Efficient Representation and Computation of Reachable Sets for Hybrid Systems

Olaf Stursberg[1] and Bruce H. Krogh[2]

[1] University of Dortmund
Germany
olaf.stursberg@uni-dortmund.de
[2] Dept. of Electrical and Computer Engineering
Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213-3890 USA
krogh@ece.cmu.edu

Abstract. Computing reachable sets is an essential step in most analysis and synthesis techniques for hybrid systems. The representation of these sets has a deciding impact on the computational complexity and thus the applicability of these techniques. This paper presents a new approach for approximating reachable sets using *oriented rectangular hulls* (ORHs), the orientations of which are determined by singular value decompositions of sample covariance matrices for sets of reachable states. The orientations keep the over-approximation of the reachable sets small in most cases with a complexity of low polynomial order with respect to the dimension of the continuous state space. We show how the use of ORHs can improve the efficiency of reachable set computation significantly for hybrid systems with nonlinear continuous dynamics.

Keywords. Convex Hull, Hybrid Dynamic Systems, Hyperrectangles, Model Checking, Polyhedra, Singular Value Decomposition.

1 Introduction

Research on hybrid systems (HSs) has led to a variety of methods for verification of properties, such as safety or liveness (e.g., [1–4]) and controller synthesis (e.g., [5–9]). For the vast majority of these methods, a common step is to compute approximations for the set of reachable states in the continuous state space. Typically, the state space is either partitioned into a finite number of subsets and an (approximate) evaluation of the continuous dynamics reveals which elements of the partition are reachable, or the continuous dynamics are used to propagate the reachable set iteratively from the set of initial states. In both cases the reachable sets are used to determine which discrete transitions are possible and to check if the given property is fulfilled or violated (possibly for specific control inputs).

The geometry chosen to represent reachable sets has a crucial effect on the efficiency of the whole procedure. Usually, the more complex the geometry of

the representation is: (i) the more costly is the storage of the sets, (ii) the more difficult it is to perform operations like union and intersection, and (iii) the more elaborate is the computation of new reachable sets, but (iv) the better the approximation of the actual set of reachable states for the HS. Choosing the geometry has to be a compromise between these impacts.

Several approaches have been proposed in the literature to over-approximate reachable sets by unions of convex objects of simple geometry such as hyperrectangles [10–12], polyhedra obtained from convex hull computations [3, 13], and ellipsoids [14, 15]. Each of these representations has strengths and weaknesses. Hyperrectangles have the advantage that they are easy to represent and the number of faces grows only linearly with the dimension, but a large number of boxes (i.e., a small grid) must be used to assure the approximations are not overly conservative. Polyhedra can give arbitrarily close approximations to convex sets, but the number of faces and vertices can grow prohibitively large and, as shown in [16, 17], the computation of polyhedra by convex hull routines becomes intractable for large sets of points in higher dimensions. Ellipsoids are attractive because the representation grows quadratically with the dimension of the continuous space. However, many ellipsoids may be needed to represent reachable sets with sufficient accuracy, and intersections and unions of ellipsoids are not ellipsoids.

This paper proposes an alternative that combines the geometrical simplicity of hyperrectangles with an orientation derived from the true reachable set rather than being fixed to the state-space axes. For a given number of points obtained from the evaluation of the dynamics, a preferred orientation is determined by the singular value decomposition (SVD) of the corresponding covariance matrix. Using this orientation, the smallest hyperrectangle that encloses all points is computed, giving as *oriented hyperrectangular hull* (ORH). When these geometrical objects are used as building blocks for the reachable set, a suitable compromise between computational complexity, approximation accuracy, and the ability to compute intersections and unions is often obtained.

In the following sections, we first identify the steps of reachability algorithms for hybrid systems for which the set representation has a crucial impact. We also explain in more detail in which cases previously proposed approaches have disadvantages with respect to efficiency. Then the concept of the ORH is introduced, and the complexity of ORH computations is discussed. Finally we show for one specific instance of reachability algorithms that the use of ORHs can reduce the computational costs drastically while retaining sufficient approximation accuracy.

2 Set Representation in Reachability Analysis of Hybrid Systems

2.1 Reachable Sets of Hybrid Systems

Throughout the paper we refer to the following definition of hybrid systems. The discussion of reachable set representation applies as well to other variations of this model that appear in literature.

Definition 1. *Syntax of a Hybrid Automaton HA*
The hybrid automaton $HA = (Z, z_0, X, X_0, inv, T, g, j, f)$ contains:

- *the finite set of locations Z with an initial location $z_0 \in Z$,*
- *the continuous state space $X \subseteq \mathbb{R}^n$, and an initial continuous set $X_0 \subseteq X$;*
- *the invariant function $inv : Z \to 2^X$ that assigns an invariant $inv(z) \subseteq X$ to each location $z \in Z$; for convenience we require that $X_0 \subseteq inv(z_0)$;*
- *the set of discrete transitions $T \subseteq Z \times Z$;*
- *the guard function $g : T \to 2^X$ that assigns a guard set $g(t) \subseteq X$ to each transition $t = (z_1, z_2) \in T$;*
- *the jump function $j : T \times X \to 2^X$ that assigns a jump set $j(t, x) \subseteq X$ to each pair $t \in T$ and $x \in g(t)$;*
- *and the flow function $f : Z \to (X \to \mathbb{R}^n)$ assigns a continuous vector field $f(z)$ to each location $z \in Z$. The continuous evolution in z is determined by the ODE $\dot{\chi}(t) = f(z, \chi(t))$ for which we assume that a unique solution exists for each $\chi(0) \in inv(z)$.*

