# Parallelotope Bundles for Polynomial Reachability*

Tommaso Dreossi
University of Udine
VERIMAG
2 Avenue de Vignate
38610 Gieres, France
tommaso.dreossi@imag.fr

Thao Dang
VERIMAG
2 Avenue de Vignate
38610 Gieres, France
thao.dang@imag.fr

Carla Piazza
University of Udine
via delle Scienze 206
33100 Udine, Italy
carla.piazza@uniud.it

## ABSTRACT

In this work we present *parallelotope bundles*, i.e., sets of parallelotopes for a symbolic representation of polytopes. We define a compact representation of these objects and show that any polytope can be canonically expressed by a bundle. We propose efficient algorithms for the manipulation of bundles. Among these, we define techniques for computing tight over-approximations of polynomial transformations. We apply our framework, in combination with the Bernstein technique, to the reachability problem for polynomial dynamical systems. The accuracy and scalability of our approach are validated on a number of case studies.

## Keywords

Reachability; dynamical system; parallelotope bundle; polytope

## 1. INTRODUCTION

In this paper we are concerned with polynomial image computation and its application to reachability analysis of dynamical systems. The image computation can be stated as follows: given a function $\mathbf{f} : D \to R$ and a set $X \subset D$, compute the image of $X$ by $\mathbf{f}$, that is the set $Y = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in X\}$. The function $\mathbf{f}$, commonly called *dynamics*, describes the flow of the system (or its approximation derived from the system vector field), where the sets $D$ and $R$ are typically $\mathbb{R}^n$ when the state variables of the dynamical system in question are real-valued. Set-based image computation is a central operation for many verification and synthesis procedures. A number of approaches in the verification of continuous and hybrid systems can be seen as extensions of numerical integration where the iterative numerical schemes are computed on sets. In addition, image computation finds applications in other problems such as control synthesis [22] or program verification through invariant computation via fixed-point iteration.

For affine functions, image computation can be efficiently computed using various set representations, such as polyhedra, ellipsoids, support functions, zonotopes (see, e.g., [10, 4, 29, 18, 23, 27, 24, 1, 15]). For nonlinear functions such as polynomials, the problem is more difficult since many properties of linear functions that facilitate image computation (such as convexity preservation) are no longer valid. One approach to handle nonlinearity is to use piecewise linear approximations [20, 5, 6, 2]. While this approach can be applied to a quite general class of flows and vector fields, other approaches focus on some special classes of functions and exploit their properties. Among these classes, polynomials have drawn a particular interest thanks to their applications in the modeling of biological processes, economic, and engineering systems.

In [13] we proposed a technique for polynomial systems using the Bernstein expansion, which allows one to efficiently approximate the image of a box domain by a polynomial. However, the box representation is restrictive in geometrical expressiveness and thus limited in approximation power. For better approximation accuracy, more general sets should be used. In [12] we introduced the idea of using parallelotopes as a trade-off between computational complexity and approximation accuracy. Indeed, using parallelotopes, the transformation to boxes can be done efficiently, and in addition, the choice of parallelotopic form can be used to fine tune the approximation. In this work, we generalize this idea to polytopes defining *parallelotope bundles* that allow one to represent polytopes as finite sets of parallelotopes. Intuitively, the strength of parallelotope bundles is that the images of the represented polytopes can be over-approximated by the intersection of the images of the parallelotopes that constitute the bundles. Hence, exploiting the transformations of single parallelotopes, thanks to parallelotope bundles we can tightly over-approximate the images of polytopes by polynomials.

Related to our work, in [8, 9], for a continuous system described by nonlinear ODEs, the flow is first approximated from an interval domain in a time interval by a Taylor polynomial around some point inside the domain. Then, this polynomial is bloated by an interval to account for the remainder terms. Taylor models support basic arithmetic operations using interval arithmetics techniques and can thus handle polynomials. Prior to this work, in [21] interval-based integration of ODEs was used for computing the reachable set of nonlinear hybrid systems. In [26] the reachable set

*This work has been partially supported by GNCS-INdAM and the ANR-INS project MALTHY.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*HSCC '16, April 12–14, 2016, Vienna, Austria.*

© 2016 ACM. ISBN 978-1-4503-3955-1/16/04...$15.00

DOI: http://dx.doi.org/10.1145/2883817.2883838

is also represented by box cells in a partition of the state space and the propagation of the system from one box to a neighboring one is conservatively determined by the flow constraints on their boundary. In terms of reachable set representation, all the above-mentioned methods use boxes. Parallelotopes have been used in [3] as domains for abstract interpretation and algorithms for linear operations, together with specific operations for program verification (assignment, union). Nevertheless, the abstract domain is defined by a single parallelotope and nonlinear operators are not yet considered. The use of zonotopes and template polyhedra adopted by [28, 27, 17, 1, 19] as set representations are also close to our work, but the box-domain requirement of the Bernstein technique makes this sets less convenient to handle, since their transformation to boxes are expensive.

The paper is organized as follows: in Section 2 we introduce basic definitions; Section 3 defines parallelotope bundles and operations on them; Section 4 is dedicated to the representation and the algorithmic manipulation of parallelotope bundles; in Section 5 a reachability algorithm based parallelotope bundles for polynomial dynamical systems is given; the proposed techniques are experimentally validated in Section 6; Section 7 ends the paper with some remarks. The proofs of this work are available at http://www-verimag.imag.fr/~dreossi/docs/papers/bundles_2015.pdf

## 2. PRELIMINARIES

A half-space $h$ of $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ characterized by a linear inequality, i.e., $h = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{dx} \leq c\}$, where $\mathbf{d} \in \mathbb{R}^n$ is a non-null vector also called *normal vector* and $c \in \mathbb{R}$ is an *offset*.

Definition 1 (Polytope). *A* polytope $Q$ *is a bounded subset of* $\mathbb{R}^n$ *such that there is a finite set* $H = \{h_1, \ldots, h_k\}$ *of half-spaces whose intersection is* $Q$*, i.e.,* $Q = \cap_{i=1}^{k} h_i$.

The linear constraints that generate the half-spaces can be organized in a matrix $D \in \mathbb{R}^{k \times n}$, called *direction matrix* (or *template*) and a vector $\mathbf{c} \in \mathbb{R}^k$, called *offset vector*. The $i$-th row $D_i$ of $D$ together with the $i$-th component $\mathbf{c}_i$ of $\mathbf{c}$ define the half-space $h_i \in H$, being its normal vector and offset, respectively. With a slight abuse of notation, we denote with $\langle D, \mathbf{c} \rangle$ the polytope generated by the direction matrix $D$ the offset vector $\mathbf{c}$. Notice that polytopes are bounded subsets of $\mathbb{R}^n$, hence not all the pairs $\langle D, \mathbf{c} \rangle$ define a polytope.

A polytope $Q$ can be represented as the intersection of different sets of half-spaces. For instance, adding to $Q$ new half-spaces that do not affect the intersection, we get a new representation of $Q$. Moreover, even without adding new half-spaces, we can get a new representation by multiplying the $i$-th row of $D$ and the $i$-th component of $\mathbf{c}$ by a positive constant. However, if needed, one can refer to the canonical representation in which all the half-spaces are necessary and the direction vectors are versors, i.e., vectors of norm one.

Parallelotopes are centrally symmetric polytopes of $\mathbb{R}^n$ having $2n$ pairwise parallel constraints.

Definition 2 (Parallelotope). *Let* $\langle \Lambda, \mathbf{c} \rangle$ *be a polytope in* $\mathbb{R}^n$ *with* $\Lambda \in \mathbb{R}^{2n \times n}$ *direction matrix such that* $\Lambda_i = -\Lambda_{i+n}$*, for* $i \in \{1, \ldots, n\}$ *and* $\mathbf{c} \in \mathbb{R}^{2n}$ *offset vector. The* parallelotope $P$ *generated by* $\Lambda$ *and* $\mathbf{c}$ *is* $P = \langle \Lambda, \mathbf{c} \rangle$.

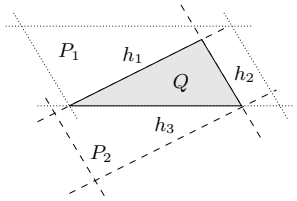



Figure 1: A polytope $Q$ and a possible decomposing bundle $\{P_1, P_2\}$, i.e., $\{P_1, P_2\}^{\cap} = Q$.

