Reachability Analysis of Polynomial Systems Using Linear Programming Relaxations*

Mohamed Amin Ben Sassi[2], Romain Testylier[1], Thao Dang[1], and Antoine Girard[2]

[1] Verimag, Université de Grenoble
[2] Laboratoire Jean Kuntzmann, Université de Grenoble
firstname.lastname@imag.fr

Abstract. In this paper we propose a new method for reachability analysis of the class of discrete-time polynomial dynamical systems. Our work is based on the approach combining the use of template polyhedra and optimization [25,7]. These problems are non-convex and are therefore generally difficult to solve exactly. Using the Bernstein form of polynomials, we define a set of equivalent problems which can be relaxed to linear programs. Unlike using affine lower-bound functions in [7], in this work we use piecewise affine lower-bound functions, which allows us to obtain more accurate approximations. In addition, we show that these bounds can be improved by increasing artificially the degree of the polynomials. This new method allows us to compute more accurately guaranteed over-approximations of the reachable sets of discrete-time polynomial dynamical systems. We also show different ways to choose suitable polyhedral templates. Finally, we show the merits of our approach on several examples.

1 Introduction

Reachability analysis has been a major research issue in the field of hybrid systems for more than a decade. Spectacular progress has been made over the past few years for a class of hybrid systems where the continuous dynamics can be described by affine differential equations [10,12,18,9]. However, dealing efficiently with systems with nonlinear dynamics remains a challenging problem that needs to be addressed. Besides, reachability analysis of non linear dynamical systems is also motivated by its numerous potential applications, in particular in systems biology [11,5,29].

In this paper, we present a new method for reachability analysis of a class of discrete-time nonlinear systems defined by polynomial maps. We follow the approach proposed in [25,7] which, by using template polyhedra, reduces the problem of reachability analysis to a set of optimization problems involving polynomials over bounded polyhedra. These are generally non-convex optimization

* This work was supported by the Agence Nationale de la Recherche (VEDECY project - ANR 2009 SEGI 015 01).

S. Chakraborty and M. Mukund (Eds.): ATVA 2012, LNCS 7561, pp. 137–151, 2012.

problems and hence it is hard to solve them exactly. However, computing lower bounds of the solutions of these optimization problems is actually sufficient to obtain guaranteed over-approximations of the reachable sets that are usually needed for safety verification. Unlike using affine lower-bound functions in [5], in this work we use piecewise affine lower-bound functions, which allows us to obtain more accurate approximations. To this end, we essentially use the Bernstein expansions of polynomials and their properties to build linear programming relaxations of the original optimization problems. This can be roughly described as follows. First, by writing polynomials in the Bernstein basis we define a set of equivalent problems. Then, using properties of Bernstein polynomials, we show that good lower bounds of the optimal value of these problems can be computed efficiently using linear programming. This provides us with an elegant approach to reachability analysis of polynomial systems.

The rest of the paper is organized as follows. In Section 2, we show a technique for computing a lower bound of a non convex optimization problem where the cost function is a multivariate polynomial and the constraints are given by a bounded polyhedron included in the unit box. We then present a result which allows us to improve the accuracy of our lower bounds and a comparison with other relaxation methods. In Section 3, we show that reachability analysis of polynomial dynamical systems can be handled by optimizing multivariate polynomials on bounded polyhedra, and this will be used to compute guaranteed over-approximations of the reachable set. The choice of the templates, the complexity of the whole approach and a comparison with other related approach are also discussed. Finally, in Section 4, we show some experimental results including the application of our approach to reachability analysis of biological systems.

2 Optimization of Polynomials Using Linear Programming

In the following, we consider the problem of computing a guaranteed lower bound of the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \ell \cdot g(y) \\ \text{over} & y \in [0,1]^n, \\ \text{subject to} & Ay \leq b. \end{array} \tag{1}$$

where $\ell \in \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^n$ is a polynomial map, $A \in \mathbb{R}^{n \times m}$ and $b \in \mathbb{R}^m$. Let $y_1, \ldots, y_n$ denote the components of $y \in [0,1]^n$ and $\delta_1, \ldots, \delta_n$ denote the degrees of g in $y_1, \ldots, y_n$. Let $\Delta = (\delta_1, \ldots, \delta_n)$; for $I = (i_1, \ldots, i_n) \in \mathbb{N}^n$, we write $I \leq \Delta$ if $i_j \leq \delta_j$ for all $j \in \{1, \ldots, n\}$. Also, let $|I| = i_1 + \cdots + i_n$ and for $y \in \mathbb{R}^n$, $y^I = y_1^{i_1} \ldots y_n^{i_n}$. Then, the polynomial map g can be written in the form:

$$g(y) = \sum_{I \leq \Delta} g_I y^I \text{ where } g_I \in \mathbb{R}^n, \ \forall I \leq \Delta.$$

Let $\delta_{max} = \max\{|I|, \text{ such that } I \leq \Delta \text{ and } g_I \neq 0\}$.

