Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices

Sergiy Bogomolov
Australian National University
Canberra, Australia

Marcelo Forets
Goran Frehse
Frédéric Viry
Univ. Grenoble Alpes, VERIMAG
Grenoble, France

Andreas Podelski
Christian Schilling
University of Freiburg
Freiburg, Germany

ABSTRACT

Approximating the set of reachable states of a dynamical system is an algorithmic yet mathematically rigorous way to reason about its safety. Although progress has been made in the development of efficient algorithms for affine dynamical systems, available algorithms still lack scalability to ensure their wide adoption in the industrial setting. While modern linear algebra packages are efficient for matrices with tens of thousands of dimensions, set-based image computations are limited to a few hundred. We propose to decompose reach set computations such that set operations are performed in low dimensions, while matrix operations like exponentiation are carried out in the full dimension. Our method is applicable both in dense- and discrete-time settings. For a set of standard benchmarks, it shows a speed-up of up to two orders of magnitude compared to the respective state-of-the-art tools, with only modest losses in accuracy. For the dense-time case, we show an experiment with more than 10.000 variables, roughly two orders of magnitude higher than possible with previous approaches.

KEYWORDS

reachability analysis, safety verification, linear time-invariant systems, set recurrence relation

ACM Reference Format:
Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Frédéric Viry, Andreas Podelski, and Christian Schilling. 2018. Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices. In *HSCC '18: 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), April 11–13, 2018, Porto, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178126.3178128

1 INTRODUCTION

Verifying safety properties for dynamical systems is an important and intricate task. For bounded time it is well known that the problem can be reduced to the computation of the reachable states. We are interested in the set-based reachability problem for affine dynamical systems [26]. Here, recurrence relations of the form

$$\mathcal{X}(k+1) = \Phi\mathcal{X}(k) \oplus \mathcal{V}(k), \quad k = 0, 1, \ldots, N \tag{1}$$

arise naturally. In the context of control engineering, the sequence of sets $\{\mathcal{V}(k)\}_k$ usually represents nondeterministic inputs or noise, $\oplus$ denotes the Minkowski sum between sets, Φ is a real $n \times n$ matrix, and the set $\mathcal{X}(0)$ accounts for uncertain initial states.

Numerous works present strategies for solving equation (1) in the form of ellipsoids [35, 36], template polyhedra such as zonotopes [4, 24] or support functions [13, 20–22, 38], or a combination [3]. The problem also generalizes to hybrid systems with piecewise affine dynamics [8, 29]. A key difficulty is scalability, as the cost of some set operations increases superlinearly with the dimension. A second challenge is the error accumulation for increasing values of N, known as the wrapping effect.

In this paper, we propose, analyze, and evaluate a novel *partial decomposition algorithm* for solving equation (1) that does not suffer from the wrapping effect if the inputs are held constant over all time. The complexity of non-decomposition approaches is mostly affected by the dimension n and grows superlinearly with it. Our method partially shifts this dependence on n to other structural properties: we perform set operations in low dimensions (unaffected by n); we effectively omit variables from the analysis if they are not involved in the property of interest; and we exploit the sparsity of Φ and its higher-order powers. However, unlike other decomposition approaches, we keep the matrix computations in high dimensions, which allows us to produce precise approximations. The strategy consists of decomposing the discrete recurrence relation (1) into subsystems of low dimensions. Then we compute the reachable states for each subsystem; these low-dimensional set operations can be performed efficiently. Finally we compose the low-dimensional sets symbolically and project onto the desired output variables. The analysis scales to systems with tens of thousands of variables, which are out of scope of state-of-the-art tools for dense-time reachability.

We apply our method to compute reachable states and verify safety properties of affine dynamical systems,

$$x'(t) = Ax(t) + Bu(t). \tag{2}$$

The initial state can be any point in a given set $\mathcal{X}_0$, and $u(t) \in \mathcal{U}(t) \subset \mathbb{R}^m$ is a nondeterministic input. Both the initial set and the set of input functions are assumed to be compact and convex. We also consider observable outputs,

$$y(t) = Cx(t) + Du(t), \tag{3}$$

HSCC '18, April 11–13, 2018, Porto, Portugal

ACM ISBN 978-1-4503-5642-8/18/04…$15.00
https://doi.org/10.1145/3178126.3178128

where C and D are matrices of appropriate dimension. In mathematical systems theory, equations (2)-(3) define what is known as a linear time-invariant (LTI) system.

Contribution. We present a new method to solve the reachability problem for affine dynamical systems with nondeterministic inputs and experimentally show that it is highly scalable under modest loss of accuracy. More precisely:

- We provide a new decomposition approach to solve equation (1) and analyze the approximation error.
- We address both the dense time and the discrete time reachability problem for general LTI systems of the form (2)-(3).
- We implement our approach efficiently and demonstrate its scalability on real engineering benchmarks. The tool, source code, and benchmark scripts are publicly available [2].

Related work. Kaynama and Oishi consider a Schur-based decomposition to compute the reach set [33, 34]. They approximate the result for subsystems by nondeterministic inputs using a static (i.e., time-unaware) box approximation. The authors also address approximation errors by solving a Sylvester equation to obtain a similarity transformation that minimizes the submatrix coupling. For systems where variables are linearly correlated in the initial states and inputs are constant, Han and Krogh propose an approximation method that uses Krylov subspace approximations [31] without explicitly decomposing the system. If the system is singularly perturbed with different time scales ("slow and fast variables"), time-scale decomposition can be applied [19, 27]. We do not consider this setting here. The reachability analysis tool Coho uses *projectahedra* – an approximate polyhedron representation consisting of all possible axis-aligned 2D projections – for set representation [28, 48]. Seladji and Bouissou define a sub-polyhedra abstract domain based on support functions [44]. Our approach can choose directions dynamically, and independently for each subsystem.

An orthogonal approach to reduce the complexity of system analysis is known as *model order reduction* (MOR) [6]. The idea is to construct a lower-dimensional model with *similar* behavior. Recently there have been efforts to combine MOR and abstraction techniques to obtain a sound overapproximation [47]. In a further approach, Bogomolov et al. [15] suggest an abstraction technique, which employs dwell time bounds. Moreover, Bogomolov et al. [14] introduce a system transformation to reduce the state space dimension based on the notion of quasi-dependent variables, which captures the dependencies between system state variables. In principle, such methods could be used as a preprocessing for our approach, where the approximation errors would then be combined.

Bak and Duggirala check safety properties and compute counterexample traces for LTI systems in a "simulation equivalent manner" [9]. A reachable set computed in this way consists of all the states that can be reached by a fixed-step simulation for any choice of the initial state and piecewise constant input. This set, however, does not include all trajectories of equation (2). The simulation equivalent reachability also involves a recurrence of the type (1), and we study its decomposed form in this work as well.

Decomposition methods have also been designed for the reachability problem of nonlinear ODEs. Chen et al. show that, using Hamilton-Jacobi methods, the (analytically) exact reach set can be reconstructed from an analysis of the subsystems for general ODE systems [17]. The system needs, however, be composed of so-called *self-contained subsystems*, which is a strong assumption. The technique is based on [40] which has no such limitation but suffers from a projection error. For general LTI systems (which we consider) an approximation error is unavoidable.

Asarin and Dang propose a decomposition approach where they project away variables and abstract them by time-unaware differential inclusions [7]. To address the overapproximation, they split these variables again into several subdomains. Chen and Sankaranarayanan apply uniform hybridization to analyze the subsystems over time and feed the results to the other subsystems as time-varying interval-shaped inputs [18]. In contrast, our reachability algorithm needs not be performed iteratively because the analysis of each subsystem is completely decoupled. Schupp et al. decompose a system by syntactic independence [43]. In our setting this corresponds to dynamics matrices of block diagonal form. For such systems the dynamical error is zero in both approaches.

The paper is organized as follows. In Section 2 we recall some basics on approximating convex sets with polyhedra and a state-of-the-art algorithm for approximating the reach sets of affine systems using the affine recurrence relation (1). In Section 3, we start by considering the decomposition of a single affine map, and then develop the more general case of an affine recurrence. The approximation error is discussed in Section 4. We present our reachability algorithm in Section 5, discuss the different techniques used to gain performance, and evaluate it experimentally in Section 6. Finally, we draw the conclusions in Section 7.

An extended version of this paper is available online [12].

2 APPROXIMATE REACHABILITY OF AFFINE SYSTEMS

In this section, we recall the state-of-the-art in approximating the reachable set of an affine dynamical system.