## 3. PARALLELOTOPE BUNDLES

We now define *parallelotope bundles*, that are sets of parallelotopes whose intersections symbolically represent polytopes. Our definition and notation are inspired by [1].

Definition 3 (Parallelotope Bundle). *A* parallelotope bundle *is a finite set of parallelotopes* $\{P_1, \ldots, P_b\}$ *whose intersection, denoted by* $\{P_1, \ldots, P_b\}^{\cap} = \bigcap_{i=1}^{b} P_i$*, is the polytope generated by* $\langle D, \mathbf{c} \rangle$*, where* $D$ *and* $\mathbf{c}$ *are the union of the templates and offsets of* $P_i$*, for* $i \in \{1, \ldots, b\}$.

Two parallelotope bundles $\{P_1, \ldots, P_b\}$ and $\{P'_1, \ldots, P'_{b'}\}$ are *equivalent* if they denote the same polytope. A bundle $\{P_1, \ldots, P_b\}$ allows us to symbolically represent a polytope $\{P_1, \ldots, P_b\}^{\cap}$ without requiring the explicit computation of the intersection of the parallelotopes $P_i$. Since we are interested in bundles as symbolic representations of polytopes, we can always replace a bundle with an equivalent one, whenever this is convenient.

Lemma 1 (Polytope Decomposition). *Let* $Q$ *be a polytope. There exists a finite set of parallelotopes* $\{P_1, \ldots, P_b\}$ *such that* $Q = \{P_1, \ldots, P_b\}^{\cap}$.

Lemma 2 (Decomposition Cardinality). $\lceil m/n \rceil$ *parallelotopes are sufficient to decompose a polytope* $Q$ *defined by* $m$ *constraints into a bundle.*

Lemma 1 states that any polytope can be represented by a parallelotope bundle while Lemma 2 fixes the maximum number of parallelotopes sufficient to decompose a polytope. Figure 1 shows a polytope $Q$ together with a possible decomposition, i.e., a bundle $\{P_1, P_2\}$ such that $\{P_1, P_2\}^{\cap} = Q$. Here $m = 3$ and $n = 2$, so $\lceil 3/2 \rceil = 2$ parallelotopes are sufficient to decompose $Q$ (in our case $P_1$ and $P_2$). In Section 5 we will describe an algorithm for decomposing a polytope into a bundle in view of accurate image approximation.

Definition 4 (Set Enclosure). *Let* $S \subset \mathbb{R}^n$ *be a compact set and* $Q = \langle D, \mathbf{c} \rangle \subset \mathbb{R}^n$ *be a polytope. The enclosure of* $S$ *with respect to* $Q$ *is defined as the polytope* $\odot(S, Q) = \langle D, \mathbf{c}' \rangle$*, where* $\mathbf{c}'_i = \max_{\mathbf{x} \in S} D_i \mathbf{x}$*, for* $i = 1, \ldots, k$.

The enclosure of $S$ with respect to $Q$ can be seen as tight over-approximation of $S$ obtained using the template of $Q$. The following properties of set enclosure can be easily proved.

Lemma 3. *Let* $S, S' \subset \mathbb{R}^n$ *be compact sets with* $S \subseteq S'$ *and* $Q = \langle D, \mathbf{c} \rangle$*,* $Q' = \langle D', \mathbf{c}' \rangle$ *be two polytopes such that* $D' \subseteq D$*. It holds that:*

*(1)* $S \subseteq \odot(S, Q)$*;*

*(2)* $\odot(S, Q) = \odot(S, \odot(S, Q)) = \odot(\odot(S, Q), Q)$*;*

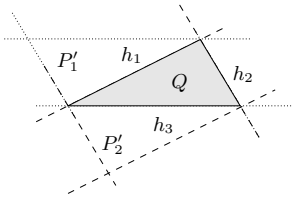



Figure 2: A polytope $Q$ and a its set bundle enclosure with respect to $\{P_1, P_2\}$ of Figure 1: $\boxdot(Q, \{P_1, P_2\}) = \{P_1', P_2'\}$.

*(3)* $\odot(S, Q) \subseteq \odot(S', Q)$;

*(4)* $\odot(S, Q) \subseteq \odot(S, Q')$.

The notion of set enclosure can be extended to bundles.

Definition 5 (Set Bundle Enclosure). *Let $S \subset \mathbb{R}^n$ be a compact set and $\{P_1, \dots, P_b\}$ be a bundle. The enclosure of $S$ with respect to the bundle $\{P_1, \dots, P_b\}$ is defined as $\boxdot(S, \{P_1, \dots, P_b\}) = \{P_1', \dots, P_b'\}$, where $P_i' = \odot(S, P_i)$, for $i = 1, \dots, b$.*

The set bundle enclosure surrounds $S$ with the parallelotopes $P_i$, producing a bundle whose parallelotopes $P_i'$ wrap $S$. The two operators are related by the following equality.

Lemma 4. $\boxdot(S, \{P_1, \dots, P_b\})^\cap = \odot(S, \{P_1, \dots, P_b\}^\cap)$.

The set bundle enclosure of $S$ with respect to the bundle $\{P_1, \dots, P_b\}$ coincides with the enclosure of $S$ with respect to the polytope $\{P_1, \dots, P_b\}^\cap$. As a consequence, we get that $\odot(\cdot, \cdot)$ and $\boxdot(\cdot, \cdot)$ are equivalent, with the difference that $\odot(\cdot, \cdot)$ returns a polytope, while $\boxdot(\cdot, \cdot)$ returns a bundle. Figure 2 shows the set bundle enclosure of the polytope $Q$ with respect to the bundle $\{P_1, P_2\}$ of Figure 1. The result of $\boxdot(S, \{P_1, P_2\})$ is the new bundle $\{P_1', P_2'\}$.

We say that $\{P_1', \dots, P_{b'}'\}$ is a *sub-bundle* of $\{P_1, \dots, P_b\}$ if $\{P_1', \dots, P_{b'}'\} \subseteq \{P_1, \dots, P_b\}$ and that two bundles are *strongly similar* if the set of normal vectors defining a parallelotope in one bundle is equal to the set of normal vectors defining a parallelotope in the other bundle. The following properties of the bundle enclosure operator immediately follow by definitions, Lemma 3, and Lemma 4.

Lemma 5. *Let $S, S' \subset \mathbb{R}^n$ be compact sets with $S \subseteq S'$, $\{P_1, \dots, P_b\}$ be a bundle, $\{P_1', \dots, P_{b'}'\}$ be one of its sub-bundles, and $\{P_1'', \dots, P_b''\}$ be a bundle strongly similar to $\{P_1, \dots, P_b\}$. It holds that:*

*(1)* $S \subseteq \boxdot(S, \{P_1, \dots, P_b\})^\cap$;

*(2)* $\boxdot(S, \{P_1, \dots, P_b\}) = \boxdot(S, \boxdot(S, \{P_1, \dots, P_b\})) = \boxdot\left(\boxdot(S, \{P_1, \dots, P_b\}), \{P_1, \dots, P_b\}\right)$;

*(3)* $\boxdot(S, \{P_1, \dots, P_b\})^\cap \subseteq \boxdot(S', \{P_1, \dots, P_b\})^\cap$;

*(4)* $\boxdot(S, \{P_1', \dots, P_{b'}'\}) \subseteq \boxdot(S, \{P_1, \dots, P_b\})$ *and* $\boxdot(S, \{P_1, \dots, P_b\})^\cap \subseteq \boxdot(S, \{P_1', \dots, P_{b'}'\})^\cap$;

*(5)* $\boxdot(S, \{P_1, \dots, P_b\})$ *is strongly similar to* $\{P_1, \dots, P_b\}$;

*(6)* $\boxdot(S, \{P_1, \dots, P_b\}) = \boxdot(S, \{P_1'', \dots, P_b''\})$.

A bundle representing a polytope may not be "minimal" in the sense that one or more paralleloptopes can be shrunk while the resulting bundle still represents the same polytope. The shrinking process removes parts of parallelotopes that are not in the polytope and it is useful for many operations, in particular image over-approximation. As we will see later, the shrinking reduces the error when the image over-approximation is performed on shrunk parallelotopes. A bundle that remains unchanged after a shirking is said to be in *canonical form*.