◇

Definition 2. *Runs and the Reachable Set of HA*
Let $S = \bigcup_{z \in Z} \bigcup_{x \in inv(z)} (z, x)$ denote the set of hybrid states (z, x) of a hybrid automaton HA. Then, each possible run of HA is a sequence $\sigma = \{s_0, s_1, s_2, \ldots\}$, iff:

- *the initial hybrid state is $s_0 = (z_0, x_0)$, with $x_0 \in X_0$,*
- *and each pair of consecutive states $(s_i, s_{i+1}) \in \sigma$ with $s_i = (z_i, x_i)$ and $s_{i+1} = (z_{i+1}, x_{i+1})$ satisfies:*
  - *either (discrete transition) $(z_i, z_{i+1}) \in T$, $x_i \in g((z_i, z_{i+1}))$, and $x_{i+1} \in j((z_i, z_{i+1}), x_i)$;*
  - *or (continuous evolution) $z_i = z_{i+1}$ and there exists $\chi : [0, \tau] \to X$, $\tau \in \mathbb{R}^{>0}$ such that $x_i = \chi(0)$, $\dot{\chi}(t) = f(z_i, \chi(t))$, $\chi(t) \in inv(z_i)$ for $t \in [0, \tau]$, and $x_{i+1} = \chi(\tau)$.*

If Σ is the set of all possible runs of HA, the reachable set *is defined by $R = \{s \mid \exists \sigma \in \Sigma : s \in \sigma\} \subseteq S$, i.e., R contains all hybrid states that are elements of at least one run σ.* ◇

2.2 Computation of Reachable Sets

In order to point out where the set representation plays a role in computing the reachable set R of HA, we consider the general reachability procedure shown in Fig. 1. Starting from the initial set S_0, R is computed iteratively.[1] In each step k those hybrid states D are added to R that are reached in the current step but were not reached before. We leave open at this point what exactly defines a step – it could be a specified time increment, space increment, or, e.g., a consecutive pair of continuous evolution and discrete transition. The operator *Reach* computes the set of states that are reachable in one step according to the semantics given in Def. 2. While the structure of this algorithm appears to be

```
given: S_0 = {z_0} × X_0
k = 0, D = S_0, R = ∅
WHILE D ≠ ∅
    k = k + 1
    R = R ∪ D
    S_k = Reach(D)
    D = S_k \ R
END
```

Fig. 1. High-level algorithm for computing R iteratively (S_0: initial set, S_k: set of states reachable within one step k, D: set of states that is reached the first time in step k).

very simple, a concrete implementation leads to the following issues:

(a) In the general case when D is an infinite set of hybrid states, the operator $Reach(D)$ requires the evaluation of an infinite number of behaviors of HA. If the ODEs cannot be solved analytically, one has to fall back to numerical (yet conservative) approximations of $Reach(D)$.
(b) The sets S_k are in general non-convex (even if analytical solutions of the ODEs exist). In order to store the reachable sets efficiently, they have to be approximated by objects of simple geometry (usually convex sets).
(c) Computing R, S_k, and D requires efficient implementations of operations like set intersection, set union, and set subtraction. Note in this context that computing $Reach(D)$ involves checking whether transition guards are enabled and applying the jump function.

In order to cope with problems (a) and (b), most of the existing algorithms compute a series of convex objects that over-approximate S_k. These algorithms include the computation of convex hulls for finite sets of points (e.g., see [13, 12]). The resulting polyhedra represent either an intermediate or the final result for approximating S_k. The choice of the geometry for the objects that establish

[1] The well known decidability results for hybrid automata (see, e.g., [18, 19]) imply this procedure might not terminate.

S_k has a crucial impact on the accuracy of the approximation as well as on the storage requirements. As an example, Fig. 2 illustrates three alternatives for hulls of a given set of points determined to be reachable from the set D. The differences in accuracy and the effort to store the objects (hyperrectangle: a matrix in $\mathbb{R}^{n\times 2}$; hyperellipsoid: a matrix in $\mathbb{R}^{n\times(n+1)}$; convex hull: a matrix in $\mathbb{R}^{q\times(n+1)}$ where q is the number of faces) are apparent.

Fig. 2. Different types of hulls for a set of points encountered while computing $S_k = Reach(D)$: 1 - hyperrectangle (axes-parallel), 2 - hyperellipsoid, 3 - convex hull.

With respect to problem (c), the efficient applicability of the set operations is another important criterion in choosing suitable objects for the set representation. For convex polyhedra, the result of intersection, union and subtraction is itself a set of of convex polyhedra and standard routines exist. If hyperellipsoids are chosen to represent R, D, and S_k, these operations do not yield hyperellipsoids and an additional approximation step is required to obtain an ellipsoidal approximation.

The effort to compute the hull also plays a dominant role when making the decision for a specific geometric object. In particular, if S_k has a complex shape, the approximation accuracy usually requires that S_k be the union of several small convex sets. The number of hull computations then becomes very large and it can be observed that the time spent on this step is a substantial portion of the overall computation time. This calls for an efficient procedure to compute the hull. Since existing reachability algorithms are limited to low-dimensional systems due to their complexity, the hull computation should especially scale well with the dimension n of the state space and the number of points in each step.

A last important criterion is the numerical stability. We have found that convex hull algorithms encounter difficulties if the set of points to be enclosed lies in a lower-dimensional subspace. Usually they return a convex hull that is 'bloated' to full dimension (i.e. some points are perturbed) or a lower-dimensional set that comprises a higher number of faces than necessary. With respect to an efficient set representation, it is certainly desirable to have a procedure that generates a hull of minimal dimension and a minimal number of faces.

3 Set Representation by Oriented Rectangular Hulls

This section first introduces the ORH as an alternative to the types of sets discussed in the previous section. We then describe the advantages of the ORH with respect to the criteria listed above.