2.1 Preliminaries

Let us introduce some notation. Let $\mathbb{I}_n$ be the identity matrix of dimension $n \times n$. For $p \geq 1$, the p-norm of an n-dimensional vector $x \in \mathbb{R}^n$ is denoted $\|x\|_p$. The norm of a set $\mathcal{X}$ is $\|\mathcal{X}\|_p = \max_{x \in \mathcal{X}} \|x\|_p$. Let $\mathcal{B}_p^n$ be the unit ball of the p-norm in n dimensions, i.e., $\mathcal{B}_p^n = \{x : \|x\|_p \leq 1\}$. The Minkowski sum of sets $\mathcal{X}$ and $\mathcal{Y}$ is $\mathcal{X} \oplus \mathcal{Y} := \{x + y : x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$. Their Cartesian product, $\mathcal{X} \times \mathcal{Y}$, is the set of ordered pairs (x, y), with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The origin in $\mathbb{R}^n$ is written $\mathbf{0}_n$. There is a relation between products of sets and Minkowski sum: if $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$, then $\mathcal{X} \times \mathcal{Y} = (\mathcal{X} \times \{\mathbf{0}_m\}) \oplus (\{\mathbf{0}_n\} \times \mathcal{Y})$. The convex hull operator is written CH. Let $\boxdot(\cdot)$ be the symmetric interval hull operator, defined for any $\mathcal{X} \subset \mathbb{R}^n$ as the n-th fold Cartesian product of the intervals $[-|\bar{x}_i|, |\bar{x}_i|]$ for all $i = 1, \ldots, n$, where $|\bar{x}_i| := \sup\{|x_i| : x \in \mathcal{X}\}$.

2.2 Polyhedral Approximation of a Convex Set

We recall some basic notions for approximating convex sets. Let $\mathcal{X} \subset \mathbb{R}^n$ be a compact convex set. The *support function* of $\mathcal{X}$ is the function $\rho_{\mathcal{X}} : \mathbb{R}^n \to \mathbb{R}$,

$$\rho_{\mathcal{X}}(\ell) := \max_{x \in \mathcal{X}} \ell^{\mathsf{T}} x. \tag{4}$$

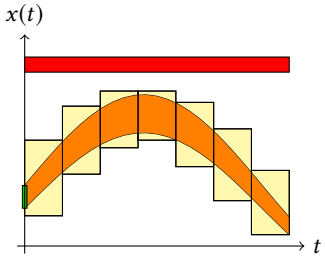

Figure 1: Illustration of a reach tube (orange) with set of initial states (green) and an approximation (yellow) that shows absence of error states (red).

The farthest points of $\mathcal{X}$ in the direction ℓ are the *support vectors*

$$\sigma_{\mathcal{X}}(\ell) := \left\{x \in \mathcal{X} : \ell^{\mathsf{T}} x = \rho_{\mathcal{X}}(\ell)\right\}. \tag{5}$$

When we speak of *the* support vector, we mean the choice of any support vector in (5). The projection of a set into a low dimensional space (a special case of $M\mathcal{X}$) can be conveniently evaluated using support functions, since $\sigma_{M\mathcal{X}}(\ell) = \sigma_{\mathcal{X}}(M^{\mathsf{T}}\ell)$. Given directions $\ell_1, \ldots, \ell_m$, a tight overapproximation of $\mathcal{X}$ is the *outer polyhedron* given by the constraints

$$\bigwedge_i \ell_i^{\mathsf{T}} x \le \rho_{\mathcal{X}}(\ell_i). \tag{6}$$

For instance, a bounding box involves evaluating the support function in $2n$ directions. More precise approximations can be obtained by adding directions. To quantify this, we use the following distance measure. A set $\hat{\mathcal{X}}$ is within Hausdorff distance ε of $\mathcal{X}$ if and only if

$$\hat{\mathcal{X}} \subseteq \mathcal{X} \oplus \varepsilon\mathcal{B}_p^n \text{ and } \mathcal{X} \subseteq \hat{\mathcal{X}} \oplus \varepsilon\mathcal{B}_p^n. \tag{7}$$

The infimum $\varepsilon \ge 0$ that satisfies (7) is called the Hausdorff distance between $\mathcal{X}$ and $\hat{\mathcal{X}}$ with respect to the p-norm, and is denoted $d_H^p(\mathcal{X}, \hat{\mathcal{X}})$. Another useful characterization of the Hausdorff distance is the following. Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ be polytopes. Then

$$d_H^p(\mathcal{X}, \mathcal{Y}) = \max_{\ell \in \mathcal{B}_p^n} |\rho_{\mathcal{Y}}(\ell) - \rho_{\mathcal{X}}(\ell)|. \tag{8}$$

In the special case $\mathcal{X} \subseteq \mathcal{Y}$, the absolute value can be removed.

By adding directions using Lotov's method [39], the outer polyhedron in (6) is within Hausdorff distance $\varepsilon\|X\|_p$ for $O(1/\varepsilon^{n-1})$ directions, and this bound is optimal. It follows that accurate outer polyhedral approximations are possible only in low dimensions. It is well-known that for $n = 2$, the bound can be lowered to $O(1/\sqrt{\varepsilon})$ directions, which is particularly efficient and the reason why we chose to decompose the system into subsystems of dimension 2.

2.3 Trajectory, Reach Set, and Reach Tube

A *trajectory* of the affine ODE with time-varying inputs (2) is the unique solution $x_{x_0,u}(t) : [0, T] \to \mathbb{R}^n$, for a given initial condition x_0 at time $t = 0$, and a given input signal u,

$$x_{x_0,u}(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}u(s)\,ds, \tag{9}$$

where we map $Bu(t)$ to $u(t)$ without loss of generality. Here T is the time horizon, which is considered to be finite in this paper. Given a set of initial states $\mathcal{X}_0$ and an input signal u, the *reach set* at time t is $\mathcal{R}(\mathcal{X}_0, u, t) := \{x_{x_0,u}(t) : x_0 \in \mathcal{X}_0\}$. This extends to a family of solutions as

$$\mathcal{R}(\mathcal{X}_0, \mathcal{U}, t) = \bigcup \left\{\mathcal{R}(\mathcal{X}_0, u, t) : u(s) \in \mathcal{U}(s), \forall\, s \in [0, t]\right\}. \tag{10}$$

The *reach tube* for a given time interval $[t_1, t_2] \subseteq [0, T]$ is the set

$$\mathcal{R}(\mathcal{X}_0, \mathcal{U}, [t_1, t_2]) := \bigcup_{t_1 \le t \le t_2} \mathcal{R}(\mathcal{X}_0, \mathcal{U}, t). \tag{11}$$

In general, the reach tube can be computed only approximately. An example reach tube and an overapproximation using boxes is shown in Fig. 1. In the next section we discuss how to compute such an overapproximation of the reach tube.

2.4 Approximation Model

The standard numerical approach for the reachability problem is to reduce it to computing a finite sequence of sets, $\{\mathcal{X}(k)\}_{k=0}^N$, that overapproximates the exact reach tube (11). We assume a given constant time step size $\delta > 0$ over the time horizon $T = N\delta$, where N is the number of time steps. With respect to the inputs, we assume that the time-varying function $\mathcal{U}(\cdot)$ from Sect. 2.3 is piecewise constant, i.e., we consider a possibly time-varying discrete sequence $\{\mathcal{U}(k)\}_k$ for all $k = 0, 1, \ldots, N$.

In the dense-time case, one is interested in covering all possible trajectories of the given continuous system. In the discrete-time case, the reach tube of the discretized system is only covered at discrete time steps, but not necessarily between time steps. In either case, starting from the system (2)-(3), we can reduce the reachability problem to the general recurrence (1), with suitably transformed initial states and nondeterministic input. These reductions can be found in previous works [21, 38] and we recall them below.

First, we recall the dense time case. All continuous trajectories are covered by the discrete approximation if

$$\mathcal{R}(\mathcal{X}_0, \mathcal{U}, [k\delta, (k+1)\delta]) \subseteq \mathcal{X}(k), \quad k = 0, 1, \ldots, N \tag{12}$$

Previous works have provided approximation models such that (12) holds [21, 37, 38]. In particular, in [21, Lemma 3] the authors intersect a first-order approximation of the interpolation error going forward in time from $t = 0$ with one that goes backward in time from $t = \delta$. Note that this forward-backward approximation is used in SpaceEx, to which we will compare our method later. Here, we consider the forward-only approximation. To guarantee that the overapproximation covers the interval between time steps, the initial set and the input sets are bloated by additive terms

$$E_\psi(\mathcal{U}(k), \delta) := \square(\Phi_2(|A|, \delta) \square (A\mathcal{U}(k)))$$
$$E^+(\mathcal{X}_0, \delta) := \square(\Phi_2(|A|, \delta) \square (A^2\mathcal{X}_0)),$$

where the matrices $\Phi_1(A, \delta)$ and $\Phi_2(A, \delta)$ are defined via

$$\Phi_1(A, \delta) := \sum_{i=0}^{\infty} \frac{\delta^{i+1}}{(i+1)!}A^i, \quad \Phi_2(A, \delta) := \sum_{i=0}^{\infty} \frac{\delta^{i+2}}{(i+2)!}A^i.$$

The required transformations for dense time are:

$$\begin{cases} \Phi \leftarrow e^{A\delta} \\ \mathcal{X}(0) \leftarrow \mathrm{CH}\big(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta\mathcal{U}(0) \oplus E_\psi(\mathcal{U}(0), \delta) \oplus E^+(\mathcal{X}_0, \delta)\big) \\ \mathcal{V}(k) \leftarrow \delta\mathcal{U}(k) \oplus E_\psi(\mathcal{U}(k), \delta), \quad \forall\, k = 0, 1, \ldots, N \end{cases} \tag{13}$$

For discrete time reachability the transformations are:

$$\begin{cases} \Phi \leftarrow e^{A\delta} \\ \mathcal{X}(0) \leftarrow \mathcal{X}_0 \\ \mathcal{V}(k) \leftarrow \Phi_1(A,\delta)\mathcal{U}(k), \quad \forall\, k = 0,1,\ldots,N \end{cases} \tag{14}$$

Note that there is no bloating of the initial states, and that the inputs are assumed to remain constant between sampling times.

