Also, we write $\frac{\mathbf{i}}{\mathbf{d}}$ for $(\mathbf{i}_1/\mathbf{d}_1, \mathbf{i}_2/\mathbf{d}_2, \ldots, \mathbf{i}_n/\mathbf{d}_n)$ and $\binom{\mathbf{i}}{\mathbf{d}}$ for $\binom{\mathbf{i}_1}{\mathbf{d}_1}\binom{\mathbf{i}_2}{\mathbf{d}_2}\ldots\binom{\mathbf{i}_n}{\mathbf{d}_n}$.

2.1 Reachable Sets

We consider a discrete-time dynamical system

$$x[k+1] = \pi(x[k]) \tag{2}$$

where the initial state $\mathbf{x}[0]$ is inside some set $X_0 \subset \mathbb{R}^n$, and X_0 is called the initial set.

Given a set $X \subset \mathbb{R}^n$, the image of X by π, denoted by $\pi(X)$, is defined as follows:

$$\pi(X) = \{(\pi_1(\mathbf{x}), \ldots, \pi_n(\mathbf{x})) \mid \mathbf{x} \in X\}.$$

The *reachable set* X_k of the system (2) at time step $k \geq 0$ is defined by the following recurrence

$$X_{k+1} = \pi(X_k)$$

where X_0 is the initial set.

2.2 Template Polyhedra

When starting from X_0, the dynamical system (2) generates a set of solutions. To characterize this set of solutions we use special convex polyhedra with fixed geometric form, called *template polyhedra* [32, 7]. In the following we give a brief introduction to template polyhedra.

A convex polyhedron is a conjunction of a finite number of linear inequalities described as $A\mathbf{x} \leq \mathbf{b}$, where A is a $m \times n$ matrix, $\mathbf{b}$ is a column vector of size m. A bounded convex polyhedron can also be represented as the convex hull of its vertices. Template polyhedra are commonly used in static analysis of programs for computing invariants. Ranges [9] and the octagon domains [25] are special template polyhedra. General template polyhedra are also used as an abstract domain to represent sets of states in [32, 7]. A *template* is a set of linear functions over $\mathbf{x} = (x_1, \ldots, x_n)$. We denote a template by an $m \times n$ matrix H, such that each row H^i corresponds to the linear function $H^i\mathbf{x}$. Given such a template H and a real-valued vector $\mathbf{c} \in \mathbb{R}^m$, a template polyhedron is defined by considering the conjunction of the linear inequalities of the form

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

We denote this polyhedron by $\langle H, \mathbf{c}\rangle$.

By varying the values of the elements of $\mathbf{c}$, one can create a family of template polyhedra corresponding to the template H. We call $\mathbf{c}$ a *polyhedral coefficient vector*. Given $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^m$, if $\forall i \in \{1, \ldots, m\} \ : \ c_i \leq c'_i$, we write $\mathbf{c} \preceq \mathbf{c}'$. Given an $m \times n$ template H and two polyhedral coefficient vectors $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^m$, if $\mathbf{c} \preceq \mathbf{c}'$ then the inclusion relation $\langle H, \mathbf{c}\rangle \subseteq \langle H, \mathbf{c}'\rangle$ holds, and we say that $\langle H, \mathbf{c}\rangle$ is not larger than $\langle H, \mathbf{c}'\rangle$.

On polyhedra, the Boolean operations (union, intersection) and common geometric operations can be done using the existing algorithms [4]. Hence, it is important to note again that a polyhedron-based reachability algorithm for continuous systems are readily extended to hybrid systems where guard and reset conditions can be described by linear constraints. 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 [32]. Manipulating general convex polyhedra is expensive especially in high dimensions. This poses a major problem in continuous and hybrid systems verification approaches using polyhedral representations.

3 Reachable Set Approximation Using Template Polyhedra

To compute the reachable set at each time step, we need to compute the image of a polyhedron P by the polynomial π. The template matrix H, 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

$$\pi(P) \subseteq \langle H, \mathbf{c} \rangle. \tag{3}$$

For safety verification purposes, exact computation of reachable sets is often not possible (due to undecidablity 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

$$\forall \mathbf{x} \in P \ : \ H\pi(\mathbf{x}) \leq \mathbf{c}$$

is sufficient for (3) to hold.

Therefore, to determine $\mathbf{c}$, one can formulate the following optimization problems

$$\forall i \in \{1, \ldots, m\}, c_i = \max(\Sigma_{k=1}^{n} H_k^i \pi_k(\mathbf{x})) \text{ subj. to } \mathbf{x} \in P. \tag{4}$$

where H^i is the i^{th} row of the matrix H and H_k^i 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 [12] and references therein). An alternative solution is to find their affine bound functions, in order to replace the polynomial optimization problem by a linear program, which can be solved more efficiently (in polynomial time) using well-developed techniques, such as Simplex and interior point techniques [31]. To this end, the Bernstein expansion can be used to compute affine bound functions of polynomials, as shown in the next section.

