Decomposing reach set computations with low-dimensional sets and high-dimensional matrices (extended version)

Bogomolov, Sergiy; Forets, Marcelo; Frehse, Goran; Podelski, Andreas; Schilling, Christian

Published in:
Information and Computation

DOI (link to publication from Publisher):
10.1016/j.ic.2022.104937

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Information and Computation 289 (2022) 104937

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/yinco

Decomposing reach set computations with low-dimensional sets and high-dimensional matrices (extended version)

Sergiy Bogomolov [a], Marcelo Forets [b], Goran Frehse [c], Andreas Podelski [d], Christian Schilling [e,*]

[a] Newcastle University, Newcastle upon Tyne, United Kingdom
[b] CURE, Universidad de la República, Uruguay
[c] ENSTA Paris, Paris, France
[d] University of Freiburg, Freiburg, Germany
[e] Aalborg University, Aalborg, Denmark

ARTICLE INFO

Article history:
Received 15 September 2021
Received in revised form 17 June 2022
Accepted 19 June 2022
Available online 30 June 2022

Keywords:
Linear time-invariant systems
Reachability analysis
Safety verification
Set recurrence relation

ABSTRACT

Approximating the set of reachable states of a dynamical system is an algorithmic way to rigorously reason about its safety. Despite progress on efficient algorithms for affine dynamical systems, available algorithms still lack scalability to ensure their wide adoption in practice. 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 are performed in the full dimension. Our method is applicable in both dense- and discrete-time settings. For a set of standard benchmarks, we show a speed-up of up to two orders of magnitude compared to the respective state-of-the-art tools, with only modest loss 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 before.

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 [1]. 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 (1) with set representations like ellipsoids [2,3], template polyhedra such as zonotopes [4,5] or support functions [6–10], or a combination [11]. The problem also generalizes to hybrid systems with

* Corresponding author.
E-mail addresses: sergiy.bogomolov@newcastle.ac.uk (S. Bogomolov), mforets@gmail.com (M. Forets), goran.frehse@ensta-paris.fr (G. Frehse), podelski@informatik.uni-freiburg.de (A. Podelski), christianms@cs.aau.dk (C. Schilling).

https://doi.org/10.1016/j.ic.2022.104937
0890-5401/

piecewise affine dynamics [12,13]. A key difficulty is scalability, as the cost of some set operations increases superlinearly with the dimension [14, Table 1]. A second challenge is the error accumulation for increasing values of N, known as the wrapping effect.

In this paper we extend the approach of the partial decomposition algorithm for solving (1) first presented in [15]. The complexity of non-decomposition approaches is mostly affected by the dimension n. 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 in 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,

$$x'(t) = Ax(t) + Bu(t), \qquad t \in [0, T], \tag{2}$$

where $T > 0$ is a given time horizon. 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,

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

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 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 discuss and extend the decomposition approach from [15], which combines low-dimensional sets with high-dimensional matrices to solve (1), and we 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 [16].

This article is the first in a two-part series on the decomposition of the symbolic analysis of linear dynamical systems. The present article provides the foundation by decomposing the symbolic analysis of a linear ODE. A preliminary version previously appeared in [15]. The second article extends the approach to systems with discrete state changes that can modify the ODE and the state vector [17].

New content for the extended version. This article is based on [15]. We have included all the proofs[1] among other major contributions:

- We extend the approach to arbitrarily-sized decompositions into subspaces (the approach in [15] was restricted to subspaces of size two).
- We extend the implementation to arbitrary low-dimensional set representations (the previous method in [15] was restricted to box directions).
- We perform an empirical evaluation of the approximation error.
- We improve the algorithm's precision when approximating the nondeterministic inputs.
- We improve the scalability of the discretization, using an efficient implementation of Krylov subspace methods, by several orders of magnitude.
- We investigate the influence of set approximations in the context of decomposition, and the influence of decompositions in higher dimension.

Combining these contributions, we are able to verify more benchmark problems and solve the previous ones faster.

Related work. Kaynama and Oishi consider a Schur-based decomposition to compute the reach set [18,19]. 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

[1] The conference paper [15] did not include proofs.

Krogh propose an approximation method that uses Krylov subspace approximations [20] 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 [21,22]. We do not consider this setting here. The reachability analysis tool Coho uses *projectahedra* – a possibly non-convex polyhedron consisting of the intersection of prisms, back-projected from 2D polygons – for set representation [23,24]. Seladji and Bouissou define a sub-polyhedra abstract domain based on support functions [25]. 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) [26]. 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 [27]. In a further approach, Bogomolov et al. [28] suggest an abstraction technique that employs dwell time bounds. Moreover, Bogomolov et al. [29] 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" [30]. 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 (2). The simulation equivalent reachability also involves a recurrence of the type (1), and we study its decomposed form in this work as well. Bak et al. have extended this approach to exploit specific problem structure, namely low input or output dimension. Using Krylov simulations, the approach can scale up to a billion dimensions [31].

