Dynamics-Aware Subspace Identification for Decomposed Aggregation in the Reachability Analysis of Hybrid Automata

Viktorio S. el Hakim
v.s.elhakim@utwente.nl
Computer Architectures for Embedded Systems
University of Twente
Enschede, The Netherlands

Marco J. G. Bekooij
marco.bekooij@nxp.com
Algorithms and Software Innovation
NXP Semiconductors
Eindhoven, The Netherlands

ABSTRACT

Hybrid automata are an emerging formalism used to model sampled-data control Cyber-Physical Systems (CPS), and analyze their behavior using reachability analysis. This is because hybrid automata provide a richer and more flexible modeling framework, compared to traditional approaches. However, modern state-of-the-art tools struggle to analyze such systems, due to the computational complexity of the reachability algorithm, and due to the introduced overapproximation error. These shortcomings are largely attributed (but not limited) to the aggregation of sets.

In this paper we propose a subspace identification approach for decomposed aggregation in the reachability analysis of hybrid automata with linear dynamics. Our key contribution is the observation that the choice of a good subspace basis does not only depend on the sets being aggregated, but also on the continuous-time dynamics of an automaton. With this observation in mind, we present a dynamics-aware subspace identification algorithm that we use to construct tight decomposed convex hulls for the aggregated sets.

Our approach is evaluated on two practically relevant hybrid automata models of sampled-data CPS that have been shown to be difficult to analyze by modern state-of-the-art tools. Specifically, we show that for these models our approach can improve the accuracy of the reachable set by up-to 10 times when compared to standard Principal Component Analysis (PCA), for which finding a fixed point is not guaranteed. We also show that while the computational complexity is increased, a fixed-point is found earlier.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Theory of computation** → Formal languages and automata theory; • **Mathematics of computing** → *Numerical analysis*; **Mathematical optimization**; • **Hardware** → **Functional verification**; • **Software and its engineering** → **Formal methods**.

KEYWORDS

hybrid automata, reachability analysis, aggregation

ACM Reference Format:
Viktorio S. el Hakim and Marco J. G. Bekooij. 2020. Dynamics-Aware Subspace Identification for Decomposed Aggregation in the Reachability Analysis of Hybrid Automata. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '20), April 22–24, 2020, Sydney, NSW, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3365365.3382200

1 INTRODUCTION

In reachability problems of hybrid automata the growth of disjoint sets is a frequently encountered issue. This growth increases the complexity of the reachability algorithm exponentially for generic hybrid automata, thus preventing any use of a reachability tool within a reasonable amount of time.

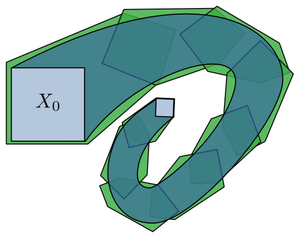

Figure 1: A flowpipe (blue) generated from the set X_0, and its overapproximation using segments (green).

The first reason for this exponential growth is due to the computation of the so-called flowpipe. Because the continuous-time trajectories are dense in time, an image of the flowpipe from an initial set of states cannot be computed exactly. Thus, during the flowpipe construction phase, it is overapproximated as a union of a set of segments, each generated for a fixed time-step, such that it covers a portion of the true flowpipe, see Figure 1. The smaller the time-step, the tighter each segment becomes at the cost of a larger number of segments.

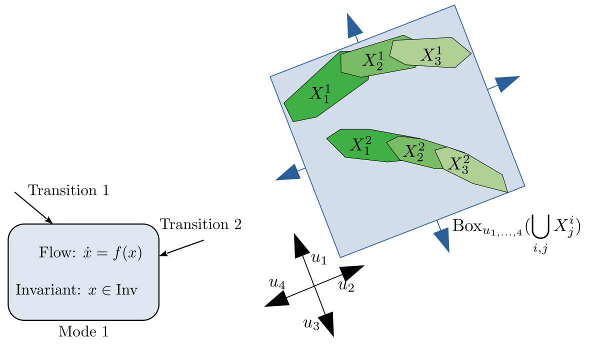

Figure 2: A mode with two incoming transitions (2a), and two sets of flowpipe segments from each transition, $X^1_{1,2,3}$ and $X^2_{1,2,3}$, respectively, aggregated using a box (2b).

The second reason is due to the accumulation of segments in a mode from incoming transitions. During the discrete transition phase, the flowpipe in each discrete state (mode) is intersected with a guard, and an image of the intersected segments under a so-called jump transformation is computed for each outgoing transition. On the receiving end, a mode with incoming transitions uses these as initial sets to compute a new flowpipe in the next iteration, see Figure 2 for a basic idea.

To prevent this exponential growth of segments, reachability algorithms typically utilize aggregation techniques, which derive significantly smaller numbers of set representations (aggregates) that tightly contain the segments, with the most popular representations being template polyhedra. However, these representations are based on fixed, manually selected templates and typically introduce a large overapproximation error, as shown in Figure 2b. This error propagates through each iteration of the algorithm, and can severely compromise its reliability by bloating the reachable set to the point where a fixed-point cannot be determined. This is particularly problematic for the verification of liveness properties, such as stability. On the other hand, convex hulls can be used to compute an exact aggregate of a union of disjoint sets, but the computational complexity is exponential with respect to the dimensionality of the sets. In such cases decomposed aggregation via subspace projections can decrease this complexity at the expense of overapproximation error. This error greatly depends on the selected subspaces.

In this paper we present a subspace identification approach for decomposed convex hull aggregation. The key contribution is the important observation that the choice of subspace basis for decomposed aggregation highly depends on the continuous dynamics. To be more precise, depending on the flow equation of the continuous-time state variables, an aggregate may contract faster, even if it is not tight with respect to the set it overapproximates. Subsequently we develop an optimization algorithm, which derives an orthogonal basis for the subspaces, such that the aggregate contracts faster. Specifically, given a set of sets, our approach finds a box aggregate, such that the overapproximated flowpipe generated from this aggregate converges faster to the exact flowpipe with respect to a set measure. Convex hulls are then computed of the projections of the sets onto the subspaces, and the decomposed sets are composed back using the Cartesian product. This decomposed style of aggregation in lower dimensional subspaces, brings a fair balance between accuracy and algorithmic complexity to the reachability algorithm. Our approach is applied to the class of hybrid automata with linear dynamics, where the flow and jump functions are linear with respect to the state variables.

In our case study we apply our technique to the reachability analysis of sampled-data control CPS [1, 2, 4, 10, 20, 24, 25, 29]. Specifically, we evaluate our approach on two practically relevant hybrid automata models where we consider full-state feedback control of the plant. We show that a good subspace basis in decomposed aggregation significantly improves the accuracy of the reachability algorithm by up-to a factor of 10.

The rest of this paper is organized as follows. In Section 2 we review related work. Section 3 gives a basic overview of our approach. Section 4 describes the reachability algorithm of hybrid automata with linear dynamics, and decomposed convex hull aggregation. In Section 5 we describe our subspace identification algorihtm. Our case study and results are presented in Section 6. Our conclusions are presented in Section 7.

2 RELATED WORK

In this section we discuss approaches that are closely related to our work and outline the differences.

A recently proposed method [6] uses PCA to determine the directions of template polyhedra overapproximations of the flowpipe segments. First, PCA is used to determine the directions of the template approximating the initial segment of a flowpipe. Subsequently, PCA is used to determine an oriented box template of the complete flowpipe, or partitions of it, because the size of the exact convex hull of the segments may grow large. However, the key difference with our work is that their subspace identification approach via PCA does not consider the continuous dynamics of the automaton. Furthermore, decomposed subspace based aggregation is not considered. Finally, in their approach aggregation is performed on individual flowpipes and their segments, rather than sets of unrelated flowpipe segments from different modes.

