Reachable set computation using box splines and integer polynomial optimization

Thao Dang

VERIMAG,
Centre Equation
2 avenue de Vignate
38610 Gières, France

Abstract. This paper is concerned with a method for over-approximating the reachable sets of dynamical systems described by polynomial difference equations. This method is based on the box spline representation of polynomials, the coefficients of which can be used to enclose all the reachable points. The coefficients correspond to the integer points inside a set and their enumeration can be expensive. In this work, we propose a way to avoid this explicit enumeration via integer polynomial optimization.

1 Introduction

In this paper we describe a method for over-approximating the reachable sets of dynamical systems described by polynomial difference equations. These equations arises in many practical systems originating from biological and chemical engineering applications. They can also arise in the control software which implements some numerical discretization of an embedded controller. In the context of embedded and hybrid systems, formal verification is an important problem especially in the design of safety-critical systems. Reachability computation can be used to verify safety properies, which essentially state that some set of unsafe states is not reachable.

The method we proposed is based on the box-spline representation of polynomials, the coefficients of which can be used to enclose all the reachable points. Compared to our previous method using the Bézier simplex representation, this method works on more general domains and does not require simplicial decompositions.

We focus on a discrete-time dynamical systems described by a difference equation of the form $x(k+1) = \pi(x(k))$ where $x(k) \in \mathbb{R}^n$ and π is a multi-variate polynomial. It should be noted that our method can be extended to a continuous-time system described by a differential equation.

Indeed, one can approximate it by a difference equation using a discretization scheme with some uncertainty term to guarantee conservativeness of the approximation.

The goal is to approximate the set of all states reachable by this system from some initial set X_0. This requires successive computations of the image of a set by the polynomial π. The main ideas of our method can be summarized as follows. We represent the polynomial in question using a box spline basis. The coefficients of this representation allow to over-approximate the image.

The paper is organized as follows. In Section 2 we recall the method using the box-spline representation to approximate the image of a set by a polynomial. In the next section we present how to use integer polynomial optimisation to avoid expensive integer points enumeration.

2 Box splines

2.1 Definition

We will present an inductive definition of box spline, and other definitions (such as, geometric or by recursion) can be found in [3].

Let n be the dimension (or the number of variables). Given an integer $k \geq n$, let $V = \{\xi_1, \ldots, \xi_k\}$ be a set of k vectors in $\mathbb{R}^n$ where $\xi_1, \xi_2, \ldots, \xi_n$ are linearly independent. Each such vector is called *direction*. We denote by $M = [\xi_1\, \xi_2\, \ldots \xi_k]$ a matrix of size $n \times k$, the columns of which are the vectors in V. For convenience of notation, we use the same letter ξ_i to refer to a vector in V or a column of the matrix M.

Without loss of generality, it could be assumed that the first n columns of M form the n-dimensional identity matrix I. The inductive definition of a box spline B_M associated with V is as follows.

We denote $s = k - n$ and start with the base case where $k = n$ and $s = 0$. Then $M_s = I$ and $B_{M_s}(x) = 1$ if $x \in [0, 1)^n$ and 0 otherwise. Note that this function is piecewise constant and has degree $s = 0$.

If we add a column in M denoted by $M \cup \xi$, then the box spline associated with the new matrix $M_s = M_{s-1} \cup \xi$ is defined as:

$$B_{M_s \cup \xi} = \mathcal{Z}_0^1 B_{M_{s-1}}(x - t\xi)dt. \tag{1}$$

Thus, each convolution in another direction v increases the degree by 1, and when $s = k - n$, B_M is a piecewise polynomial of degree $k - n$.

We use the well-known Zwart-Powell box spline [3] to illustrate the above definition. First, we consider the following matrix of size 2×3

$$M = \begin{pmatrix} 1\,0\,1 \\ 0\,1\,1 \end{pmatrix}$$

In this case we have $n = 2$, $k = 3$. We start with $k = n = 2$ and the identity matrix formed by two vectors $\xi_1 = (1, 0)$ and $\xi_2 = (0, 1)$ defines a unit box (see Figure 2.1-(a)). By adding the vector $\xi_3 = (1, 1)$ the matrix M defines a mesh inside a zonotope, as shown in Figure 2.1-(b). Using the definition (1), we obtain the box spline B_M which is the hat function shown in Figure 2.1.