2.1 Using the Bernstein Form

The main theoretical ingredient of our approach is the Bernstein expansion of polynomials [2,3]. The polynomial map g in its Bernstein form is given by:

$$g(y) = \sum_{I \leq \Delta} h_I B_{\Delta,I}(y) \text{ where } h_I \in \mathbb{R}^n, \ \forall I \leq \Delta \tag{2}$$

and the Bernstein polynomials are defined for $I \leq \Delta$ as follows:

$$B_{\Delta,I}(y) = \beta_{\delta_1,i_1}(y_1) \dots \beta_{\delta_n,i_n}(y_n)$$

with for $j = 1, \dots n$, $i_j = 0, \dots, \delta_j$: $\beta_{\delta_j,i_j}(y_j) = \binom{\delta_j}{i_j} y_j^{i_j}(1-y_j)^{\delta_j - i_j}$.

The coefficients h_I of g in the Bernstein basis can be evaluated explicitly from the coefficients g_I of g in the canonical basis using the following explicit formula: for all $I \leq \Delta$,

$$h_I = \sum_{J \leq I} \frac{\binom{i_1}{j_1} \dots \binom{i_n}{j_n}}{\binom{\delta_1}{j_1} \dots \binom{\delta_n}{j_n}} g_J. \tag{3}$$

We also propose an alternative approach for computing the coefficients h_I using the interpolation at the points $\frac{J}{\Delta'} = (\frac{j_1}{\delta_1}, \dots, \frac{j_n}{\delta_n})$ for $J \leq \Delta$:

$$\sum_{I \leq \Delta} h_I B_{\Delta,I}(\tfrac{J}{\Delta}) = g(\tfrac{J}{\Delta})$$

Let us denote B_Δ the matrix whose lines are indexed by $J \leq \Delta$ and columns are indexed by $I \leq \Delta$ with coefficients $B_{\Delta,I}(\frac{J}{\Delta})$. Let h be the matrix whose lines indexed by $I \leq \Delta$ are $h_I^\top$ and g the matrix whose lines indexed by $J \leq \Delta$ are $g(\frac{J}{\Delta})^\top$.Then, the previous equation under matricial form can be written as $\mathsf{B}_\Delta \mathsf{h} = \mathsf{g}$. From standard polynomial interpolation theory, the matrix B_Δ is invertible and the Bernstein coefficients are given by

$$\mathsf{h} = \mathsf{B}_\Delta^{-1} \mathsf{g}. \tag{4}$$

For determining a linear programming relaxation of (1), the most useful properties of the Bernstein polynomials are the following:

Proposition 1. *The Bernstein polynomials satisfy the following properties:*

1. *For all* $y \in \mathbb{R}^n$, $\sum_{I \leq \Delta} B_{\Delta,I}(y) = 1$ *and* $\sum_{I \leq \Delta} \frac{I}{\Delta} B_{\Delta,I}(y) = y$.
2. *For all* $y \in [0,1]^n$, $0 \leq B_{\Delta,I}(y) \leq B_{\Delta,I}(\frac{I}{\Delta})$
   where $B_{\Delta,I}(\frac{I}{\Delta}) = \prod_{j=1}^{j=n} \binom{\delta_j}{i_j} \frac{i_j^{i_j}(\delta_j - i_j)^{\delta_j - i_j}}{\delta_j^{\delta_j}}$.

2.2 Linear Programming Relaxation

We can use the previous proposition to derive a linear programming relaxation of the problem (1):

Proposition 2. *Let $\underline{p}^*$ be the optimal value of the linear program:*

$$\begin{array}{lll} \textit{minimize} & \sum_{I \leq \Delta} (\ell \cdot h_I) z_I & \\ \textit{over} & z_I \in \mathbb{R}, & I \leq \Delta, \\ \textit{subject to} & 0 \leq z_I \leq B_{\Delta,I}(\frac{I}{\Delta}), & I \leq \Delta, \\ & \sum_{I \leq \Delta} z_I = 1, & \\ & \sum_{I \leq \Delta} (A\frac{I}{\Delta}) z_I \leq b. & \end{array} \tag{5}$$

Then, $\underline{p}^* \leq p^*$ where p^* is the optimal value of problem (1).

Proof. Using the Bernstein expansion (2) of g and the first property in Proposition 1, we can rewrite the problem (1) under the form

$$\begin{array}{ll} \text{minimize} & \sum_{I \leq \Delta} (\ell \cdot h_I) B_{\Delta,I}(y) \\ \text{over} & y \in [0,1]^n, \\ \text{subject to} & \sum_{I \leq \Delta} (A\frac{I}{\Delta}) B_{\Delta,I}(y) \leq b. \end{array}$$

Then, let y^* be the optimum of the problem (1), and let $z_I = B_{\Delta,I}(y^*)$ for all $I \leq \Delta$. It is clear from Proposition 1 that these satisfy the constraints of (5), therefore the optimal value of (5) is necessary smaller than that of (1). □

Now we will show how to improve the precision at the expense of an increase in computational cost. The linear program (5) is a relaxation of the optimization problem (1). Then, the lower bound $\underline{p}^*$ on the optimal value p^* is generally not tight. We first remark that g can be seen as a higher order polynomial, possibly by adding monomials of higher degree with zero coefficients.