In the tool SpaceEx [12, 14] they use support function representations of the flowpipes, and a combination of template polyhedra and convex hulls are used for aggregation. Their aggregation and clustering strategy on the flowpipe level, which is referred to as the `STC` scenario in the tool, determines the time-step dynamically for each of the segments. As demonstrated by the authors, this can significantly reduce the number of segments, with a slightly increased overapproximation error. However, in contrast to our approach SpaceEx does not allow automatic selection of the template directions for aggregation. Instead, they are fixed and manually selected by the user. Additionally, the clustering technique presented in [14] is applied to segments during flow-pipe construction. Finally, decomposed aggregation is not used in the tool.

In [27] the authors propose an efficient Counterexample-Guided Abstraction Refinement (CEGAR) based approach to dynamically reduce overapproximation error. The idea, albeit similar to ours, is to automatically select suitable parameters of the reachability algorithm, so that verification can be performed more efficiently and accurately. The parameters considered are the state set representations, their spatial and algorithmic complexity, the time-step, the aggregation strategy and others. Then the reachability algorithm is executed repeatedly, starting with an initial configuration of parameters, and are refined with each execution until the verification process is complete, or the search space is exhausted. However, their approach is different from ours because the refinement of parameters occurs on each complete execution of the algorithm. In contrast, our subspace identification approach is applied on each iteration of the reachability algorithm. Additionally, they do not consider decomposed aggregation in their approach. As such, an optimal subspace basis for aggregation is not included in the parameter search space.

In [3, 9] the authors propose an aggregation and a de-aggregation technique to reduce the number of segments, such that a low overapproximation error is maintained. More precisely, the technique identifies spuriously aggregated sets, which are then de-aggregated so that the error is reduced. The set representations of the segments that are considered are generalized stars, while aggregation is performed by a combination of template polyhedra and convex hulls. However, the key difference with our approach is that they do not provide a method to automatically determine template directions. Additionally, while they utilize symbolic orthogonal projections [18], their aggregation approach is not decomposed using subspaces. Thus, a subspace identification technique is not considered.

3 BASIC IDEA

In this section we provide a basic overview of aggregation methods and our approach.

3.1 Aggregation using template polyhedra

Template polyhedra is a very popular way to aggregate sets and have been extensively used in state-of-the-art reachability tools, such as HyLaa [3, 9], SpaceEx [12] and Flow* [6, 7], due to their simplicity. The basic idea is to find support hyperplanes to the aggregated set, and form a convex polyhedron from their intersection, see Figure 2b. First an aggregated set $X \subset \mathbb{R}^n$ is projected onto a set of normal vectors $\{u_1, \ldots u_N\}$, called directions, and subsequently upper bounds, $\{b_1, \ldots, b_N\}$, are derived of the projections, i.e. $b_i = \max_{x \in X} \langle u_i, x \rangle$. Here $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. The pairs (u_i, b_i) then define the hyperplanes.

The box template is particularly useful, because its set of directions, $\{\pm u_1, \ldots, \pm u_n\}$, are orthogonal, and therefore form an orthogonal subspace basis for $\mathbb{R}^n$. Thus, when a set X is projected onto the subspaces spanned by the directions, the template can be incrementally refined to produce a tighter polyhedron, by adding more directions in the respective subspaces. This incremental approach has been used in SpaceEx [12], however here they use the standard basis.

However, the choice of directions heavily influences how well the reachability algorithm performs, because of overapproximation error. This choice is left to the user in state-of-the-art tools, and thus it often leads to very poor results. While PCA-based approaches have been used to dynamically determine directions [6], we show in our case study that PCA is not sufficiently accurate for practical applications.

3.2 Aggregation using convex hulls

Another approach for set aggregation is to compute convex hulls of sets. A big advantage over template polyhedra, is that convex hulls are the tightest possible aggregates of sets, since a convex hull consists of all the possible convex combinations of the sets' points. Unfortunately the computational complexity of the convex hull algorithm depends on the representation used for the aggregated sets, and their dimensionality. The algorithm has been shown to be most efficient for point sets, and $\mathcal{V}$-representations of polyhedra [5], however even then the complexity is exponential with respect to dimension.

3.3 Decomposed aggregation

An emerging approach that exploits the complexity/accuracy trade-off in aggregation using convex hulls, template polyhedra, etc, is decomposed aggregation, which we apply in this paper. The basic idea is to project the set onto lower dimensional subspaces of the parent space, perform the aggregation on the projections for each subspace, and compose the aggregated projections back into the original space using the Cartesian product. More formally, suppose that $X \subset \mathbb{R}^n$ is a set to aggregate, and let $\mathcal{W}_1, \ldots, \mathcal{W}_p$ be orthogonal subspaces of $\mathbb{R}^n$, such that $\mathbb{R}^n = \mathcal{W}_1 \oplus \cdots \oplus \mathcal{W}_p$. Then $P_{\mathcal{W}_i}(X)$ is a projection of X onto $\mathcal{W}_i$. Now let $\text{Aggr}(P_{\mathcal{W}_1}(X))$ be the new aggregate (box, convex hull, etc) of the projection of X onto subspace $\mathcal{W}_i$. The decomposed aggregate of X is then $\text{Aggr}(P_{\mathcal{W}_1}(X)) \times \cdots \times \text{Aggr}(P_{\mathcal{W}_p}(X))$.

The main advantage of decomposed aggregation is that the complexity to compute lower-dimensional aggregates is much lower at the expense of increased overapproximation error. The trade-off is controlled by selecting the size of the subspace partition.

3.4 Indentifying subspaces

Besides selecting the subspace partion, overapproximation error can be further reduced by correctly selecting the subspaces. As such, the identification of a suitable basis for the subspaces is crucial for the accurate performance of the reachability algorithm, and is usually done using PCA. A key observation that we have made, is that the choice of a good

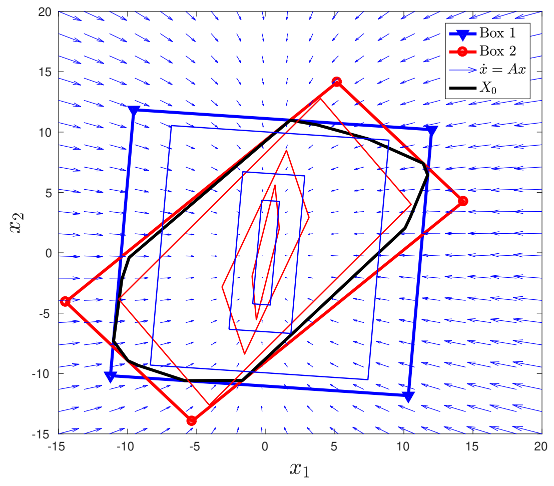

Figure 3: Contraction of two box aggregates.

basis required to derive a tight decomposed aggregate does not only depend on the sets being aggregated, but also on the continuous dynamics. Specifically, how fast the aggregate contracts depends also on the flow functions that govern the evolution of the state variables. This contraction of the aggregate directly affects how its approximated flowpipe converges to the exact flowpipe. We observe that if it contracts faster, then the propagated overapproximation error due to successive aggregations is reduced. We demonstrate this observation with the following example.