Definition 6 (Bundle Canonical Form). *A bundle $\{P_1, \dots, P_b\}$ is in* canonical form *if and only if:*

$$\boxdot(\{P_1, \dots, P_b\}^\cap, \{P_1, \dots, P_b\}) = \{P_1, \dots, P_b\}.$$

Intuitively, a bundle $\{P_1, \dots, P_b\}$ is in canonical form if the enclosure of its symbolic polytope $P^\cap = \{P_1, \dots, P_b\}^\cap$ with respect to $\{P_1, \dots, P_b\}$ does not affect the parallelotopes $P_i$, for $i \in \{1, \dots, b\}$. The canonical form of a bundle $\{P_1, \dots, P_b\}$ can be obtained by enclosing its polytope $P^\cap$ with respect to its parallelotopes $P_i$. The bundle $\{P_1', P_2'\}$ of Figure 2 is in canonical form, since it is the result of the bundle enclosure $\boxdot(Q, \{P_1, P_2\}) = \{P_1', P_2'\}$ where $Q = \{P_1, P_2\}^\cap$. In virtue of Lemma 5 item *(2)*, the result of a bundle enclosure is always in canonical form. In other terms, the operator $\boxdot(\cdot, \cdot)$ can be exploited for canonizing bundles, as stated by the following result.

Lemma 6 (Canonization). *Let $\{P_1, \dots, P_b\}$ be a bundle. The bundle $\boxdot(\{P_1, \dots, P_b\}^\cap, \{P_1, \dots, P_b\})$ is in canonical form and it is equivalent to $\{P_1, \dots, P_b\}$.*

A bundle in canonical form is a "minimal" representation of the polytope with respect to a given set of parallelotope directions, since all the offsets are shifted towards the constraints of the polytope. The advantage of dealing with bundles in canonical form will become clearer on images approximation.

In the following we show the advantage of bundles in image approximation. We start by proving some inclusions that hold on the images of bundles by a continuous function. Note that these properties hold for all continuous functions, and in the case of polynomials, they are particularly useful for our image approximation problem, because we can indeed effectively enclose the image of a parallelotope.

Lemma 7 (Bundle Image). *Let $\{P_1, \dots, P_b\}$ be a bundle with $P^\cap = \{P_1, \dots, P_b\}^\cap$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function. The following inclusions hold:*

$$\mathbf{f}(P^\cap) \subseteq \boxdot(\mathbf{f}(P^\cap), \{P_1, \dots, P_b\})^\cap \subseteq \quad (7.1)$$
$$\subseteq \textstyle\bigcap_{i=1}^b \boxdot(\mathbf{f}(P_i), \{P_1, \dots, P_b\})^\cap \subseteq \quad (7.2)$$
$$\subseteq \textstyle\bigcap_{i=1}^b \boxdot(\mathbf{f}(P_i), \{P_i\})^\cap \quad (7.3)$$

Figure 3 shows the intersection of the enclosures of $\mathbf{f}(P_1)$ and $\mathbf{f}(P_2)$ with respect to $\{P_1, P_2\}$; Figure 4 shows the enclosure of $\mathbf{f}(P_1)$ with respect to $P_1$ intersected with the enclosure of $\mathbf{f}(P_2)$ with respect to $P_2$. Note how the over-approximation of the first method is tighter than the second one.

Intuitively Lemma 7 suggests us two possible ways of approximating the image of a polytope $P^\cap$ represented by a bundle $\{P_1, \dots, P_b\}$. In case (7.2), we can over-approximate $\mathbf{f}(P^\cap)$ with the bundle enclosures of the transformed parallelotopes $\mathbf{f}(P_i)$ with respect to $\{P_1, \dots, P_b\}$. In case (7.3), we can consider the set enclosures of each parallelotope image $\mathbf{f}(P_i)$ with respect to its original directions. In both cases we only need to be able to compute the images of the parallelotopes and their enclosures.

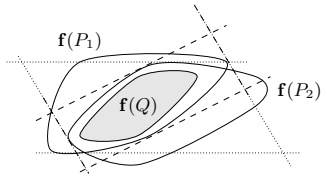



Figure 3: $\boxdot(\mathbf{f}(P_1), \{P_1, P_2\})^{\cap} \cap \boxdot(\mathbf{f}(P_2), \{P_1, P_2\})^{\cap}$.

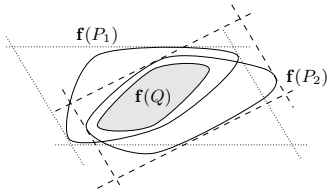



Figure 4: $\odot(\mathbf{f}(P_1), P_1) \cap \odot(\mathbf{f}(P_2), P_2)$.

Notice however that the bundle $\{P_1, \ldots, P_b\}$ may not be in canonical form. The following result shows that if we approximate the image of $\{P_1, \ldots, P_b\}^{\cap}$ exploiting Lemma 7 on the canonical bundle $\boxdot(\{P_1, \ldots, P_b\}^{\cap}, \{P_1, \ldots, P_b\})$ we get tighter approximations.

THEOREM 1 (CANONICAL BUNDLE IMAGE). *Let us consider a bundle $\{P_1, \ldots, P_b\}$, with $P^{\cap} = \{P_1, \ldots, P_b\}^{\cap}$, and a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$. Let also $\{P'_1, \ldots, P'_b\} = \boxdot(\{P_1, \ldots, P_b\}^{\cap}, \{P_1, \ldots, P_b\})$. The following relations hold among the over-approximations of $\mathbf{f}(P^{\cap})$:*

$$\boxdot(\mathbf{f}(P^{\cap}), \{P'_1, \ldots, P'_b\})^{\cap} = \boxdot(\mathbf{f}(P^{\cap}), \{P_1, \ldots, P_b\})^{\cap} \subseteq$$
$$\textstyle\bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P'_i), \{P'_1, \ldots, P'_b\})^{\cap} \subseteq \bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P_i), \{P_1, \ldots, P_b\})^{\cap}$$
$$\textstyle\bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P'_i), \{P'_i\})^{\cap} \subseteq \quad \bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P_i), \{P_i\})^{\cap}$$

As a consequence, having to compute the image of a generic compact set $S$, one can first over approximate $S$ through the bundle enclosure operator, which returns a bundle in canonical form, and then exploit the above results to over-approximate the image of $S$.

# 4. BUNDLE DATA STRUCTURE

In this section we define a data structure and some methods for the compact representation and transformation of bundles in canonical form.

A parallelotope bundle in canonical form can be compactly represented with the tuple $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T \rangle$ where:

- $L \in \mathbb{R}^{k \times n}$ is the *directions matrix* that contains the directions used to build the parallelotopes. The $i$-th row $L_i$ of $L$ represents a direction;
- $\overline{\mathbf{d}} \in \mathbb{R}^k$ is the *upper offsets vector*. The $i$-th element of $\overline{\mathbf{d}}$, associated with the $i$-th direction $L_i$, constitutes the half-space $L_i\mathbf{x} \leq \overline{\mathbf{d}}_i$;
- $\underline{\mathbf{d}} \in \mathbb{R}^k$ is the *lower offsets vector*. The $i$-th element of $\underline{\mathbf{d}}$, associated with the $i$-th direction $L_i$, constitutes the half-space $-L_i\mathbf{x} \leq \underline{\mathbf{d}}_i$;
- $T \in \{1, \ldots, k\}^{b \times n}$ is the *templates* matrix that represents the set of the parallelotope templates. Each element in $T$ is a reference to a direction in $L$ and offsets in $\overline{\mathbf{d}}$ and $\underline{\mathbf{d}}$. A row in $T$ constitutes a set of half-spaces that generates a parallelotopes.