In the following, g is considered as a polynomial with degrees $K = (k_1, \ldots, k_n)$ where $\Delta \leq K$, then g can be written in the Bernstein form:

$$g(y) = \sum_{I \leq K} h_I^K B_{K,I}(y),$$

We will use the following result [19]:

Proposition 3. *For $I \leq K$,*

$$\|h_I^K - g(\tfrac{I}{K})\| = O(\tfrac{1}{k_1} + \cdots + \tfrac{1}{k_n}).$$

Now let $\underline{p}_K^*$ be the optimal value of the linear program (5) with degrees K instead of Δ. We want to determine the limit of $\underline{p}_K^*$ when all the k_i go to infinity.

Let $\underline{c}_K^*(y)$ be the optimal value of the linear program:

$$\begin{array}{ll} \text{minimize} & \sum_{I \leq K} (\ell \cdot h_I^K) z_I \\ \text{over} & z_I \in \mathbb{R}, \qquad I \leq K, \\ \text{subject to} & 0 \leq z_I \leq B_{K,I}(\frac{I}{K}), \; I \leq K, \\ & \sum_{I \leq K} z_I = 1, \\ & y = \sum_{I \leq K} \frac{I}{K} z_I \end{array} \tag{6}$$

Then,

$$\underline{p}^*_K = \min_{\substack{y\in[0,1]^n \\ Ay\leq b}} \underline{c}^*_K(y) \leq p^*.$$

Now let $C(\ell \cdot g)$ be the convex hull of the function $\ell \cdot g$ taken over the set $[0,1]^n$ (see e.g. [13]). Formally, $C(\ell \cdot g)$ is a convex function over $[0,1]^n$ such that for all $y \in [0,1]^n$, $C(\ell \cdot g)(y) \leq \ell \cdot g(y)$ and for all functions h convex over $[0,1]^n$ and such that for all $y \in [0,1]^n$, $h(y) \leq g(y)$, then for all $y \in [0,1]^n$, $h(y) \leq C(\ell \cdot g)(y)$. In other words, $C(\ell \cdot g)(y)$ is the largest function convex over $[0,1]^n$ bounding $\ell \cdot g$ from below. In particular, if $\ell \cdot g$ is convex then $C(\ell \cdot g) = \ell \cdot g$. Now, let p^*_C be the optimal value of the following optimization problem

$$\begin{array}{ll} \text{minimize} & C(\ell \cdot g)(y) \\ \text{over} & y \in [0,1]^n, \\ \text{subject to} & Ay \leq b. \end{array}$$

It is clear that $p^*_C \leq p^*$ with equality if $\ell \cdot g$ is convex. We can now state the main result of the section:

Proposition 4. *For all $K \geq \Delta$, $\underline{p}^*_K \leq p^*_C$ and*

$$|\underline{p}^*_K - p^*_C| = O(\tfrac{1}{k_1} + \cdots + \tfrac{1}{k_n}).$$

Proof. It is not hard to show that $\underline{c}^*_K$ is convex over $[0,1]^n$ and under-estimates $\ell \cdot g$. Then, by definition of the convex hull of a function we have: $\underline{c}_K{}^*(y) \leq C(\ell \cdot g)(y)$ for all $y \in [0,1]^n$. This gives directly that $\underline{p}^*_K \leq p^*_C$.

Now let $y \in [0,1]^n$, and let $z_I{}^*$, $I \leq K$ be an optimal solution of (6), then we have:

$$\underline{c}^*_K(y) = \sum_{I\leq K} (\ell \cdot h^K_I) z_I{}^* \text{ and } y = \sum_{I\leq K} z_I{}^* \frac{I}{K}.$$

Using properties of the convex hull of a function we have:

$$\begin{aligned} C(\ell \cdot g)(y) &= C(\ell \cdot g)\left(\textstyle\sum_{I\leq K} \frac{I}{K} z_I{}^*\right) \\ &\leq \sum_{I\leq K} C(\ell \cdot g)\left(\tfrac{I}{K}\right) z_I{}^* \leq \sum_{I\leq K} \ell \cdot g\left(\tfrac{I}{K}\right) z_I{}^*. \end{aligned}$$

It follows that:

$$0 \leq C(\ell \cdot g)(y) - \underline{c}^*_K(y) \leq \sum_{I\leq K} [\ell \cdot g(\tfrac{I}{K}) - \ell \cdot h^K_I] z_I{}^*,$$

Finally, using Proposition 3 we have for $y \in [0,1]^n$:

$$0 \leq C(\ell \cdot g)(y) - \underline{c}^*_K(y) \leq O(\tfrac{1}{k_1} + \cdots + \tfrac{1}{k_l})$$

which yields $|\underline{p}^*_K - p^*_C| = O(\tfrac{1}{k_1} + \cdots + \tfrac{1}{k_n})$. □

In other words, when we artificially increase the degrees of the polynomial g, we improve the computed lower bound. However, we cannot do better than p^*_C which is the optimal value of a convexified version of problem (1). We remark that if the function $\ell \cdot g$ is convex then we can approach the optimal value p^* arbitrarily close.

2.3 Related Work in Linear Relaxations

Our method of using the Bernstein form to replace expensive polynomial programming by linear programming is close to the one proposed in [7]. As mentioned earlier, the main improvement over this work is that our method uses a piecewise affine lower-bound function (the function $\underline{c}_K^*(y)$ defined in (6)) which is more accurate than a single affine lower bound function used in [7]. Moreover, our approach allows us to work directly with arbitrary polyhedral domains, while the Bernstein form in [7] works only for the unit box and this requires rectangular approximations that introduce additional error.