Consider a 2-dimensional state-space with the variable $x \in \mathbb{R}^n$ evolving according to the flow equation $\dot{x} = Ax, A \in \mathbb{R}^{n\times n}$, see Figure 3. Here are shown the vector field of A, an arbitrary compact set X_0, and two box aggregates, Box 1 and 2, of X_0 with respect to two orthogonal subspace bases. Thus the orthogonal vectors that span the subspaces are normal to the faces of the respective box aggregates. Box 2 specifically is derived using PCA. Additionally we show several contracting trajectory "snapshots" of the boxes at discrete moments in time. Note that a trajectory is generated by taking a point as initial state in the flow equation and simulating the system. What is observed, is that the points of Box 1 are approaching the origin faster than the ones of Box 2, despite that Box 2 seems tighter than the first one. Another interesting observation, is that Box 1 contracts along the flow field. As we show later in the paper the most important consequence is that the flowpipe of Box 1 converges faster to the flowpipe of X_0 than Box 2, and as such the cumulative overapproximation error is reduced.

By taking this important observation into account, our approach attempts to find a subspace basis which produces the tightest box aggregate that also contracts faster. The template is refined further via a decomposed convex hull, described later in the paper. Our approach thus has a clear advantage over standard PCA, since PCA does not use a contraction measure/metric in its optimization problem. As a result, PCA cannot guarantee that the reachable set contracts faster, or at all, an effect observed in our case study.

4 REACHABILITY OF HYBRID AUTOMATA WITH LINEAR DYNAMICS

In this section we provide a basic introduction to hybrid automata with linear dynamics and their reachability. We also define the decomposed convex hull operator.

4.1 Definition

We adopt a similar definition of a hybrid automaton with linear dynamics from [28]:

Definition 4.1 (Syntax of hybrid automata with linear dynamics). A hybrid automaton with linear dynamics is the tuple $\mathcal{H} = (\mathcal{Q}, q_0, \mathcal{X}, X_0, E, \mathcal{I}, \mathcal{G}, f, R)$, where $\mathcal{Q} = \{q_1 \dots q_l\}$ is a finite set of modes, with initial mode q_0; $\mathcal{X} \subseteq \mathbb{R}^n$ is a continuous-time state-space with a state variable $x(t) \in \mathbb{R}^n$ (t is omitted throughout the paper for clarity), and initial set $X_0 \subseteq \mathcal{X}$; $E \subseteq \mathcal{Q} \times \mathcal{Q}$ is a set of discrete transitions; $\mathcal{I} : \mathcal{Q} \to 2^{\mathcal{X}}$ assigns a polyhedral invariant set $\mathcal{I}(q) = \{x \in \mathcal{X} \mid C_q x \le c_q\}$ for each mode $q \in \mathcal{Q}$; similarly, $\mathcal{G} : E \to 2^{\mathcal{X}}$ assigns a guard set $\mathcal{G}(e) = \{x \in \mathcal{X} \mid G_e x \le g_e\}$ for each transition $e \in E$; $f : \mathcal{Q} \times \mathcal{X} \to \mathcal{X}, f(q, x) = A_q x, A_q \in \mathbb{R}^{n\times n}$, is a vector field assigned for each mode, such that if $x \in \mathcal{I}(q)$ for the active mode $q \in \mathcal{Q}$, then $\dot{x} = f(q, x)$; $R : E \times \mathcal{X} \to \mathcal{X}, R(e, x) = J_e x + j_e, J_e \in \mathbb{R}^{n\times n}, j_e \in \mathbb{R}^n$, is an affine jump transformation for an enabled transition $e = (q, q')$, such that if $x \in \mathcal{G}(e)$, then the new state is $x' = R(e, x)$ if the transition occurs.

We do not provide a description of the formal semantics for hybrid automata, such as their execution, since this falls out of the scope of the paper, and instead refer the reader to [3, 12, 28].

4.2 Reachability of hybrid automata

A standard reachability algorithm of hybrid automata can be summarized with the following steps:

(1) Compute an invariant-satisfying flowpipe from the current initial set for each mode.
(2) For each mode and each outgoing transition, intersect the flowpipe with a guard, and apply the jump transformation to the intersection. The result is used as initial set in the next iteration per destination mode.
(3) For each mode, apply aggregation to its union of new initial sets.
(4) Repeat steps 1-4 until a fixed point condition is satisfied.

A more detailed description of the algorithm is provided in Algorithm 1.

4.3 Flowpipe computation

Because understanding how a flowpipe is computed is important for our approach, we briefly describe the standard

Algorithm 1 Reachability of hybrid automata

```
function $(Q, X) \leftarrow$ REACHABILITY$(\mathcal{H}, T, \hat{k})$
    $Q_0 \leftarrow \{q_0\}$                                        ▷ Initialization
    $X_0^{q_0} \leftarrow X_0$
    $\forall q \in \mathcal{Q} \setminus \{q_0\} : X_0^q \leftarrow \emptyset$
    for $k = 1, \ldots, \hat{k}$ do
        for $q \in Q_{k-1}$ do                                ▷ Flowpipe computation
            $\Psi_k^q \leftarrow$ flowpipe$(X_{k-1}^q, T) \cap \mathcal{I}(q)$
        end for
        $Q_k \leftarrow Q_{k-1}$
        $\forall q \in \mathcal{Q} : X_k^q \leftarrow \emptyset$
        for $e = (q, q') \in E, q \in Q_{k-1}$ do             ▷ Transitions
            $X_k^{q'} \leftarrow X_k^{q'} \cup R(e, \Psi_k^q \cap \mathcal{G}(e))$
            $Q_k \leftarrow Q_k \cup \{q'\}$.
        end for
        for $q \in Q_k$ do                                    ▷ Aggregation
            $X_k^q \leftarrow$ aggregate$(X_k^q)$
        end for
        $X_k \leftarrow \bigcup_{q \in Q_k} X_k^q$
        if $(Q_k, X_k) = (Q_{k-1}, X_{k-1})$ then             ▷ Fixed point
            return $(Q_k, X_k)$.
        end if
    end for
    return $(Q_{\hat{k}}, X_{\hat{k}})$.
end function
```

computation process here. For each reachable mode $q \in \mathcal{Q}$ in the current iteration k, the reachable set of the continuous time variables at q, say $X_k^q \subseteq \mathcal{X}$, is taken as initial set. This initial set is then used to derive the set of all of the possible forward reachable trajectories from X_k^q up-to some time $t \in \mathbb{R}_+$, which is the exact flowpipe. Here we remind that a trajectory, $\xi : \mathbb{R} \times \mathcal{X} \to \mathcal{X}$, is a function that satisfies the differential equation $\dot{x} = A_q x|_{x=\xi}$, with initial state $x(0) = x_0 \in X_k^q$. For clarity, we will refer to the initial set as X_0 instead of X_k^q, and let $A = A_q$ for the rest of this section. Because the flow function is linear, this trajectory is exactly derived as $\xi(t, x_0) = e^{At}x_0$. Thus, with a little abuse of notation, the exact flowpipe from the set X_0 at any time t is:

$$\Psi(t, X_0) = e^{At}X_0.$$

Because this set is dense in time it cannot be computed exactly, and instead it is overapproximated by a set of segments. First, the trajectory is discretized using a time step $\delta \in \mathbb{R}_+$. Then an initial segment Ω_0 is computed, such that $\forall t \in [0, \delta] : \Psi(t, X_0) \subseteq \Omega_0$. The most common approach is by bloating, where:

$$\Omega_0 = \mathrm{conv}(X_0 \cup e^{A\delta}X_0) \oplus \mathcal{B}_\alpha,$$

where $\alpha \in \mathbb{R}_+$ is a bloating constant, $\oplus$ is the Minkowski sum, conv$(\cdot)$ is the convex hull of a set and $\mathcal{B}_r$ is a ball with radius $r > 0$, see [15]. Then each segment for any $N \in \mathbb{N}$ is recursively computed as:

$$\Omega_i = e^{A\delta}\Omega_{i-1},\ i = 1, \ldots, N. \tag{1}$$

As shown in [15, 16], $\forall t \in [0, N\delta] : \Psi(t, X_0) \subseteq \bigcup_{i=0}^N \Omega_i$. We note that even though making δ smaller results into tighter segments, there is a saturation point beyond which the overapproximation error is not significantly reduced, while the number of segments increases substantially. Most importantly, this tightening of the flowpipes does not improve their subspace identification and aggregation, since their topology and structure remain unchanged.