Decomposition methods have also been designed for the reachability problem of nonlinear ODEs. Chen et al. consider controller synthesis and show that, using Hamilton-Jacobi methods, the (analytically) exact reach set can be reconstructed from an analysis of the subsystems for general ODE systems [32]. The system needs, however, to be composed of so-called *self-contained subsystems*, which is a strong assumption. In our setting of LTI systems, this corresponds to a dynamics matrix with block structure

$$\begin{pmatrix} A_1 & 0 & A_3 \\ 0 & B_2 & B_3 \\ 0 & 0 & C_3 \end{pmatrix}$$

in case of two subsystems. Our approach cannot make use of this structure due to a discretization, which results in a (discrete) system that in general does not preserve the structure. The technique is based on [33] which has no such limitation but suffers from a projection error.

Asarin and Dang propose a decomposition approach that projects variables away and abstracts them by time-unaware differential inclusions [34]. To address the overapproximation, the authors 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 [35]. Their method needs to iteratively check that the selected variables lie in the given intervals. A key difference with our work is that we do not replace variables by unknown inputs; instead we decompose the reach sets into subspaces. Moreover, our reachability algorithm is specific for LTI systems and exploits the system structure to be more efficient and more precise; our method also needs not be performed iteratively because the analysis of the subspaces is completely decoupled. Schupp et al. decompose a system by syntactic independence [36]. In our setting, this corresponds to dynamics matrices of block diagonal form. For such systems the dynamical error is zero in both approaches.

Our approach can handle blocks of arbitrary size, including blocks of different sizes in the same model. In the special case of 1-dimensional blocks, the approach is in a sense equivalent to a reach set approximation using interval arithmetic (see, e.g., [37] for a comprehensible overview). This connection merits some comments: One difference is that the traditional version is not free of the wrapping effect [38]. This is due to the fact that it targets nonlinear dynamics, for which the wrapping effect cannot be entirely avoided. The analytic error bounds given in this paper provide an indication of the kind of precision that can be achieved when combining interval arithmetic with a wrapping-free reach set algorithm. To achieve the soundness that traditional interval arithmetic reachability provides, we would have to construct the matrix exponential as an interval matrix and compute a sound upper bound on the support function in order to carry out the projection. For the matrix exponential, several sound implementations exist, either based on sound ODE solvers (e.g., VNODE-LP [39]) or directly on sound arithmetic (e.g., Arb [40]). For the computation of the support function, this is equivalent to finding sound upper bounds on the solution of a linear program. An efficient solution of this problem has been proposed, e.g., by Chen et al. [41].

Structure of the article. The paper is organized as follows. In Sect. 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 Sect. 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 Sect. 4. We present our reachability algorithm in Sect. 5, discuss the different techniques used to gain performance, and evaluate it experimentally in Sect. 6. Finally, we draw the conclusions in Sect. 7.

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 $\mathbb{I}_n$ 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 p-norm of a set $\mathcal{X} \subset \mathbb{R}^n$ is $\|\mathcal{X}\|_p = \sup_{x \in \mathcal{X}} \|x\|_p$, and the diameter is $\Delta_p = \sup_{x,y \in \mathcal{X}} \|x - y\|_p$. 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 the 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-fold Cartesian product of the intervals $[-|\bar{x}_i|, |\bar{x}_i|]$ for all $i = 1, \ldots, n$, where $|\bar{x}_i| := \sup_{x \in \mathcal{X}} |x_i|$.

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 $\rho_{\mathcal{X}} : \mathbb{R}^n \to \mathbb{R}$,

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

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

$$\sigma_{\mathcal{X}}(\ell) := \left\{ x \in \mathcal{X} : \ell^{\mathrm{T}} x = \rho_{\mathcal{X}}(\ell) \right\}. \tag{4}$$

When we speak of *the* support vector, we mean the choice of any support vector in (4). The following properties of support functions and support vectors are well-known [42] and will be used in the sequel.