Let us now briefly discuss the complexity of our method and compare it with existing relaxation methods. In fact, the linear program (5) has polynomial complexity in its number of decision variables which is $N_\Delta = (\delta_1+1)\times\cdots\times(\delta_n+1)$. However, it should be noted that $N_\Delta = O({\delta_{max}}^n)$ which is exponential in the dimension n of the state space. Another known relaxation method which uses also linear programming is the reformulation-linearization-technique method (RLT) introduced by Sherali in [26,27]. The linear program given by this method is in fact a linearized version of the problem (1) by adding new variables where the constraints are given by exploiting all possible products of the original ones with respect to a fixed degree δ. In general, its number of decision variables is the same comparing to our method but the number constraints is much greater. We should mention that thanks to Bernstein properties our method can be more precise.

Also, one could use a method based on semi-definite programming and the theory of moments [17]. It should be noticed that the size of the semi-definite programs that need to be solved would be similar to the size of the linear program we solve. The quality of the lower bound obtained by semi-definite programming would certainly be better and it has been showed that an asymptotic convergence to the optimal value can be obtained; however, the approach would require more computational resources.

3 Reachability Analysis of Polynomial Dynamical Systems

Let us consider a disrete-time dynamical system of the following form:

$$x_{k+1} = f(x_k),\ k \in \mathbb{N},\ x_k \in \mathbb{R}^n,\ x_0 \in X_0 \tag{7}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a polynomial map with degrees $\Delta = (\delta_1, \ldots, \delta_n)$ and X_0 is a bounded polyhedron in $\mathbb{R}^n$. In this paper, we are concerned with bounded-time reachability analysis of system (7). This consists in computing the sequence $X_k \subseteq \mathbb{R}^n$ of reachable sets at time k of the system up to some time $K \in \mathbb{N}$. It is straightforward to verify that this sequence satisfies the following induction relation $X_{k+1} = f(X_k)$. It should be noticed that even though the first element X_0 is a polyhedron, in general the other elements of the sequence are not. Actually, they are generally not even convex.

In this work, we over-approximate these sets using bounded polyhedra $\overline{X}_k$. Such a sequence can clearly be computed inductively by setting $\overline{X}_0 = X_0$ and by ensuring that for all $k = 0, \ldots, K-1$, $f(\overline{X}_k) \subseteq \overline{X}_{k+1}$. Hence, we first focus on the computation of a polyhedral over-approximation $\overline{X}_{k+1}$ of the image of a bounded polyhedron $\overline{X}_k$ by a polynomial map f.

Now, the problem we address is stated as follows: given a polyhedron $\overline{X}_k$, we want to compute $\overline{X}_{k+1}$ such that $f(\overline{X}_k) \subseteq \overline{X}_{k+1}$. In the following we propose a solution to this problem based on the use of template polyhedra and optimization.

3.1 Template Polyhedra

To represent $\overline{X}_{k+1}$, we use a template polyhedron. A template is a set of linear functions over $x \in \mathbb{R}^n$. We denote a template by a matrix $A \in \mathbb{R}^{m \times n}$, given such a template A and a coefficient vector $b \in \mathbb{R}^m$, we define the template polyhedron

$$Poly(A, b) = \{x \in \mathbb{R}^n |\ Ax \leq b\}$$

where the inequality is to be understood component-wise. By varying the value of b, we obtain a family of template polyhedra corresponding to the template A. Essentially, the template A defines the directions of the faces and the vector b define their positions. The advantage of using template polyhedra over general convex polyhedra is that the Boolean operations (union, intersection) and common geometric operations can be performed more efficiently. Indeed, such operations are crucial for most verification procedures based on reachable set computations.

In the following, we assume that $\overline{X}_{k+1}$ is a bounded polyhedron with a given template $A_{k+1} \in \mathbb{R}^{m \times n}$. We add the subscript $k+1$ to emphasize that the template may not be the same for all the polyhedra $\overline{X}_{k+1}$, $k = 0, \ldots, K-1$. Then,

$$\overline{X}_{k+1} = Poly(A_{k+1}, b_{k+1})$$

where the vector $b_{k+1} \in \mathbb{R}^m$ needs to be determined at each iteration. The choice of the templates A_{k+1} will be discussed later in Section 3.3.

From the previous discussion, it appears that the computation of the set $\overline{X}_{k+1}$ reduces to determining values for the vectors b_{k+1}. Let $b_{k+1,i}$ denote the i-th element of these vectors; and $A_{k+1,i}$ denote the i-th line of matrices A_{k+1} .