The cost of solving the general recurrence (1) with either the data (13) or (14), to compute an approximation of the reach set or the reach tube, increases superlinearly with the dimension of the system and the desired approximation error. In the rest of this paper, we will consider a decomposition of the system to reduce the computational cost.

3 DECOMPOSITION

In this section, we present a novel approach for solving the general recurrence (1) using block decompositions.

3.1 Cartesian Decomposition

From now on, let $\mathcal{X} \subset \mathbb{R}^n$ be a compact and convex set. To simplify the discussion, we assume n to be even. We characterize the decomposition of $\mathcal{X}$ into $b := n/2$ sets of dimension two as follows. Let π_i be the *projection matrix* that maps a vector $x \in \mathbb{R}^n$ to its coordinates in the i-th block, $x_i = \pi_i x$. The *Cartesian decomposition* of $\mathcal{X}$ is the set

$$\mathsf{dcp}(\mathcal{X}) := \pi_1\mathcal{X} \times \cdots \times \pi_b\mathcal{X}.$$

We call a set *decomposed* if it is identical to its Cartesian decomposition. For instance, the symmetric interval hull $\boxdot(\mathcal{X})$ is a decomposed set, since it is the Cartesian product of one-dimensional sets, i.e., intervals. Throughout the paper, we will highlight decomposed sets with the symbol $\hat{\cdot}$ (as in $\hat{\mathcal{X}}, \hat{\mathcal{Y}}$). Note that decomposition distributes over Minkowski sum:

$$\mathsf{dcp}(\mathcal{X} \oplus \mathcal{Y}) = \mathsf{dcp}(\mathcal{X}) \oplus \mathsf{dcp}(\mathcal{Y}). \tag{15}$$

If $\mathcal{X}$ is a polyhedron in constraint form, the projections can be very costly to compute, which amounts to quantifier elimination. However, using the methods in Sect. 2.2, we can efficiently compute an overapproximation. The overapproximation can be coarse, e.g., a bounding box, or ε-close in the Hausdorff norm for a given value of ε. Since the choice of approximation is of no particular importance to the remainder of the paper, we simply assume an operator

$$\hat{\mathcal{X}}_1 \times \cdots \times \hat{\mathcal{X}}_b = \widehat{\mathsf{dcp}}(\mathcal{X})$$

that overapproximates the Cartesian decomposition with a decomposed set $\hat{\mathcal{X}}_1 \times \cdots \times \hat{\mathcal{X}}_b$ such that $\mathsf{dcp}(\mathcal{X}) \subseteq \widehat{\mathsf{dcp}}(\mathcal{X})$.

3.2 Decomposing an Affine Map

Suppose that a compact and convex set $\mathcal{V} \subset \mathbb{R}^n$ is given, and let Φ be a real $n \times n$ matrix. Consider the n-dimensional *affine map*

$$\mathcal{X}' = \Phi\mathcal{X} \oplus \mathcal{V} = \begin{pmatrix} \Phi_{11} & \cdots & \Phi_{1b} \\ \vdots & \ddots & \vdots \\ \Phi_{b1} & \cdots & \Phi_{bb} \end{pmatrix} \mathcal{X} \oplus \mathcal{V}, \tag{16}$$

where Φ_{ij} denotes the 2×2 submatrix of Φ in row i and column j, counting from top to bottom and from left to right. We call such a submatrix a *block*, and $[\Phi_{i1}\Phi_{i2}\cdots\Phi_{ib}]$ a *row-block*. We assume without loss of generality that n is even. Hence, every row-block of Φ consists of $b := n/2$ two-dimensional blocks.

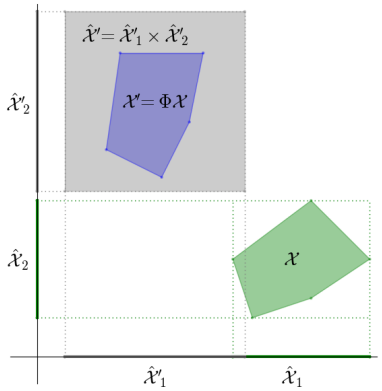

Figure 2: The Cartesian decomposition of $\mathcal{X}$ (green) in two blocks of size one is the set $\widehat{\mathsf{dcp}}(\mathcal{X}) = \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2$. The decomposed image of the map $\mathcal{X}' = \Phi\mathcal{X}$ (blue) is the set $\hat{\mathcal{X}}'$ (gray) obtained by the application of Eq. (17) for each block $\hat{\mathcal{X}}'_j$.

The *decomposed image* of the map (16) is obtained in two steps. First, we transform the full-dimensional sets $\mathcal{X}$ and $\mathcal{V}$ into Cartesian products of two-dimensional sets $\hat{\mathcal{X}}_1, \ldots, \hat{\mathcal{X}}_b$, using the operator $\widehat{\mathsf{dcp}}$ described in the previous section. Second, we construct the two-dimensional sets

$$\hat{\mathcal{X}}'_i := \bigoplus_{j=1}^{b} \Phi_{ij}\hat{\mathcal{X}}_j \oplus \hat{\mathcal{V}}_i, \quad \forall\, i = 1,\ldots,b. \tag{17}$$

We call the Cartesian product $\hat{\mathcal{X}}' = \hat{\mathcal{X}}'_1 \times \cdots \times \hat{\mathcal{X}}'_b$ the decomposed image of (16). For each i, $\hat{\mathcal{X}}'_i$ only depends on the i-th row-block of Φ. We illustrate in Fig. 2 the decomposed image of an affine map over a polygon.

We now compare the cost of (16) and (17). Let us denote the cost of computing the image of an $n\times n$ linear map by $C_\odot(n,m)$ and the cost of computing the Minkowski sum of two n-dimensional sets by $C_\oplus(n,m)$, where m is a parameter that depends on the set representation. The asymptotic complexity of performing the above operations for common set representations is shown in Table 1; we refer to [23–25, 41] for further details. Since one Minkowski sum and one linear map are involved in (16), we have that

$$\text{Cost (16)} \in O(C_\odot(2b,m) + C_\oplus(2b,m)).$$

On the other hand, the aggregated cost for the i-th block in (17) is $bC_\oplus(2,m') + bC_\odot(2,m')$, where m' is the parameter for complexity (m) in two dimensions. The total cost is thus

$$\text{Cost (17)} \in O(b^2C_\odot(2,m') + b^2C_\oplus(2,m')).$$

Whenever $C_\odot$ and $C_\oplus$ depend at least quadratically on the dimension, and since $m' \ll m$, the cost of the decomposed image (17) is asymptotically smaller than the cost of the non-decomposed image (16).

Table 1: Complexity of set operations involved in the affine map computation by decomposition.

	polyhedra		zonotopes	supp. fun.
	m constraints	m vertices	m generat.	m direct.
$C_{\odot}(n, m)$	$O(mn^2 + n^3)$	$O(mn^2)$	$O(mn^2)$	$O(mn^2\mathcal{L})$
$C_{\oplus}(n, m)$	$O(2^n)$	$O(m^2n)$	$O(n)$	$O(m\mathcal{L})$

$\mathcal{L}$ is the cost of evaluating the support function of $\mathcal{X}$. For polyhedra in constraint representation we assume that Φ is invertible; otherwise the complexity is $O(m^n)$. Note that m is not comparable between different representations.

3.3 Decomposing an Affine Recurrence

Let us reconsider the affine recurrence in Eq. (1). We can rewrite it into b row-blocks, as in Sect. 3.2, with given compact and convex sequence $\{\mathcal{V}(k)\}_k \subset \mathbb{R}^n$ for $k \geq 0$, and an initial set $\mathcal{X}(0)$:

$$\mathcal{X}(k+1) = \begin{pmatrix} \Phi_{11} & \cdots & \Phi_{1b} \\ \vdots & \ddots & \vdots \\ \Phi_{b1} & \cdots & \Phi_{bb} \end{pmatrix} \mathcal{X}(k) \oplus \mathcal{V}(k). \tag{18}$$

In this recurrence, the approximation error of the k-th step is propagated, and possibly amplified, in step $k+1$. This can be partly avoided by using a non-recursive form [26]. We present two scenarios, which differ in whether the sequence of input sets is constant or not. Let Φ^k_{ij} be the submatrix of Φ^k corresponding to the indices of the submatrix Φ_{ij} of Φ.