Consider for instance the bundle $\{P'_1, P'_2\}$ in canonical form of Figure 2 where $P'_1 = \langle \Lambda'_1, \mathbf{c}'_1 \rangle$ and $P'_2 = \langle \Lambda'_2, \mathbf{c}'_2 \rangle$ with:

$$\Lambda'_1 = \begin{pmatrix} 1.6 & 1 \\ 0 & 1 \\ -1.6 & -1 \\ 0 & -1 \end{pmatrix} \mathbf{c}'_1 = \begin{pmatrix} 10 \\ 3.1 \\ -1 \\ -1 \end{pmatrix} \Lambda'_2 = \begin{pmatrix} 1.6 & 1 \\ -0.5 & 1 \\ -1.6 & -1 \\ 0.5 & -1 \end{pmatrix} \mathbf{c}'_2 = \begin{pmatrix} 10 \\ 1 \\ -1 \\ 1.7 \end{pmatrix}$$

This bundle can be represented by the tuple $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T \rangle$ where:

$$L = \begin{pmatrix} 1.6 & 1 \\ 0 & 1 \\ -0.5 & 1 \end{pmatrix} \overline{\mathbf{d}} = \begin{pmatrix} 10 \\ 3.1 \\ 1 \end{pmatrix} \underline{\mathbf{d}} = \begin{pmatrix} -1 \\ -1 \\ 1.7 \end{pmatrix} T = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}.$$

With this representation we avoid the storage of redundant directions shared by different parallelotopes. In doing so, a single operation on an entry in the tuple, indirectly affects several parallelotopes in the bundle. Moreover, for each parallelotope we store only $n$ directions against $2n$ constraints, since we know that parallel constraints can be obtained by reversing the signs of the normal vectors. Note that each direction $L_i$ is associated with a unique upper and lower offset $\overline{\mathbf{d}}_i$ and $\underline{\mathbf{d}}_i$. This means that if two parallelotopes share a direction, the constraints defined by this direction coincide in the two parallelotopes. Hence, this data structure does not allow us to represent all the possible bundles (for instance the one shown in Figure 1), but it is expressive enough to capture all the canonical bundles (like the one of Figure 2).

We now show how the operations presented in Section 3 can be defined on our data structure $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T \rangle$. We begin with the decomposition of a polytope (see Definition 3).

METHOD 1 (POLYTOPE DECOMPOSITION). *Let $Q \subset \mathbb{R}^n$ be a polytope defined by $m$ constraints. Let $L \in \mathbb{R}^{k \times n}$ be a matrix containing all the versors of $Q$ without repetitions, i.e., the elements of $L$ are pairwise linearly independent. To generate the $i$-th decomposing parallelotope, it is sufficient to pick $n$ directions $L_{j_1}, \ldots, L_{j_n}$ from $L$ and store their indices in the template matrix $T_i = (j_1, \ldots, j_n)$. By Lemmas 1 and 2, we have to generate at most $\lceil m/n \rceil$ parallelotopes such that the union of the picked directions is a cover of the constraints of $Q$. Finally, the offset vectors $\overline{\mathbf{d}}, \underline{\mathbf{d}} \in \mathbb{R}^k$ can be obtained by enclosing $Q$ with respect to the constructed parallelotopes as described in Method 2.*

We now show how to compute the set bundle enclosure.

METHOD 2 (SET BUNDLE ENCLOSURE $\boxdot(\cdot, \cdot)$). *The enclosure of a bounded set $S \subset \mathbb{R}^n$ with respect to a canonical bundle $\{P_1, \ldots, P_b\}$ stored as $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T \rangle$ can be obtained by updating the upper and lower offset vector as $\overline{\mathbf{d}}_i = \max_{\mathbf{x} \in S} L_i\mathbf{x}$ and $\underline{\mathbf{d}}_i = \max_{\mathbf{x} \in S} -L_i\mathbf{x}$, for $i = 1, \ldots, k$.*

The described methods work only on canonical bundles and return a compact representation of a canonical bundle. The enclosure of a polytope with respect to a bundle requires the resolution of $2k$ linear programs. Thus, the canonization of a bundle can be done by solving a series of linear programs where only the offsets of the constraints that do not participate to the intersection are modified.

The transformation of a bundle through a continuous function can be rather difficult, depending on the transforming function. If the function is linear, it is possible to exactly compute the image of each parallelotope and then obtain

the exact bundle transformation. Things are more complex when the function is nonlinear and the geometric properties of the parallelotopes are not preserved. We will now describe two methods based on Theorem 1, requiring only computations on single parallelotopes (besides being able to implement Method 2). As stated by Lemma 7 item *(3)*, an over-approximation of a bundle transformation $\mathbf{f}(P^{\cap})$, with $P^{\cap} = \{P_1, \ldots, P_b\}$, can be obtained by enclosing each image $\mathbf{f}(P_i)$ with $P'_i = \boxdot(\mathbf{f}(P_i), \{P_i\})^{\cap}$ and then considering the intersection $\bigcap_{i=1}^{b} P'_i$ (see (7.3)). We call this approximation *one-for-one (OFO)*, since each parallelotope in the bundle is independently approximated.

Method 3 (One-for-One Image (OFO)). *The* one-for-one approximation *of the bundle* $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T\rangle$ *can be obtained by retrieving each parallelotope* $P_i$*, computing the enclosures* $P'_i = \odot(\mathbf{f}(P_i), P_i)$*, and then computing the canonization of* $\{P'_1, \ldots, P'_b\}^{\cap} = P'^{\cap}$*, that is* $\boxdot(P'^{\cap}, \{P_1, \ldots, P_b\})$.

The polytope provided by the OFO method corresponds to the polytope $\bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P_i), \{P_i\})^{\cap}$ of Lemma 7 item *(3)*.

In order to obtain a finer over-approximation, it is possible to change the template in the approximation process, i.e., we can fix a new template to enclose $\mathbf{f}(P_i)$. As suggested by Lemma 7 item *(2)*, we can exploit all the directions of the bundle, i.e., instead of looking for a new template for each parallelotope, we can bound each set $\mathbf{f}(P_i)$ with all the directions of the transformed bundle. We call this approximation *all-for-one (AFO)* since all the directions of the bundle are used to approximate the image of a single parallelotope.

Method 4 (All-for-One Image (AFO)). *The* all-for-one *approximation of the bundle* $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T\rangle$ *can be obtained by retrieving each parallelotope* $P_i$*, computing the set bundle enclosures* $\{P'_{i\,1}, \ldots, P'_{i\,b}\} = \boxdot(\mathbf{f}(P_i), \{P_1, \ldots, P_b\})$*, for* $i = 1, \ldots, b$*, and enclosing the polytope* $\bigcap_{i=1}^{b} \{P'_{i\,1}, \ldots, P'_{i\,b}\}^{\cap}$ *with respect to the transformed bundle, i.e., computing the bundle enclosure* $\boxdot(\bigcap_{i=1}^{b} \{P'_{i\,1}, \ldots, P'_{i\,b}\}^{\cap}, \{P_1, \ldots, P_b\})$.

The AFO transformation produces a bundle whose polytope corresponds to $\bigcap_{i=1}^{b} \boxdot(\mathbf{f}(P_i), \{P_1, \ldots, P_b\})^{\cap}$ of Lemma 7 item *(2)*. By Lemma 7, the AFO approximation is finer than the of OFO one. Clearly the precision has a cost: the OFO method requires $b(2n) + k$ optimizations against the $b(2k) + k$ optimizations of the AFO approach (recall that $k \geq n$).

Both the approximation methods are based on a series of enclosures. The offsets of the constraints necessary to obtain the enclosures can be attained by solving optimization problems of the form $\overline{\mathbf{d}}_j = \max_{\mathbf{x} \in \mathbf{f}(P_i)} L_j \mathbf{x}$. If the transformation function $\mathbf{f}$ is nonlinear, these optimization problems might be computationally expensive. In the next section we expose a method, based on the Bernstein coefficients, to efficiently deal with optimizations of polynomial functions.

# 5. POLYNOMIAL REACHABILITY

In this section we recall a technique to over-approximate the image of a parallelotope in the case of polynomial functions. The technique is based on Bernstein coefficients and has been described in our previous works [13, 14, 12]. Together with the methods defined in the previous section, this gives us an algorithm for the reachability computation of discrete-time polynomial dynamical systems.

A discrete-time polynomial dynamical system can be described by difference equations as follows:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k) \\ \mathbf{x}_0 &\in X_0 \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of state variables and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a vector of $n$ multi-variate polynomials of the form $\mathbf{f}_i : \mathbb{R}^n \to \mathbb{R}$, for each $i \in \{1, \ldots, n\}$. The set $X_0 \subseteq \mathbb{R}^n$ is called the *initial set*.