3.1 The Bernstein Expansion

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

$$\pi(\mathbf{x}) = \sum_{\mathbf{i} \in I_{\mathbf{d}}} \mathbf{a_i} \mathbf{x}^{\mathbf{i}}$$

where $\mathbf{a_i}$ is a vector in $\mathbb{R}^n$; $\mathbf{i}$ and $\mathbf{d}$ are two multi-indices of size n such that $\mathbf{i} \leq \mathbf{d}$; $I_{\mathbf{d}}$ is the set of *all* multi-indices $\mathbf{i} \leq \mathbf{d}$, that is $I_{\mathbf{d}} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{d}\}$. 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

$$\mathcal{B}_{\mathbf{d},\mathbf{i}}(\mathbf{x}) = \beta_{\mathbf{d}_1,\mathbf{i}_1}(x_1) \ldots \beta_{\mathbf{d}_n,\mathbf{i}_n}(x_n)$$

where for a real number y, $\beta_{\mathbf{d}_j,\mathbf{i}_j}(y) = \binom{\mathbf{d}_j}{\mathbf{i}_j} y^{\mathbf{i}_j}(1-y^{\mathbf{d}_j - \mathbf{i}_j})$.

Then, for all $\mathbf{x} \in \mathcal{B} = [0,1]^n$, the polynomial π can be written using the Bernstein expansion as follows:

$$\pi(\mathbf{x}) = \sum_{\mathbf{i} \in I_{\mathbf{d}}} \mathbf{b_i} \mathcal{B}_{\mathbf{d},\mathbf{i}}(\mathbf{x}) \tag{5}$$

where for each $\mathbf{i} \in I_\mathbf{d}$ the Bernstein coefficient $\mathbf{b_i}$ is defined as

$$\mathbf{b_i} = \sum_{\mathbf{j} \leq \mathbf{i}} \frac{\binom{\mathbf{i}}{\mathbf{j}}}{\binom{\mathbf{d}}{\mathbf{j}}} \mathbf{a_j}. \tag{6}$$

The following lemma states some important properties of the Bernstein coefficients.

Lemma 1 *1. (Convex-hull property)*

$$Conv\{(\mathbf{x}, \pi(\mathbf{x})) \ : \ \mathbf{x} \in \mathcal{B}\} \subseteq Conv\{(\mathbf{i}/\mathbf{d}, \mathbf{b_i}) \mid \mathbf{i} \in I_\mathbf{d}\}.$$

The points $\mathbf{b_i}$ are called the control points of π.

2. The above enclosure yields $\forall \mathbf{x} \in \mathcal{B} \ : \ \pi(\mathbf{x})) \in \square(\{\mathbf{b_i} \mid \mathbf{i} \in I_\mathbf{d}\})$ where $\square$ denotes the bounding box of a point set.

3. (Sharpness of some special coefficients)

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

where $I^0_\mathbf{d}$ is the set of all the vertices of $[0, \mathbf{d}_1] \times [0, \mathbf{d}_2] \ldots [0, \mathbf{d}_n]$.

With respect to our reachability problem that requires computing the image of a set by a polynomial, the Bernstein expansion is of particular interest.

For example, using the second property, the coefficients of the Bernstein expansion can be used to over-approximate the image of the unit box $\mathcal{B}$ by the polynomial π. Furthermore, as we will show in Section 4, these coefficients can be used to efficiently compute an affine approximation of the polynomial.

It is important to note that the expansion (5) is valid only if $\mathbf{x}$ is inside the unit box. Even if our initial set X_0 is inside the unit box $\mathcal{B}$, after the first step, the polyhedral approximation of the reachable set can be outside the unit box. Therefore, we need to consider the problem of computing the image of a general convex polyhedron P. To this end, we first consider the case where the set P is the unit box and then show how the solution can be extended to general convex polyhedra.

4 Computing Bound Functions Over the Unit Box Domain

We first formally define bound functions.

Definition 1 (Upper and lower bound functions) *Given $f : \mathbb{R}^n \to \mathbb{R}$, the function $v : \mathbb{R}^n \to \mathbb{R}$ is called an upper bound function of f w.r.t. a set $X \subset \mathbb{R}^n$ if $\forall \mathbf{x} \in X : \ f(\mathbf{x}) \leq v(\mathbf{x})$. A lower bound function can be defined similarly.*

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

Lemma 2 *Given two sets $X, Y \subseteq \mathbb{R}^n$ such that $Y \subseteq X$, if υ is an upper (lower) bound function of f w.r.t. X, then υ is also an upper (lower) bound function of f w.r.t. Y.*

Note that we can easily compute an upper bound function of π by computing a lower bound functions for $-\pi$ using this method and then multiply each resulting function by -1.

To compute bound functions, we use the method based on the Bernstein expansion, published in [18]. Computing convex lower bound functions for polynomials is a problem of great interest, especially in global optimization. The reader is referred to [18, 19, 14] 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 $\mathcal{B}$. The reason is that it employs the expression of the control points of the Bernstein expansion in (6) 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 P is included in the unit box.