Constant input sets. Assuming that the sets $\mathcal{V}$ do not depend on k, the non-recurrent form of (18) is:

$$\begin{cases} \mathcal{X}(k) = \Phi^k \mathcal{X}(0) \oplus \mathcal{W}(k) \\ \mathcal{W}(k+1) = \mathcal{W}(k) \oplus \Phi^k \mathcal{V}, \quad \mathcal{W}(0) := \{\mathbf{0}_n\}. \end{cases}$$

The decomposed map, for $i = 1, \ldots, b$, is:

$$\begin{cases} \hat{\mathcal{X}}_i(k) = \bigoplus_{j=1}^{b} \Phi^k_{ij} \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) = \hat{\mathcal{W}}_i(k) \oplus [\Phi^k_{i1} \cdots \Phi^k_{ib}]\mathcal{V}, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_2\}. \end{cases} \tag{19}$$

Note that the set $[\Phi^k_{i1} \cdots \Phi^k_{ib}]\mathcal{V}$ in (19) is of low dimension and corresponds to the i-th block.

Time-varying input sets. Assuming that the sequence of inputs depends on k, the non-recurrent form of (18) is:

$$\begin{cases} \mathcal{X}(k) = \Phi^k \mathcal{X}(0) \oplus \mathcal{W}(k) \\ \mathcal{W}(k+1) = \Phi\mathcal{W}(k) \oplus \mathcal{V}(k), \quad \mathcal{W}(0) := \{\mathbf{0}_n\}. \end{cases}$$

The decomposed map, for $i = 1, \ldots, b$, is:

$$\begin{cases} \hat{\mathcal{X}}_i(k) = \bigoplus_{j=1}^{b} \Phi^k_{ij} \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) = \bigoplus_{j=1}^{b} \Phi_{ij} \hat{\mathcal{W}}_j(k) \oplus \hat{\mathcal{V}}_i, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_2\}. \end{cases} \tag{20}$$

4 APPROXIMATION ERROR

In general, the reduction in the computational cost of the decomposed image comes at the price of an approximation error for $\hat{\mathcal{X}}_i(k)$. We discuss the two sources of this error. The first one is due to the decomposition of the initial states. For discrete time reachability, the initial set $\mathcal{X}_0$ remains unchanged under the transformations (14). In practice, $\mathcal{X}_0$ often has the shape of a hyperrectangle, and hence there is no approximation error. However, for dense time reachability the transformations (13) do not preserve an initially decomposed set, and $\widehat{\mathsf{dcp}}$ invariably introduces an approximation error. If the constraints on $\mathcal{X}(0)$ are known, an upper bound on the Hausdorff distance $d^p_H(\mathcal{X}(0), \hat{\mathcal{X}}(0))$ can be obtained using support functions [39]. The second source of the approximation error is the step-wise decomposition of the inputs. This can be either a linear combination with respect to a row-block, as in (19), or a single block as in (20). For a stable matrix Φ, in either case, the error propagated to $\hat{\mathcal{W}}_i(k+1)$ goes to zero for $k \to \infty$. In the rest of the section, we discuss these errors in more detail.

4.1 Error of a Decomposed Affine Map

We now turn to the question how big the decomposition error is in the decomposed affine map (17) compared to (16). To simplify the discussion, we omit $\mathcal{V}$ without loss of generality, since we can rephrase (16) with an augmented state space where $\mathcal{X}^* \leftarrow \mathcal{X} \times \mathcal{V}$ and $\Phi^* \leftarrow [\Phi\ I]$. Then $\mathcal{X}' = \Phi^*\mathcal{X}^*$.

We proceed in two steps. First, we bound the error for a set that is already decomposed. Then we bound the distance between the image of the decomposed and the original set. The total error follows from a triangle inequality.

Proposition 1. *Let $\hat{\mathcal{X}}$ be a decomposed set, and let $\bar{\mathcal{X}}'$ be the image of $\hat{\mathcal{X}}$ under the linear map $\bar{\mathcal{X}}' = \Phi\hat{\mathcal{X}}$ and let $\hat{\mathcal{X}}'$ the image of $\hat{\mathcal{X}}$ under the decomposed map* (17)*. Then $\bar{\mathcal{X}}' \subseteq \hat{\mathcal{X}}'$ and*

$$d^p_H(\bar{\mathcal{X}}', \hat{\mathcal{X}}') = \max_{\|d\|_p \leq 1} \sum_{i,j} \rho_{\hat{\mathcal{X}}_j}(\Phi^T_{ij} d_i) - \rho_{\hat{\mathcal{X}}_j}\left(\sum_k \Phi^T_{kj} d_k\right) \tag{21}$$

where the max *is taken over $d = d_1 \times \cdots \times d_b$ in the unit ball of the p-norm, and $\Phi^T_{ij} := (\Phi_{ij})^T$.*

Corollary 1. *If only one Φ_{ij} per column is nonzero, then the error is zero. A special case of such a matrix is the (real) Jordan form if all eigenvalues have multiplicity 1.*

We can simplify this bound to clarify the relationship between the error and the norm of the matrix blocks Φ_{ij}. To quantify the error associated to the i-th block, let us introduce the number Δ_i to be the diameter of $\hat{\mathcal{X}}_i$, i.e., the smallest number such that for any d,

$$\rho_{\hat{\mathcal{X}}_i}(d) + \rho_{\hat{\mathcal{X}}_i}(-d) \leq \|d\|_{\frac{p}{p-1}} \Delta_i.$$

For instance, if $\hat{\mathcal{X}}_i$ is an interval hull, then Δ_i is the width of the largest interval. Intuitively, the approximation error is small if the off-diagonal entries of Φ are small. Quantifying this using (21), we get that the error bound is a weighted sum of the diameters of the state sets:

Proposition 2. *For $j = 1, \ldots, b$, let $q_j := \arg\max_i \|\Phi_{ij}\|_p$ (the index of the block with the largest matrix norm in the j-th column-block), so that $\alpha_j := \max_{i \neq q_j} \|\Phi_{ij}\|_p$ is the second largest matrix norm*

in the j-th column-block. Let $\alpha_{\max} := \max_j \alpha_j$ *and* $\Delta_{\text{sum}} := \sum_{j=1}^{b} \Delta_j$. *The error of the decomposed map is*

$$d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') \leq (b-1) \sum_{j=1}^{b} \alpha_j \Delta_j \leq \frac{n}{2} \alpha_{\max} \Delta_{\text{sum}}. \tag{22}$$

We now come to the second step. We compare the image $\mathcal{X}' = \Phi\mathcal{X}$ with the decomposed image $\hat{\mathcal{X}}'$, including both the decomposition error from $\mathcal{X}$ to $\hat{\mathcal{X}}$ and the error introduced by the decomposed map (17). We use a simple lemma:

Lemma 2. *Let* $\mathcal{X}' = \Phi\mathcal{X}$ *and* $\bar{\mathcal{X}}' = \Phi\hat{\mathcal{X}}$, *where* $\mathcal{X} \subseteq \hat{\mathcal{X}}$. *The distance between the images is* $d_H^p(\mathcal{X}', \bar{\mathcal{X}}') \leq \|\Phi\|_p d_H^p(\mathcal{X}, \hat{\mathcal{X}})$.

Combining Lemma 2 with Prop. 2 and the triangle inequality $d_H^p(\mathcal{X}', \hat{\mathcal{X}}') \leq d_H^p(\mathcal{X}', \bar{\mathcal{X}}') + d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}')$, we get the following total error bound on the decomposed image computation:

Proposition 3. $d_H^p(\mathcal{X}', \hat{\mathcal{X}}') \leq (b-1) \sum_{j=1}^{b} \alpha_j \Delta_j + \|\Phi\|_p d_H^p(\mathcal{X}, \hat{\mathcal{X}})$.

The above bound gives us an idea about the error of the decomposed affine map, without having to do any high-dimensional set computations. We now apply it to affine recurrences.

4.2 Error of a Decomposed Affine Recurrence

For any Φ, there exist constants K_Φ and α_Φ such that

$$\|\Phi^k\|_p \leq K_\Phi \alpha_\Phi^k, \qquad k \geq 0.$$

If $\Phi = e^{A\delta}$, one choice is $\alpha_\Phi = e^{\lambda\delta}$ with λ the spectral abscissa (largest real part of any eigenvalue of A), although it may not be possible to compute the corresponding K_Φ efficiently. In this case, $\alpha_\Phi \leq 1$ if the system is stable. Another choice is to let $\alpha_\Phi = e^{\mu\delta}$, with μ the logarithmic norm of A and $K_\Phi = 1$. In this case, α_Φ may be larger than 1 even for stable systems. Note that in both cases $\alpha_\Phi \to 1$ as $\delta \to 0$. For concreteness we continue with the first formulation in the remaining section.