Given an initial set $X_0$, we are interested in computing the bounded time *reachable set* of the dynamical system, i.e, the set of states visited by the dynamical system up to a fixed time horizon $K \in \mathbb{N}$. The reachable set can be obtained as the solution of the recursion $X_{i+1} = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in X_i\}$, for $i = 0, \ldots, K$. In this approach, the computation of the reachable set can be reduced to a series of images of sets by the polynomial $\mathbf{f}$. This means that if we represent with a bundle the set $X_i$, we can reduce the single step evolution $X_{i+1} = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in X_i\}$ to a bundle transformation.

Algorithm 1 shows our reachability algorithm based on parallelotope bundles. For brevity, the bundle $\{P_1, \ldots, P_b\}$ computed at the $i$-th iteration is abbreviated by $\mathcal{B}_i$.

**Algorithm 1** Bundle Reachability

```
1: function REACH(X_0)                          ▷ X_0 polytope
2:     B_0 ←DECOMPOSE(X_0)                      ▷ Init. bundle
3:     for i = 1, ..., K do
4:         B_i ←TRANSFORM(f, B_{i-1})
5:         B_i ←DECOMPOSE(B_i)                  ▷ Optional
6:     end for
7: end function
```

The algorithm receives in input a polytope $X_0$ that is decomposed into the bundle $\mathcal{B}_0$ (Line 2). Then, it enters in a loop where at each iteration it over-approximates the set of states reachable at time $i$ through the transformation of the bundle $\mathcal{B}_{i-1}$ with respect to the dynamics $\mathbf{f}$ (Line 4). The transformation performed by the function Transform can be either the OFO (see Method 3) or the AFO (see Method 4). In both cases, the transformation produces a bundle $\mathcal{B}_i$ in canonical form that over-approximates the states reachable by the dynamical system from $\mathcal{B}_{i-1}$. Finally, the symbolic polytope of the computed bundle $\mathcal{B}_i$ can be decomposed (Line 5), obtaining a new bundle whose parallelotopes combine the directions differently from $\mathcal{B}_{i-1}$. The decomposition is optional, but it might improve the precision in the over-approximation of the future transformations, since the over-approximating parallelotopes might be smaller than the ones produced by the transformation. In the following we will discuss in detail the functions Transform and Decompose. We begin with the polynomial transformation since, as we will discover later, the decomposition is strictly related to the way we transform the bundles.

*Transformation.* The operation at the basis of the transformation of a parallelotope $P$ is the nonlinear optimization problem of the form $\mathbf{c}'_i = \max_{\mathbf{x} \in \mathbf{f}(P)} \Lambda_i \mathbf{x}$, or equivalently, $\mathbf{c}'_i = \max_{\mathbf{x} \in P} \Lambda_i \mathbf{f}(\mathbf{x})$, where $\Lambda_i \in \mathbb{R}^n$ is a generic direction. Solving this problem, or finding a tight upper-bound of $\mathbf{c}'_i$, means being able to find the offset of a constraint with normal vector $\Lambda_i$ tangent or close to the set $\mathbf{f}(P)$. In our previous works [14, 12], inspired by [13], we developed

a method to bound polynomials over boxes and parallelotopes. The method, based on a particular property of the Bernstein coefficients, is summarized in the following.

A polynomial $\pi : \mathbb{R}^n \to \mathbb{R}$ can be expressed in the common power basis as $\pi(\mathbf{x}) = \sum_{\mathbf{i}\in I} \mathbf{a_i}\mathbf{x}^{\mathbf{i}}$, where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^n$ is a vector or variables, $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_n) \in \mathbb{N}^n$ is a multi-index, $I$ is the multi-index set of $\pi$, and $\mathbf{a_i} \in \mathbb{R}$ are the polynomial coefficients. In the following, we write $\mathbf{d}/\mathbf{i}$ for $(\mathbf{d}_1/\mathbf{i}_1, \ldots, \mathbf{d}_n/\mathbf{i}_n)$ and $\binom{\mathbf{d}}{\mathbf{i}}$ for the product $\binom{\mathbf{d}_1}{\mathbf{i}_1} \ldots \binom{\mathbf{d}_n}{\mathbf{i}_n}$. The same polynomial $\pi$ can be represented using the Bernstein basis as $\pi(\mathbf{x}) = \sum_{\mathbf{i}\in I} \mathbf{b_i}\mathcal{B}_{(\mathbf{d},\mathbf{i})}(\mathbf{x})$, where $\mathbf{d} \in \mathbb{N}^n$ is the degree of $\pi$, i.e., the multi-index that dominates the multi-indices in $I$, $\mathbf{b_i} = \sum_{\mathbf{j}\le\mathbf{i}} \binom{\mathbf{i}}{\mathbf{j}}/\binom{\mathbf{d}}{\mathbf{j}}\mathbf{a_j}$ are the *Bernstein coefficients*, and $\mathcal{B}_{(\mathbf{d},\mathbf{i})}(\mathbf{x}) = \beta_{(\mathbf{d}_1,\mathbf{i}_1)}(\mathbf{x}_1) \ldots \beta_{(\mathbf{d}_n,\mathbf{i}_n)}(\mathbf{x}_n)$ is the $\mathbf{i}$-th *Bernstein basis* where $\beta_{(\mathbf{d}_j,\mathbf{i}_j)}(x) = \binom{\mathbf{d}_j}{\mathbf{i}_j} x^{\mathbf{i}_j}(1-x)^{\mathbf{d}_j-\mathbf{i}_j}$. The points $(\mathbf{i}/\mathbf{d}, \mathbf{b_i}) \in \mathbb{R}^{n+1}$ are called *Bernstein control points*.

Bernstein coefficients own the useful *range enclosing property* stating that for all the $\mathbf{x} \in [0,1]^n$, $\min_{\mathbf{i}\in I} \mathbf{b}_i \le \pi(\mathbf{x}) \le \max_{\mathbf{i}\in I} \mathbf{b}_i$. This means that the image of the unit box $\pi([0,1]^n)$ is bounded by the minimum and maximum Bernstein coefficients. Hence, if we want to bound a polynomial over the unit box, instead of solving a nonlinear optimization problem, we can compute the Bernstein coefficients $\mathbf{b_i}$ and take their maximum and minimum. The following lemma [13] bounds the distance between a polynomial and its Bernstein control points, or in other words, the error between the maximum and minimum of a polynomial and the bounds provided by its Bernstein coefficients. This lemma provides us a criterion for decomposing a bundle and for dividing sets in order to achieve better precision.

LEMMA 8. *[13] Let $C_\pi : \mathbb{R}^n \to \mathbb{R}$ be the piecewise linear function defined by the Bernstein control points of the polynomial $\pi : \mathbb{R}^n \to \mathbb{R}$, with respect to the box $[0,1]^n$. For all $\mathbf{x} \in [0,1]^n$*

$$| \pi(\mathbf{x}) - C_\pi(\mathbf{x}) | \le \max_{\mathbf{x}\in[0,1]^n; i,j\in\{1,\ldots,n\}} | \partial_i\partial_j\pi(\mathbf{x}) | \qquad (2)$$

*where $|\cdot|$ is the infinity norm on $\mathbb{R}^n$.*

Several convergent subdivision procedures for reducing the gap between bounds and optimums have been proposed [16, 25]. Note that the range enclosing property works only on the unit-box domain. If we want to apply this property to a generic box or a parallelotope $X \subset \mathbb{R}^n$, we can define a linear transformation $\mathbf{v} : [0,1]^n \to X$ that maps the unit box to $X$, and exploit the Bernstein range enclosing property on the function $\pi(\mathbf{v}(\mathbf{x})) : [0,1]^n \to \mathbb{R}$. With this technique we can define a procedure BOUND($\pi$,$X$) that receives in input a polynomial $\pi$ and a box or parallelotope $X$, computes the linear transformation $\mathbf{v} : [0,1]^n \to X$, composes $\pi$ with $\mathbf{v}$, computes the Bernstein coefficients of $\pi(\mathbf{v}(\mathbf{x}))$, and returns the maximum Bernstein coefficients $b^* \in \mathbb{R}$. By the range enclosing property, $b^*$ is such that $b^* \ge \max_{\mathbf{x}\in X} \pi(\mathbf{x})$.

The function BOUND can be used to bound our optimization problem and compute the transformation of a bundle: given a parallelotope $P$, the determination of $\mathbf{c}'_i$ for a direction $\Lambda_i$ such that $\Lambda_i\mathbf{x} \le \mathbf{c}'_i \supseteq \mathbf{f}(P)$, can be obtained with the procedure $\mathbf{c}'_i = \text{BOUND}(\Lambda_i\mathbf{f}(\mathbf{x}), P)$.