3.1 Definition of the ORH

We begin with some basic definitions and notation. Let x be vector of continuous variables defined in the Euclidean space $\mathbb{R}^{n\times 1}$. A *half-space* is given by $S := \{x \mid c \cdot x \leq d,\ c \in \mathbb{R}^{1\times n},\ d \in \mathbb{R}\}$. For a given half-space S, let $B = \{x \mid\ c \cdot x = d\}$ denote the corresponding *bounding hyperplane*. If the intersection of a finite set $\mathcal{S} = \{S_1, \ldots, S_q\}$ of half-spaces with pairwise different normal vectors c is non-empty and bounded in $\mathbb{R}^n$, it determines a convex *polyhedron* $P := \{x \mid \exists \mathcal{S} : x \in S_j\ \forall\ S_j \in \mathcal{S}\}$. The corresponding set of bounding hyperplanes is denoted by $\mathcal{B}$.

Let $\mathcal{X} = \{x^1, x^2, \ldots, x^p\}$ be a finite set of vectors in $\mathbb{R}^{n\times 1}$. A polyhedron P is called a *polyhedral hull* of $\mathcal{X}$, denoted by $PH(\mathcal{X})$, iff $x^i \in P$ for all $x^i \in \mathcal{X}$, $i \in \{1, \ldots, p\}$.

Given a polyhedral hull $PH(\mathcal{X})$, the set of *vertices* of PH is denoted by $\mathcal{V} = \{v_1, \ldots, v_r\}$, where each vertex $v := \{x \mid \exists\ \mathcal{B}' \subseteq \mathcal{B}, |\mathcal{B}'| \geq n,\ x \in \mathbb{R}^n : x \in B_j\ \forall\ B_j \in \mathcal{B}'\}$ is determined by the intersection of at least n bounding hyperplanes.

The polyhedral hull PH is called a *convex hull*, denoted by $CH(\mathcal{X})$, iff $\mathcal{V} \subseteq \mathcal{X}$ applies. A convex hull CH is called a *rectangular hull*, denoted by $RH(\mathcal{X})$, iff $\mathcal{S}$ is the intersection of $q = 2 \cdot n$ half-spaces such that for each $S_j = \{x \mid c_j \cdot x \leq d_j\}$: *(i)* there exists an $S_{k\neq j} = \{x \mid c_k \cdot x \leq d_k\}$ with $c_j = -c_k$, and *(ii)* for each $S_{i\neq j, i\neq k} \in \mathcal{S}$, $\langle c_j, c_i^T \rangle = 0$. Let $\mathcal{E} = \{e_1, \ldots, e_n\}$ be the set of directions of the axes of $\mathbb{R}^n$, i.e., $e_i = (0^{1\times i-1}, 1, 0^{1\times n-i})$, $i \in \{1, \ldots, n\}$. In the special case that for the normal vector c of each half-space of $\mathcal{S}$ there exists an $e_i \in \mathcal{E}$ with $\langle \pm e_i, c^T \rangle = 0$, we call $RH(\mathcal{X})$ an *axes-parallel rectangular hull*, denoted by $ARH(\mathcal{X})$.

The attractive feature of a rectangular hull $RH(\mathcal{X})$ is that the number of bounding hyperplanes increases only linearly with n and is independent of p, the number of points it encloses. The definition of $RH(\mathcal{X})$ does not, however, specify an orientation of the hull (i.e, of the choice of the normal vectors), except of the special case of $ARH(\mathcal{X})$. For the latter, it is easy to imagine a set of points for which the axes-parallel orientation is not suitable in the sense of a tight enclosure of $\mathcal{X}$ (see Fig. 2). To overcome this problem, we now introduce an efficient procedure for choosing the parameters c and d of the halfspaces defining $RH(\mathcal{X})$.

The principle is to derive the orientation from the distribution of the points within the space $\mathbb{R}^n$. The elements of $\mathcal{X}$ are interpreted as p sampled evaluations of the dynamics of HA. The arithmetic mean of the samples $x^m = \frac{1}{p}\sum_{i=1}^{p} x^i$ is chosen as the origin of a set of translated samples: $\overline{\mathcal{X}} = \{\overline{x}^1, \ldots, \overline{x}^p\}$, $\overline{x}^i = x^i - x^m$. To characterize the distribution of $\overline{\mathcal{X}}$ we define the sample covariance matrix as follows.

Given the set $\overline{\mathcal{X}}$ of p translated samples of the vector of n continuous variables x, the sampling matrix is denoted by:

$$\overline{X} = \begin{pmatrix} \overline{x}_{1,1} & \cdots & \overline{x}_{1,p} \\ \vdots & \ddots & \vdots \\ \overline{x}_{n,1} & \cdots & \overline{x}_{n,p} \end{pmatrix} \tag{1}$$

where $\overline{x}_{i,j} = x_i^j - x_i^m$ is the j-th sample of the i-th variable. For two components of the translated state vector, $\overline{x}_i = x_i - x_i^m$ and $\overline{x}_k = x_k - x_k^m$, the *sample covariance* is defined as:

$$Cov(\overline{x}_i, \overline{x}_k) = \frac{1}{p-1} \sum_{j=1}^{p} \overline{x}_{i,j} \cdot \overline{x}_{k,j}. \tag{2}$$

In the context of reachable set computations for hybrid systems, the correlation of two variables x_i, x_k (resulting in $Cov(\overline{x}_i, \overline{x}_k) \neq 0$) follows from the fact that $\mathcal{X}$ is obtained from the evaluation of coupled ODEs to generate x_i, x_k.