For constant inputs sets, (19) is a linear map of the decomposed initial states $\hat{\mathcal{X}}(0)$ plus a decomposed input $\hat{\mathcal{W}}(k)$, which is itself obtained from a sequence of decomposed linear maps. Applying Prop. 3 gives the following result.

Proposition 4. *Let the decomposition error of the initial states* $\mathcal{X}(0)$ *be bounded by* $\varepsilon^x \geq d_H^p\big(\mathcal{X}(0), \hat{\mathcal{X}}(0)\big)$, *and let the decomposition error of* $\mathcal{V}$ *be bounded by* $\varepsilon^v \geq d_H^p\big(\mathcal{V}, \hat{\mathcal{V}}\big)$. *Let* Δ_j^x *be the diameter of* $\hat{\mathcal{X}}_j(0)$, *and* $\Delta_{\text{sum}}^x = \sum_{j=1}^{b} \Delta_j^x$. *Let* Δ_j^v *be the diameter of* $\hat{\mathcal{V}}_j$, *and* $\Delta_{\text{sum}}^v = \sum_{j=1}^{b} \Delta_j^v$. *Then the approximation error due to decomposition, at step k, is bounded by*

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \leq K_\Phi \Big(\alpha_\Phi^k \big(b\Delta_{\text{sum}}^x + \varepsilon^x \big) + \big(b\Delta_{\text{sum}}^v + \varepsilon^v \big) \alpha_\Phi \frac{1 - \alpha_\Phi^{k-1}}{1 - \alpha_\Phi} \Big) + \varepsilon^v.$$

If $\alpha_\Phi < 1$ *(stable system), the error is bounded for all k by*

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \leq K_\Phi \Big(b\Delta_{\text{sum}}^x + \varepsilon^x + \big(b\Delta_{\text{sum}}^v + \varepsilon^v \big) \frac{\alpha_\Phi}{1 - \alpha_\Phi} \Big) + \varepsilon^v.$$

In conclusion, the approximation error is linear in the width of the initial states and the inputs, and in the decomposition errors of the initial states and the input sets. For unstable systems, or time steps not large enough, the input set can become the dominating source of error, e.g., considering cases with $\alpha_\Phi > \frac{1}{2}$.

4.3 Error of a Decomposed Reach Tube Approximation

The decomposed reach tube approximation consists of the affine recurrence (19), with suitable sets $\mathcal{X}(0)$ and $\mathcal{V}$. The error bound follows from Prop. 4 and the decomposition errors for $\mathcal{X}(0)$ and $\mathcal{V}$.

In the discrete time case (14), the initial states $\mathcal{X}(0)$ of the affine recurrence (19) are identical to the initial states $\mathcal{X}_0$ of the model, so their decomposition error is

$$\varepsilon^x = d_H^p(\mathcal{X}_0, \hat{\mathcal{X}}_0).$$

However, $\mathcal{V} = \Phi_1(A, \delta)\mathcal{U}$. Let $\hat{\mathcal{U}} = \mathsf{dcp}(\mathcal{U})$. By Lemma 2 we get

$$\varepsilon^v = \|\Phi_1(A, \delta)\|_p d_H^p(\mathcal{U}, \hat{\mathcal{U}}).$$

In the dense time case (13), the initial states of the affine recurrence (19) are $\mathcal{X}(0) = \mathrm{CH}\big(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta\mathcal{U} \oplus E_\psi(\mathcal{U}, \delta) \oplus E^+(\mathcal{X}_0, \delta)\big)$, and $\mathcal{V} = \delta\mathcal{U} \oplus E_\psi(\mathcal{U}, \delta)$. Recall from (15) that decomposition distributes over Minkowski sum. We get

$$\varepsilon^v = \delta d_H^p(\mathcal{U}, \hat{\mathcal{U}}).$$

The decomposition error for the initial states is more complex and harder to estimate. We consider the idealized case where the system is stable with $\alpha_\Phi = e^{-\lambda\delta}$, $\lambda > 0$, for an infinitesimal time step $\delta \to 0$. Then $\alpha_\Phi \to 1 - \lambda\delta$ and $\frac{\alpha_\Phi}{1-\alpha_\Phi} \to \frac{1}{\lambda\delta}$, so that the decomposition error due to the inputs does not go to zero in Prop. 4. Let $\Delta_{\mathcal{X}_0}$, $\Delta_{\mathcal{U}}$ be the sum of the diameters of decomposed sets of $\mathcal{X}_0$ and $\mathcal{U}$. Let $\varepsilon_0^x = d_H^p(\mathcal{X}_0, \hat{\mathcal{X}}_0)$ and $\varepsilon_0^v = d_H^p(\mathcal{U}, \hat{\mathcal{U}})$. For both the discrete time and the dense time case, $\varepsilon^x \to \varepsilon_0^x$, $\Delta_{\text{sum}}^x \to \Delta_{\mathcal{X}_0}$, $\Delta_{\text{sum}}^v \to \delta\Delta_{\mathcal{U}}$ and $\varepsilon^v \to \delta\varepsilon_0^v$. Then Prop. 4 gives a nonzero upper bound

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \leq K_\Phi \Big(b\Delta_{\mathcal{X}_0} + \varepsilon_0^x + \big(b\Delta_{\mathcal{U}} + \varepsilon_0^v \big) \frac{1}{\lambda} \Big) + O(\delta).$$

This indicates that a small time step may be problematic for systems with large time constants (small λ).

5 ALGORITHM & IMPLEMENTATION

We now describe the decomposition method from Sect. 3 from an algorithmic view and discuss some crucial details for our implementation in *Julia* [11]. Given an LTI system in the form (2)-(3), we first apply a suitable approximation model from Sect. 2.4 (`discretize`). Then we execute the corresponding decomposed recurrence from Sect. 3.3 to compute the reach tube (`reach`) or to check a safety property. Finally, we project onto output variables (`project`).

One of the main principles we exploit in our implementation is that of lazy (on-demand) evaluation. We use lazy (i.e., symbolic) set representations for most of the set operations, in particular for Minkowski sum, linear map, and Cartesian product. Common sets such as hypercubes in different norms, polyhedra, and polygons each are represented by specific types. Each type has to provide a function to compute the support vector in a given direction. The operations can be nested symbolically without actually evaluating them. Then, we can compute the support vector of the (nested) lazy set on demand. The advantage of lazy data structures is that we may save unnecessary evaluations at the cost of higher memory consumption. In practice, we use a careful balance between lazy sets

Algorithm 1: Function `reach`.

Input: $\mathcal{D} = (\Phi, \mathcal{V}(\cdot), \mathcal{X}(0))$: discrete system
N: total number of steps
blocks: list of block indices
Output: $\{\hat{\mathcal{X}}(k)\}_k$: array of 2D reach tubes

1 $\hat{\mathcal{X}}(0) \leftarrow \widehat{\mathsf{dcp}}(\mathcal{X}(0))$;
2 *all_blocks* $\leftarrow$ `get_all_block_indices`(dim(Φ));
3 $P \leftarrow \mathbb{I}_{\dim(\Phi)}$;
4 $Q \leftarrow \Phi$;
5 $\hat{\mathcal{V}}_{\text{tmp}} \leftarrow []$;
6 **for** $b_i \in$ *blocks* **do**
7 $\quad \hat{\mathcal{V}}_{\text{tmp}}[b_i] \leftarrow \{\mathbf{0}_2\}$;
8 **end**
9 **for** $k = 1$ **to** $N - 1$ **do**
10 $\quad \hat{\mathcal{X}}_{\text{tmp}} \leftarrow []$;
11 $\quad$ **for** $b_i \in$ *blocks* **do**
12 $\quad\quad \hat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \{\mathbf{0}_2\}$;
13 $\quad\quad$ **for** $b_j \in$ *all_blocks* **do**
14 $\quad\quad\quad \hat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \hat{\mathcal{X}}_{\text{tmp}}[b_i] \oplus Q[b_i, b_j] \odot \hat{\mathcal{X}}(0)[b_j]$;
15 $\quad\quad$ **end**
16 $\quad\quad \hat{\mathcal{V}}_{\text{tmp}}[b_i] \leftarrow \mathtt{approx}(\hat{\mathcal{V}}_{\text{tmp}}[b_i] \oplus P[b_i, :] \odot \mathcal{V}(k-1))$;
17 $\quad\quad \hat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \mathtt{approx}(\hat{\mathcal{X}}_{\text{tmp}}[b_i] \oplus \hat{\mathcal{V}}_{\text{tmp}}[b_i])$;
18 $\quad$ **end**
19 $\quad \hat{\mathcal{X}}(k) \leftarrow \hat{\mathcal{X}}_{\text{tmp}}$;
20 $\quad P \leftarrow Q$;
21 $\quad Q \leftarrow Q \cdot \Phi$;
22 **end**

and concrete sets, i.e., the nesting depth is fixed (depending on the model dimension n). The alternative to using lazy data structures is to make the representation explicit after each operation, potentially involving an overapproximation.