4.1 Using a Convex Hull Lower Facet

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

$$\boldsymbol{l}(\mathbf{x}) = \mathbf{b}^0$$

where

$$\mathbf{b}^0 = \min\{\mathbf{b_i} \mid \mathbf{i} \in I_\mathbf{d}\}.$$

Better bound functions can be derived using the following two methods. The first step of this method, proposed in [19], involves computing the affine lower bound function whose corresponding hyperplane passes through this control point $\mathbf{b}^0$. 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. To summarize this algorithm, 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 $\mathbf{b_i}$ to $\mathbf{b}^0$ in the direction $\mathbf{u}^1$

$$\forall \mathbf{i} \in I_\mathbf{d} : \mathbf{i}[1] \neq \mathbf{i}^0[1],\ g^1_\mathbf{i} = \frac{\mathbf{b_i} - \mathbf{b}^0}{\mathbf{i}[1]/\mathbf{d}[1] - \mathbf{i}^0[1]/\mathbf{d}[1]}$$

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

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

- Iteration $j = 2, \ldots, n$.
  - Compute the direction $\bar{\mathbf{u}}^j = (\beta_1, \ldots, \beta_{j-1}, 0, \ldots, 0)$ such that $\bar{\mathbf{u}}^j \frac{\mathbf{i}^k - \mathbf{i}^0}{d} = 0$ 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 $\mathbf{b_i}$ to $\mathbf{b}^0$ in the direction $\mathbf{u}^j$

$$\forall \mathbf{i} \in I_{\mathbf{d}} : \frac{\mathbf{i}[1] - \mathbf{i}^0[1]}{\mathbf{d}}\mathbf{u}^j \neq 0, g^j_{\mathbf{i}} = \frac{\mathbf{b_i} - l^{j-1}(\mathbf{i}/\mathbf{d})}{(\mathbf{i}/\mathbf{d} - \mathbf{i}^0/\mathbf{d})\mathbf{u}^j}$$

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

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

4.2 Using a Linear Least Squares Approximation

The essence of the second method, proposed in [14], 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 can be 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 are denoted similarly. Let A be a matrix of size $n_b \times (n+1)$ (n is the number of state variables of the dynamical systems in question) such that its elements are defined as follows. For all $1 \leq j \leq n_b$ and $1 \leq k \leq n$,

$$A^j_k = \frac{\mathbf{i}^j_k}{\mathbf{d}_k}$$

and $A^j_{n+1} = 1$. Let ζ be the solution of the following linear least squares approximation problem

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

Then, the affine function

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

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

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

This results in the following lower bound function

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

5 Computing Affine Bound Functions Over Polyhedral Domains

As mentioned earlier, the methods to compute affine bound functions for polynomials in Section 4 can be applied only when the set P is inside the unit box $\mathcal{B}$ anchored at the origin. To extend it to polyhedral domains, we transform the polyhedra to the unit box by two methods: (1) via an (oriented) box approximation, and (2) by rewriting the polynomials using a change of variables.

5.1 Using a Box Approximation

If we over-approximate P with a box B, it is then possible to derive a formula expressing the Bernstein coefficients of π over 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 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 (6) of the Bernstein expansion. This is explained more formally in the following.

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

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

Lemma 3 *Let $\gamma = \pi \ o \ \tau$. Then, $\pi(P) \subseteq \gamma(\mathcal{B})$.*

Proof. By the definition of the composition γ, $\gamma(\mathcal{B}) = \{\pi(\tau(\mathbf{x})) \mid \mathbf{x} \in \mathcal{B}\}$. Additionally, $\tau(\mathcal{B}) = B$. Therefore, $\gamma(\mathcal{B}) = \pi(B)$. Since the polyhedron P is included in its bounding box B, we thus obtain $\pi(P) \subseteq \pi(B) = \gamma(\mathcal{B})$.
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 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) [20]. A detailed description of the method can be found in [10].

5.2 Using a Change of Variables

The polyhedron P can also be mapped to the unit box $\mathcal{B}$ by a change of variables as follows. We assume that the polyhedron P is bounded and let

$V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l\}$ be the set of its vertices. We first express the coordinates of a point x inside the polyhedron P as a linear combination of the vertices of P, that is

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

such that

$$\forall j \in \{1, \ldots, l\}\ \alpha_j \geq 0 \tag{7}$$

$$\sum_{j=1}^{l} \alpha_j = 1. \tag{8}$$

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 (8) 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 (7-8) indicate that γ is inside the unit box $B_{\tilde{\alpha}}$ 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 P using the Bernstein expansion of ξ over the $B_{\tilde{\alpha}}$.

6 Reachable Set Computation

6.1 Image computation

We now show how the above affine bound functions can be used to solve the optimization problems (4) in order to determine the coefficients of a template polyhedron over-approximating the reachable set. The functions to optimize in (4) can be seen as the compositions of polynomials π_k. Since every coefficient H_k^i is constant, each

$$\boldsymbol{s}^i(\mathbf{x}) = \Sigma_{k=1}^{n} H_k^i \pi_k(\mathbf{x})$$

is simply a polynomial and we can compute its bound functions. The template polyhedral coefficients can hence be computed by solving the following optimization problems

$$\forall i \in \{1, \ldots, m\}, c_i = max(\boldsymbol{s}^i(\mathbf{x})) \text{ subj. to } \mathbf{x} \in P; \tag{9}$$

However, such compositions often result in more monomial terms in the polynomials $\boldsymbol{s}^i$ and thus more Bernstein coefficients to consider. In the following we propose a way to avoid such compositions by bounding each element of the above sum separately, which costs less computation time but induces greater overall error. 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. the initial polyhedron P.