Lemma 1. *If for all* $i = 1, \ldots, m$

$$-b_{k+1,i} \leq \min_{x \in \overline{X}_k} -A_{k+1,i} f(x) \tag{8}$$

then $f(\overline{X}_k) \subseteq \overline{X}_{k+1}$ *where* $\overline{X}_{k+1} = Poly(A_{k+1}, b_{k+1})$

Proof. Let $y \in f(\overline{X}_k)$. For $i = 1, \ldots, m$, it is clear that we have

$$A_{k+1,i} y \leq \max_{x \in \overline{X}_k} A_{k+1,i} f(x).$$

Then, by remarking that $\max_{x \in \overline{X}_k} A_{k+1,i} f(x) = -\min_{x \in \overline{X}_k} -A_{k+1,i} f(x)$ and by (8) we obtain $A_{k+1,i} y \leq b_{k+1,i}$. □

Let us remark that the computation of the minimal values in equations (8) involves optimizing a generally non-convex multi-variable polynomial function on a bounded polyhedron. This is a difficult problem in general; however Lemma 1 shows that the computation of a lower bound for the minimal values is sufficient to obtain an over-approximation of $f(\overline{X}_k)$. Thus, we can see that the computation of $\overline{X}_{k+1}$ can be done by computing guaranteed lower bounds on the optimal values of minimization problems involving multi-variable polynomial functions on a bounded polyhedron. A solution to this problem, based on linear programming, is proposed in the previous section when $x \in [0,1]^n$. We will see how this result can be adapted to our reachability problem.

3.2 Reachability Algorithm

According to the previous discussion, in each step $k \in \mathbb{N}$ we have to compute a lower bound of the value

$$p_{k+1,i}{}^* = \min_{x \in \overline{X}_k} -A_{k+1,i} \cdot f(x) \text{ for all } i = 1, \ldots, m.$$

For doing so, we will propose an algorithm with essentially three steps: in the first one we compute a bounding parallelotope that will be necessary for the second step. The second one will consist in a change of variable allowing us to recast our optimization problem on the form of the one given by (1). In the last step, lower bound will be obtained using the linear program given by Proposition 2 and then an over approximation of the reachable set is computed.

Step 1: Bounding Parallelotope Computation
As $\overline{X}_k$ is a bounded polyhedron of $\mathbb{R}^n$ we can write it in the form $\overline{X}_k \cap Q_k$ where Q_k is its bounding parallelotope given by $Q_k = Poly(\tilde{C}_k, \tilde{d}_k)$ with

$$\tilde{C}_k = \begin{bmatrix} C_k \\ -C_k \end{bmatrix}, \; \tilde{d}_k = \begin{bmatrix} \overline{d}_k \\ -\underline{d}_k \end{bmatrix}.$$

$C_k \in \mathbb{R}^{n \times n}$ is an invertible matrix, $\underline{d}_k \in \mathbb{R}^n$ and $\overline{d}_k \in \mathbb{R}^n$. We assume that the matrix direction C_k is given as an input (a method describing its computation will be given later) and we compute the component $\overline{d}_{k,i}$ and $\underline{d}_{k,i}, i = 1, \ldots n$ of the vector position $\tilde{d}_k$ as optimal solutions of the following linear programs :

$$\overline{d}_{k,i} = \max_{x \in \overline{X}_k} C_{k,i} \cdot x \text{ and } \underline{d}_{k,i} = \max_{x \in \overline{X}_k} -C_{k,i} \cdot x \quad \forall i = 1, \ldots n$$

Step 2: Change of Variable
Now, let's proceed to the following change of variable $x = q_k(y)$ where the affine map $q_k : \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$q_k(y) = {C_k}^{-1} D_k y + {C_k}^{-1} \underline{d_k}$$

where D_k is the diagonal matrix with entries $\overline{d}_{k,i} - \underline{d}_{k,i}$. The change of variable q_k essentially maps the unit cube $[0,1]^n$ to Q_k. This change of variable is the main

reason for defining $\overline{X}_k$ as the intersection of a polyhedron and a parallelotope. We then define the polynomial map g_k as $g_k(y) = f(q_k(y))$. Finally, let $A'_k \in \mathbb{R}^{n\times m}$ and $b'_k \in \mathbb{R}^m$ be given by

$$A'_k = A_k {C_k}^{-1} D_k,\ b'_k = b_k - A{C_k}^{-1}\underline{d}.$$

Remark 1. It is clear that g_k is a polynomial map. As for the degrees $\Delta' = (\delta'_1, \ldots, \delta'_n)$ of g_k in the variables $y_1, \ldots, y_n$, we shall discuss two different cases depending on the nature of parallelotope Q_k. If Q_k is an axis-aligned box (i.e. if C_k is a diagonal matrix), then the degrees of g_k are the same as f: $\Delta' = \Delta$. This is not the case in general, when Q_k is not axis-aligned; in that case, the change of variable generally increases the degrees of the polynomial and g_k can be regarded as a polynomial of degrees $\Delta' = (\delta_{max}, \ldots, \delta_{max})$.

Step 3: Solving the Optimization Problem
After the change of variable we easily find the equivalent optimization problem:

$$\begin{array}{ll} \text{minimize} & -A_{k+1,i} \cdot g_k(y) \\ \text{over} & y \in [0,1]^n, \\ \text{subject to} & A'_k y \leq b'_k. \end{array}$$

Then, thanks to Proposition 2, a lower bound $-b_{k+1,i}$ can be found.
The reachable set at the step $k+1$ will be $\overline{X}_{k+1} = Poly(A_{k+1}, b_{k+1})$.

3.3 Choice of the Templates

In this section, we discuss the choice of the templates A_k for the polyhedra $\overline{X}_k$ and C_k for the parallelotope Q_k used to over-approximate the reachable sets.