4.4 Decomposed convex hull aggregation

As discussed earlier, aggregation is applied to the flowpipe segments by grouping them with a new set representation. Since in this paper we use the so called decomposed convex hull to derive this representation, we provide a formal definition below.

Definition 4.2 (Decomposed convex hull). Let $\mathcal{J} = \{J_1, \ldots, J_m\}$ be a partition of $\{1, \ldots, n\}$, such that $i \neq j \implies J_i \cap J_j = \emptyset$ and $\bigcup_i J_i = \{1, \ldots, n\}$. Furthermore, let $U \in \mathbb{R}^{n \times n}$ be an orthogonal matrix, then for each $J_i = \{j_1, \ldots, j_p\} \in \mathcal{J}$ we construct a matrix $U_{J_i} \in \mathbb{R}^{n \times p}$, such that $U_{J_i} = \begin{pmatrix} u_{j_1} & \cdots & u_{j_p} \end{pmatrix}$, where $u_j \in \mathbb{R}^{n \times 1}$ is the j-th column of U. Then the decomposed convex hull of a set $X \subset \mathbb{R}^n$, is the Cartesian product:

$$\mathrm{dconv}(X, U, \mathcal{J}) = \bigtimes_{i=1}^{m} U_{J_i} \,\mathrm{conv}(U_{J_i}^\top X). \tag{2}$$

We call the set $\mathcal{J}$ a subspace partition, and use it to select the projections of X onto the subspaces spanned by U, that are individually aggregated using the convex hull. The result is a set of convex hulls in each subspace, that are composed back in the original space using the Cartesian product. For example, when $\mathcal{J} = \{\{1\}, \{2\}, \ldots, \{n\}\}$, dconv$(X, I, \mathcal{J})$ is the standard basis bounding box of X, where I is the identity matrix.

5 SUBSPACE IDENTIFICATION

In this section we present our subspace identification approach.

5.1 Notation and definitions

We assume that $\mathbb{R}^n$ is equipped with the infinity vector norm $\|x\| = \max_i |x_i|, x \in \mathbb{R}^n$, and define the norm of a compact subset $A \subset \mathbb{R}^n$ induced by $\|\cdot\|$ as $\|A\| = \max_{a \in A \cup \{0\}} \|a\|$. We denote with $\mathrm{O(n)} = \{S \in \mathbb{R}^{n \times n} \mid S^\top S = SS^\top = I\}$ the set of orthogonal matrices, where $\cdot^\top$ is the transpose of a matrix, and with $\mathrm{SO}(n) = \{S \in \mathrm{O(n)} \mid \det(S) = 1\}$ the set of rotation matrices. Given a $U \in \mathrm{O(n)}$ with columns u_i, and a set $X \subseteq \mathbb{R}^n$, then:

$$\mathrm{Box}_U(X) = \{x' \in \mathbb{R}^n \mid \forall i \in \{1, \ldots, n\} : \\ \langle u_i, x' \rangle \leq \max_{x \in X} \langle u_i, x \rangle \text{ and } \langle u_i, x' \rangle \geq \min_{x \in X} \langle u_i, x \rangle\}, \tag{3}$$

is the bounding box of X with respect to U, with $\pm u_i$ normal to its facets. The matrix-valued function skew : $\mathbb{R}^{n(n-1)/2} \to$

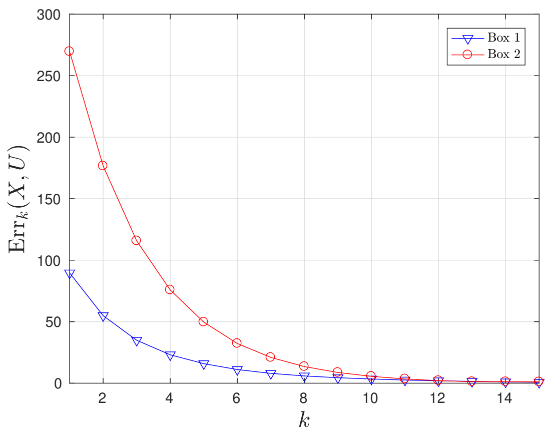

Figure 4: Convergence error for the example in Figure 3.

$\mathbb{R}^{n\times n}$ is defined as:

$$\operatorname{skew}(v) = \begin{pmatrix} 0 & v_1 & \cdots & v_{n-1} \\ -v_1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & v_{n(n-1)/2} \\ -v_{n-1} & \cdots & -v_{n(n-1)/2} & 0 \end{pmatrix},$$

and it constructs a skew-symmetric matrix from a $n(n-1)/2$-dimensional vector.

5.2 Identification using PCA

PCA is the most popular dimensionality reduction and subspace identification approach, used also in reachability algorithms [6, 28]. For the sake of completeness, we give a minimal definition of the method here.

Given a finite set of points, $\{x_1, \ldots, x_N\} \subset \mathbb{R}^n$, represented by a matrix $X = \begin{pmatrix} x_1 & \cdots & x_N \end{pmatrix} \in \mathbb{R}^{n\times N}$, then the standard 2-norm PCA optimization problem is:

$$\operatorname*{argmax}_U \{\frac{1}{2}\operatorname{tr}(U^\top X X^\top U) \mid U \in \mathrm{O(n)}\}, \tag{4}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix. The columns of optimal matrix U span 1-dimensional subspaces, such that the variance of the points' projections onto the subspaces is maximized. Equivalently, U forms a basis for $\operatorname{aff}(X) - x_i, i = 1, \ldots, N$, where $\operatorname{aff}(\cdot)$ is the affine hull of a set. The optimization problem is efficiently solved with the singular value decomposition:

$$X = U\Sigma V^\top. \tag{5}$$

An issue with the standard 2-norm PCA is its sensitivity to outliers [23], and thus a key reason for its poor performance in aggregation due to the possible existence of spread out disjoint sets.

5.3 Dynamics-aware identification

Earlier in the paper, we argued that a good subspace basis is very beneficial for the selection of directions of template polyhedra, and for computing decomposed convex hulls. The initial intuition is to choose a basis, such that the aggregate tightly contains the aggregated set. This is usually formulated as the problem of finding a minimum volume bounding aggregate, such as an ellipsoid or box. However, we observed that the continuous dynamics of the automaton also influence the choice of optimal basis, and demonstrated that counter to the initial intuition, a tighter template is not necessarily better in terms of convergence. We also highlighted the significance of the bounding box template as a building block for more refined aggregates. Most importantly, its directions form an orthogonal subspace basis. For these reasons, our dynamics-aware subspace identification approach uses bounding boxes to derive an orthogonal basis. We now formulate and define the optimization problem of our approach.