We consider the following optimization problem:

$$\forall i \in \{1, \ldots, m\}, c_i = \Sigma_{k=1}^{n} H_k^i \omega_k. \tag{10}$$

where the term $H_k^i \omega_k$ is defined as follows:

- If the element $H_k^i > 0$, $H_k^i \omega_k = H_k^i \max u_k(\mathbf{x})$ subj. to $\mathbf{x} \in P$;
- If the element $H_k^i \leq 0$, $H_k^i \omega_k = H_k^i \min l_k(\mathbf{x})$ subj. to $\mathbf{x} \in P$.

The following lemma is a direct result of (10).

Lemma 4 *If a polyhedral coefficient vector $\mathbf{c} \in \mathbb{R}^m$ satisfies (10). Then $\pi(P) \subseteq \langle H, \mathbf{c} \rangle$.*

Proof. It is not hard to see that the solution c_i of the optimization problems (10) is greater than or equal to the solution of (4). Hence, if $\mathbf{c}$ satisfies (10), then

$$\forall i \in \{1, \ldots, m\}\ \forall \mathbf{x} \in P :\ \Sigma_{k=1}^{n} H_k^i \pi_k(\mathbf{x}) \leq c_i.$$

This implies that $\forall \mathbf{x} \in P :\ H\pi(\mathbf{x}) \leq \mathbf{c}$, that is the image $\pi(P)$ is included in the template polyhedron $\langle H, \mathbf{c} \rangle$.

We remark that if all the bound functions in (10) are affine and P is a convex polyhedron, $\mathbf{c}$ can be computed by solving $2n$ linear programming problems.

6.2 Reachable Set Computation Algorithm

Algorithm 1 summarizes the main steps of our approach for over-approximating the reachable set of the system (2) where the initial set X_0 is a bounded polyhedron in $\mathbb{R}^n$. The template is an input of the algorithm. In the current implementation of the algorithm, either templates fixed a-priori by the user or templates forming regular sets are used.

To unify 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.

The procedure *UnitBoxMap* is used to determine the function β. This function is then composed with the polynomial π, the result of which is the polynomial γ. The affine lower and upper bound functions l and u of γ are then computed, using the Bernstein expansion of γ over the corresponding unit box. The function *PolyApp* determines the polyhedral coefficient vector $\mathbf{c}$ by solving the linear programs where the optimization domain is the unit box.

Algorithm 1 Reachable set computation

/ Inputs: convex polyhedron X_0, polynomial π, templates H */*

$k = 0$
repeat
  $\beta = UnitBoxMap(X_k)$ */* Compute the function mapping the unit box $\mathcal{B}$ to the polyhedron X_k */*
  $\gamma = \pi \ o \ \beta$
  $(u, l) = BoundFunctions(\gamma)$ */* Compute the affine bound functions */*
  $\bar{\mathbf{c}} = PolyApp(u, l, H)$ */* Compute the polyhedral coefficient vector */*
  $X_{k+1} = \langle H, \bar{\mathbf{c}} \rangle$ */* Construct the template polyhedron and return it */*
  $k++$
until $k = k_{max}$

The polyhedral coefficient vector $\bar{\mathbf{c}}$ is then used to define a template polyhedron X_{k+1}.

Based on the analysis so far, we can state the correctness of Algorithm 1.

Theorem 1 *Let $\langle H, \bar{\mathbf{c}} \rangle$ be the template polyhedron returned by Algorithm 1. Then $\pi(P) \subseteq \langle H, \bar{\mathbf{c}} \rangle$.*

We remark that, when using a box approximation, u and l are upper and lower bound functions of γ with respect to the unit box $\mathcal{B}$. It is not hard to see that $\tau^{-1}(X_k) \subseteq \mathcal{B}$ where τ^{-1} is the inverse of τ. Using the property of bound functions, u and l are also bound functions of γ with respect to $\tau^{-1}(X_k)$. Hence, if we solve the optimization problems over the domain $\tau^{-1}(X_k)$ (which is often smaller than $\mathcal{B}$), using Lemma 4, the resulting polyhedron is still an over-approximation of $\pi(X_k)$. This remark can be used to obtain more accurate results.

7 Approximation Error and Computation Cost

In this section we briefly discuss precision and complexity of the proposed methods. 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 [36, 6] states an important property of the Bernstein expansion.

Lemma 5 *Let $C_{\pi,B}$ be the piecewise linear function defined by the Bernstein control points of π with respect to the box B. Then, for all $\mathbf{x} \in B$,*

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

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

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 [18]. In our current work we are searching a subdivision of the box 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 [26].

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 used similarly 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.

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.

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 variables to map the polyhedra to the unit box. This direction is part of our current work.

8 Experimental Results

We have implemented our methods in a prototype tool. The implementation uses the library **lpsolve**[2] for linear programming. The tool can be combined with reachability analysis algorithms to verify hybrid systems with polynomial continuous dynamics. In the following, we demonstrate the methods with three examples: a control system (modeled as a hybrid system) and two biological systems (modeled as continuous systems). The time efficiency of the tool was also evaluated by considering a number of randomly generated polynomials.