Lemma 1. *For all compact convex sets $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, for all $n \times n$ real matrices M, all scalars λ, and all vectors $\ell \in \mathbb{R}^n$, we have:*

- $\rho_{\lambda\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(\lambda\ell)$, $\sigma_{\lambda\mathcal{X}}(\ell) = \lambda\sigma_{\mathcal{X}}(\lambda\ell)$
- $\rho_{M\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(M^{\mathrm{T}}\ell)$, $\sigma_{M\mathcal{X}}(\ell) = M\sigma_{\mathcal{X}}(M^{\mathrm{T}}\ell)$
- $\rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell)$, $\sigma_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \sigma_{\mathcal{X}}(\ell) \oplus \sigma_{\mathcal{Y}}(\ell)$
- $\rho_{\mathcal{X}\times\mathcal{Y}}(\ell) = \ell^{\mathrm{T}}\sigma_{\mathcal{X}\times\mathcal{Y}}(\ell)$,
  $\sigma_{\mathcal{X}\times\mathcal{Y}}(\ell) = (\sigma_{\mathcal{X}}(\ell_1), \sigma_{\mathcal{Y}}(\ell_2))$, $\ell = (\ell_1, \ell_2)$
- $\rho_{\mathrm{CH}(\mathcal{X}\cup\mathcal{Y})}(\ell) = \max(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell))$,
  $\sigma_{\mathrm{CH}(\mathcal{X}\cup\mathcal{Y})}(\ell) = \arg\max_{x,y}(\ell^{\mathrm{T}}x, \ell^{\mathrm{T}}y), \quad x \in \sigma_{\mathcal{X}}(\ell), y \in \sigma_{\mathcal{Y}}(\ell)$

In particular, 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 $\sigma_{M\mathcal{X}}(\ell) = M\sigma_{\mathcal{X}}(M^{\mathrm{T}}\ell)$. Given directions $\ell_1, \ldots, \ell_m$, a tight overapproximation of $\mathcal{X}$ is the *outer polyhedron* given by the constraints

$$\bigwedge_i \ell_i^{\mathrm{T}} x \leq \rho_{\mathcal{X}}(\ell_i). \tag{5}$$

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

$$\hat{\mathcal{X}} \subseteq \mathcal{X} \oplus \varepsilon\mathcal{B}_p^n \text{ and } \mathcal{X} \subseteq \hat{\mathcal{X}} \oplus \varepsilon\mathcal{B}_p^n. \tag{6}$$

The infimum $\varepsilon \geq 0$ that satisfies (6) 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. Recall that a compact set $\mathcal{X}$ is a *polytope* if there is a finite set of half-spaces whose intersection is $\mathcal{X}$. If $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ are polytopes, then

$$d_H^p(\mathcal{X}, \mathcal{Y}) = \max_{\ell \in \mathcal{B}_p^n} |\rho_{\mathcal{Y}}(\ell) - \rho_{\mathcal{X}}(\ell)|. \tag{7}$$

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

By adding directions using Kamenev's method [43,44], the outer polyhedron in (5) is within Hausdorff distance $\varepsilon\|X\|_p$ for $\mathcal{O}(1/\varepsilon^{n-1})$ directions, and this bound is optimal. It follows that accurate outer polyhedral approximations are only feasible in low dimensions. It is well-known that for $n = 2$, the bound can be lowered to $\mathcal{O}(1/\sqrt{\varepsilon})$ directions [43].

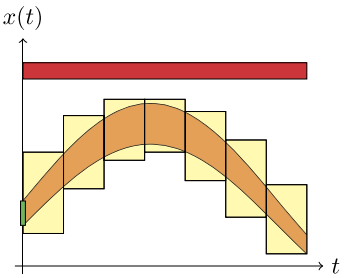


Fig. 1. Illustration of a reach tube (orange) with a set of initial states (green) and an approximation (yellow) that shows absence of error states (red). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

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,

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

where we map $Bu(t)$ to $u(t)$ without loss of generality. Here T is the time horizon, which we consider 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

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

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

$$\mathcal{R}(\mathcal{X}_0, \mathcal{U}, [t_1, t_2]) := \bigcup_{t_1 \le t \le t_2} \mathcal{R}(\mathcal{X}_0, \mathcal{U}, t). \tag{8}$$

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 (8). 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 $k = 0, 1, \ldots, N$. Note that while the input *set* $\mathcal{U}(k)$ is constant, the input *signal* $u(t)$ can still vary nondeterministically at arbitary times.

Starting from the continuous system (2)-(3), we are interested in reducing the reachability problem to the recurrence (1), with suitably transformed initial states and nondeterministic inputs. Such transformations are called approximation models (see [45] for a review). One can consider two distinct problems, which we call dense-time and discrete-time reachability, respectively. In the discrete-time case, the reach tube of the continuous system is only covered at finitely many time points, but not necessarily in between. On the other hand, the dense-time case corresponds to covering all possible trajectories of the given continuous system for every point between $[0, T]$. Next we describe the approximation models used in this article in more detail.