We start first by defining a contraction measure of a set. Let $X \subset \mathbb{R}^n$ be a set, $U \in \mathrm{O(n)}$ an orthogonal basis matrix and $\Phi \in \mathbb{R}^{n\times n}$ a discrete-time state-transition matrix. Then by (1), the discrete-time flowpipe of X as initial set is $X_{k+1} = \Phi X_k, k = 0, 1, \ldots$, with $X_0 = X$. A similar relation holds for the bounding box $X_U = \mathrm{Box}_U(X)$. We then construct standard basis boxes $\mathrm{Box}_I(X)$ and $\mathrm{Box}_I(X_U)$, and use them to measure the contraction of the flowpipe. We call the sequence $(\mathrm{Err}_k(X, U))_{k=1}^L, L \in \mathbb{N}$, where:

$$\mathrm{Err}_k(X, U) = \operatorname{vol}(\mathrm{Box}_I(\Phi^k X_U)) - \operatorname{vol}(\mathrm{Box}_I(\Phi^k X)), \tag{6}$$

the contraction error between the discrete flow-pipe of X and its box template X_U with respect to U, where $\operatorname{vol}(X)$ is the m-dimensional volume of a set, $m = \dim \operatorname{aff}(X)$. The reason we define the error this way, is because $\operatorname{vol}(\Phi^k X_U) = \det(\Phi)^k \operatorname{vol}(X_U)$. Thus in this case our problem would be reduced to just finding the minimum volume bounding box, which is independent of the matrix Φ. On the other hand measuring the volume difference with respect to the standard basis also allows measuring the contraction due to Φ. We also note that this is one of many estimates of the contraction error. However this error estimate is easy to compute for box templates, as we show later, compared to others.

Two error sequences for the box example in Figure 3 are shown in Figure 4, given matrices U_1 and U_2. The plot indicates that the sequence of errors indeed converges to zero faster for the first box, compared to the second. More importantly, each value of the first sequence is smaller than each value of the second sequence, which indicates that U_1 is a desirable basis matrix. This is despite that $\operatorname{vol}(\mathrm{Box}_{U_1}(X)) > \operatorname{vol}(\mathrm{Box}_{U_2}(X))$. We can thus give a formal definition to the subspace identification problem.

Definition 5.1 (Dynamics aware subspace identification). A subspace basis represented by a matrix $U^* \in \mathrm{O(n)}$ is optimal for a set $X \subset \mathbb{R}^n$ in the sense of the contraction error, if $\forall U \in \mathrm{O(n)} : \sum_{k=1}^L \mathrm{Err}_k(X, U^*) \leq \sum_{k=1}^L \mathrm{Err}_k(X, U)$ for some $L \in \mathbb{N}$.

More informally want to find a matrix U^*, such that this error is minimized for every segment in the flowpipe over a finite discrete time interval $[1, L]$. This is a scalarized multi-objective optimization problem with equal weighting for each objective. In this case the initial box template represented by U^* converges the fastest to the set X. This can be formulated as the following optimization problem:

$$\underset{U}{\operatorname{argmin}}\{\sum_{k=0}^{L}\operatorname{vol}(\operatorname{Box}_I(\Gamma\Phi^k X_U)) \mid U \in \mathrm{O(n)}\}, \tag{7}$$

where we additionally introduce a scaling matrix $\Gamma \in \mathbb{R}^{n\times n}$, and thus summation is done over $k \in \{0,\ldots,L\}$.

5.4 Simplification using zonotopes

Here we derive an analytical expression for the objective function of (7) by using a zonotope representation of $\Phi^k X_U$, for which the error sequence is easy to compute.

Given a matrix $V \in \mathbb{R}^{n\times m}$ and a point $c \in \mathbb{R}^n$, then a zonotope is the set:

$$Z(c,V) = \{c + Vz \mid z \in [-1,1]^m\}, \tag{8}$$

which is the image of the m-dimensional unit hypercube under V, with center c. Note that $\Phi Z(c,V) = Z(\Phi c, \Phi V)$. Additionally $\operatorname{Box}_I(Z(c,V)) = [m_1, M_1] \times \cdots \times [m_n, M_n]$, where for all $i \in \{1,\ldots,n\} : m_i = |c_i| - \sum_{j=1}^{m}|V_{ij}|$, and $M_i = |c_i| + \sum_{j=1}^{m}|V_{ij}|$. Thus:

$$\operatorname{vol}(\operatorname{Box}_I(Z(c,V))) = \prod_{i=1}^{n}(M_i - m_i) = 2^n \prod_{i=1}^{n}\sum_{j=1}^{m}|V_{ij}|. \tag{9}$$

We can thus represent $\Phi^k X_U$ for all $k \in \{0,\ldots,L\}$ as:

$$\Phi^k X_U = \frac{1}{2}Z(\Phi^k U d_X^+, \Phi^k U \operatorname{diag}(d_X^-)), \tag{10}$$

where for all $i \in \{1,\ldots,n\}$:

$$d_{X\,i}^{\pm} = \max_{x\in X}\{\langle u_i, x\rangle\} \pm \min_{x\in X}\{\langle u_i, x\rangle\}. \tag{11}$$

This makes sense, because a box is simply a zonotope with V being an orthogonal matrix. Then for any matrix $G \in \mathbb{R}^{n\times n}$, $\operatorname{vol}(\operatorname{Box}_I(GX_U))$ can be derived using equations (9), (10), and (11) as the function:

$$g(U,G,X) = \operatorname{prod}(\operatorname{abs}(GU)\operatorname{abs}(d_X^-)), \tag{12}$$

where $\operatorname{abs}(\cdot)$ is the element-wise or component-wise absolute value function of a matrix or vector, respectively, and $\operatorname{prod}(\cdot)$ is the product of components of a vector. Thus the optimization problem (7) is reformulated to the equivalent simplified problem:

$$\underset{U}{\operatorname{argmin}}\{f(U) = \sum_{k=0}^{L} g(U, \Gamma\Phi^k, X) \mid U \in \mathrm{O(n)}\} \tag{13}$$

5.5 Optimization using Sequential Monte Carlo

The optimization problem is non-linear and non-convex, while the objective function of (13), $f(U)$, is non-differentiable. Additionally, standard gradient methods are not easily applied here due to the orthogonality constraint. For this reason, we use an evolutionary algorithm to derive sub-optimal solutions of the problem using Sequential Monte Carlo (SMC) [8]. The algorithm can be summarized in the following steps:

(1) Generate a set of N candidate solutions (particles) $\{U_1,\ldots,U_N\}$.
(2) Evaluate the objective function and compute a weight for each candidate, i.e. $w_i = f(U_i), i \in \{1,\ldots,N\}$.

Algorithm 2 Dynamics-aware subspace identification

```
1: function $U \leftarrow$ OPTIMBASIS($X, \Phi, \Gamma, L, N, \rho, r$)
2:     $U_P \leftarrow \mathrm{PCA}(X)$                          ▷ Using equation (5)
3:     $X' \leftarrow U_P^\top X$.
4:     for $i \in \{1,\ldots,N\}$ do                    ▷ Initialize particles
5:         $\theta \sim \mathbb{U}(-r\pi/2, r\pi/)^{n(n-1)/2}$
6:         $U_i \leftarrow \exp(\mathrm{skew}(\theta))$
7:     end for
8:     $v \leftarrow \infty, U \leftarrow I$
9:     loop
10:        for $i \in \{1,\ldots,N\}$ do                ▷ Compute weights
11:            $w^i \leftarrow \sum_{k=0}^{L} g(U_P U_i, \Gamma\Phi^k, X')$
12:        end for
13:        if $\min_i\{w_i\} \geq v$ then
14:            return $U_p U$
15:        end if
16:        $v \leftarrow \min_i\{w_i\}$                  ▷ Update objective value
17:        $j = \mathrm{argmin}_i\{w_i\}$
18:        $U \leftarrow U_j$
19:        for $i \in \{1,\ldots,N\}$ do                ▷ Construct pdf
20:            Map $w_i$ to the interval $[0,1]$
21:            $w_i \leftarrow e^{-\rho w_i}$
22:        end for
23:        $\forall i \in \{1,\ldots,N\} : c_i \leftarrow \frac{\sum_{j=1}^{i} w_j}{\sum_{j=1}^{N} w_j}$       ▷ Construct cdf
24:        $t \sim \mathbb{U}(0, \frac{1}{N})$
25:        for $i \in \{1,\ldots,N\}$ do     ▷ Resample (Systematic)
26:            $j \leftarrow \mathrm{argmin}_k\{c_k - t \mid c_k - t \geq 0\}$
27:            $t \leftarrow t + \frac{1}{N}$
28:            $U_i^* \leftarrow U_j$
29:        end for
30:        for $i \in \{1,\ldots,N\}$ do                ▷ Perturb particles
31:            $\theta \sim \mathbb{U}(-r\pi/2, r\pi/2)^{n(n-1)/2}$
32:            $U_i \leftarrow \exp(\mathrm{skew}(\theta))U_i^*$
33:        end for
34:    end loop
35: end function
```

(3) Construct a discrete Probability Density Function (PDF) from the weights, and use it to resample the particle set.
(4) Perturb the particles in order to avoid sample impoverishment.
(5) Repeat steps 2-4 until the objective function stops decreasing for all of the particles.