[2] http://lpsolve.sourceforge.net/

Our algorithms were realized in C++ programs which were compiled with GCC 4.4.0. The experimental results were obtained by executing the programs in a mono-threaded mode, on a machine using the Linux-based operating system Ubuntu with Intel Core^{TM} 2 Duo processor (2.4GHz, 2Go RAM).

8.1 Duffing Oscillator

The first example we present is the Duffing oscillator taken from [21, 12]. This is a nonlinear oscillator of second order and its continuous-time dynamics is described by

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

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 [12], 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 [12], 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

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

The controlled system is thus modeled as a hybrid automaton [1] 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 1 which are coherent with the phase portrait in [12]. 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 80 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 3.96s. We also used this example to compare the two methods of computing bound functions and observed that they produced equally accurate results.

8.2 Michaelis-Menten Enzyme Kinetics

The second example is the well-known Michaelis-Menten enzyme kinetics, taken from [11]. The kinetic reaction of this signal transduction pathway is represented in Figure 2, where E is the concentration of an enzyme that combines with a substrate S to form an enzyme substrate complex ES. In the next step, the

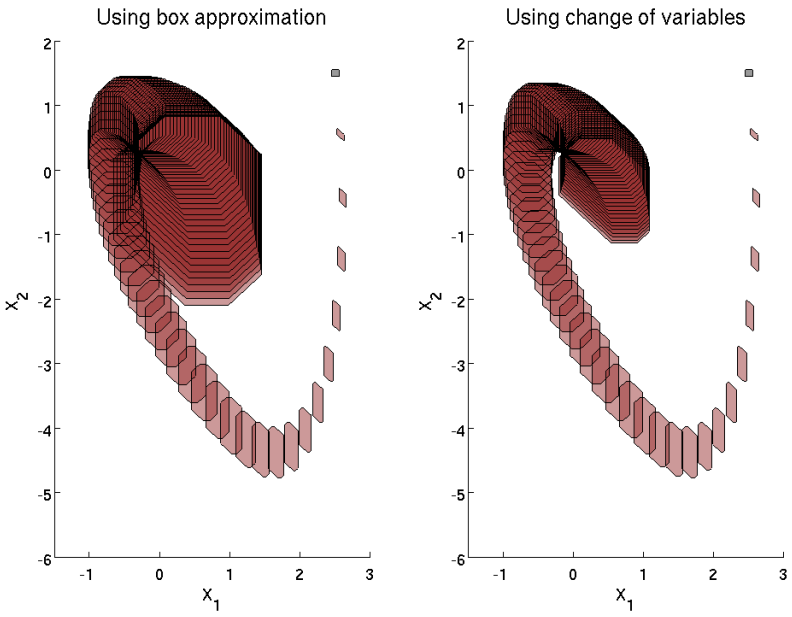

Figure 1: The Duffing oscillator: the reachable set computed using a change of variables is more accurate than the one computed using a box approximation.

complex can be dissociated into E and S or it can further proceed to form a product P. This pathway kinetics can be described by the following ODEs

$$S + E \underset{\theta_2}{\overset{\theta_1}{\rightleftarrows}} ES \overset{\theta_3}{\rightarrow} E + P$$

Figure 2: Michaelis-Menten enzyme kinetics

where x_1, x_2, x_3 and x_4 are the concentrations of S, E, ES and P

$$\begin{aligned}
\dot{x}_1 &= -\theta_1 x_1 x_2 + \theta_2 x_3 \\
\dot{x}_2 &= -\theta_1 x_1 x_2 + (\theta_2 + \theta_3) x_3 \\
\dot{x}_3 &= \theta_1 x_1 x_2 + (\theta_2 + \theta_3) x_3 \\
\dot{x}_4 &= \theta_3 x_3
\end{aligned}$$

Using a second order Runge Kutta discretization with step size 0.3, we obtain the following 4-variate polynomial system

$$\begin{aligned}
\pi_1(\mathbf{x}) &= x_1 - 0.053838 x_1 x_2 + 0.001458 x_1^2 x_2 + 0.001458 x_1 x_2^2 + \\
&\quad -3.9366 \cdot 10^{-5} x_1^2 x_2^2 + 0.005775 x_3 - 0.002025 x_1 x_3 - 0.000162 x_2 x_3 + \\
&\quad 5.9049 \cdot 10^{-5} x_1 x_2 x_3 - 6.075 \cdot 10^{-6} x_3^2 \\
\pi_2(\mathbf{x}) &= x_2 - 0.051975 x_1 x_2 + 0.001458 x_1^2 x_2 + 0.001458 x_1 x_2^2 \\
&\quad -3.9366 \cdot 10^{-5} x_1^2 x_2^2 + 0.0721875 x_3 - 0.002025 x_1 x_3 - 0.000162 x_2 x_3 + \\
&\quad 5.9049 \cdot 10^{-5} x_1 x_2 x_3 - 6.075 \cdot 10^{-6} x_3^2 \\
\pi_3(\mathbf{x}) &= 0.051975 x_1 x_2 - 0.001458 . x_1^2 x_2 - 0.001458 x_1 x_2^2 + \\
&\quad 3.9366 \cdot 10^{-5} x_1^2 x_2^2 + 0.927812 x_3 + 0.002025 x_1 x_3 + 0.000162 x_2 x_3 \\
&\quad -5.9049 \cdot 10^{-5} x_1 x_2 x_3 + 6.075 \cdot 10^{-6} x_3^2 \\
\pi_4(\mathbf{x}) &= 0.001863 x_1 x_2 + 0.0664125 x_3 + x_4 .
\end{aligned}$$