5.1 Discretization

In the case of a dense-time model, the `discretize` step transforms the system $(A, \mathcal{U}(\cdot), \mathcal{X}_0)$ to its discrete counterpart $(\Phi, \mathcal{V}(\cdot), \mathcal{X}(0))$. Recall from Sect. 2.4 that both the definition of the discretized input sequence, $\mathcal{V}(\cdot)$, and the discretized initial states, $\mathcal{X}(0)$, depend on the approximation model (dense time vs. discrete time). The set transformations are performed lazily for all but the symmetric interval hull operator, while the matrix exponentiation can be either explicit or lazy. We support exponentiation techniques for large and sparse matrices (e.g., $n = 10,000$), which has a major impact on runtime performance and memory cost. Instead of computing the matrix exponential $\Phi = e^{A\delta}$ explicitly, we can evaluate the action of a matrix on an n-dimensional vector. We use `Expokit.jl` [1], a Julia implementation of *Expokit* [45].

5.2 Reach Tube Approximation

After we have obtained a discretized system, we use Algorithm 1 to compute an approximation of the reach tube. As an additional input the algorithm receives an array of block indices (*blocks*) that we are interested in. For simplicity we assume that the elements of *blocks* have the form $[j, j+1]$ for even j.

The result, a reach tube for each time interval k, is represented by the array $\{\hat{\mathcal{X}}(k)\}_k$. The type of each entry $\hat{\mathcal{X}}(k)$ itself is an array of polygons representing the 2D reach tubes. To reconstruct the full-dimensional reach tube for time interval k, the result has to be interpreted as a Cartesian product, i.e., $\bigotimes_{b_i} \hat{\mathcal{X}}(k)[b_i]$. Initially, $\hat{\mathcal{X}}(0)$ just contains the decomposed initial states (line 1).

The list *all_blocks* just consists of all the 2D block indices. We maintain the matrix Q to be the matrix Φ raised to the power of k, i.e., $Q = \Phi^k$ at step k; similarly, $P = \Phi^{k-1}$. For clarity we use $Q[b_i, b_j]$ instead of Q_{ij} as in Sect. 3, and similarly, $P[b_i, :]$ denotes the whole row-block b_i.

The main loop starting in line 9 computes the reach tubes for each k. We write $\oplus$ and $\odot$ to denote lazy set representation of Minkowski sum and linear map, respectively. The array $\hat{\mathcal{X}}_{\text{tmp}}$ is filled with two-dimensional reach tubes in the inner loop (lines 11 to 18) for each block in *blocks*. Line 16 computes the current input convolution, which is added in line 17. The function `approx` overapproximates its argument (a two-dimensional lazy set) to a polygon in constraint representation using Lotov's method (see Sect. 2.2). The fact that we approximate in line 16 is a design decision; in principle we could keep the elements of $\hat{\mathcal{V}}_{\text{tmp}}$ a lazy set, but experiments have shown that the precision gain is marginal.

We use a different implementation for models with dense and sparse matrices Φ, respectively. For instance, the loop around line 14 only has to be executed if the submatrix $\Phi^k[b_i, b_j]$ is non-zero. As the linear algebra back-end we use either a BLAS-compatible library [5] or a native Julia implementation for sparse matrices following Gustavson [30]. These optimizations have a major impact on the runtime (around one order of magnitude).

5.3 Efficient 2D Approximation

For non-decomposed approaches, manipulating polygons or polyhedra involves using an external linear programming (LP) back-end, possibly in high-dimensional space. Since Algorithm 1 mostly operates on 2D sets, we can use the following efficient implementation for evaluating the support vector.

Since vectors in the plane can be ordered by the angle with respect to the positive real axis, we can efficiently evaluate the support vector of a polygon in constraint representation by comparing normal directions, provided that its edges are ordered. We use the symbol $\preceq$ to compare directions, where the increasing direction is counter-clockwise. The following lemma provides an algorithm to find the support vector.

Lemma 3. *Let $\mathcal{X}$ be a polygon described by m linear constraints $a_i^\mathsf{T} x \le b_i$, ordered by the normal vectors (a_i), i.e., $a_i \preceq a_{i+1}$ for all $i \in \{1, \ldots, m\}$, where we identify a_{m+1} with a_1. Let $\ell \in \mathbb{R}^2 \setminus \{\mathbf{0}_2\}$. Then there exists $i \in \{1, \ldots, m\}$ such that $a_i \preceq \ell \preceq a_{i+1}$ and every optimal solution $\bar{x}$ of the linear program $\rho_{\mathcal{X}}(\ell) = \max\{\ell^\mathsf{T} x : x \in \mathcal{X}\}$ satisfies $\bar{x} \in \{x : a_i^\mathsf{T} x \le b_i\} \cap \{x : a_{i+1}^\mathsf{T} x \le b_{i+1}\}$.*

For the evaluation (Sect. 6) we use a box approximation. We note that our implementation of the `approx` function works for general set approximations; for box approximation we could also use an optimized implementation, e.g., using interval arithmetic.

5.4 Projection onto Output Variables

After `reach` has terminated, returning an array of Cartesian products of two-dimensional sets, we usually need to observe some output variables $y(t)$, as in the LTI system Eq. (3). A list with the output variables, or more generally, a $2 \times n$ projection matrix, can

be passed to the function `project`. This projection matrix is used if we want to observe, e.g., a linear combination of states.

5.5 Checking Safety Properties

For checking safety properties, we can improve Algorithm 1. If we are only interested in tracking a handful of variables, our approach naturally supports the computation of only some of the blocks. Complexity-wise this saves us a factor of b when tracking a constant number of blocks. Consider a six-dimensional model with the property $2x_1 - 3x_5 < 10$. A naive approach would compute the reachable states for blocks 1 and 3, i.e., upper and lower bounds for x_1, x_2, x_5, and x_6. However, we are only interested in the upper bound for x_1 and the lower bound for x_5. We modify the algorithm in two ways: First, we replace line 19 by a function that computes the support for the direction of interest. Second, we make the `approx` function in line 17 the identity (i.e., keep the lazy set). The reason is that we can evaluate the support directly on the lazy set, so there is no need for an additional overapproximation.

6 EVALUATION

We evaluate our implementation (called Algo. 1 here) on a set of SLICOT benchmarks [10, 16, 46] which reflect "real world" applications. Some of the original models are differential algebraic equations (DAEs), in which case we only kept the ODE part, i.e., the coefficient matrices A and B, which is consistent with related literature on reach set approximation. We use an Intel i5 3.50 GHz CPU and 16 GB RAM, running Linux and Julia v0.6.

6.1 Reach Tube Computation

We compare Algo. 1 to the state-of-the-art support function algorithm LGG in SpaceEx. We consider two cases: one dimension, where we only compute the reach tube in one variable, or full dimensions, where the whole reach tube is computed. For implementation reasons, we actually compute the reach set for at least one block (two variables), even in the 1D case, in which SpaceEx computes the bounds for a single variable only. In that sense, the comparison is biased in favor of SpaceEx. The results are given in Table 2.

We compare the precision using the bounds on a single variable (column Var. in the table) at the last time step. The SpaceEx bounds are taken as the baseline and we report the relative deviation. We expect a lower precision than SpaceEx for two reasons: First, our reach tube is a Cartesian product of 2D sets; this induces an error that is inherent to the decomposition method, as explained in Sect. 4. Second, SpaceEx uses a forward-backward interpolation model, which is more sophisticated than the forward-only model from Sect. 2.4; we note that our method could also use the SpaceEx model without requiring any other changes. In the experiments, the precision is moderately below that of SpaceEx. For the PDE model, the approximation error is relatively high. For the beam model our analysis is not only faster but also more precise. For all models tested, we observe a speedup; as expected, the improvement is more evident for large and sparse models. SpaceEx gave up or crashed with a segmentation fault for the three largest models.

6.2 Verification of Safety Properties

As described in Sect. 5.5, we can check safety properties in the form of (conjunctions and disjunctions of) linear inequalities over the state variables. In Table 3 we compare our results to those of Hylaa [9], a simulation-based verification tool in discrete time. Hylaa assumes that the inputs are constant between time steps, and we stick to this assumption for the purpose of comparison. We used the same time step as in the evaluation of [9] and were able to verify all safety properties except for the models ISS and FOM. Hylaa verified all benchmarks.[1] For seven out of the nine examples we observe a speedup which ranges from ×3 up to ×79. As expected, our approach scales best for the models whose properties only involve a few variables; PDE/FOM involve all variables, and here Hylaa is faster; ISS involves half of the variables, and here we still achieve a speedup of factor 7.