The algorithm is also known as the Particle Filter (PF), and it is a widely used method in signal processing due to its effectiveness for estimating the state of non-linear and non-Gaussian systems. Unfortunately, it has a high computational cost and may suffer from the curse of dimensionality.

Fortunately $f(U)$ is relatively cheap to evaluate, and "periodic" with respect to U. Specifically, for any n-dimensional rotation R around a $(n-2)$-dimensional subspace of $\mathbb{R}^n$ by π it holds that $f(U) = f(RU)$, due to the symmetry of the box template. This property considerably reduces the search space of the algorithm. In our experiments, we observe that a very small number of particles is required to reach

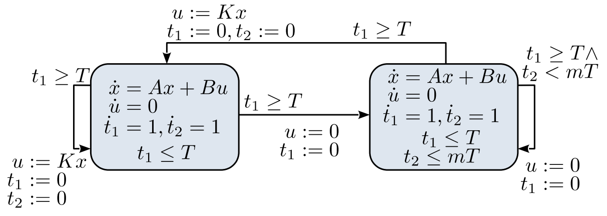

Figure 5: Hybrid automaton of a sampled-data CPS with information loss.

a good solution. The complete algorithm is summarized in Algorithm 2.

Here we would like to point out that an important component of the algorithm is the perturbation of the particle matrices, such that they remain orthogonal. Perturbation is required in order to ensure that the algorithm is not "stuck" in a local a minimum, and is achieved by generating randomized rotation matrices $R_i \in \mathrm{SO(n)},\ i \in \{1, \dots, N\}$ [17]. We use a very simple method to generate random rotations: first a sample is drawn from the multivariate uniform distribution, i.e. $\theta \sim \mathbb{U}(-r\pi/2, r\pi/2)^{n(n-1)/2}$, where $0 < r \leq 1$; next we construct the skew symmetric matrix $S = \mathrm{skew}(\theta)$; finally the rotation matrix is $R = \exp(S)$, where $\exp(\cdot)$ is the matrix exponential, and is used to perturb a given particle.

Second, while we try to improve over the standard PCA, we still use it in our algorithm to initialize the particle set. This is done for the following reasons: 1) an initial basis matrix U_P is fast to compute; 2) it reduces the search space considerably.

6 CASE STUDY

In this section we present our case study, where we evaluate and compare our approach with PCA on the reachability of two types of sampled-data control CPS, modeled using hybrid automata.

6.1 Motivation

Sampled-data control CPS [1, 2, 4, 10, 20, 24, 25, 29] are systems where the controller is implemented digitally on a computer, and is typically executed on a networked multiprocessor architecture. As such, the sampling and actuation moments are determined by the control software and temporal disturbances, such as varying processing latency, processor workload, task scheduling, etc. Traditionally, during the design and analysis of sampled-data control CPS, temporal bounds and workload characterizations of the system are derived first using well known real-time analysis techniques, and are used afterwards to design and analyze the controller separately. This in turn is done using classical techniques from control theory, such as Common Quadratic Lyapunov Function (CQLF), Linear Matrix Inequalities (LMIs), etc [20, 25, 29]. However, these techniques are quite limiting when considering a system involving complex interactions. For example, in a networked control system [25, 29], a timed change in the network topology affects the transmission delay, packet loss and the sampling period. Such effects cannot be

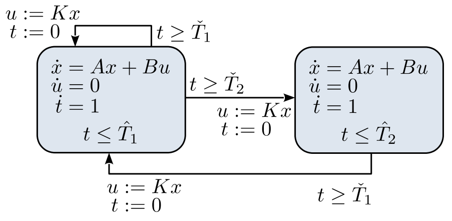

Figure 6: Hybrid automaton of a sampled-data CPS with uncertain sampling rate.

described and analyzed easily using traditional design methods. Hybrid automata however provide a more compatible modeling framework for sampled-data CPS, such that they can be analyzed using reachability [1, 4, 10, 13].

Unfortunately, the reachability problem for Hybrid Automata (HA) is in general undecidable and properties such as asymptotic stability can be verified only for a certain subset of models [10]. The subset of such models that we consider in this case study are timed models, where the switching between modes is entirely time-driven. Many practical system archetypes exist that can be modeled and analyzed using such models, a few of which we consider in this paper and describe below. We believe that these models demonstrate the importance of good aggregation in reachability, since it can greatly affect the accuracy of the reachability algorithm, an observation we demonstrate in our results. In fact, as shown in [4, 10, 11], state-of-the-art tools such as SpaceEx and Flow* consistently fail to analyze such models.

6.2 Control with uncertain sampling rate

For our first case, we consider a cyber-physical control system with uncertainty in the sampling and actuation times. The following assumptions are made for the model: 1) upper and lower bounds on the difference between the sampling moments are known; 2) the bounds are allowed to switch to different values. Such a model can be used to describe and analyze a multiprocessor based computer-controlled system with a switching workload characterization. To be more precise, the sampling moments are determined by the execution time of the control algorithm. On the other hand the difference between the bounds is dictated by the processors' workloads. For example, one may assume that the difference is large when the workload on the processors is high, and small otherwise. Such characterizations have been shown to be more accurate approximations for analysis, than the single upper and lower bound characterization on the execution time, i.e. the so-called Worst-Case Execution Time (WCET) characterization [19].

The hybrid automaton for this system is shown in Figure 6. Here we use a so-called "clock" variable t that progresses linearly in time to model the sampling and switching behavior of the controller. Two modes are used to represent two workloads of the processor, low and high load, with sampling bounds $\check{T}_1 < \hat{T}_1$ and $\check{T}_2 < \hat{T}_2$, respectively, with the first assumed the normal mode of operation. Thus if the ideal sampling period of the controller is T, then we let

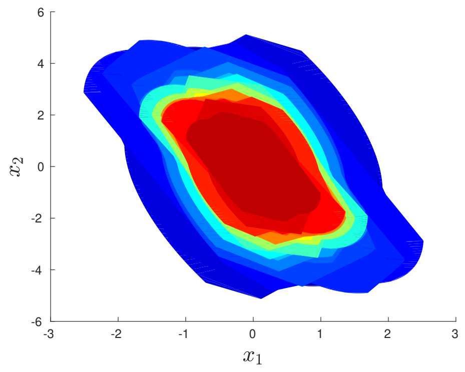

Figure 7: Reachable set of the first hybrid automaton from run 1 for variables $x_{1,2}$ at iteration $k = 1, \ldots, 15$ (blue to red).