We further add a new vector $\xi_4 = (-1, 1)$ to form a matrix M' as follows:

$$M' = \begin{pmatrix} 1\,0\,1\,-1 \\ 0\,1\,1\;\;1 \end{pmatrix}.$$

This defines a new mesh (see Figure 2.1-(d)) and the corresponding box spline $B_{M'}$ is depicted in Figure 2.1. This function is called Zwart-Powell box spline.

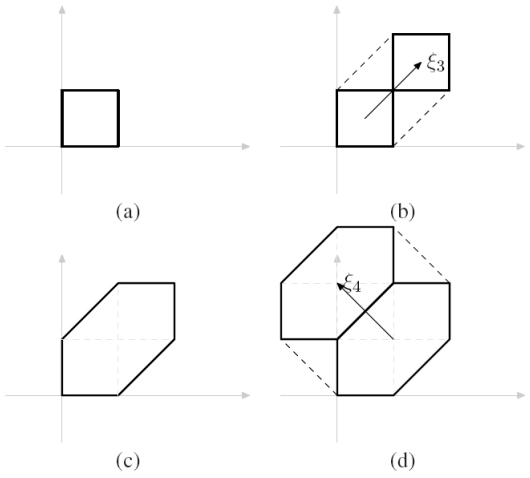

Fig. 1. Supports of some box splines.

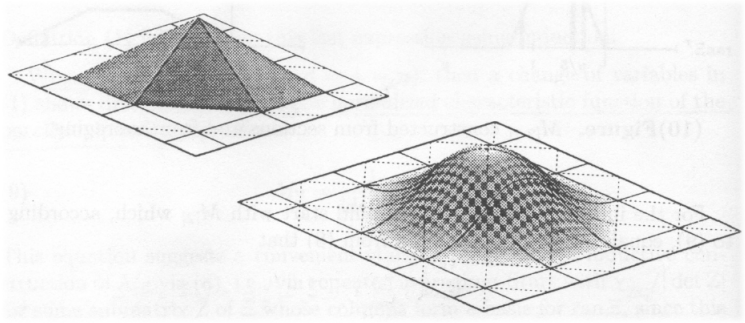

Fig. 2. The hat function (left) and the Zwart-Powell box spline (right).

2.2 Basic properties

The box splines have the following properties which follow directly from their definition.

For all $x \in [\xi_1\, \xi_2\, \ldots \xi_k][0,1)^k$, $B_M(x) > 0$. The support of $B_M(x)$ is

$$Z = [\xi_1 \xi_2 \ldots \xi_k][0,1]^k. \tag{2}$$

that is the sum of the vectors in V, or *the zonotope with V as the set of its generators*. The box spline B_M is symmetric with respect to the center of its support.

It should be noted that the box spline B_M is *piecewise polynomial*. Indeed, it is a polynomial of degree *at most* $(k-n)$ within each element of the simplicial mesh defined by V over the support of B_M. The following lemma states two important properties of B_M.

Let ρ de the minimal number of vectors that need to be removed from V so that they do not span $\mathbb{R}^n$. The box spline B_M is $(\rho - 2)$ times continuously differentiable.

Again, we assume that the vectors in $V = \{v_1, \ldots, v_k\}$ span $\mathbb{R}^n$. We define an *integer shift* of the box spline $B_M(x - \mathbf{j})$ with $\mathbf{j} \in \mathcal{Z}^n$. Since $B_M(x)$ is non-negative and the sum of all the integer shifts of $B_M(x)$ is 1, the integer shifts of any box spline $B_M(x)$ form a *partition of unity*.

We are now interested in the space of polynomials spanned by all shifts of B_M. We define the *index set* for each $x \in \mathbb{R}^n$ as follows:

$$\mathcal{I}_M(x) = \{\mathtt{i} \in \mathcal{Z}^n \mid B_M(x - \mathtt{i}) \neq 0\}.$$

Note that this set is finite for any x; therefore, for simplicity we write infinite linear combinations of the integer shifts while keeping in mind that the combinations are only over the associated index set.