The *sample covariance matrix*, which represents the distribution of $\mathcal{X}$ in $\mathbb{R}^n$, is then written as:

$$Cov(\overline{x}) = \begin{pmatrix} Cov(\overline{x}_1, \overline{x}_1) & \cdots & Cov(\overline{x}_1, \overline{x}_n) \\ \vdots & \ddots & \vdots \\ Cov(\overline{x}_n, \overline{x}_1) & \cdots & Cov(\overline{x}_n, \overline{x}_n) \end{pmatrix} = \frac{1}{p-1} \cdot \overline{X} \cdot \overline{X}^T. \tag{3}$$

To obtain a suitable orientation for a rectangular hall $RH(\mathcal{X})$ from $Cov(\overline{x})$, we use the technique known as *principal component analysis* (PCA) in literature. As described in the early publications [20–22], PCA usually aims at finding the dominating correlations between large sets of variables for given large sets of data. By considering the dominating correlations only, the original data can be represented by a small yet meaningful set of variables (the principal components). In our setting however, we use PCA to derive an orientation to define a particular $RH(\mathcal{X})$ for $\mathcal{X}$. The orientation is obtained from the *singular value decomposition* of $Cov(\overline{x})$, given by

$$Cov(\overline{x}) = U \cdot \Sigma \cdot V^T, \tag{4}$$

where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{n \times n}$ are unitary matrices. If $r = rank(Cov(\overline{x}))$, the matrix of singular values is

$$\Sigma = \begin{pmatrix} \Sigma_r & \mathbf{0}^{r \times n-r} \\ \mathbf{0}^{n-r \times r} & \mathbf{0}^{n-r \times n-r} \end{pmatrix}, \text{ with } \Sigma_r = diag(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}, \tag{5}$$

and the *singular values* σ are ordered such that $\sigma_1 \geq \ldots \geq \sigma_r > 0$.[2]

Since $Cov(\overline{x})$ is symmetric, $U = V$. If we write $U = [U_r, U_{n-r}]$ with $U_r \in \mathbb{R}^{n \times r}$ and $U_{n-r} \in \mathbb{R}^{n \times n-r}$, the columns of U_r define an orthonormal basis of the r-dimensional subspace that contains all points in $\mathcal{X}$. We use the directions defined by this basis to determine the orientation of $RH(\mathcal{X})$:

[2] It can be shown that the singular values are given by $\sigma_i = \sqrt{\lambda_i}$, where λ_i, $i = 1, \ldots, r$, are the nonzero eigenvalues of $Cov(\overline{x}) \cdot Cov(\overline{x})^T$, which are all real and non-negative.

Definition 3. Oriented Rectangular Hull
Let $\mathcal{X} = \{x^1, x^2, \ldots, x^p\}$ be a given set of samples, $\overline{\mathcal{X}}$ the set of translated samples with a covariance matrix $Cov(\overline{x}) = U \cdot \Sigma \cdot V^T$. We write $U_{\bullet,i}$ to denote the i-th column of U. The oriented rectangular hull, denoted $ORH(\mathcal{X})$, is the rectangular hull $RH(\mathcal{X})$ defined by the set $\mathcal{S} = \{S_1, S_2, \ldots, S_{2 \cdot n}\}$ of halfspaces such that for a $\varepsilon \in \mathbb{R}^{\geq 0}$ and $\forall\ i \in \{1, \ldots, n\}$:

$$- \ \exists\ S_j \in \mathcal{S}: \quad S_j = \{x \mid U_{\bullet,i}^T \cdot x \leq \max_{\overline{x} \in \overline{\mathcal{X}}} \{U_{\bullet,i}^T \cdot \overline{x}\} + U_{\bullet,i}^T \cdot x^m + \varepsilon\},\ \textit{and:}$$

$$- \ \exists\ S_k \in \mathcal{S}: \quad S_k = \{x \mid -U_{\bullet,i}^T \cdot x \leq -\min_{\overline{x} \in \overline{\mathcal{X}}} \{U_{\bullet,i}^T \cdot \overline{x}\} - U_{\bullet,i}^T \cdot x^m + \varepsilon\}$$

$\diamond$

Note that for a rank deficiency ($i > r$), the two halfspaces S_j and S_k lead to a hull ORH that is 'flat' in the direction of $U_{\bullet,i}$ (if $\varepsilon = 0$). If a full dimensional hull is deemed to be numerically more stable within the computation of $S_k = Reach(D)$, a small tolerance $\varepsilon > 0$ can be chosen.

3.2 Assessment of ORH

The criteria for assessing hulls, as listed at the end of Sec. 2, are now applied to ORHs. Figure 3 shows four distributions of points in $\mathbb{R}^2$ and the corresponding ORHs. It is apparent that, if the distribution of points has a preferred orientation (upper row of figures), one of the axes of ORH represents the corresponding direction (determined by $U_{\bullet,1}$). If all points lie in a lower-dimensional subspace (as in Fig. 3.b), the expansion in the null space is just given by $2 \cdot \varepsilon$. If the distribution of points is completely symmetric with respect to x^m (i.e., two or more singular values are identical), the orthonormal basis U_r is not uniquely defined, and an additional criterion should be employed. For the points shown in Fig. 3.c, both rectangles correspond to a valid singular value decomposition of $Cov(\overline{x})$. To obtain the solution marked by the solid line, which is clearly the better solution (and equivalent to the convex hull), the additional requirement is that the vertices of ORH coincide with the given points. The point set in Fig. 3.c does not have a visible preferred orientation either, but has a random distribution. It is not obvious, however, that a rectangular hull with a more suitable orientation exists.