$\check{T}_1 \le t \le \hat{T}_1 = T$ in the first mode and $\check{T}_2 = T \le t \le \hat{T}_2$ in the second. The controller may switch to the high load mode at any time, as is usually the case for real-time multiprocessor based control systems [26]. The variable x is the state of the plant, and we consider full-state feedback control via the piece-wise constant actuation u (Zero-Order Hold (ZOH)).

6.3 Periodic control with sample loss

In the second case we consider a networked control system with periodic sampling and actuation. The model is very similar to the first case. However, here we assume that sampling data may be lost during transmission, due to packet drop. For example in the case of a full-state feedback control, only a subset of the state-variables may be received. A packet may be lost at any time, but to keep the model realistic, we also assume that no-more than m consecutive packet drops can occur. A similar system was studied in [20].

The hybrid automaton of this model is shown in Figure 5, and is very similar to the one discussed previously. However, the key difference is that we add an additional clock variable t_2 to monitor the number of packet losses. The variable t_1 models the sampling and actuation times of the controller, as usual. The first mode is the standard, periodic mode of operation. In the second mode, the complete sensor data packet is lost, and no actuation is applied to the plant, i.e. $u = 0$.

6.4 Evaluation method

To evaluate our approach, we use the implementation of the reachability algorithm as described in [11], because it allows computing the reachable set of clock and other variables independently. Thus it is suitable for the analysis of our models. However instead of the standard convex hull algorithm used in their implementation, we use our own decomposed convex hull and subspace identification algorithm. Specifically, at line 12 of Algorithm 2 in [11], the operation Conv is replaced by dconv(·) in equation (2), and preceded by OPTIMBASIS from Algorithm 2, or by PCA as defined in equation (5), to derive the projection matrix. Note that aggregation in this case is only applied to non-clock variables. Algorithm 2 and [11] are implemented using MATLAB, and are executed on a computer with 16GB RAM and a quadcore Intel™ processor. We use a time step $\delta = 0.01$ for all of the evaluation runs of the models. The parameters used for our algorithm are summarized in Table 1, and are selected heuristically. To demonstrate the effect of the overapproximation error on the reachable set X_k of continuous variables (excluding the clocks), we compute the set norm $e_k = \|X_k\|$ (see Section 5) on each iteration of the algorithm, with $X_0 = [-1, 1]^n$.

Parameter	Aut 1	Aut2
N	100	100
ρ	40	40
r	1/30	1/30
L	5	10
Γ	$e^{A(7\times\delta)}$	$e^{A(7\times\delta)}$
Φ	$e^{A\delta}$	$e^{A\delta}$

Table 1: Parameters used for Algorithm 2.

The plant model used for both hybrid automata is described by the following matrices:

$$A = \begin{pmatrix} 19.24 & 10.72 & -5.67 \\ -84.95 & -13.56 & 18.53 \\ 50.7 & 11.53 & -13.68 \end{pmatrix}, B = \begin{pmatrix} -3 \\ 0.5 \\ 1 \end{pmatrix}.$$

A discrete time full-state feedback control matrix has been designed using MATLAB's command `dlqr` for the plant, given a sampling time $T = 0.1s$, as $K = \begin{pmatrix} -1.39 & -0.463 & 1.0446 \end{pmatrix}$. Additionally for the first hybrid automaton, $\check{T}_1 = 0.06s$ and $\hat{T}_2 = 0.15s$, while for the second $m = 2$.

6.5 Results

Run	Time (PCA)		Time (our)		$\mathcal{J}$
	Aut 1	Aut 2	Aut 1	Aut 2	
1	0.31s	0.59s	2.02s	5.57s	$\{\{1\},\{2\},\{3\},\{4\}\}$
2	0.8s	0.81s	6.18s	9.16s	$\{\{1,2\},3,4\}$
3	0.43s	0.77s	6.42s	9.81s	$\{1,\{2,3\},4\}$
4	0.3s	0.6s	1.75s	5.16s	$\{1,2,\{3,4\}\}$
5	0.86s	1.04s	7.6s	9.92s	$\{\{1,3\},2,4\}$
6	5.72s	2.88s	66.3s	26.77s	$\{\{1,2,3\},4\}$

Table 2: Evaluation run times.

A total of six runs are performed for each hybrid automaton using standard PCA and our approach for subspace identification prior to aggregation. The plots of the set norm of the reachable set for each automaton are shown in Figure 8 and Figure 9, respectively. The run times, as well as the subspace partitions used for each run, are summarized in Table 2. The reachable set of the first automaton from run 1 is shown in Figure 7. From the figure it is clear that our subspace identification algorithm outperforms the standard PCA with respect to the norm of the reachable set, which can be seen to expand indefinitely in e.g. run 1. An exception to this is run 6, where both methods perform equally well. This is because in this case the convex hull is used on all 3 dimensions of the state variables of the plant, and aggregation does not benefit

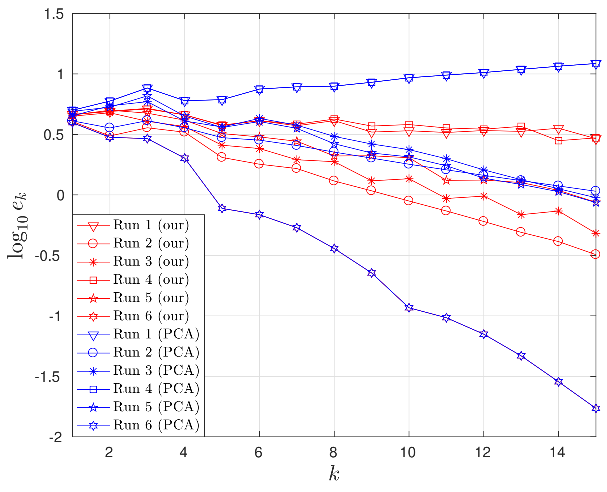

Figure 8: Results for the first hybrid automaton.

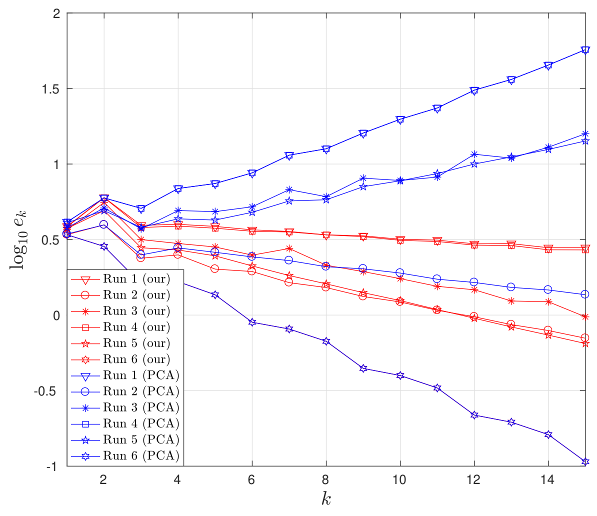

Figure 9: Results for the second hybrid automaton.

from decomposition. What this shows is that the reachability algorithm greatly benefits from decomposed aggregation if a good subspace basis is used. Additionally, one can see that the convergence rate is also determined by the subspace partition $\mathcal{J}$. Thus, as a future consideration, it is beneficial to understand how to select the correct partition automatically, instead of manually.

However, our method is greatly outperformed by PCA in terms of run-time. Indeed, a drawback of our method is its computational complexity, as observed from the run-times of the reachability algorithm. While the implementation of Algorithm 2 may not be optimal, we believe that the key reason for this shortcoming is because of the point set representation used in [11]. We observe that the number of points can grow quite large, such that even for a small number of particles, our algorithm slows down considerably. The decomposed convex hull also slows down due to the number of points. We thus consider the following future improvements:

(1) Use a different set representation, such as zonotopes [15, 16] or ellipsoids [22].
(2) Instead of using all of the points, one can consider the so-called core set [21], which a subset of the original set that is sufficient to find an optimal solution.
(3) Use a combination of convex hulls and template polyhedra.