Again for this example, the method using a change of variables produced slightly more precise results but took more time. The computation time of this method for 20 steps is 153.5s while the method using a box approximation took only 11.7s. The reason for this discrepancy is that the polynomials have many monomial terms, which causes a large number of Bernstein coefficients to consider.

The reachable set computed by the method using a change of variables, for all the initial states inside a ball centered at $(12, 12, 0, 0)$ with radius 10^{-4}, is shown in Figure 3. In order to compare with the result in [11], the figures depict the temporal evolution of the first variable for the first few steps. The horizontal axis is time. In the vertical axis, the minimal and maximal values of the variable are shown. This result is coherent with the simulation result in [11].

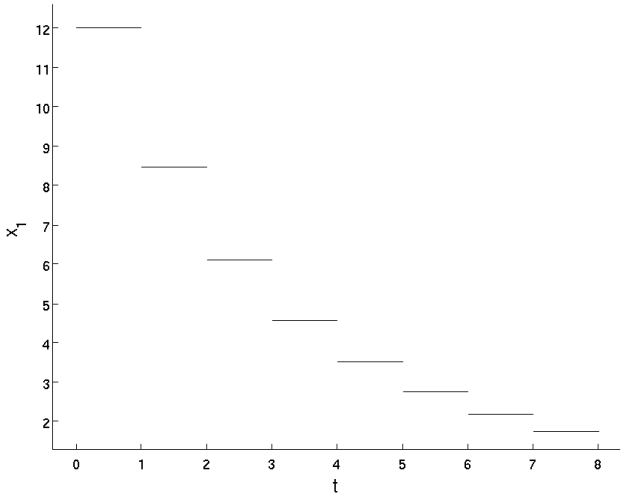

Figure 3: Michaelis-Menten enzyme kinetics. The evolution of the reachable set (projected on the first variable) after 7 steps computed by the method using a change of variables.

8.3 FitzHugh-Nagumo Neuron Model

The FitzHugh-Nagumo neuron model describing the electrical activity of a neuron [30] can be expressed by a polynomial dynamical system

$$\dot{x} = x - x^3 - y + 7/8 \quad (11)$$

$$\dot{y} = 0.08(x + 0.7 - 0.8y) \quad (12)$$

We now study an Euler time discretization scheme of the above differential equation with the step size 0.2. The initial set is an octagon included in the bounding box $[0.9, 1.1] \times [2.4, 2.6]$. Figure 4 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. The computation time of the method using a box approximation after 500 steps is 5.79s and that of the method using a change of variables is 12.73s.

Note that the use of template polyhedra provide only over-approximation of the exact reachable set. To improve the accuracy, we can increase the number of template directions. Figure 5 shows two analysis results with 8 and 20 template directions. We can observe a significant gain of precision when using 20 template directions, and the time computation is linearly increased (15.43s).

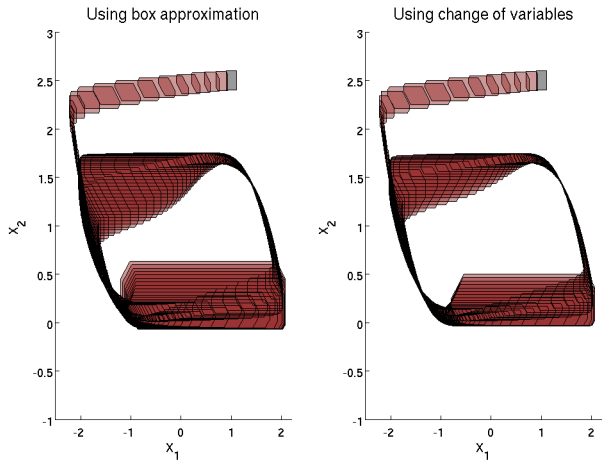

Figure 4: FitzHugh-Nagumo neuron model. The evolution of the reachable set computed using the two methods: using a box approximation and using a change of variables. The result obtained by the method using a change of variables is more accurate.