We also applied Algo. 1 to the benchmarks in dense time, which is not handled by Hylaa. We were able to successfully verify seven out of the nine benchmarks. The successful instances are shown in Table 4. In particular, we were able to verify a model with 10,000 variables in less than 7 minutes, where 98% of the time was spent in the discretization.

In both the discrete-time and dense-time cases, Algo. 1 fails to verify the same instances, namely ISS and FOM, because it is not precise enough. In both cases, the property involves a linear combination of state variables from different blocks. It is not unsurprising that this leads to a higher error, in particular because in these experiments we have only considered box directions. An alternative approach, which we have not investigated yet, would be to use a refined 2D `approx` function that introduces more constraints to the polygonal approximation of the reach set.

6.3 Sparsity

Each LTI system has its own structural properties, describing how each state influences the system dynamics. The SLICOT models have different sparsity patterns, which we exploited effectively in Algo. 1. We measure the sparsity of Φ as the number of 2×2 blocks with at least one non-zero element, divided by the total number of blocks (b^2). As a rule of thumb, for a given row-block the cost increases linearly in the number of occupied blocks. For models such as Heat and Beam, the sparsity is 0%, meaning that the matrix is completely dense, while for models such as ISS and FOM it is 97.8% and 99.8%, respectively. We note that the matrix power operation does not necessarily preserve the sparsity pattern, although it does in some particular cases, e.g., if Φ is block upper-triangular.

The efficiency with respect to the sparsity pattern is manifest in the small runtimes for sparse models, compared to higher runtimes for dense models. In contrast, non-decomposed methods cannot make full use of the sparsity since they rely on a high-dimensional LP even for evaluating the support vector in a single direction. This explains the very high speedup of ×50 for ISS. Moreover, the 1006-dimensional FOM model is analyzed in about the same time.

[1] We had to modify Hylaa (reduced the time horizon chunk size `max_steps_in_mem` from 527 to 400) for the FOM model to prevent out-of-memory problems.

Table 2: Reach tube computation in dense time. The number of time steps is 2×10^4 with step size $\delta = 1 \times 10^{-3}$ for both Algo. 1 and SpaceEx.

Model	n	Var.	Discretize (sec)	Runtime (sec) one state variable			Runtime (sec) all state variables			O.A. %
				Algo. 1	SpaceEx	speedup	Algo. 1	SpaceEx	speedup	
Motor	8	x_5	4.89×10^{-4}	1.06	1.90	1.8	4.46	9.29	2.1	21.53
Building	48	x_{25}	9.20×10^{-3}	4.49	9.54	2.1	1.15×10^2	2.24×10^2	1.9	6.50
PDE	84	x_1	3.30×10^{-2}	4.43	6.17×10^1	13.9	1.58×10^2	4.75×10^3	30.1	81.59
Heat	200	x_{133}	2.09×10^{-1}	2.47×10^1	1.02×10^2	4.1	2.32×10^3	5.68×10^3	2.4	0.05
ISS	270	x_{182}	2.03×10^{-1}	2.46	7.91×10^1	32.1	1.60×10^2	8.12×10^3	50.8	14.52
Beam	384	x_{89}	1.28	5.40×10^1	3.32×10^2	6.1	6.81×10^3	3.80×10^4	5.6	−30.35
MNA1	578	x_1	6.16	1.40×10^2	†	n/a	1.80×10^4	†	n/a	n/a
FOM	1006	x_1	4.70	1.06×10^1	†	n/a	2.92×10^3	†	n/a	n/a
MNA5	10913	x_1	3.68×10^2	1.38×10^3	†	n/a	T.O.	†	n/a	n/a

"Discretize" stands for the discretization time. "Runtime" stands for the total runtime. "O.A. %" stands for overapproximation in percent, which is computed as the increase in the bounds computed with Algo. 1 for the variable reported under "Var.", measured at the last time step, relative to the SpaceEx bounds. "†" marks a crash and "T.O." marks a timeout (10^5 sec).

Table 3: Verification of safety properties in discrete time. The number of time steps is 4×10^3 with step size $\delta = 5 \times 10^{-3}$ for both Algo. 1 and Hylaa.

Model	n	Property	Runtime (sec)					Verified
			Algo. 1			Hylaa	speedup	
			Discretize	Check	Total			
Motor	8	$x_1 \notin [0.35, 0.4] \vee x_5 \notin [0.45, 0.6]$	4.5×10^{-4}	2.48×10^{-1}	2.48×10^{-1}	1.6	6.5	yes
Building	48	$x_{25} < 6 \times 10^{-3}$	9.87×10^{-3}	5.20×10^{-1}	5.30×10^{-1}	2.5	4.7	yes
PDE	84	$y_1 < 12$	1.62×10^{-2}	2.22×10^1	2.22×10^1	3.5	0.2	yes
Heat	200	$x_{133} < 0.1$	1.48×10^{-1}	4.08	4.23	1.38×10^1	3.3	yes
ISS	270	$y_3 \in [-7, 7] \times 10^{-4}$	1.87×10^{-1}	2.12×10^1	2.14×10^1	1.53×10^2	7.1	no
Beam	384	$x_{89} < 2100$	3.66×10^{-1}	6.60	6.97	1.69×10^2	24.2	yes
MNA1	578	$x_1 < 0.5$	1.54	1.82×10^1	1.97×10^1	2.88×10^2	14.6	yes
FOM	1006	$y_1 < 185$	4.48	4.56×10^2	4.60×10^2	3.30×10^2	0.7	no
MNA5	10913	$x_1 < 0.2 \wedge x_2 < 0.15$	2.32×10^2	2.03×10^2	4.35×10^2	3.44×10^4	79.1	yes

Variables y_i denote outputs, as in (3), consisting of linear combinations of state variables x_i (involving all variables for PDE/FOM and half of the variables for ISS). The last column shows if we could verify the property for the given time step.

Table 4: Verification of safety properties in dense time.

Model	n	Property	δ	Runtime (sec)
Motor	8	$x_1 \notin [0.35, 0.4] \vee x_5 \notin [0.45, 0.6]$	1×10^{-3}	1.62
Building	48	$x_{25} < 6 \times 10^{-3}$	3×10^{-3}	8.76×10^{-1}
PDE	84	$y_1 < 12$	3×10^{-4}	1.03×10^3
Heat	200	$x_{133} < 0.1$	1×10^{-3}	1.48×10^1
Beam	384	$x_{89} < 2100$	5×10^{-5}	8.57×10^2
MNA1	578	$x_1 < 0.5$	4×10^{-4}	2.87×10^2
MNA5	10913	$x_1 < 0.2 \wedge x_2 < 0.15$	3×10^{-1}	3.92×10^2

Step sizes selected such that the property is satisfied. The time horizon is 20.

7 CONCLUSIONS

We have revisited the fundamental set-based recurrence relation that arises in the study of reachability problems with affine dynamics and nondeterministic inputs. We combined high dimensional linear algebra with low dimensional set computations and a state-of-the-art reachability algorithm. This approach is advantageous against the "curse of dimensionality": Reformulating the recurrence as a sequence of independent low-dimensional problems, we can effectively scale to high order systems. The overapproximation is conservative due the decomposition, and we have characterized the influence of initial states, inputs, dynamics, and time step with an analytical upper bound. We have evaluated our method on a set of real-world models from control engineering, involving many coupled variables. Numerical results show a speedup of up to two orders of magnitude with respect to state-of-the-art approaches that are non-decomposed. Apart from one exception, the overapproximation is within 22% of the non-decomposed solution. In the dense-time case, our approach can handle systems with substantially more variables than the state-of-the-art tool SpaceEx, by almost two orders of magnitude. Note that in this paper we have only tested box directions to represent two-dimensional sets, since the accuracy seemed sufficient. The investigation of a method producing more accurate low-dimensional projections, arbitrarily close to the exact projection, could deliver even more precise results.

Our approach can be parallelized, since the computations for each block are independent (see the loop in line 11 of Algorithm 1). Using a separate thread for each block, this will give a speedup of $n/2$. SpaceEx can also be parallelized, for bounding boxes with a theoretical speedup of up to $2n$. Comparing a parallelized Algorithm 1 with parallelized SpaceEx, we could theoretically see the speedup in Table 2 reduced from 50× to 12× (ISS benchmark). However, in practice, the speedup from parallelizing the LGG algorithm used in SpaceEx turns out much more modest [42]. We have only discussed partitions of two-dimensional blocks, and sequentially

for each pair of rows. However, there is no theoretical restriction in considering blocks of dimensions one or three for the explicit computations, which should give further gains in speed for the former and gains in precision for the latter. Furthermore, allowing overlapping blocks leads to relative completeness for software [32].