Lemma 1. *If the box spline B_M is r times continuously differentiable. Then, for any polynomial* $\mathbf{a}$ *of degree* $(r+1)$*, the function*

$$\pi(x) = \sum_{\mathtt{i} \in \mathcal{Z}^n} \mathbf{a}(\mathtt{i}) B_M(x - \mathtt{i}). \tag{3}$$

is a polynomial of degree $(r+1)$*, and any polynomial can be represented as in (3) by choosing* M *appropriately. The coefficients* $\mathbf{a}(\mathtt{i})$ *are called the* control points*, and the function* $\mathbf{a}$ *is called the weight function.*

Lemma 2. *Given a point* x*,* $\pi(x)$ *lies in the convex hull of the control points corresponding to the index set* $\mathcal{I}_M(x)$*:*

$$\pi(x) = conv\{\mathbf{a}(\mathtt{i}) \mid \mathtt{i} \in \mathcal{I}_M(x)\}.$$

This property, called the *convex-hull property*, will be used for our image computation problem.

2.3 Index set

As mentioned earlier, when writing the infinite sum, it indeed suffices to consider the integer points in the index set. Given x, the index set $\mathcal{I}(x)$ associated with the box spline B_M is $\mathcal{I}_M(x) = \{\mathtt{i} \in \mathcal{Z}^n \mid B_M(x - \mathtt{i}) \neq 0\}$. It is not hard to prove that

$$\mathcal{I}_M(x) = \mathcal{Z}^n \cap (x - M[0,1]^k). \tag{4}$$

The index set for all x inside some set $X \subseteq \mathbb{R}^n$, denoted by $\mathcal{I}_M(X)$, is defined as follows:

$$\begin{aligned} \mathcal{I}_M(X) &= \{\mathtt{i} \in \mathcal{Z}^n \mid \exists x \in X : B_M(x - \mathtt{i}) \neq 0\} \\ &= \mathcal{Z}^n \cap (X \oplus (-M[0,1]^k)). \end{aligned}$$

where $\oplus$ denotes the Minkowski sum. Then the computation of the index set $\mathcal{I}_M(X)$ amounts to enumerating all the points with integer coordinates inside the Minkowski sum of P and the zonotope $-M[0,1]^k$.

3 Reachable set computation

We consider a dynamical discrete-time system

$$x_{k+1} = \pi(x_k)$$

where π is a polynomial map and $x_0 \in X_0$. Given a set $X \subset \mathbb{R}^n$, we write:

$$\pi(X) = \{\pi(x) \mid x \in X\}.$$

Let $P^k = \pi(P^{k-1})$ with $P^0 = X_0$. The reachable set of the above system from the initial set X_0 is defined as:

$$R = \bigcup_k P^k$$

In order to compute R, we need a method for computing the image of a set by a polynomial. To this end, we write the polynomial π using the a box spline representation, provided that the box spline B_M must be of appropriate degree. We then use the convex-hull property of the box spline representation to over-approximate the image $\pi(X)$ of a given set X. To compute the control points, we derive a symbolic expression of the weight function $\mathbf{a}$, and then determine the index set, that is the set of integer points at which the integer shifts in the box spline representation (3) is non-null.

We assume that we have chosen an appropriate matrix M. This means that the condition on M, stated in Lemma 1, which guarantees that the associated box spline B_M can reproduce all the polynomials of the required degree.

In the previous work [2], we proposed a method for enumerating the index set associated with the Minkowski sum $conv\{P^i\} \oplus (-M[0,1]^k)$. To each integer point in the index set, we apply the weight function $\mathbf{a}$ to obtain the corresponding control point. The disadvantage of this method is that the integer point enumeration is expensive and moreover they can be numerous. The goal of this work is to remedy this problem by formula it as an integer polynomial optimization problem. We first describe a method for computing the weight function symbolically and show that the resulting is a polynomial. Hence, if using a family of half-spaces with a given normal vector, finding a half-space that tightly contains the control points of interest can be formulated as a problem of optimizing a polynomial over a set of integer points inside a polytope.

4 Computing the weight function

Any polynomial can be decomposed to a linear combination of monomials, it is possible to compute the control points for each monomial and then combine them.

More concretely, if the polynomial π is a linear combination of two monomials m_1 and m_2, i.e. $\pi = km + k'm'$. Then, if $m(x) = \sum_{\mathtt{i} \in \mathcal{Z}^n} \mathbf{a}(\mathtt{i}) B_M(x - \mathtt{i})$ and $m'(x) = \sum_{\mathtt{i} \in \mathcal{Z}^n} \mathbf{a}'(\mathtt{i}) B_M(x - \mathtt{i})$, then $\pi(x) = \sum_{\mathtt{i} \in \mathcal{Z}^n} (\mathbf{a}(\mathtt{i}) + \mathbf{a}'(\mathtt{i})) B_M(x - \mathtt{i})$.