Dynamical Templates for Polyhedra $\overline{X}_k$
Let us consider the template A_k for the polyhedron $\overline{X}_k = \{x \in \mathbb{R}^n |\ A_k \cdot x \leq b_k\}$. We propose a method which involves determining dynamical templates based on the dynamics of the system.

In the next iteration, we want to compute a new template A_{k+1} that reflects as much as possible the changes of the shape of $\overline{X}_k$ under the polynomial f. For that purpose, we use a local linear approximation of the dynamics of the polynomial dynamical system (7) given by the first order Taylor expansion around the centroid x^*_k of the last computed polyhedron $\overline{X}_k$:

$$f(x)\ \approx L_k(x) = f(x^*_k) + J(x^*_k)(x - x^*_k)$$

where J is the Jacobian matrix of the function f. Let us denote $F_k = J(x^*_k)$ and $h_k = f(x^*_k) - J(x^*_k)x^*_k$, then in a neighborhood of x^*_k the nonlinear dynamics can be roughly approximated by $x_{k+1} = F_k x_k + h_k$. Assuming that F_k is invertible, this gives $x_k = F_k^{-1} x_{k+1} - F_k^{-1} h_k$. Transposing the constraints on x_k given by $\overline{X}_k$ to x_{k+1}, we obtain

$$A_k F_k^{-1} x_{k+1} \leq b_k + A_k F_k^{-1} h_k$$

Then, it appears that a reasonable template for $\overline{X}_{k+1}$ should be $A_{k+1} = A_k F_k^{-1}$. This new template A_{k+1} can then be used in next iteration for the computation of the polyhedron $\overline{X}_{k+1}$ using the method described in the previous section. Let us remark that this choice implies that our reachability algorithm is exact if f is an affine map.

Dynamical Templates for Parallelotope Q_k
It can be useful in some cases to take static axis aligned boxes for Q_k (i.e. C_k is the identity matrix for all $k = 0, \ldots, K$). This allows us to preserve the degrees of the polynomials when making the change of basis. However, as explained before, using static templates may produce increasingly large approximation errors. Similar to the polyhedra $\overline{X}_k$, the accuracy will be better if we use dynamical templates which take in consideration the dynamic of the system (essentially the rotation effects).

We should mention that the image of an oriented rectangular box Q_k by the linear map F_k is not necessarily an oriented rectangular box so we can not use directly the matrix F_k^{-1} computed previously. To solve this problem we will use an popular technique in interval analysis [22] based on the QR-Decomposition of matrices. Essentially, F_k will be written as the product of two matrices $F_k = \mathsf{Q_k R_k}$ where $\mathsf{Q_k}$ is an orthogonal matrix and R is an upper triangular one. Then, to choose the template C_{k+1} of the next oriented box Q_{k+1}, we apply our rotation transformation matrix $\mathsf{Q_k}$ to the given rectangular box Q_k which is equivalent to choose the template $C_{k+1} = C_k \mathsf{Q_k^\top}$. 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 Q_k.

3.4 Computation Cost and Related Work

We have presented two approaches to compute the Bernstein form of the polynomial after a change of variable: either we compute explicitly the change of variable and then use equation (3) or we proceed by using equation (4). Both methods have polynomial time and space complexity in $N_{\Delta'} = (\delta'_1 + 1) \times \cdots \times (\delta'_n + 1)$. The size of the matrix $\mathsf{B}_{\Delta'}$ introduced in the interpolation method is $(\delta'_1+1)\times\cdots\times(\delta'_n+1)$. Let us remark that in the context of reachability analysis, the inverse matrix $\mathsf{B}_{\Delta'}^{-1}$ has to be computed only once and can be used in each iteration to compute Bernstein coefficients by a simple matrix vector product operation.

In fact, we shall see from numerical experiments that both methods have their advantages depending on the form of the parallelotope Q. If Q is an axis-aligned box, we have already seen that the change of variable does not increase the degrees of the polynomial. So, the matrix $\mathsf{B}_{\Delta'}$ has a reasonnable size and the computation of the coefficients h_I using equation (4) is generally more efficient. If Q is a general parallelotope, then $\mathsf{B}_{\Delta'}$ might be much larger. In that case, since several coefficients f_I of the polynomial f might actually be zero, it will be more efficient to compute explicitly the change of variable and use equation (3) to determine the coefficients h_I.

Then, as we mentioned before, the linear program has polynomial complexity in its number of decision variables which is also $N_{\Delta'}$. Therefore, the complexity of the overall procedure is polynomial in $N_{\Delta'}$, and so is the complexity of the reachability procedure described in Lemma 1.

Finally, following the discussion in Remark 1, we would like to highlight that $N_{\Delta'}$ might be much smaller when the parallelotope Q is an axis-aligned box. Indeed, in that case $N_{\Delta'} = (\delta_1 + 1) \times \cdots \times (\delta_n + 1)$ whereas in the general case we have $N_{\Delta'} = (\delta_{max} + 1)^n$. This point might be taken into consideration in the reachability algorithm when choosing the template for parallelotopes Q_{k+1}.

One could also use interval analysis [14] for computing the reachable sets of polynomial systems. However, these methods are generally used with rectangular domains. Moreover, our approach obtains enclosures that are always finer than those obtained using Bernstein coefficients and it has been shown that these are generally more accurate than those computed using interval arithmetics [20]. Interval analysis methods can be improved using preconditionning techniques (see e.g.[22]), however these can also be used in our approach as shown in the previous section.