In order to assess the accuracy of representing $\mathcal{X}$ by an ORH, we compare its volume $Vol(ORH(\mathcal{X}))$ to that of other types of polyhedral hulls. The volume may seem to be a questionable measure for accuracy since polyhedral hulls of completely different shape can have the same volume. However, relat-

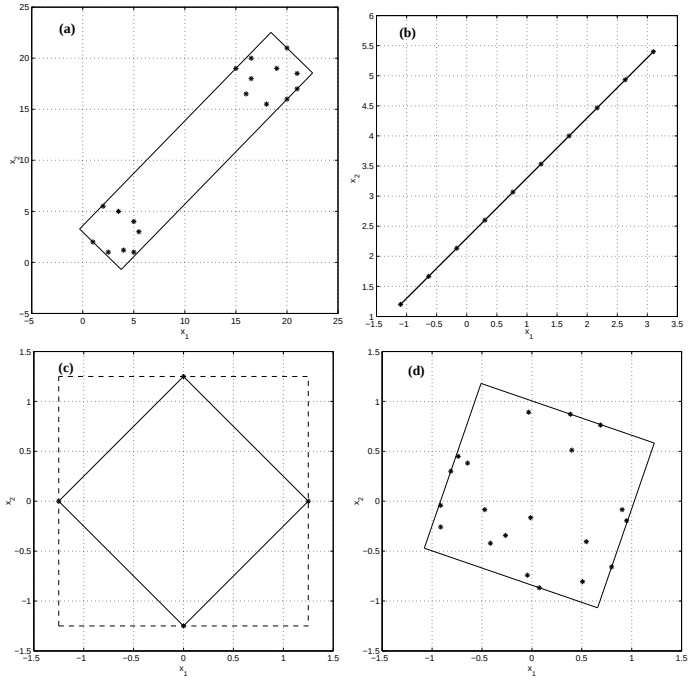

Fig. 3. Oriented rectangular hulls for different example sets of points.

Table 1. Volumes of different polyhedral hulls for the example sets in Fig. 3 (with $\varepsilon = 0$ for Fig. 3.b)

Fig. 3:	(a)	(b)	(c)	(d)
$ORH(\mathcal{X})$	152.99	0	3.13	3.20
$CH(\mathcal{X})$	134.99	0	3.13	2.41
$ARH(\mathcal{X})$	400.00	17.64	6.25	3.28

ing $Vol(ORH(\mathcal{X}))$ to the volume of the convex hull $Vol(CH(\mathcal{X}))$ gives some estimation for the degree by which $ORH(\mathcal{X})$ overapproximates the set $\mathcal{X}$. This is true since $CH(\mathcal{X})$ is by definition the exact polyhedral hull that contains $\mathcal{X}$. Table 1 shows the volumes of $ORH(\mathcal{X})$, $CH(\mathcal{X})$, and $ARH(\mathcal{X})$ for the four cases chosen in Fig. 3. In the two cases with non-symmetrical distribution of points (a, d), $Vol(ORH(\mathcal{X}))$ exceeds $Vol(CH(\mathcal{X}))$ by 13.3%, and 32.8% respectively. Of course, the restriction to a rectangular hull with $2 \cdot n$ faces leads in most cases to a less accurate representation of $\mathcal{X}$ than is given by $CH(\mathcal{X})$. This disadvantage has to be related, however, to the computational effort required to obtain the hull, as discussed below. If the volume of a polyhedral hull is taken as a meaningful measure, one can pose the question whether computing the rectangular hull that encloses all points in $\mathcal{X}$ with a minimum volume $RH_{minVol}(\mathcal{X})$ can be suitable alternative to $ORH(\mathcal{X})$. Algorithms to compute $RH_{minVol}(\mathcal{X})$ exist (see, e.g., [23]). While $RH_{minVol}(\mathcal{X})$ is by definition a better approximation (in terms of the volume) than $ORH(\mathcal{X})$, the difference vanishes in most cases if $\mathcal{X}$ has a preferred direction. (Such a preferred orientation is usually obtained if the continuous dynamics of HA is evaluated for a suitable time increment). The reason for not proposing the computation of $RH_{minVol}(\mathcal{X})$ within a reachability algorithm is the computational costs: Existing techniques to determine $RH_{minVol}(\mathcal{X})$ involve the computation of convex hulls or a nonlinear optimization with non-differentiable cost function. Both approaches seem in general not to be competitive to the ORH computation with respect to complexity.

For the ORH, only the evaluation of Eq. 3 and the determination of the halfspaces according to Def. 3 depends on $|\mathcal{X}|$. The computation of the SVD depends on the dimension of the state space as $O(n^3)$ [24]. Table 2 lists the times to compute ORHs for a set of randomly distributed points, where n and $|\mathcal{X}|$ vary. The times for computing $CH(\mathcal{X})$ for the same sets using a Quickhull-algorithm [25] are included for comparison. These computations could not be finished within one hour for the configurations in the last column. The numbers show that the complexity barrier applies for ORH at much higher dimensions than for convex hulls, and that the number of points affects the ORH computation much less than they effect the computation of $CH(\mathcal{X})$. Although not included in the table, it should be noted that the computation of ellipsoid containing a given set of points involves the solution of optimization problems [26].

In order to asses the storage requirements, we note that an ORH is determined by $C \cdot x \leq d$, with matrices $C \in \mathbb{R}^{2 \cdot n \times n}$ and $d \in \mathbb{R}^{2 \cdot n}$. (In the case of rank deficiency, $2 \cdot (n - r)$ inequalities can be replaced by $n - r$ equalities.) Therefore, the amount of data to be stored grows quadratically with the dimension n and is

Table 2. Computation times. The shown values are the mean CPU-times in seconds of 5 experiments each. (Implementation: Matlab; Machine: Pentium III, 700MHz).