We use the extension of the Marsden identity to derive an analytic expression of $\mathbf{a}$. We consider a monomial of the form $x_1^{r_1} \dots x_n^{r_n}$ where each r_i is a non-negative integer, and we denote it as $m_r(x) = x^r$ where $r = (r_1, \dots, r_n)$ is called the multi-index. We define $r' \prec r$ iff $r \neq r'$ and $\forall i \in \{1, \dots, n\} : r'_i \leq r_i$. Similarly, the difference $r - r'$ can be defined componentwise, that is $r - r' = (r_1 - r'_1, \dots, r_n - r'_n)$.

Given a monomial m_r, the goal is to determine the weight function $\mathbf{a}_r$ such that the monomial can be represented using the box spline B_M, that is, $x^r = \sum_{\mathtt{i}} \mathbf{a}(\mathtt{i}) B_M(x - \mathtt{i})$. We define the operator

$$\mu : f \rightarrow \sum_{\mathtt{i}} B_M(\mathtt{i}) f(-\mathtt{i}). \tag{5}$$

The (symbolic) computation of the function $\mathbf{a}$ can be done using the following recurrence [3]:

$$\begin{cases} \mathbf{a}_r & = m_r - \sum_{r' \prec r} \mu(m_{r-r'})\, \mathbf{a}_{r'} \\ \mathbf{a}_{(0,\dots,0)} & = 1. \end{cases} \tag{6}$$

The generation of $\mu(m_{r'})$ for all $r' \prec r$ can be done before starting the recursion (6). For a fixed r, we need to compute the expresions of $c_{r'}$ for all $r' \prec r$, and this requires computing the application of the operator μ on them.

From the construction of the weight function we can see that the result is a polynomial. ∎

5 Polynomial optimization over integer points

The improvements we propose in this work are based on the following observations. First, we remark that the enumeration of the integer points inside a polytope (represented as the Minkowski sum of the convex hull of a set of points and a zonotope) can be very expensive. In addition, we have to evaluate the weight function at every integer point. Indeed, as we can see in Section 4, the weight function is a polynomial, and thus we can formulate the problem of approximate the convex hull of the control points directly as a problem of optimizing a polynomial function over a set of integer points. This optimization problem can then be solved

using a method of encoding integer points by the Barnikov generating functions [1].

More precisely, the problem we are addressing now is stated as follows: Given a polytope Q and a polynomial function $\mathbf{a}$, compute a polytope G such that

$$chull\{\mathbf{a}(\mathtt{i}) \mid \mathtt{i} \in Q\} \subseteq G \tag{7}$$

In particular, on one hand we want Q to be as small as possible. On the other hand,, we want Q to have a manageable geometric complexity. Indeed, for various operations needed for the verification of hybrid systems (such as Boolean operations over the reachable sets), discrete transition can make the complexity of the reachable sets grow very fast. One way to handle such situations is to use polyhedral representations with fixed structures (at the price of compromising accuracy). In this work, we propose to use template polyhedra (see for example in [4]).

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 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 $\mathbf{d} \in \mathbb{R}^m$, a template polyhedron is defined by considering the conjunction of the linear inequalities of the form $\bigwedge_{i=1,\ldots,m} H^i\mathbf{x} \leq d_i$. We denote this polyhedron by $\langle H, \mathbf{d}\rangle$.

We now suppose that we seek G as a template polyhedron with a given H. Determining G means determining the coefficient vector $\mathbf{d}$.

It is not hard to see that the following condition is sufficient for (7) to hold:

$$\forall \mathtt{i} \in Q \ : \ H\mathbf{a}(\mathtt{i}) \leq \mathbf{d}$$

Therefore, to determine $\mathbf{d}$, one can formulate the following optimization problem:

$$\forall i \in \{1, \ldots, m\}, d_i = \max(\Sigma_{k=1}^{n} H_k^i \mathbf{a}_k(\mathtt{i})) \text{ subj. to } \mathtt{i} \in Q. \tag{8}$$

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 over a set of integer points. In the next section we focus on this problem.