Also, a popular approach for nonlinear systems is to characterize reachable sets using Hamilton-Jacobi formulation [21], and then solve the resulting partial differential equations which requires expensive computations. Recent results of this approach can be seen in [15]. Another approach is based on a discretization of the state-space for abstraction (see e.g [28,16]) and approximation especially using linearization (see e.g [1,6]). For the particular class of polynomial systems,direct methods for reachability analysis [4,7] without state-space discretization has been proposed . The improvement over [7] has also been discussed in Section 2.2.

Concerning set representation, the work presented in this paper draws inspiration from the approach using template polyhedra [25]. In the hybrid systems verification, polynomial optimization can also be used to compute barrier certificates [24,25], and algebraic properties of polynomials are used to compute polynomial invariants [28] and to study computability issues of the image computation in [23].

4 Experimental Results

We implemented our reachability computation method and tested it on various examples. To solve linear programs, we use the publicly available lp_solve library.

4.1 FitzHugh-Nagumo Neuron Model

The first example we studied is a discrete time version of the FitzHugh-Nagumo model [8] which is a polynomial dynamical system modelling the electrical activity of a neuron:

$$\begin{cases} x_1(k+1) = x_1(k) + h\left((x_1(k) - \frac{x_1(k)^3}{3} - x_2(k) + I\right) \\ x_2(k+1) = x_2(k) + h\left(0.08(x_1(k) + 0.7 - 0.8x_2(k))\right) \end{cases}$$

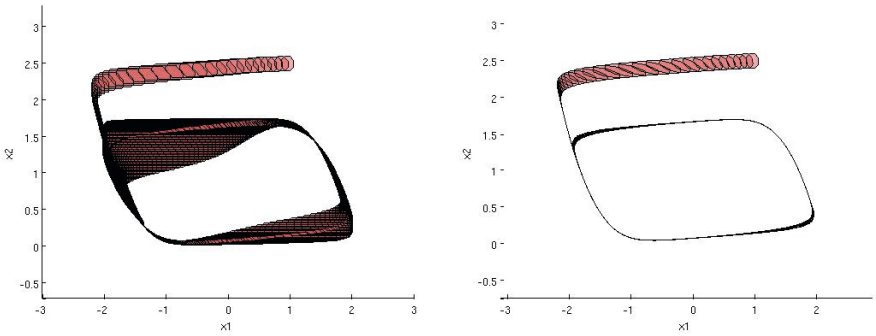

Fig. 1. Reachability computation for the FitzHugh-Nagumo neuron model using static (left) and dynamical (right) template polyhedra $\overline{X}_k$ with static bounding boxes

where the model parameter I is equal to $\frac{7}{8}$ and the time step $h = 0.05$. This system is known to have a limit cycle.

Figure 1 shows two reachable set evolutions where the initial set is a regular octagon included in the bounding box $[0.9, 1.1] \times [2.4, 2.6]$. The figure on the left was computed using static templates for polyhedra $\overline{X}_k$ where dynamical templates are used for the figure on the right. In both cases, we use axis-aligned boxes for Q_k. For a better readability, the reachable sets are plotted once every 5 steps. We observed a limit cycle after 1000 iterations. The computation time is 1.16 seconds using a static template and 1.22 seconds using the dynamical templates. We can see from the figure a significant precision improvement obtained by using dynamical templates, at little additional cost.

4.2 Prey Predator Model and Performance Evaluation

Now we consider the generalized Lotka-Volterra equations modelling 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 $A \in \mathbb{R}^{n \times n}$.

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 2 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.

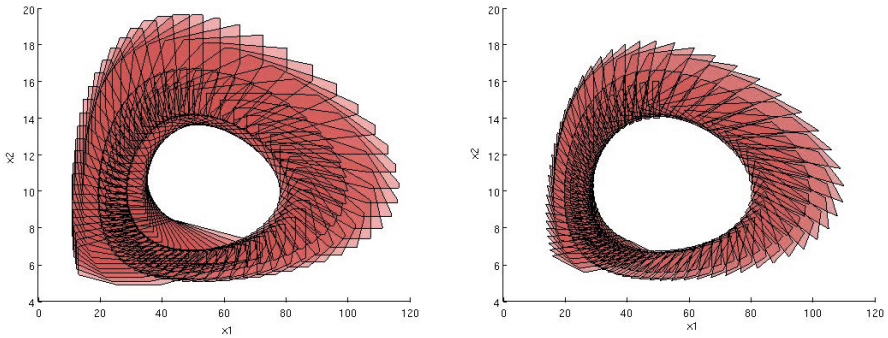

Fig. 2. Reachability computation for a 2 dimensional predator-prey model using dynamical template polyhedra $\overline{X}_k$ with axis aligned (left) and oriented (right) bounding boxes

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 1).

Table 1. Computation time for one reachable set computation iteration for some generated Lotka-Voltera systems

dim	explicit	interpol	$\mathsf{B}^{-1}_{\Delta'}$	dim	explicit	interpol	$\mathsf{B}^{-1}_{\Delta'}$
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

We observe that the interpolation method provides more effective results than the explicit computation of Bernstein coefficient but requires to compute the matrix $\mathsf{B}^{-1}_{\Delta'}$ before starting the analysis. A similar evaluation was done using oriented boxes but the results show that this method is not tractable over 4 dimension due to the degree elevation of polynomials by the change of variable when we don't use axis-aligned boxes.