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

$$\mathcal{R}(\mathcal{X}_0, \mathcal{U}, [k\delta, (k+1)\delta]) \subseteq \mathcal{X}(k), \quad \forall\, k = 0, 1, \ldots, N. \tag{9}$$

Previous works have provided approximation models such that (9) holds [42,6,7]. In particular, in [7, 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

$$E_\psi(\mathcal{U}(k),\delta) := \boxdot(\Phi_2(|A|,\delta) \boxdot (A\mathcal{U}(k))) \tag{10}$$

$$E^+(\mathcal{X}_0,\delta) := \boxdot(\Phi_2(|A|,\delta) \boxdot (A^2\mathcal{X}_0)), \tag{11}$$

where $|A|$ represents the component-wise absolute value of A and the matrices $\Phi_1(A,\delta)$ and $\Phi_2(A,\delta)$ are defined via

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

The required transformations for dense time are:

$$\begin{cases} \Phi \leftarrow e^{A\delta} \\ \mathcal{X}(0) \leftarrow \mathrm{CH}\big(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta\mathcal{U}(0) \oplus E_\psi(\mathcal{U}(0),\delta) \oplus E^+(\mathcal{X}_0,\delta)\big) \\ \mathcal{V}(k) \leftarrow \delta\mathcal{U}(k) \oplus E_\psi(\mathcal{U}(k),\delta), \quad \forall\, k = 0,1,\ldots,N \end{cases} \tag{12}$$

For discrete-time reachability the transformations are:

$$\begin{cases} \Phi \leftarrow e^{A\delta} \\ \mathcal{X}(0) \leftarrow \mathcal{X}_0 \\ \mathcal{V}(k) \leftarrow \Phi_1(A,\delta)\mathcal{U}(k), \quad \forall\, k = 0,1,\ldots,N \end{cases} \tag{13}$$

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 (12) or (13), 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 [14]. 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 our 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. We characterize the decomposition of $\mathcal{X}$ into b sets as follows. Let $x \in \mathbb{R}^n$ and let $\{\pi_i\}_{i=1}^b$ be a set of contiguous projectors onto the coordinates of x, i.e., such that any $x \in \mathcal{X}$ can be uniquely written as $x = [\pi_1 x, \ldots, \pi_b x]$. We call the set

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

the *Cartesian decomposition* of $\mathcal{X}$. We call a set *decomposed* if it is identical to its Cartesian decomposition for some $\{\pi_i\}_{i=1}^b$. 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 $\hat{}$ (as in $\hat{\mathcal{X}}, \hat{\mathcal{Y}}$). Note that decomposition distributes over Minkowski sum:

$$\mathtt{dcp}(\mathcal{X} \oplus \mathcal{Y}) = \mathtt{dcp}(\mathcal{X}) \oplus \mathtt{dcp}(\mathcal{Y}). \tag{14}$$

If $\mathcal{X}$ is a polytope in constraint form, the projections can be very costly to compute, as this 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 the approximation is of no particular importance to the remainder of the paper, we simply assume an operator

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

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

3.2. Decomposing an affine map

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

$$\mathcal{X}' = \Phi\mathcal{X} \oplus \mathcal{V} = \begin{pmatrix} \Phi_{11} & \cdots & \Phi_{1b} \\ \vdots & \ddots & \vdots \\ \Phi_{b1} & \cdots & \Phi_{bb} \end{pmatrix} \mathcal{X} \oplus \mathcal{V}, \tag{15}$$

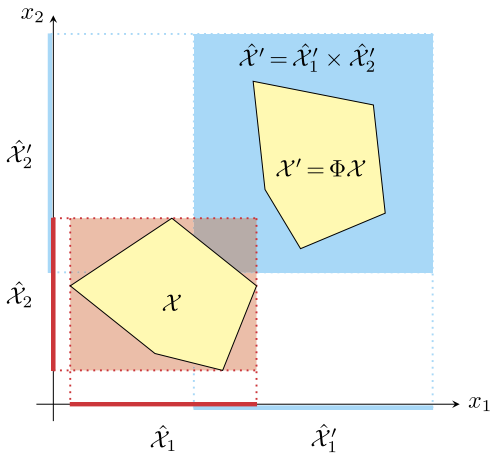


Fig. 2. The Cartesian decomposition of $\mathcal{X}$ into two blocks of size one is the set $\widehat{\mathtt{dcp}}(\mathcal{X}) = \hat{\mathcal{X}}_1 \times \hat{\mathcal{X}}_2$ (red). The decomposed image of the map $\mathcal{X}' = \Phi\mathcal{X}$ is the set $\hat{\mathcal{X}}'$ (blue) obtained by applying (16) to each interval $\hat{\mathcal{X}}_j$.