$\|\mathcal{X}\| = 20, n:$	2	4	6	8	10	12
$ORH(\mathcal{X})$	0.05	0.06	0.08	0.17	1.81	77.4
$CH(\mathcal{X})$	0.05	0.62	8.80	93.61	$> 10^3$	–
$n = 5, \|\mathcal{X}\|:$	20	40	60	80	100	1000
$ORH(\mathcal{X})$	0.06	0.06	0.06	0.06	0.07	0.53
$CH(\mathcal{X})$	2.42	9.20	28.41	51.09	87.55	–

independent of $|\mathcal{X}|$. In comparison, the number of inequalities required to define $CH(\mathcal{X})$ depends on the number of vertices determining the convex hull, and can be significantly higher even for low dimensions (e.g., for the points in the right lower example in Fig. 3). As mentioned in Sec. 2.2, hyperellipsoids can be represented by $(x-b)^T \cdot C \cdot (x-b) \leq d$, $C \in \mathbb{R}^{n \times n}$, $d \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, i.e., the required memory also grows quadratically with n.

As for the suitability of a particular type of hull for operations like intersection, union, and set subtraction, the complexity of all these operations crucially depends on the number of faces of a polyhedral hull, or the dimensions of C and c specifying ellipsoids respectively. Hence, the discussion above for the required memory applies when assessing the complexity of set operations. Note that union and set subtraction can lead to non-convex results, and thus an approximating step is required to yield a set that is of the same type as the original sets.

Finally, the issue of handling lower-dimensional sets deserves a comment. When applying convex hull algorithms, we have observed that existing approaches lead to difficulties if all points in $\mathcal{X}$ are in a lower-dimensional subspace. Most algorithms perturb the points slightly to compute a full dimensional convex hull first, and then the result is projected onto the original lower-dimensional space. The consequence is a hull that contains identical rows in the C- and d-matrix, and extra vertices. These identical rows can cause numerical problems in many operations. A remedy for this problems is to first transform $\mathcal{X}$ into the lower-dimensional space, compute a convex-hull in this space, and to transform the vertices back into full dimension. The ORH computation does not require the projection into a lower-dimensional space. As given in Def. 3, the concept of transforming into a space of appropriate dimension is implicit, and the tolerance ε can be chosen to control the effects of numerical errors.

4 Oriented Rectangles in Reach Set Computations

4.1 Reachability Computation based on ORH

This section describes for one specific instance of reachable set algorithms what the impact of the set representation is, and shows that the use of ORH can be favorable. Referring to the algorithm in Fig. 1, we now assume that a step (denoted by k) is given by a continuous evolution followed by a discrete transition. The operator $S_k = \mathit{Reach}(D)$ first determines the set of hybrid states which are

reachable from states in D by continuous evolution until a transition is enabled and then applies applies the transition. The first step must conservatively compute all possible evolutions starting from D. We here employ the algorithm used in the tool CheckMate, as described in [28]. This algorithm can be written as follows:

```
given: D, △T ∈ ℝ
S_k = ∅
Z_D = {z | z ∈ Z : s = (z, x) ∈ D}
FOR ALL  z ∈ Z_D
    Q = {x | ∃ s = (z, x) ∈ D}
    G = Q
    WHILE  G ∩ inv(z) ≠ ∅
        V = evolve_vertices(z, G, △T)
        H = determine_hull(G,V)
        G = bloat_hull(z, H)
        Q = Q ∪ (G ∩ inv(z))
    END
    S_Q = {s | ∃x ∈ Q :  s = (z, x) ∈ S)
    S_trans = execute_transition(S_Q)
    S_k = S_k ∪ S_Q ∪ S_trans
END
```

Fig. 4. A high-level algorithm to compute the set $S_k = Reach(D)$.

The set Z_D denotes the locations that correspond to D. Q is the set of continuous states x that form a subset of D together with a particular $z \in Z_D$, and it is given as convex polyhedral set.[3] Starting from Q, the part of the invariant $inv(z)$ is computed which is reachable by continuous evolution. Let G be a set of continuous states that is initialized to Q. The computation then involves three phases that are applied stepwise until the current location invariant is completely left. The function *evolve_vertices* first simulates the vertices of G for a given timestep $\triangle T$. The result is the set V of vertices. A function *determine_hull* then computes a polyhedral hull H containing V and all vertices of G. This is the point where the choice between convex hull, ORH, or another polyhedral hull has to be made. The third step, denoted by a function *bloat_hull*, expands H such that all trajectories emerging from G are completely contained in the bloated hull. This is achieved by pushing the faces of H outwards by solving the following optimization problem: Let $\mathcal{B}$ again be the set of bounding hyperplanes of the hull H. Then, the solution of:

$$\max_{\substack{\chi(0) \in G \\ \tau \in [0, \triangle T]}} \{c \cdot \chi(t)\} \quad \texttt{s.t.} \quad \chi(t) = \chi(0) + \int_0^\tau f(z, \chi(\tau)) \cdot d\tau \tag{6}$$

[3] Here we assume for simplicity that Q is one polyhedral object. In general it can be a list of polyhedral sets and the WHILE-loop has to be applied to each of these.

for every bounding hyperplane $B = \{x \mid c \cdot x = d\} \in \mathcal{B}$ leads to a *bloated hull* that entirely contains the set of points reachable from G within the timestep $\triangle T$. It is important to note that one optimization is carried out for each bounding hyperplane of the hull, and that the orientations of B (determined by the vectors c) are obtained from the hull computation. It is thus obvious that the chosen type of hull plays a key role in determining the computation required to construct the approximation for the reachable set.