5 Conclusion

In this paper we proposed an approach for computing reachable sets of polynomial dynamical systems. This approach combines optimization and set representation using template polyhedra. The novelty of our results lies in a efficient method for relaxing a polynomial optimization problem to a linear programming problem. On the other hand, by exploiting the evolution of the system we proposed a way to determine templates dynamically so that the reachable sets can be approximated more accurately. The approach was implemented and our experimental results are promising, compared to the existing results (see e.g. [7]). We intend to continue this work in a number of directions. One direction involves an extension of the approach to systems with parameters and uncertain inputs. Additionally, the evolution of templates can be estimated locally around each facet rather than globally at the centroid of a template polyhedron.

References

1. Asarin, E., Dang, T., Girard, A.: Hybridization methods for the analysis of nonlinear systems. Acta Informatica 43(7), 451–476 (2007)
2. Bernstein, S.: Collected Works, vol. 1. USSR Academy of Sciences (1952)
3. Bernstein, S.: Collected Works, vol. 2. USSR Academy of Sciences (1954)
4. Dang, T.: Approximate Reachability Computation for Polynomial Systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 138–152. Springer, Heidelberg (2006)
5. Dang, T., Le Guernic, C., Maler, O.: Computing Reachable States for Nonlinear Biological Models. In: Degano, P., Gorrieri, R. (eds.) CMSB 2009. LNCS, vol. 5688, pp. 126–141. Springer, Heidelberg (2009)
6. Dang, T., Maler, O., Testylier, R.: Accurate hybridization of nonlinear systems. In: HSCC 2010, pp. 11–20 (2010)
7. Dang, T., Salinas, D.: Image Computation for Polynomial Dynamical Systems Using the Bernstein Expansion. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 219–232. Springer, Heidelberg (2009)
8. FitzHugh, R.: Impulses and physiological states in theoretical models of nerve membrane. Biophysical J. 1, 445–466 (1961)
9. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable Verification of Hybrid Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)
10. Girard, A., Le Guernic, C., Maler, O.: Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 257–271. Springer, Heidelberg (2006)
11. Halasz, A., Kumar, V., Imielinski, M., Belta, C., Sokolsky, O., Pathak, S., Rubin, H.: Analysis of lactose metabolism in e.coli using reachability analysis of hybrid systems. IET Systems Biology 1(2), 130–148 (2007)
12. Han, Z., Krogh, B.H.: Reachability Analysis of Large-Scale Affine Systems Using Low-Dimensional Polytopes. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 287–301. Springer, Heidelberg (2006)

13. Horst, R., Tuy, H.: Global optimazation: Deterministic approaches, 2nd edn. Springer (1993)
14. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer (2001)
15. Kaynama, S., Oishi, M., Mitchell, I., Dumont, G.A.: The continual reachability set and its computation using maximal reachability techniques. In: CDC (2011)
16. Kloetzer, M., Belta, C.: Reachability Analysis of Multi-affine Systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 348–362. Springer, Heidelberg (2006)
17. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM Journal of Optimization 11(3), 796–817 (2001)
18. Le Guernic, C., Girard, A.: Reachability Analysis of Hybrid Systems Using Support Functions. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 540–554. Springer, Heidelberg (2009)
19. Lin, Q., Rokne, J.G.: Interval approxiamtions of higher order to the ranges of functions. Computers Math 31, 101–109 (1996)
20. Martin, R., Shou, H., Voiculescu, I., Bowyer, A., Wang, G.: Comparison of interval methods for plotting algebraic curves. Computer Aided Geometric Design (19), 553–587 (2002)
21. Mitchell, I., Tomlin, C.J.: Level Set Methods for Computation in Hybrid Systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 310–323. Springer, Heidelberg (2000)
22. Nedialkov, N.S., Jackson, K.R., Corliss, G.F.: Validated solutions of initial value problems for ordinary differential equations. Applied Mathematics and Computation (105), 21–68 (1999)
23. Platzer, A., Clarke, E.M.: The Image Computation Problem in Hybrid Systems Model Checking. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 473–486. Springer, Heidelberg (2007)
24. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Transactions on Automatic Control 52(8), 1415–1429 (2007)
25. Sankaranarayanan, S., Dang, T., Ivančić, F.: Symbolic Model Checking of Hybrid Systems Using Template Polyhedra. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 188–202. Springer, Heidelberg (2008)
26. Sherali, H.D., Tuncbilek, C.H.: A global optimization algorithm for polynomial programming using a reformulation-linearization technique. Journal of Global Optimization 2, 101–112 (1991)
27. Sherali, H.D., Tuncbilek, C.H.: New reformulation-linearization/convexification relaxations for univariate and multivariate polynomial programming problems. Operation Research Letters 21, 1–9 (1997)
28. Tiwari, A., Khanna, G.: Nonlinear Systems: Approximating Reach Sets. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 600–614. Springer, Heidelberg (2004)
29. Yordanov, B., Belta, C.: Cdc. In: A Formal Verification Approach to the Design of Synthetic Gene Networks (2011)