6 Polynomial optimization over integer points

As we have seen that enumeration of integer point can be very expensive, A. Barnikov proposed a method for encoding all the integer points inside

a polytope as a rational functions (see [1]). Integer points are encoded as exponent vectors of monomials. For example, the point $(\mathtt{i}_1, \ldots, \mathtt{i}_n)$ is encoded by the monomial $x_1^{\mathtt{i}_1} \ldots x_n^{\mathtt{i}_n}$. The set of integer points is then represented by a polynomial:

$$h_Q(x) = \sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} x^{\mathtt{i}}.$$

This (exponentially large) sum of monomials $h_Q(x)$ can be written as a polynomially large sum of rational functions of the form:

$$h_Q(x) = \sum_{k \in K} E_k \frac{x^{\mathbf{u}_k}}{\prod_{j=1} n(1 - x^{\mathbf{v}_{kj}})} \tag{9}$$

where $E_i\{1, 1\}$ and $\mathbf{u}, \mathbf{v}_{jk} \in Z^n$ for all i and j. Indeed, it was proven that K is a polynomial-size set.

The problem now is to encode the values of a polynomial g in a similar manner. Essentially, the differential operators associated to g can be used to encode

$$\sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} g(\mathtt{i}) x^{\mathtt{i}}.$$

Indeed, we consider the differential operator $x_j \frac{\partial}{\partial x_j}$. First, for a simple case the polynomial $g(x) = x_j$. Then,

$$x_j \frac{\partial}{\partial x_j} h_Q(x) = \sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} x_j \frac{\partial x^{\mathtt{i}}}{\partial x_j} = \sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} \mathtt{i}_j x^{\mathtt{i}}.$$

By differentiating the generating function $h_Q(x)$ in form of rational function (9) we will obtain the Barnikov representation $h_{Q,g,k}(x)$ of the function $\sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} g^k(\mathtt{i}) x^{\mathtt{i}}$ and it was proven that this representation is polynomial in the degree of g (for a fixed dimension n). We can extend to above argument to more general monomials $g(x) = x^{\mathbf{j}}$.

We are now able to encode the values of $\sum_{\mathtt{i} \in Q \cap \mathcal{Z}^n} g(\mathtt{i}) x^{\mathtt{i}}$. over the integer points inside the polytope Q. The remaining question is to use this representation to solve our initial optimization problem. To this end, we first note that

$$L^k = (h_{Q,g,k}(1)/h_{Q,g,0}(1))^{1/k}$$

and

$$U^k = (h_{Q,g,k}(1))^{1/k}$$

are respectively a lower and upper bound of the maximal value of g over the integer points in Q. And therefore we can compute a sequence of (L^k, U^k) until they approach to the same integer value.

It is important to emphasize that the above works for non-negative polynomials g. For polynomials with negative values, one needs to estimate an upper and a lower bound on each variable x_j and therefore a bound on the absolute value of each monomial and then to shift the polynomial accordingly.

7 Reachable set computation algorithm

We can now summarize our developments so far into a reachable set computation algorithm

Algorithm 1 Reachable set computation

$Q^0 = X_0$
$Z = (-M[0,1]^k)$
$i = 0$
repeat
  forall$(m \in \{1, \ldots, m\})$
    $\mathbf{d}^m = \max\{H^m \mathbf{a}(\mathtt{i}) \mid \mathtt{i} \in P^i \oplus Z\}$ /* *computing polyhedral coefficient vector* */
  endforall
    $P^{i+1} = \{x \mid Hx \leq \mathbf{d}\}$ /* computing the template polyhedron */
  $i = i + 1$
until $i > K_{max}$

8 Conclusion

In this work we present an improvement in the algorithm using box splines to approximate reachable sets. This improvement will reduce considerably the complexity of the algorithm since the Barnikov representation has a polynomial size while the number of integer points is exponentially large. It remains to validate this via experiments with concrete examples, which is part of our ongoing work.

References

1. A.I. Barnikov and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. In *New Perspectives in Algebraic Combinatorics*, page 91147. Math. Sci. Res. Inst. Publ. 38, Cambridge Univ. Press, Cambridge, 1999.

2. T. Dang. Box splines and reachability computation for polynomial systems. In *Technical report VERIMAG, Grenoble*, Jan 2009.
3. Carl de Boor, Klaus Hülig, and Sherman Riemenschneider. *Box splines*. Springer-Verlag New York, Inc., New York, NY, 1993.
4. 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.