The result of the bloating operation(the part which is inside of $inv(z)$) is added to Q. By repeating this three-phase procedure until $G \cap inv(z) = \emptyset$ applies, the subset of $inv(z)$ that is reachable from the initial Q is conservatively approximated.[4] The function *execute_ transition* determines which outgoing transitions of z are enabled. This step includes computing the intersection of Q with the corresponding transition guards. Applying the jump function (see Def. 1) to these intersections leads then to the hybrid state set S_{trans}. S_Q is the set of hybrid states that are formed by $z \in Z_D$ and Q. The sets S_{trans} and S_Q are added to S_k at the end of each cycle of the `FOR`-loop. When this loop terminates, S_k is the returned.

4.2 Example: Van der Pole System

We illustrate the algorithm by the example of Van der Pole equations with an additional clock variable:

$$
\begin{aligned}
&Z = \{z_1, z_2, z_3\}, \quad z_0 = z_1, \quad X = \mathbb{R}^3, \quad X_0 = [0.6, 0.9] \times [0.6, 0.9] \times 0, \\
&inv(z_1) = [-2, 2] \times [-2, 2] \times [0, 9], \quad inv(z_2) = [2, 5] \times [-2, 2] \times [0, 12], \\
&inv(z_3) = [-2, 2] \times [-2, 2] \times [9, 12] \\
&T = \{(z_1, z_2), (z_1, z_3)\}, \\
&g((z_1, z_2)) = 2 \times [-2, 2] \times [0, 9], \quad g((z_1, z_3)) = [-2, 2] \times [-2, 2] \times 9, \\
&j((z_1, z_2), x) = x, \quad j((z_1, z_3), x) = x, \\
&f(z_1) = (x_2, \frac{x_2}{5} \cdot (x_1^2 - 1) - x_1, 1)^T, \quad f(z_2) = f(z_3) = (0, 0, 0)^T
\end{aligned}
\tag{7}
$$

For this system we have posed the verification problem whether the location z_2 is reachable from the initial hybrid set specified by z_0 and X_0. The reachable set is shown in Fig. 5 for two different choices of hulls (i.e., two different functions *determine_hull*). The set of grey-shaded polyhedra in the left figure is the result obtained with convex hull approximation, while the right figure shows the result using ORH. The plane at $x_3 = 9$ (vertical axis) corresponds to the guard $g((z_1, z_3))$. The figures reveal that the reachable set approximation is slightly larger when the ORH is used, but the result is not significantly different from the result using the convex hull. It is also apparent that the number of bounding hyperplanes for each set G is large if convex hulls are used, while six bounding

[4] Conservativeness is achieved under the assumptions for Eq. 6 that the optimization leads to the global optimum and that the continuous trajectory $\chi(t)$ is exactly computable. In practice, both are achieved within a specified tolerance.

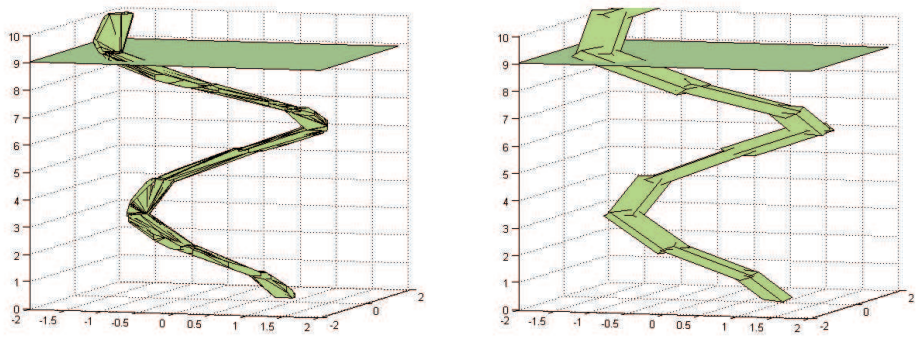

Fig. 5. Reachable set for the 3-dim. Van der Pole system ($\triangle T = 1$): left – convex hulls, right – oriented rectangular hulls.

hyperplanes suffice in each step in the ORH case. Hence, the number of optimizations according to Eq. 6 carried out in each step is considerably lower using ORHs.

Table 3 shows computation times for the two alternatives in comparison for different timesteps $\triangle T$. In order to evaluate the computational complexity for varying dimensions of this example, we have introduced further clock variables. This means that continuous variables x_4 and x_5 with dynamics as for x_3 are defined (and the sets X and X_0, as well as the invariants and guards are extended accordingly). The results in Table 3 show that for a fixed dimension a reduction of the timestep increases the CPU-times roughly linearly (a constant overhead is required for model compilation etc.). For a constant $\triangle t$, the CPU-time for the use of convex hulls leads to a quick increase with the dimension and prohibits the verification for dimension much larger than $n = 5$ for this example. An increase is also observed for the use of ORH, but for the same configurations the computation time is considerably smaller.

Table 3. Computation times in seconds for the Van der Pole system for varying dimensions n and timesteps $\triangle T$. (Implementation: Matlab; Machine: Pentium IV, 1.5GHz).

n :	3	3	3	4	4	4	5	5	5
$\triangle T$:	1	0.6	0.2	1	0.6	0.2	1	0.6	0.2
$ORH(\mathcal{X})$	82.7	116.8	335.1	119.9	183.7	561.84	198.4	297.6	855.4
$CH(\mathcal{X})$	214.4	434.0	1051.4	559.9	1140.5	2689.5	1483.8	3257.8	$> 5 \cdot 10^4$

5 Conclusions

Set representation is a crucial component of reachable set algorithms for hybrid systems. Among the different options for defining hulls of point sets $\mathcal{X}$, the oriented rectangular hulls have been identified as a choice that (a) is efficiently computable, (b) can be stored with small memory requirements, (c) is suitable for operations like set intersection, union, and subtraction, (d) can represent $\mathcal{X}$ with sufficient accuracy in many cases, (e) scales favorably (over convex hulls) with the dimension and the cardinality of $\mathcal{X}$, and (f) can lead to considerable computational savings due to the relatively small number of bounding hyperplanes when used in reachability algorithms. Although the latter point has been demonstrated only for one specific algorithm, the advantages should be similar for other procedures in which the geometry of the hulls determines the computational effort.