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

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

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

where Φ_{ij} denotes the 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*. Note that the dimension of each block is determined by the given Cartesian decomposition of the set $\mathcal{X}$, i.e., $\dim\Phi_{ij} = \dim\pi_i\mathcal{X} \times \dim\pi_j\mathcal{X}$.

The *decomposed image* of the map (15) is obtained in two steps: Cartesian decomposition and affine map in lower-dimensional subspaces. The first step transforms the full-dimensional sets $\mathcal{X}$ and $\mathcal{V}$ into Cartesian products of low-dimensional sets $\hat{\mathcal{X}}_1,\ldots,\hat{\mathcal{X}}_b$, using the operator $\widehat{\mathtt{dcp}}$ described in the previous section. The second step constructs the low-dimensional sets

$$\hat{\mathcal{X}}'_i := \bigoplus_{j=1}^{b} \Phi_{ij}\hat{\mathcal{X}}_j \oplus \hat{\mathcal{V}}_i, \quad \forall\, i = 1,\ldots,b. \tag{16}$$

We call the Cartesian product $\hat{\mathcal{X}}' = \hat{\mathcal{X}}'_1 \times \cdots \times \hat{\mathcal{X}}'_b$ the decomposed image of (15). For each i, $\hat{\mathcal{X}}'_i$ only depends on the i-th row-block of Φ. In Fig. 2 we illustrate the decomposed image of an affine map over a polygon.

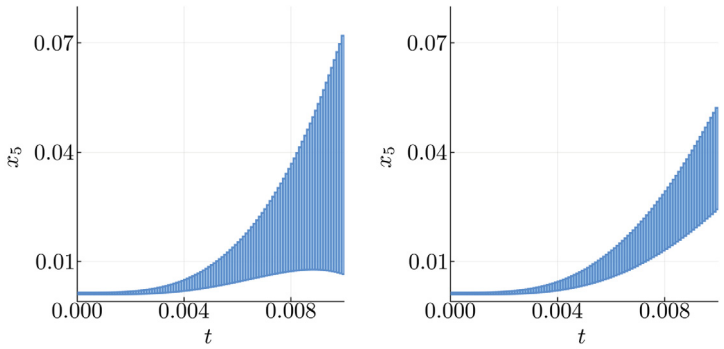
We now compare the cost of (15) and (16), assuming that every row-block of Φ is d-dimensional, and $b = n/d$ is an integer, without loss of generality. Let us denote the cost of computing the image of an $n\times n$ linear map by $C_\odot(n,m)$ and the cost of computing the Minkowski sum of two n-dimensional sets by $C_\oplus(n,m)$, where m is a parameter that depends on the set representation. For example, a square has dimension $n=2$ and $m=4$ vertices (resp. constraints), but it can also be represented as a zonotope with $m=2$ generators. The asymptotic complexity of performing the above operations for common set representations is shown in Table 1; we refer to [46,47,4,48] for further details. Since one Minkowski sum and one linear map are involved in (15), we have that the cost (recall that $n = db$) is in

$$\mathcal{O}(C_\odot(db,m) + C_\oplus(db,m)).$$

On the other hand, the aggregated cost for the i-th block in (16) is $bC_\oplus(d,m') + bC_\odot(d,m')$, where m' is the parameter for complexity (m) in d dimensions. The total cost for all b blocks is thus in

$$\mathcal{O}(b^2C_\odot(d,m') + b^2C_\oplus(d,m')).$$

Whenever $C_\odot$ and $C_\oplus$ depend at least quadratically on the dimension, and given that $m' \ll m$, the cost of the decomposed image (16) is asymptotically smaller than the cost of the non-decomposed image (15).



(a) With wrapping effect (b) Without wrapping effect

Fig. 3. Wrapping effect on the "Motor" example for variable x_5 with 100 steps of size $\delta = 1e{-}4$.