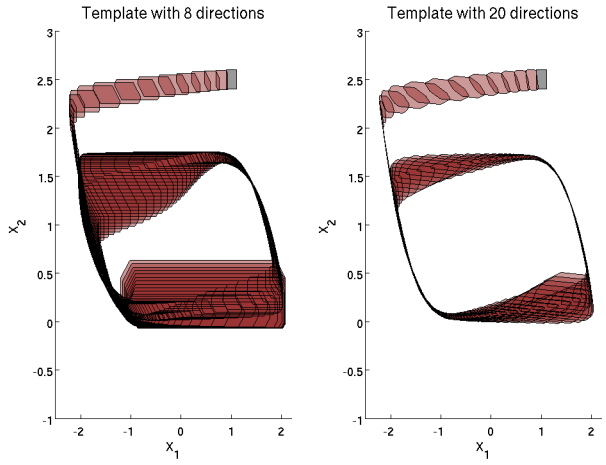

Figure 5: FitzHugh-Nagumo neuron model: The reachable set approximations with 8 template directions (left) and with 20 template directions (right), both results were computed using the box approximation method.

dim	degree d	nb monomials of degree d	nb templates	time (s) method BA	time (s) method CV
2	2	4	4	0.004	0.001
2	3	6	4	0.002	0.008
2	4	8	4	0.005	0.01
3	2	6	6	0.009	0.011
3	3	9	6	0.023	0.043
3	4	12	6	0.068	0.158
4	2	8	8	0.041	0.065
4	3	12	8	0.184	0.62
4	4	16	8	0.87	6.11167
5	2	10	10	0.265	0.501
5	3	15	10	15.44	1.48444
6	2	12	12	1.031	4.508
7	2	14	14	5.889	51.334

Table 1: Computation time for randomly generated polynomial systems in various dimensions and degrees. The second column contains the degree d of the polynomials, and third column contains their number of monomial of degree d.

8.4 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. The polynomial coefficients were randomly chosen between $[-1, 1]$. 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 by BA) and for the method using a change of variables (abbreviated by CV) are shown in Table 1.

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. In fact, 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 bound functions: using a lower convex hull facet (abbreviated by CHF) and using the least squares approximation (abbreviated by LSA). The average bound function computation time for one step of the reachability algorithm is shown in Table 2. In this experiment we used box templates and we generate random quadratic polynomial systems with 5 monomials. We were not able to test systems of di-

dim	time (s) method LSA	time (s) method CHF
2	0.00005	0.0005
3	0.00016	0.00016
4	0.00275	0.00263
5	0.0117	0.0116
6	0.0463	0.0441
7	0.1497	0.1191
8	0.8012	0.4837
9	4.755	1.591

Table 2: The computation times of computing a bound function on randomly generated polynomial systems using the LSA method (second column) and the CHF method (third column). In this experiment, the polynomials are quadratic with 5 second-order monomials).

mensions higher than 9 because polynomial composition becomes prohibitively costly. This issue can be handled by computing the Bernstein coefficients by interpolation instead of explicit polynomial composition, which is indeed a topic of our current research. The method using a least squares approximation requires solving n systems of linear equations in dimensions increasing from 1 to n, and the one using a lower convex hull facet 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, concerning linear system solving, the complexity of the method using a lower convex hull facet is roughly $O((n-1)^2n^2/4)$ while the complexity of the other is $O((n+1)^2)$. However, the LSA method requires costly matrix mulplication when the number of control points in large. This is the main reason why the LSA method is less efficient than the CHF method in high dimensions.

9 Related Work

Our reachability analysis approach is similar to a number of existing ones for continuous and hybrid systems in the use of linear approximation (such as, [38, 3, 13, 24, 37]. Its novelty resides in the efficient way of computing linear approximations. A common method to approximate a non-linear function by a piecewise linear one, as in the hybridization approach [3] for hybrid systems, requires non-linear optimization. Our approach exploits the Bernstein expansion of polynomials to replace expensive polynomial programming by linear programming.

A similar idea, which involves using the coefficients of the Bézier simplex representation, was used in [34] to compute the image of a convex polyhedron. If using the methods proposed in this paper with a sufficient number of templates to assure the same precision as the convex hull in our previous Bézier

method [34], then the convergence of both methods is quadratic. However the Bézier method requires expensive triangulation operations, and geometric complexity of resulting sets may grow step after step. Combining template polyhedra and bound functions allows a good accuracy-cost compromise.

Besides constrained global optimization, other important applications of the Bernstein expansion include various control problems [17] (in particular, robust control). The approximation of the range of a multivariate polynomial over a box and a polyhedron is also used in program analysis and optimization (for example [16, 8]). In the hybrid systems verification, polynomial optimization is used to compute barrier certificates [29]. Algebraic properties of polynomials are used to compute polynomial invariants [35] and to study the computability of image computation in [28]. Finally, systems with uncertainties are also studied from the point of view of control, optimization and simulation [15].

10 Conclusion

The reachability computation methods we proposed in this paper combine the ideas from optimization and the Bernstein expansion. These results are readily applicable to hybrid systems with polynomial continuous dynamics.

The performance of the methods was demonstrated using a number of randomly generated examples. These encouraging results also show an important advantage of the methods: thanks to the use of template polyhedra, the complexity and precision of the method are more controllable than those using polyhedra as symbolic set representations.

There are a number interesting directions to explore. Different tools from geometric modeling could be exploited to improve the efficiency of the method. For example, polynomial composition can be done for sparse polynomials more efficiently using the blossoming technique [33]. In addition to more experimentation on other hybrid systems case studies, we intend to explore a new application domain, which is verification of embedded control software. In fact, multivariate polynomials arise in many situations when analyzing programs that are automatically generated from practical embedded controllers.