We now take advantage of the function BOUND to define our bundle transformation methods. The OFO transformation of a bundle $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T\rangle$, as exposed in Method 3, can be obtained by retrieving each parallelotope $P_i = \langle\Lambda, \mathbf{c}\rangle$, for $i = 1, \ldots, b$, computing the new offsets $\mathbf{c}'_j = \text{BOUND}(\Lambda_j\mathbf{f}(\mathbf{x}), P_i)$, for $j = 1, \ldots, 2n$, and defining the over-approximating parallelotope $P'_i = \langle\Lambda, \mathbf{c}'\rangle \supseteq \mathbf{f}(P_i)$. Finally, the canonization of the transformed bundle $\{P'_1, \ldots, P'_b\}$ can be obtained by solving a family of linear programs of the form $\max_{\mathbf{x}\in P'^{\cap}} \Lambda_i\mathbf{x}$, where $\Lambda_i$ belongs to the template matrices of $P'_j$ and $P'^{\cap} = \{P'_1, \ldots, P'_b\}^{\cap}$ is the polytope of the computed bundle.

The AFO transformation of a bundle $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T\rangle$, as defined in Method 4, can be done as follows. For each parallelotope of the bundle $P_i$, for $i = 1, \ldots, b$, we have to compute the enclosure $\{P'_{i\,1}, \ldots, P'_{i\,b}\} = \boxdot(\mathbf{f}(P_i), \{P_1, \ldots, P_b\})$. An over-approximation of $P'_{i_m}$, with $m \in 1, \ldots, b$, is the parallelotope $\langle\Lambda, \mathbf{c}'\rangle$ where $\Lambda$ is the template of $P_{i_m}$ and $\mathbf{c}'_j = \text{BOUND}(\Lambda_j\mathbf{f}(\mathbf{x}), P_{i_m})$, for all $j = 1, \ldots, 2n$. Finally, the canonization $\boxdot(\bigcap_{i=1}^b \{P'_{i\,1}, \ldots, P'_{i\,b}\}^{\cap}, \{P_1, \ldots, P_b\})$ can be computed by solving a group of linear programs of the form $\max_{\mathbf{x}\in P'^{\cap}} \Lambda_j\mathbf{x}$, where $\Lambda_j$ belongs to the template matrices of $P'_{i_m}$ and $P'^{\cap} = \bigcap_{i=1}^b \{P'_{i\,1}, \ldots, P'_{i\,b}\}^{\cap}$ is the intersection of the polytopes represented by the computed bundles.

***Decomposition.*** Since in our reachability algorithm we may be interested in decomposing a polytope in a bundle (see Algorithm 1), we define a function DECOMPOSE that receives in input a bundle in canonical form $\langle L, \overline{\mathbf{d}}, \underline{\mathbf{d}}, T\rangle$ (whose polytope $P^{\cap}$ has to be decomposed) and reorganizes the templates matrix $T$ creating a new collection of parallelotopes around the polytope $P^{\cap}$. The goal of the decomposition is to create a set of small parallelotopes whose intersection is $P^{\cap}$. There are two reasons why we want small parallelotopes:

1. smaller parallelotopes $P_i$ lead to a smaller bundle image $\{\mathbf{f}(P_1), \ldots, \mathbf{f}(P_d)\}$ and then to a more accurate over-approximation $\mathbf{f}(P^{\cap})$ (see, e.g., Theorem 1);
2. the shorter the largest side length of $P_i$, the more accurate the over-approximation introduced by the Bernstein coefficients (see Lemma 8).

Hence, the aspects to take into account in the construction of the parallelotopes are the volume and the maximum side length. Moreover, we do not have to forget that the set of the parallelotope directions must cover the directions of polytope to be decomposed (see Definition 3). Finding the best decomposition in terms of volume and maximum length minimization is computationally expensive and might not be possible (recall that the set cover problem is NP-hard).

In order to efficiently find a good decomposition, we propose a heuristic technique that constructs the parallelotopes while trying to minimize the volumes and maximum side lengths. The proposed heuristic starts from a decomposition, applies a series of random changes to the templates matrix, and keeps only the best one accordingly to an evaluation function that we will soon define. The procedure is repeated until a fixed number of iterations is reached.

Given a bundle $P^{\cap} = \{P_1, \ldots, P_b\}$, the evaluation function should take into account the volumes and side lengths of the parallelotopes $P_i$, for $i \in \{1, \ldots, b\}$. The exact computation of the volume of a parallelotope is rather expensive, since it is equal to the determinant of a $n \times n$ matrix. To lighten the computation, we approximate the volume of $P = \langle\Lambda, \mathbf{c}\rangle$ with the product of the distances of its constraints:

$$\tilde{v}(P) = \prod_{i=1}^{n} \delta(\Lambda_i\mathbf{x} \le \mathbf{d}_i, \Lambda_{i+n}\mathbf{x} \le \mathbf{d}_{i+n}) \qquad (3)$$

where $\delta(\Lambda_i \mathbf{x} \leq \mathbf{d}_i, \Lambda_{i+n} \mathbf{x} \leq \mathbf{d}_{i+n}) = |\mathbf{d}_i - \mathbf{d}_{i+n}|/\|\Lambda_i\|$ and $\|\cdot\|$ is the Euclidean norm.

The computation of the side lengths of a parallelotope passes inevitably through the determination of its vertices, an operation that can be computationally expensive. Instead of calculating the exact lengths, we opt for a faster heuristic that guesses the lengths of a parallelotope from the angles of the directions of its constraints. Intuitively, in the two-dimensional case, having fixed two parallel lines, the lengths of the edges not lying on the two fixed lines are minimal when the added directions and the fixed ones are orthogonal. Thus, we define the notion of *orthogonal proximity* $\theta(\Lambda_i, \Lambda_j) = \widehat{\Lambda_i, \Lambda_j} \pmod{\pi/2}$, where $\widehat{\Lambda_i, \Lambda_j}$ is the angle between $\Lambda_i$ and $\Lambda_j$, i.e., $\widehat{\Lambda_i, \Lambda_j} = \arccos \frac{\Lambda_i \Lambda_j}{\|\Lambda_i\| \|\Lambda_j\|}$. The orthogonal proximity of a parallelotope $P = \langle \Lambda, \mathbf{c} \rangle$ is defined as:

$$\Theta(P) = \max_{i,j \in \{1,\ldots,2n\}} \theta(\Lambda_i, \Lambda_j). \tag{4}$$

Exploiting the notions of approximated volume $\tilde{v}$ and orthogonal proximity $\Theta$, we define the evaluation function $w$ for a bundle as:

$$w(\{P_1, \ldots, P_b\}) = \max_{i \in \{1,\ldots,b\}} \alpha \tilde{v}(P_i) + (1 - \alpha)\Theta(P_i) \tag{5}$$

where $\alpha \in [0, 1]$ is a tunable parameter.

# 6. EXPERIMENTAL RESULTS

We implemented a C++ tool called *Sapo* that manipulates parallelotope bundles and computes the reachable set of polynomial dynamical systems. The tool is structured in three main blocks: the bundle handler that stores and works with bundles; the base converter that computes the Bernstein coefficients of polynomials (the coefficients are symbolically calculated using our improved matrix method [14]); the reachability layer, where a dynamical system is specified and its reachable set is computed. Our tool relies on the libraries GiNaC[1] for the symbolic manipulation of formulas and GLPK (GNU Linear Programming Kit)[2] for the resolution of linear programs.

In the following we present four case studies. All the experiments were carried out on a laptop computer Intel Core(TM) Duo (2.40 GHz, 4GB RAM) running Ubuntu 12.04. The tool and the full descriptions of the case study configurations can be found at the link https://github.com/tommasodreossi/sapo.