3.3. Decomposing an affine recurrence

Let us reconsider the affine recurrence (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 $\mathcal{X}(0)$:

$$\mathcal{X}(k+1) = \begin{pmatrix} \Phi_{11} & \cdots & \Phi_{1b} \\ \vdots & \ddots & \vdots \\ \Phi_{b1} & \cdots & \Phi_{bb} \end{pmatrix} \mathcal{X}(k) \oplus \mathcal{V}(k). \tag{17}$$

In this recurrence, the approximation error of the k-th step is propagated, and possibly amplified, in step $k+1$ (known as the wrapping effect). In Fig. 3 we show the impact on the example model "Motor" [49] (see Sect. 6). This effect can be partly avoided by using a non-recursive form [1]. (In Section 5.5 we show how the wrapping effect can be avoided completely.) We present two scenarios, which differ in whether the sequence of input sets is constant. Let Φ_{ij}^k 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 (17) 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:

$$\begin{cases} \hat{\mathcal{X}}_i(k) = \bigoplus_{j=1}^{b} \Phi_{ij}^k \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) = \hat{\mathcal{W}}_i(k) \oplus [\Phi_{i1}^k \cdots \Phi_{ib}^k]\mathcal{V}, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_n\}. \end{cases} \tag{18}$$

Note that the set $[\Phi_{i1}^k \cdots \Phi_{ib}^k]\mathcal{V}$ in (18) 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 (17) 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:

$$\begin{cases} \hat{\mathcal{X}}_i(k) = \bigoplus_{j=1}^{b} \Phi_{ij}^k \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) = \bigoplus_{j=1}^{b} \Phi_{ij} \hat{\mathcal{W}}_j(k) \oplus \hat{\mathcal{V}}_i, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_n\}. \end{cases} \tag{19}$$

Compared to equation (17), the solutions of equations (18) and (19) are both more precise (due to non-recursiveness) and more efficient to compute (due to low dimensionality). Fig. 4 visualizes the reach-tube computation in the two-dimensional case.

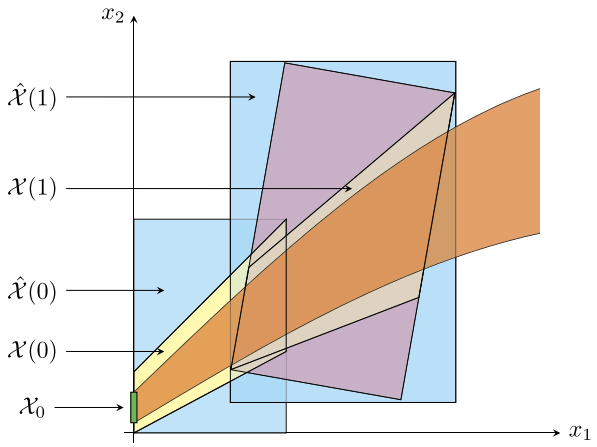


Fig. 4. Visualization of the first two reach tube computations in two dimensions. The first yellow set is the discretization $\mathcal{X}(0)$ of the initial states $\mathcal{X}_0$. The second yellow set is the (high-dimensional) propagated set $\mathcal{X}(1)$. We decompose into one-dimensional intervals. The approximation $\hat{\mathcal{X}}(0)$ is the lower left blue rectangle. Next we compute the propagated intervals. Their Cartesian product forms the approximation $\hat{\mathcal{X}}(1)$. Observe that $\hat{\mathcal{X}}(1)$ is not the smallest axis-aligned box containing $\mathcal{X}(1)$, but the smallest axis-aligned box containing the set resulting from propagating $\hat{\mathcal{X}}(0)$ (purple).

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 (13). 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 (12) do not preserve an initially decomposed set, and $\widehat{\texttt{dcp}}$ invariably introduces an approximation error. If the constraints on $\mathcal{X}(0)$ are known, an upper bound on the Hausdorff distance $d_H^p(\mathcal{X}(0), \hat{\mathcal{X}}(0))$ can be obtained using support functions [44]. 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 (18) or a single block as in (19). 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 (16) compared to (15). To simplify the discussion, we omit $\mathcal{V}$ without loss of generality, since we can rephrase (15) 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, $\bar{\mathcal{X}}'$ be the image of $\hat{\mathcal{X}}$ under the linear map $\bar{\mathcal{X}}' = \Phi\hat{\mathcal{X}}$, and $\hat{\mathcal{X}}'$ be the image of $\hat{\mathcal{X}}$ under the decomposed map (16). Then $\bar{\mathcal{X}}' \subseteq \hat{\mathcal{X}}'$ and*

$$d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') = \max_{\|\ell\|_p \le 1} \sum_{i,j} \rho_{\hat{\mathcal{X}}_j}(\Phi_{ij}^{\mathrm{T}} \ell_i) - \rho_{\hat{\mathcal{X}}_j}\left(\sum_k \Phi_{kj}^{\mathrm{T}} \ell_k\right) \tag{20}$$

where the max *is taken over $\ell = \ell_1 \times \cdots \times \ell_b$ in the unit ball of the p-norm, and $\Phi_{ij}^{\mathrm{T}} := (\Phi_{ij})^{\mathrm{T}}$.*