Additionally, the rate of convergence of our algorithm can be improved, while reducing the number of particles, by considering gradient approximations since we observe that for a large number of points, the objective function of (13) is relatively smooth.

Finally, we observed that the manual selection of parameters for the algorithm is difficult and greatly influences the accuracy of the reachable set. The most important parameters are the scaling matrix Γ, and the number of discrete-time iterations L.

7 CONCLUSION

In this paper, we presented an approach for identifying subspaces for accurate decomposed set aggregation in reachability analysis of hybrid automata. Specifically, our approach allows applying decomposed aggregation of the sets in the identified subspaces using convex hulls, such that the over-approximating aggregate contracts faster. While identifying a good subspace basis using our approach comes with a larger computational cost, it is compensated by the fact that a fixed point is found earlier. We demonstrated this in our case study by applying our approach on two practical sampled-data control CPS models, and making a direct comparison with PCA. In particular, we showed that with our approach the reachability algorithm achieves up-to 10-times tighter reachable sets, and that with PCA a fixed-point is not guaranteed to be found.

The key observation that we have made in our work, is that the choice of subspace basis used to derive a template aggregate is dependent on the continuous dynamics of the automaton. More precisely, when an initial set is aggregated for a mode using the box template, then its points contract faster or slower towards the origin, depending on the orientation of the box. We exploit this observation in our approach, and developed an algorithm to determine a suitable orthogonal basis, such that the flowpipe of the box template converges faster to the flowpipe of the aggregated set. The algorithm is based on the SMC optimization technique, because the objective value of the optimization problem is highly non-linear and non-convex.

For future work, we would like to improve the implementation of our optimization algorithm to solve the dynamics-aware subspace identification problem more efficiently. Additionally, we would like to evaluate our approach on more set representations. Finally, we want to investigate how to select the correct parameters of the algorithm automatically.

REFERENCES

[1] M. Al Khatib, A. Girard, and T. Dang. 2015. Stability Verification of Nearly Periodic Impulsive Linear Systems using Reachability Analysis. *IFAC-PapersOnLine* 48, 27 (2015), 358–363.

[2] M. Al Khatib, A. Girard, and T. Dang. 2017. Self-Triggered Control for Sampled-data Systems using Reachability Analysis. *IFAC-PapersOnLine* 50, 1 (2017), 7881–7886.

[3] S. Bak and P. S. Duggirala. 2017. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *Proc. Int'l Conference on Hybrid systems: Computation and Control*. ACM, 173–178.

[4] Stanley Bak and Taylor T Johnson. 2015. Periodically-Scheduled Controller Analysis using Hybrid Systems Reachability and Continuization. In *Proc. Real-Time Systems Symposium*. IEEE, 195–205.

[5] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. 1996. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software (TOMS)* 22, 4 (1996), 469–483.

[6] Xin Chen and Erika Ábrahám. 2011. Choice of Directions for the Approximation of Reachable Sets for Hybrid Systems. In *International Conference on Computer Aided Systems Theory*. Springer, 535–542.

[7] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An Analyzer for Non-linear Hybrid Systems. In *Computer Aided Verification*, Natasha Sharygina and Helmut Veith (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 258–263.

[8] Arnaud Doucet, Nando de Freitas, and Neil Gordon. 2001. *An Introduction to Sequential Monte Carlo Methods*. Springer New York, New York, NY, 3–14. https://doi.org/10.1007/978-1-4757-3437-9_1

[9] Parasara Sridhar Duggirala and Stanley Bak. 2019. Aggregation Strategies in Reachable Set Computation of Hybrid Systems. In *Special issue of ACM Transactions on Embedded Computing Systems (TECS) associated with 16th International Conference on Embedded Software (EMSOFT)*.

[10] Viktorio S El Hakim and Marco J. G. Bekooij. 2018. Stability Verification of Self-Timed Control Systems using Model-Checking. In *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 312–319.

[11] Viktorio S El Hakim and Marco J. G. Bekooij. 2019. Reachability Analysis of Hybrid Automata with Clocked Linear Dynamics. In *SCOPES*. 27–36.

[12] G. Frehse et al. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV) (LNCS)*. Springer.

[13] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle. 2014. Formal Analysis of Timing Effects on Closed-loop Properties of Control Software. In *Proc. Real-Time Systems Symposium*. IEEE, 53–62.

[14] G. Frehse, R. Kateja, and C. Le Guernic. 2013. Flowpipe Approximation and Clustering in Space-Time. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 203–212.

[15] A. Girard. 2005. Reachability of Uncertain Linear Systems Using Zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 291–305.

[16] A. Girard, C. Le Guernic, and O. Maler. 2006. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 257–271.

[17] Simon Godsill and Tim Clapp. 2001. *Improvement Strategies for Monte Carlo Particle Filters*. Springer New York, New York, NY, 139–158. https://doi.org/10.1007/978-1-4757-3437-9_7

[18] Willem Hagemann. 2014. Reachability Analysis of Hybrid Systems Using Symbolic Orthogonal Projections. In *Proc. Int'l Conference on Computer Aided Verification*. Springer, 407–423.

[19] Joost P. H. M. Hausmans, Stefan J. Geuns, Maarten H. Wiggers, and Marco J. G. Bekooij. 2013. Two Parameter Workload Characterization for Improved Dataflow Analysis Accuracy. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 117–126.

[20] Pratyush Kumar, Dip Goswami, Samarjit Chakraborty, Anuradha Annaswamy, Kai Lampka, and Lothar Thiele. 2012. A Hybrid Approach to Cyber-Physical Systems Verification. In *Proc. Design Automation Conference (DAC)*. ACM, 688–696.

[21] Piyush Kumar and E Alper Yildirim. 2005. Minimum-Volume Enclosing Ellipsoids and Core Sets. *Journal of Optimization Theory and Applications* 126, 1 (2005), 1–21.

[22] A. A. Kurzhanskiy and P. Varaiya. 2007. Ellipsoidal Techniques for Reachability Analysis of Discrete-time Linear Systems. *IEEE Trans. Automat. Control* 52, 1 (2007), 26–38.

[23] Nojun Kwak. 2008. Principal Component Analysis Based on L1-Norm Maximization. *IEEE transactions on pattern analysis and machine intelligence* 30, 9 (2008), 1672–1680.

[24] Michael Lemmon, Thidapat Chantem, Xiaobo Sharon Hu, and Matthew Zyskowski. 2007. On Self-triggered Full-Information H-infinity Controllers. In *Proc. Int'l Conference on Hybrid systems: Computation and Control*. Springer, 371–384.

[25] J.P. Maschuw and D. Abel. 2010. Longitudinal Vehicle Guidance in Networks with changing Communication Topology. *IFAC Proceedings Volumes* 43, 7 (2010), 785–790.

[26] S. Quinton, M. Hanke, and R. Ernst. 2012. Formal Analysis of Sporadic Overload in Real-Time Systems. In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*. EDA Consortium, 515–520.

[27] S. Schupp and E. Ábrahám. 2018. Efficient Dynamic Error Reduction for Hybrid Systems Reachability Analysis. In *Proc. Int'l Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 287–302.

[28] Olaf Stursberg and Bruce H. Krogh. 2003. Efficient Representation and Computation of Reachable Sets for Hybrid Systems. In *Proc. Int'l Conference on Hybrid systems: Computation and Control*. Springer, 482–497.

[29] Q. Zhu and H. Zeng. 2012. Stability Analysis of Multi-rate Switched Networked Systems with Short Time Delay. In *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*. IEEE, 642–646.