***Test Model.*** Our first experiment is based on the following illustrative 2-dimensional dynamical system:

$$\begin{aligned} x_{k+1} &= x_k + (0.5x_k^2 - 0.5y_k^2)\Delta \\ y_{k+1} &= y_k + (2x_k y_k)\Delta \end{aligned} \tag{6}$$

The directions constituting the bundles are $L_0 = (1, 0), L_1 = (0, 1), L_2 = (-1, 1)$, $L_3 = (1, 1)$, the initial set is a box with $x \in [0.05, 0.1]$ and $y \in [0.99, 1.00]$, and $\Delta = 0.01$. Figure 5 shows the reachable sets computed with the different techniques plotted over time up to 25 steps. Figure 5a shows the sets computed using the OFO and AFO transformations (in white and gray, respectively), without the bundle decomposition. In both cases the bundle is composed by two parallelotopes obtained by coupling $L_0$ with $L_1$ and $L_2$ with $L_3$, respectively. The picture shows that the AFO transformation is finer than the OFO one. The OFO computation took 0.14s, the AFO 0.21s. Figure 5b compares the sets computed using the AFO transformation with (in black) and without (in gray) the bundle decomposition. In the decomposition function, the parameter $\alpha$ is equal to 0.5 and the number of decompositions randomly generated at each step is 500. The computation without decomposition took 0.22s against 1.94s of the one with decomposition. Note how the black flow is always included in the gray one, meaning that decomposition, applied with the AFO transformation, is finer than non-decomposed AFO and OFO transformations. Finally, Figure 5c depicts the AFO transformation with decomposition for $\alpha = 0$ (in gray) and $\alpha = 1$ (in white), computed in 1.92s and 1.95s (also here 500 decompositions are generated at each step). This experimental evaluation shows how the parameter $\alpha$ affects the reachable set computation. In this case, it is difficult to establish which is the best technique, since there is not a strict inclusion. However, the areas of the sets computed with $\alpha = 0$ are smaller than the ones with $\alpha = 1$.

***SIR Epidemic Model.*** The second case study we consider is a 3-dimensional dynamical system that shows the benefits of using multiple directions and parallelotopes. We study the classic SIR epidemic model, where a population of individuals is divided in three compartments: $s$, the healthy individuals but *susceptible* to the disease; $i$, the *infected* individuals; $r$ the individuals *removed* from the system (e.g., recovered). Two parameters regulate the evolution of the system variables: $\beta$, the contraction rate and $\gamma$, where $1/\gamma$ is the mean infective period. $\Delta$ is the discretization step. The dynamics of the SIR model are the following:

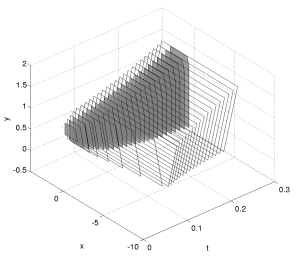
$$\begin{aligned} s_{k+1} &= s_k - (\beta s_k i_k)\Delta \\ i_{k+1} &= i_k + (\beta s_k i_k - \gamma i_k)\Delta \\ r_{k+1} &= r_k + (\gamma i_k)\Delta \end{aligned} \tag{7}$$

For this case we applied the AFO transformation without bundle decomposition. First, we computed the reachable set using a single axis-aligned template. Then, we added 5 directions not aligned with the axis and grouped them in 4 different templates. In both cases we computed the reachable sets up to 60 steps. Figure 6 shows the computed results, i.e., the single template computation (in white) and the four templates one (in black). In both cases the population is normalized and the initial set is the box with $s \in [0.79, 0.80]$, $i \in [0.19, 0.20]$, and $r = 0.00$. The chosen parameter values are $\beta = 0.34$, $\gamma = 0.05$, and $\Delta = 0.5$. The single parallelotope computation required 0.05s against the 1.04s of the 4 parallelotopes one. From the figure we can observe that multiple templates lead to a much finer result: the black flow is always included in the white one.

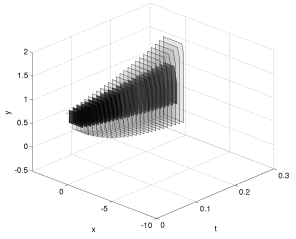
***Honeybees Site Choice.*** In our third case study, we analyze a model that describes the decision-making process mechanism adopted by a swarm of honeybees to choose one among two different nest-sites. In this model [7], a population of honeybees is divided in five groups: $x$, the neutral bees that have not chosen a site; $y_1$ and $y_2$, evangelic bees dancing for the first and second site, respectively; $z_1$ and $z_2$, non-evangelic bees that have been converted to the first

[1] http://www.ginac.de/

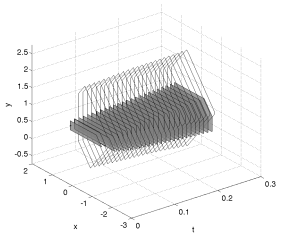
[2] http://www.gnu.org/software/glpk/glpk.html



(a) OFO (white, 0.13s) and AFO (gray, 0.24s) transformations.



(b) AFO transformation without (gray, 0.24s) and with (black, 0.97s) decomposition ($\alpha = 0.5$).



(c) AFO transformation with decomposition. $\alpha = 0$ (gray, 0.95s) and $\alpha = 1$ (white, 0.98s).

Figure 5: Reachable set of a 2-dimensional test system.

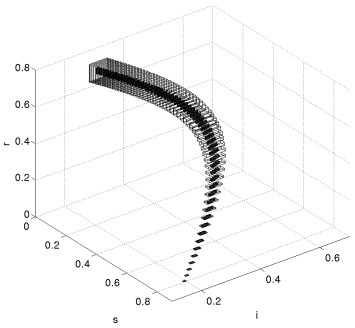
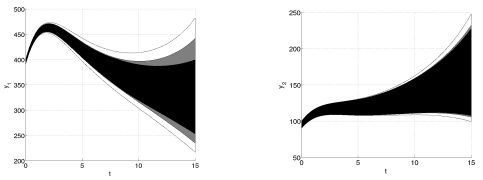


Figure 6: Reachable set of 3-dimensional SIR model. Sets have been computed with 1 temp/3 dirs (in white, 0.12s), and 4 temps/6 dirs (in black, 2.83s).

or second site, respectively, but who do not dance. The dynamics of the system are the following:

$$
\begin{aligned}
x_{k+1} &= x_k + (-\beta_1 x_k y_{1_k} - \beta_2 x_k y_{2_k})\Delta \\
y_{1_{k+1}} &= y_{1_k} + (\beta_1 x_k y_{1_k} - \gamma y_{1_k} + \delta\beta_1 y_{1_k} z_{1_k} + \alpha\beta_1 y_{1_k} z_{2_k})\Delta \\
y_{2_{k+1}} &= y_{2_k} + (\beta_2 x_k y_{2_k} - \gamma y_{2_k} + \delta\beta_2 y_{2_k} z_{2_k} + \alpha\beta_2 y_{2_k} z_{1_k})\Delta \\
z_{1_{k+1}} &= z_{1_k} + (\gamma y_{1_k} - \delta\beta_1 y_{1_k} z_{1_k} - \alpha\beta_2 y_{2_k} z_{1_k})\Delta \\
z_{2_{k+1}} &= z_{2_k} + (\gamma y_{2_k} - \delta\beta_2 y_{2_k} z_{2_k} - \alpha\beta_1 y_{1_k} z_{2_k})\Delta
\end{aligned}
\tag{8}
$$

The parameters $\beta_1$ and $\beta_2$ are the persuasion parameters, i.e., how vigorously the evangelic bees $y_1$ and $y_2$ dance; $\delta$ is the per capita rate at which the bees spontaneously leave the neutral and non-dancing groups $x, z_1, z_2$ for the dancing classes $y_1, y_2$; $\gamma$ is the per capita rate of ceasing to dance from the dancing classes $y_1, y_2$ to the non-dancing ones $x_1, x_2$; $\alpha$ is the proportionality of switching back spontaneously to the neutral state $x$; $\Delta$ is the discretization step. Similarly to the previous case study, the goal of this test is to study the scalability of our method in terms of number of directions and templates, and verify eventual improvements in the precision of the computed reachable set. For the simulation of this model we choose as initial set the box with $x_0 = 500, y_1 \in [390, 400], y_2 \in [90, 100], z_1 = z_2 = 0$. The parameter values are $\beta_1 = \beta_2 = 0.001, \gamma = 0.3, \delta = 0.5, \alpha = 0.7$, and $\Delta = 0.01$. Figure 7 shows the projections of the dancing bees $y_1$ and $y_2$ computed with three different configurations up to 1500 steps. The bundles have been transformed with



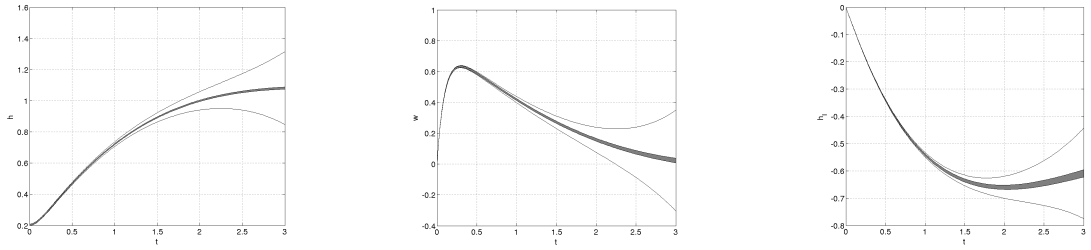
(a) Dancing bees $y_1$. (b) Dancing bees $y_2$.