Similarity transformations, such as Schur or Jordan transformations, could be applied to the system's dynamics just after the discretization. This will eventually have an impact on the accumulated error, on the performance (since the number of non-zero blocks would change), or both. Characterizing the advantage of using similarity transformations for a given dynamics matrix, initial states, and inputs is left for future study.

ACKNOWLEDGMENTS

This work was partially supported by the European Commission under grant no. 643921 (UnCoVerCPS), by the Metro Grenoble through the project NANO2017, by the Air Force Office of Scientific Research under award no. FA2386-17-1-4065, and by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force. M.F. acknowledges stimulating discussions with Alexandre Rocca and Cesare Molinari.

REFERENCES

[1] 2017. Expokit. https://github.com/acroy/Expokit.jl. (2017).
[2] 2017. JuliaReach. https://github.com/JuliaReach. (2017).
[3] Matthias Althoff and Goran Frehse. 2016. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *CDC*. IEEE, 7439–7446.
[4] Matthias Althoff and Bruce H. Krogh. 2014. Reachability Analysis of Nonlinear Differential-Algebraic Systems. *IEEE Trans. Automat. Contr.* 59, 2 (2014), 371–383.
[5] Edward Anderson, Zhaojun Bai, Jack J. Dongarra, Anne Greenbaum, AlanMcKenney, Jeremy Du Croz, Sven Hammarling, James Demmel, Christian H. Bischof, and Danny C. Sorensen. 1990. LAPACK: a portable linear algebra library for high-performance computers. In *Supercomputing*. IEEE Computer Society, 2–11.
[6] Athanasios C Antoulas, Danny C Sorensen, and Serkan Gugercin. 2001. A survey of model reduction methods for large-scale systems. *Contemporary mathematics* 280 (2001), 193–220.
[7] Eugene Asarin and Thao Dang. 2004. Abstraction by Projection and Application to Multi-affine Systems. In *HSCC*. Springer, 32–47.
[8] Eugene Asarin, Thao Dang, Oded Maler, and Olivier Bournez. 2000. Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems. In *HSCC*. Springer, 20–31.
[9] Stanley Bak and Parasara Sridhar Duggirala. 2017. Simulation-Equivalent Reachability of Large Linear Systems with Inputs. In *CAV*. Springer, 401–420.
[10] Peter Benner, Volker Mehrmann, Vasile Sima, Sabine Van Huffel, and Andras Varga. 1999. SLICOT – A subroutine library in systems and control theory. In *Applied and computational control, signals, and circuits*. Springer, 499–539.
[11] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* 59, 1 (2017), 65–98.
[12] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Andreas Podelski, Christian Schilling, and Frédéric Viry. 2018. Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices. *CoRR* abs/1801.09526 (2018).
[13] Sergiy Bogomolov, Goran Frehse, Mirco Giacobbe, and Thomas A. Henzinger. 2017. Counterexample-guided refinement of template polyhedra. In *TACAS*. Springer, 589–606.
[14] Sergiy Bogomolov, Christian Herrera, Marco Muñiz, Bernd Westphal, and Andreas Podelski. 2014. Quasi-dependent variables in hybrid automata. In *HSCC*. ACM, 93–102.
[15] Sergiy Bogomolov, Corina Mitrohin, and Andreas Podelski. 2010. Composing Reachability Analyses of Hybrid Systems for Safety and Stability. In *ATVA*. Springer, 67–81.
[16] Younes Chahlaoui and Paul Van Dooren. 2005. Benchmark examples for model reduction of linear time-invariant dynamical systems. In *Dimension Reduction of Large-Scale Systems*. Springer, 379–392.
[17] Mo Chen, Sylvia L. Herbert, and Claire J. Tomlin. 2017. Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition. In *ICRA*. IEEE, 87–92.
[18] Xin Chen and Sriram Sankaranarayanan. 2016. Decomposed Reachability Analysis for Nonlinear Systems. In *RTSS*. IEEE, 13–24.
[19] Asen L. Dontchev. 1992. Time-scale decomposition of the reachable set of constrained linear systems. *MCSS* 5, 3 (1992), 327–340.
[20] Goran Frehse, Sergiy Bogomolov, Marius Greitschus, Thomas Strump, and Andreas Podelski. 2015. Eliminating spurious transitions in reachability with support functions. In *HSCC*. ACM, 149–158.
[21] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*. Springer, 379–395.
[22] Goran Frehse, Rajat Kateja, and Colas Le Guernic. 2013. Flowpipe approximation and clustering in space-time. In *HSCC*. ACM, 203–212.
[23] Komei Fukuda. 2004. From the zonotope construction to the Minkowski addition of convex polytopes. *J. Symb. Comput.* 38, 4 (2004), 1261–1272.
[24] Antoine Girard. 2005. Reachability of Uncertain Linear Systems Using Zonotopes. In *HSCC*. Springer, 291–305.
[25] Antoine Girard and Colas Le Guernic. 2008. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes* 41, 2, 8966 – 8971.
[26] Antoine Girard, Colas Le Guernic, and Oded Maler. 2006. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *HSCC*. Springer, 257–271.
[27] Elena V. Goncharova and Alexander I. Ovseevich. 2009. Asymptotics for Singularly Perturbed Reachable Sets. In *LSSC*. Springer, 280–285.
[28] Mark R. Greenstreet and Ian Mitchell. 1999. Reachability Analysis Using Polygonal Projections. In *HSCC*. Springer, 103–116.
[29] Colas Le Guernic and Antoine Girard. 2009. Reachability Analysis of Hybrid Systems Using Support Functions. In *CAV*. Springer, 540–554.
[30] Fred G. Gustavson. 1978. Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition. *ACM Trans. Math. Softw.* 4, 3 (1978), 250–269.
[31] Zhi Han and Bruce H. Krogh. 2006. Reachability Analysis of Large-Scale Affine Systems Using Low-Dimensional Polytopes. In *HSCC*. Springer, 287–301.
[32] Jochen Hoenicke, Rupak Majumdar, and Andreas Podelski. 2017. Thread modularity at many levels: a pearl in compositional verification. In *POPL*. ACM, 473–485.
[33] Shahab Kaynama and Meeko Oishi. 2010. Overapproximating the reachable sets of LTI systems through a similarity transformation. In *ACC*. 1874–1879.
[34] Shahab Kaynama and Meeko Oishi. 2011. Complexity reduction through a Schur-based decomposition for reachability analysis of linear time-invariant systems. *Int. J. Control* 84, 1 (2011), 165–179.
[35] Alexander B. Kurzhanski and Pravin Varaiya. 2000. Ellipsoidal Techniques for Reachability Analysis. In *HSCC*. Springer, 202–214.
[36] Alexander A. Kurzhanskiy and Pravin Varaiya. 2006. Ellipsoidal Toolbox (ET). In *CDC*. 1498–1503.
[37] Colas Le Guernic. 2009. *Reachability analysis of hybrid systems with linear continuous dynamics*. Ph.D. Dissertation. Université Grenoble 1 - Joseph Fourier.
[38] Colas Le Guernic and Antoine Girard. 2010. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems* 4, 2 (2010), 250 – 262. IFAC World Congress 2008.
[39] Alexander Vladimirovich Lotov and Alexis I Pospelov. 2008. The modified method of refined bounds for polyhedral approximation of convex polytopes. *Computational Mathematics and Mathematical Physics* 48, 6 (2008), 933–941.
[40] Ian M. Mitchell and Claire Tomlin. 2003. Overapproximating Reachable Sets by Hamilton-Jacobi Projections. *J. Sci. Comput.* 19, 1-3 (2003), 323–346.
[41] David Monniaux. 2010. Quantifier Elimination by Lazy Model Enumeration. In *CAV*. Springer, 585–599.
[42] Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. 2015. XSpeed: Accelerating reachability analysis on multi-core processors. In *Haifa Verification Conference*. Springer, 3–18.
[43] Stefan Schupp, Johanna Nellen, and Erika Ábrahám. 2017. Divide and Conquer: Variable Set Separation in Hybrid Systems Reachability Analysis. In *QAPL@ETAPS*. 1–14.
[44] Yassamine Seladji and Olivier Bouissou. 2013. Numerical Abstract Domain Using Support Functions. In *NASA Formal Methods*. Springer, 155–169.
[45] Roger B. Sidje. 1998. Expokit: A Software Package for Computing Matrix Exponentials. *ACM Trans. Math. Softw.* 24, 1 (1998), 130–156.
[46] Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. 2016. Large-Scale Linear Systems from Order-Reduction. In *ARCH*, Vol. 43. EasyChair, 60–67.
[47] Hoang-Dung Tran, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 2017. Order-reduction abstractions for safety verification of high-dimensional linear systems. *Discrete Event Dynamic Systems* 27, 2 (2017), 443–461.
[48] Chao Yan and Mark R. Greenstreet. 2008. Faster projection based methods for circuit level verification. In *ASP-DAC*. IEEE, 410–415.