References

[1] 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.

[2] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierachical modeling and analysis of embedded systems. *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software*, 2002.

[3] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica.*, 43(7):451–476, 2007.

[4] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[5] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. on Automatic Control*, 43(1):31–45, 1998.

[6] C. de Boor. B-form basics. *Geometric Modeling*, (2):27–49, 1987.

[7] L. Chen, A. Miné, J. Wang, and P. Cousot. Interval polyhedra: An abstract domain to infer interval linear relationships. In *SAS*, volume 5673 of *Lecture Notes in Computer Science*, pages 309–325. Springer, 2009.

[8] F. Clauss and I.Yu. Chupaeva. Application of symbolic approach to the bernstein expansion for program analysis and optimization. *Program. Comput. Softw.*, 30(3):164–172, 2004.

[9] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming, Dunod, Paris*, pages 106–130, 1976.

[10] T. Dang and D. Salinas. Image computation for polynomial dynamical systems using the Bernstein expansion. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, CAV 2009*, volume 5643 of *Lecture Notes in Computer Science*, pages 219–232. Springer, 2009.

[11] F. He, L. F. Yeung, and M. Brown. Discrete-time model representation for biochemical pathway systems. *IAENG International Journal of Computer Science*, 34(1), 2007.

[12] 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.

[13] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10():263–279, 2008.

[14] 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.

[15] A. Rauh and E. Auer, editors. *Modeling, Design, and Simulation of Systems with Uncertainties*, Cambridge Monographs on Applied and Computational Mathematics. Springer-Verlag, 2011.

[16] I. Tchoupaeva. A symbolic approach to bernstein expansion for program analysis and optimization. In *13th International Conference on Compiler Construction, CC 2004*, pages 120–133. Springer, 2004.

[17] J. Garloff. Application of Bernstein Expansion to the Solution of Control Problems. In J. Vehi and M. A. Sainz, editor, *Workshop on Applications of Interval Analysis to Systems and Control*, pages 421–430, 2004.

[18] 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.

[19] 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.

[20] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[21] D.W. Jordan and P. Smith. *Nonlinear Ordinary Differential Equations*. Oxford Applied Mathematics and Computer Science. Oxford University Press, 1987.

[22] F. Kratz, O. Sokolsky, G.J. Pappas, and I. Lee. R-charon : a modeling language for reconfigurable hybrid systems. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, LNCS 3927, pages 392–406. Springer, 2006.

[23] R. Lohner. On the ubiquity of the wrapping effect in the computation of error bounds. In A. Facius, U. Kulisch, and R. Lohner, editors, *Perspectives on Enclosure Methods*, pages 201–217. Springer-Verlag, Vienna, Austria, 2001.

[24] M. Althoff and O. Stursberg and M. Buss. Reachability Analysis of Nonlinear Systems Using Trajectory Piecewise Linearized Models. In *Proc. of the 47th IEEE Conference on Decision and Control*, pages 4042–4048, 2009.

[25] A. Miné. A new numerical abstract domain based on difference-bound matrices. In *PADO II*, LNCS 2053, pages 155–172. Springer, 2001.

[26] A.B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations. Technical report, INRIA, August 2005. Technical report, Research report, 5658.

[27] N. S. Nedialkov and K. R. Jackson. A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations. In A. Facius, U. Kulisch, and R. Lohner, editors, *Perspectives on Enclosure Methods*, pages 219–264. Springer-Verlag, Vienna, Austria, 2001.

[28] A. Platzer and E.M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design*, 35(1):98–120, 2009.

[29] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2004.

[30] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical J.*, 1:445–466, 1961.

[31] S. Boyd and S. Vandenberghe. *Convex optimization*. Cambridge Uni. Press, 2004.

[32] S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Verification, Model-Checking and Abstract-Interpretation (VMCAI 2005)*, LNCS 3385. Springer, 2005.

[33] H.-P. Seidel. Polar forms and triangular B-spline surfaces. In *Blossoming: The New Polar-Form Approach to Spline Curves and Surfaces, SIGGRAPH '91 Course Notes 26*, Lecture Notes in Computer Science, pages 8.1–8.52. ACM SIGGRAPH, 1991.

[34] T. Dang. Approximate Reachability Computation for Polynomial Systems. In João P. Hespanha and Ashish Tiwari, editor, *Hybrid Systems: Computation and Control HSCC*, volume 3927 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2006.

[35] A. Tiwari and G. Khanna. Nonlinear systems: Approximating reach sets. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 600–614. Springer, 2004.

[36] W. Dahmen. Subdivision algorithms converge quadratically. *Journal of Computational and Applied Mathematics*, (2):145–158, 1986.

[37] Y. Chao and F. Ouchet and L. Fesquet and K. Morin-Allory. Formal Verification of C-element Circuits;. In *17th IEEE International Symposium on synchronous Circuits and Systems (ASYNC)*, 2011.

[38] Z. Han and B.H. Krogh. Reachability Analysis of Nonlinear Systems Using Trajectory Piecewise Linearized Models. In *Proc. of the American Control Conference*, pages 1505–1510, 2006.