The ORH represents a good approximation of point sets $\mathcal{X}$ particularly when $\mathcal{X}$ has a preferred orientation. Since $\mathcal{X}$ is obtained from simulating the hybrid systems over a time span $\triangle T$, a straightforward approach is to choose $\triangle T$ such that $\mathcal{X}$ has a preferred orientation (if at all possible for the given dynamics). Our current work aims at addressing this issue. One approach is to simulate until the decomposition of the covariance matrix leads to singular values that differ to a specified extent. The value of $\triangle T \in \mathbb{R}$ would then be adjusted to the evolution of the hybrid system. In addition, this modification would avoid the cases in which a missing preferred orientation of $\mathcal{X}$ leads to poor accuracy of the $ORH(\mathcal{X})$ (in comparison to $CH(\mathcal{X})$).

Acknowledgment

This research has been financially supported in part by the U.S. Army Research Office and the U.S. Defense Advanced Projects Research Agency.

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science **138** (1995) 3–34
2. Bemporad, A., Morari, M.: Verification of hybrid systems via mathematical programming. In: Hybrid Systems: Computation and Control. Volume 1569 of LNCS., Springer (1999) 31–45
3. Greenstreet, M.R., Mitchell, I.: Reachability analysis using polygonal projections. In: Hybrid Systems: Computation and Control. Volume 1569 of LNCS., Springer (1999) 103–116
4. Asarin, E., Bournez, O., Dang, T., Maler, O.: Approximate reachability analysis of piecewise-linear dynamical systems. In: Hybrid Systems – Computation and Control. Volume 1790 of LNCS., Springer (2000) 20–31
5. Wong-Toi, H.: The synthesis of conbtrollers for linear hybrid automata. In: Proc. 36^{th} IEEE Conf. Decision and Control. (1997) 4607–4612

6. Cury, J., Krogh, B., Niinomi, T.: Synthesis of supervisory controller for hybrid systems based on approximating automata. IEEE Transaction on Automatic Control **43** (1998) 564–569
7. Asarin, E., Bournez, O., Dang, T., Maler, O., Pnueli, A.: Effective synthesis of switching controllers for linear systems. Proc. of the IEEE **88** (2000) 1011–1025
8. Tomlin, C., Lygeros, J., Sastry, S.: A game theoretic approach to controller design for hybrid systems. Proc. of the IEEE **88** (2000) 949–970
9. Xia, H., Pang, Y., Trontis, A., Spathopoulos, M.: Eventuality synthesis for controlled linear automata. In: Proc. American Control Conf. (2002) 160–165
10. Bournez, K., Maler, O., Pnueli, A.: Orthogonal polyhedra: Representation and computation. In: Hybrid Systems: Computation and Control. Volume 1569 of LNCS., Springer (1999) 46–60
11. Puri, A., Borkar, V., Varaiya, P.: ϵ-approximations of differential inclusions. In: Hybrid Systems III. Volume 1066 of LNCS., Springer (1996) 363–376
12. Stursberg, O.: Analysis of switched continuous systems based on discrete approximation. In: Proc. 4^{th} Int. Conf. on Automation of Mixed Processes. (2000) 73–78
13. Chutinan, A., Krogh, B.: Verification of infinite-state dynamic systems using approximate quotient transition systems. IEEE Trans. on Automatic Control **46** (2001) 1401–1410
14. Botchkarev, O., Tripakis, S.: Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In: Hybrid Systems: Computation and Control. Volume 1790 of LNCS., Springer (2000) 73–88
15. Kurzhanski, A., Varaiya, P.: Ellipsiodal techniques for reachability analysis. In: Hybrid Systems: Computation and Control. Volume 1790 of LNCS., Springer (2000) 202–214
16. Chazelle, B.: An optimal convex hull algorithm in any fixed dimension. Discrete Comput. Geom. **10** (1993) 377–409
17. Avis, D., Bremner, D., Seidel, R.: How good are convex hull algorithms? Comput. Geom.: Theory and Appl. **7** (1997) 265–301
18. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journ. Computer and System Sciences **57** (1998) 94–124
19. Lafferriere, G., Pappas, G.J., Yovine, S.: A new class of decidable hybrid systems. In: Hybrid Systems: Computation and Control. Volume 1569 of LNCS., Springer (1999) 137–151
20. Pearson, K.: On lines and panes of closest fit to systems of points in space. Phil. Mag. **6** (1901)
21. Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24** (1933) 417–441
22. Jolliffe, I., ed.: Principal Component Analysis. Series in Statistics. Springer (1986)
23. Barequet, G., Har-Peled, S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. J. of Algorithms **38** (2001) 99–109
24. Klema, V.C., Laub, A.J.: The singular value decomposition: its computation and some applications. IEEE Trans. on Automatic Control **25** (1980) 164–176
25. Barber, C., Dobkin, D., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. on Mathematical Software **22** (1996) 469–483
26. Vandenberghe, L., Boyd, S., Wu, S.: Determinant maximization with linear matrix inequalities. SIAM Journ. Matrix Analysis and Applications **19** (1998) 499–533
27. Chutinan, A., Krogh, B.H.: Computational techniques for hybrid system verification. IEEE Trans. on Automatic Control **48** (2003)