Figure 7: Projections of reachable set of 5-dimensional honeybees decision-making model. Sets have been computed with 1 temp/5 dirs (in white, 6.57s), 2 temps/6 dirs (in gray, 26.90s), and 3 temps/7 dirs (in black, 81.27s).

the AFO method and no decomposition. In the first configuration (in white), the computation has been carried out with a single template composed by 5 axis-aligned directions (6.57s); the second (in gray) involved 2 templates composed by 6 directions, some of which were not aligned with the $x$ and $y_1$ axis (26.90s). In the third configuration we defined 3 templates composed by 7 directions, some of which not aligned with $x, y_1$, and $y_2$ axis (81.27s). From Figure 7 we can see how the precision of the computed reachable set increases with the addition of directions and templates.

***Quadcopter.*** In our last study we focus on the scalability of our method in terms of system dimension. In this case, we consider the model of a quadrotor drone composed by 17 variables regulated by quadratic dynamics. The model consists of 13 dynamics that drive the drone itself, plus 4 dynamics modeling its controller. The state variables of the drone include the inertial position $(p_n, p_e, h)$, linear velocity $(u, v, w)$, Euler angles expressed using quaternions $(q_0, q_1, q_2, q_3)$, and angular velocity $(p, q, r)$, while the controller variables involve some parameters of position, speed, and angle $(h_I, u_I, v_I, \psi_I)$. Given a reference height $h_r$, horizontal speeds $u_r, v_r$, and nose angle $\psi_r$, the goal of the controller is to bring the drone from its actual configuration to the one specified by the reference values. The detailed description of the model and its dynamics can be found in [11]. All the parameters (such as mass, axis moment of inertia, propeller masses, etc.) have been set accordingly with the real quadcopter CrazyFlie Nano by Bitcraze[3]. The chosen initial conditions are $h_0 \in [0.20, 0.21]$, $q_0 = 1$, and all

[3]https://www.bitcraze.io/



(a) Height ($h$). (b) Vertical speed ($w$). (c) Controller height ($h_I$).

Figure 8: Projections of reachable set of 17-dimensional quadcopter model. Sets have been computed with 1 temp/17 dirs (in white, 17.74s), 2 temps/18 dirs (in gray, 39.07s).

the other variables are set to zero. The reference height is $h_r = 1$, while speeds and angle are $u_r = v_r = \psi_r = 0$. We computed the reachable set up to 300 steps, corresponding to 3s of flight. We adopted 2 configurations, both based on AFO transformation without the bundle decomposition: the first consists in a single box template with axis-aligned constraints, the second has an additional parallelotope involving the dimensions that more vary during the flight (height, vertical speed, angle quaternions, and controller height). Figure 8 shows the projections of the computed reachable sets. The figure reports the evolutions over time of height $h$ (Figure 8a), vertical speed $w$ (Figure 8b), and the height computed by the controller $h_I$ (Figure 8c), obtained with a single (in white) and two templates (in gray). The first technique took 9.40s of computations, the second 20.32s. Note how a single additional template sensibly improves the precision of the computed reachable set and avoids wrapping effects.

# 7. CONCLUSION

In this work we defined parallelotope bundles and a family of operations that allowed us to define a reachability algorithm for polynomial dynamical systems. We showed the effectiveness of our method studying four nontrivial systems with polynomial dynamics. The obtained data shown the significant precision improvements with respect to the single box/parallelotope inclusion methods.

This work lays the foundation for future developments. An almost direct extension can be towards the reachability analysis of parametric dynamical systems, where a set of parameter values is provided in input. A more complicated problem would be the parameter synthesis involving bundles. Here, it is asked to refine the given parameter set so that the system satisfies a specification. Finally, note that all the defined operations on bundles are easily parallelizable. It could be interesting to implement a parallel bundle manipulator and investigate on the scalability in terms of system dimension and reachable set precision.

# 8. REFERENCES


[1] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *Conference on Decision and Control and European Control Conference, CDC-ECC*, pages 6814–6821. IEEE, 2011.

[2] M. Althoff and B. H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Trans. Automat. Contr.*, 59(2):371–383, 2014.

[3] G. Amato and F. Scozzari. The abstract domain of parallelotopes. *Electron. Notes Theor. Comput. Sci.*, 287:17–28, Nov. 2012.

[4] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control, HSCC*, pages 20–31. Springer, 2000.

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

[6] E. Asarin, T. Dang, O. Maler, and R. Testylier. Using redundant constraints for refinement. In *Automated Technology for Verification and Analysis, ATVA*, pages 37–51, 2010.

[7] N. Britton, N. Franks, S. Pratt, and T. Seeley. Deciding on a new home: how do honeybees agree? *Royal Society of London B: Biological Sciences*, 269(1498):1383–1388, 2002.

[8] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Real-Time Systems Symposium, RTSS*, pages 183–192. IEEE, 2012.

[9] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer Aided Verification, CAV*, pages 258–263, 2013.

[10] A. Chutinan and B. H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Conference on Decision and Control, CDC*, volume 2, pages 2089–2094. IEEE, 1998.

[11] A. E. C. da Cunha. Benchmark: Quadrotor attitude control. In *Applied Verification for Continuous and Hybrid Systems, ARCH*, 2015.

[12] T. Dang, T. Dreossi, and C. Piazza. Parameter synthesis using parallelotopic enclosure and applications to epidemic models. In *Hybrid Systems and Biology, HSB*, pages 67–82, 2014.

[13] T. Dang and R. Testylier. Reachability analysis for polynomial dynamical systems using the Bernstein expansion. *Reliable Computing*, 17(2):128–152, 2012.

[14] T. Dreossi and T. Dang. Parameter synthesis for polynomial biological models. In *Hybrid Systems: Computation and Control, HSCC*, pages 233–242, 2014.

[15] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang,

and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification, CAV*, pages 379–395. Springer, 2011.

[16] J. Garloff and A. P. Smith. Investigation of a subdivision based algorithm for solving systems of polynomial equations. *Nonlinear Analysis: Theory, Methods & Applications*, 47(1):167–178, 2001.

[17] K. Ghorbal, E. Goubault, and S. Putot. The zonotope abstract domain taylor1+. In *Computer Aided Verification, CAV*, pages 627–633, 2009.

[18] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control, HSCC*, pages 257–271. Springer, 2006.

[19] E. Goubault. Static analysis by abstract interpretation of numerical programs and systems, and FLUCTUAT. In *Static Analysis Symposium, SAS*, pages 1–3, 2013.

[20] Z. Han and B. Krogh. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *American Control Conference, ACC*, pages 6 pp.–, June 2006.

[21] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HYTECH: hybrid systems analysis using interval numerical methods. In *Hybrid Systems: Computation and Control, HSCC*, pages 130–144, 2000.

[22] E. K. Kostousovat. Control synthesis via parallelotopes: optimzation and parallel compuations. *Optimization Methods and Software*, 4(14):267–310, 1 2001.

[23] A. A. Kurzhanskiy, P. Varaiya, et al. Ellipsoidal toolbox. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-46*, 2006.

[24] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[25] B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations. *J. Symb. Comput.*, 44(3):292–306, 2009.

[26] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Trans. Embedded Comput. Syst.*, 6(1), 2007.

[27] S. Sankaranarayanan, T. Dang, and F. Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 188–202. Springer, 2008.

[28] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Verification, Model Checking, and Abstract Interpretation, VMCAI*, pages 25–41, 2005.

[29] O. Stursberg and B. H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Hybrid Systems: Computation and Control, HSCC*, pages 482–497. Springer, 2003.