Proof. The support function of $\bar{\mathcal{X}}'$ on $\ell \in \mathbb{R}^n$ is, applying the properties in Lemma 1,

$$\begin{aligned}\rho_{\bar{\mathcal{X}}'}(\ell) &= \rho_{\Phi\hat{\mathcal{X}}}(\ell) = \rho_{\hat{\mathcal{X}}_1 \times \cdots \times \hat{\mathcal{X}}_b}(\Phi^{\mathrm{T}}\ell) \\ &= \sum_j \rho_{\hat{\mathcal{X}}_j}\left(\pi_j(\Phi^{\mathrm{T}}\ell)\right) = \sum_j \rho_{\hat{\mathcal{X}}_j}\left(\sum_k \Phi_{kj}^{\mathrm{T}} \ell_k\right).\end{aligned}$$

On the other hand,

$$\rho_{\hat{\mathcal{X}}'}(\ell) = \rho_{\hat{\mathcal{X}}_1' \times \cdots \times \hat{\mathcal{X}}_b'}(\ell) = \sum_i \rho_{\hat{\mathcal{X}}_i'}(\ell_i) = \sum_{i,j} \rho_{\hat{\mathcal{X}}_j}\left(\Phi_{ij}^{\mathrm{T}} \ell_i\right).$$

The result is obtained plugging these expressions into (7). □

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

Note that the converse is not true. For a simple counterexample, if $\hat{\mathcal{X}} = \{x\}$ consists of a single point, its image under *any* map also consists of a single point.

We can simplify the bound from Proposition 1 to clarify the relationship between the error and the norm of the matrix blocks Φ_{ij}. To quantify the error associated with 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 ℓ, and some[2] $p \geq 1$,

$$\rho_{\hat{\mathcal{X}}_i}(\ell) + \rho_{\hat{\mathcal{X}}_i}(-\ell) \leq \|\ell\|_{\frac{p}{p-1}} \Delta_i. \tag{21}$$

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 influence using (20), we get that the error bound is a weighted sum of the diameters of the state sets:

Proposition 2. *For $j = 1, \dots, b$, 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*

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

For the proof of Proposition 2 we need the following intermediate result. We can reduce the bound on the approximation error by freely selecting one specific row-block for each column-block of Φ. In the j-th column-block, we denote this selection by q_j.

Lemma 2. *The approximation error is bounded by*

$$d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') \leq \max_{\|\ell\|_p \leq 1} \sum_j \sum_{i \neq q_j} \rho_{\hat{\mathcal{X}}_j}(\Phi_{ij}^{\mathrm{T}} \ell_i) + \rho_{\hat{\mathcal{X}}_j}\left(-\Phi_{ij}^{\mathrm{T}} \ell_i\right).$$

Proof. We use the property of support functions that

$$\rho_{\mathcal{X}}(u+v) \geq \rho_{\mathcal{X}}(u) - \rho_{\mathcal{X}}(-v).$$

With this we can bound, picking any $q \in 1, \dots, b$,

$$\rho_{\hat{\mathcal{X}}_j}\left(\sum_k \Phi_{kj}^{\mathrm{T}} \ell_k\right) \geq \rho_{\hat{\mathcal{X}}_j}\left(\Phi_{qj}^{\mathrm{T}} \ell_q\right) - \sum_{k \neq q} \rho_{\hat{\mathcal{X}}_j}\left(-\Phi_{kj}^{\mathrm{T}} \ell_k\right).$$

We let q be a function of j and substitute the above in (20) with $k := i$:

$$\begin{aligned}
d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') &\leq \max_{\|\ell\|_p \leq 1} \sum_j \sum_i \rho_{\hat{\mathcal{X}}_j}(\Phi_{ij}^{\mathrm{T}} \ell_i) - \rho_{\hat{\mathcal{X}}_j}\left(\Phi_{q_j j}^{\mathrm{T}} \ell_{q_j}\right) \\
&\quad + \sum_{i \neq q_j} \rho_{\hat{\mathcal{X}}_j}\left(-\Phi_{ij}^{\mathrm{T}} \ell_i\right) \\
&= \max_{\|\ell\|_p \leq 1} \sum_j \sum_{i \neq q_j} \rho_{\hat{\mathcal{X}}_j}(\Phi_{ij}^{\mathrm{T}} \ell_i) + \rho_{\hat{\mathcal{X}}_j}\left(-\Phi_{ij}^{\mathrm{T}} \ell_i\right) \\
&\quad - \rho_{\hat{\mathcal{X}}_j}\left(\Phi_{q_j j}^{\mathrm{T}} \ell_{q_j}\right) + \rho_{\hat{\mathcal{X}}_j}\left(\Phi_{q_j j}^{\mathrm{T}} \ell_{q_j}\right) \\
&= \max_{\|\ell\|_p \leq 1} \sum_j \sum_{i \neq q_j} \rho_{\hat{\mathcal{X}}_j}(\Phi_{ij}^{\mathrm{T}} \ell_i) + \rho_{\hat{\mathcal{X}}_j}\left(-\Phi_{ij}^{\mathrm{T}} \ell_i\right). \quad \square
\end{aligned}$$

[2] Here, $\|\ell\|_{\frac{p}{p-1}}$ is the Hölder conjugate to the norm $\|\ell\|_p$. Inequality (21) can be deduced from [42, Prop. 2.2].

The bound is actually tight. Intuitively speaking, the judicious selection of q_j allows us to eliminate the block Φ_{ij} that contributes most to the error bound. Then we can bound the approximation error by

$$d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') \le \max_{\|\ell\|_p \le 1} \sum_j \sum_{i \ne q_j} \|\Phi_{ij}^{\mathrm{T}} \ell_i\|_{\frac{p}{p-1}} \Delta_j. \tag{23}$$

Proof of Proposition 2. First, we apply to (23) that

$$\|\Phi_{ij}^{\mathrm{T}} \ell_i\|_{\frac{p}{p-1}} \le \|\Phi_{ij}^{\mathrm{T}}\|_{\frac{p}{p-1}} \|\ell_i\|_{\frac{p}{p-1}} = \|\Phi_{ij}\|_p \|\ell_i\|_{\frac{p}{p-1}}.$$

This gives us

$$\begin{aligned} d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') &\le \max_{\|\ell\|_p \le 1} \sum_j \sum_{i \ne q_j} \alpha_j \|\ell_i\|_{\frac{p}{p-1}} \Delta_j \\ &= \max_{\|\ell\|_p \le 1} \sum_j \alpha_j \Delta_j \sum_{i \ne q_j} \|\ell_i\|_{\frac{p}{p-1}}. \end{aligned}$$

With $\sum_{i \ne q_j} \|\ell_i\|_{\frac{p}{p-1}} \le (b-1)\|\ell\|_{\frac{p}{p-1}}$ we get

$$d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}') \le \max_{\|\ell\|_p \le 1} (b-1)\|\ell\|_{\frac{p}{p-1}} \sum_j \alpha_j \Delta_j.$$

For $p \le 2$, it is known that $\|\ell\|_{\frac{p}{p-1}} \le \|\ell\|_p$, which leads to (22). The result holds for any $p \ge 1$ since $\|x\|_1 \ge \|x\|_p$. □

It remains to 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 (16). We use a simple lemma:

Lemma 3. *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 $d_H^p(\mathcal{X}', \bar{\mathcal{X}}') \le \|\Phi\|_p d_H^p(\mathcal{X}, \hat{\mathcal{X}})$.*

Proof. The claim follows from the definition of the Hausdorff distance (Eq. (6)) and the property $\|\Phi x\|_p \le \|\Phi\|_p \|x\|_p$, $x \in \mathbb{R}^n$, which holds for any compatible matrix norm. □

Combining Lemma 3 with Proposition 2 and the triangle inequality $d_H^p(\mathcal{X}', \hat{\mathcal{X}}') \le d_H^p(\mathcal{X}', \bar{\mathcal{X}}') + d_H^p(\bar{\mathcal{X}}', \hat{\mathcal{X}}')$, we get the following total error bound on the decomposed image computation:

Proposition 3.

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

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 the bound to affine recurrences.

4.2. Error of a decomposed affine recurrence

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

$$\|\Phi^k\|_p \le K_\Phi \alpha_\Phi^k, \qquad k \ge 0.$$

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, $\alpha_\Phi \le 1$ 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 conciseness we continue with the first formulation in the remaining section.

For constant inputs sets, (18) 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 Proposition 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, $\varepsilon(k) := d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big)$, is bounded by*

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

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

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

Proof. Using Eq. (18), and that $\widehat{\mathtt{dcp}}$ distributes over Minkowski sum,

$$\begin{aligned} &d_H^p\big(\underbrace{\widehat{\mathtt{dcp}}(\Phi^k\mathcal{X}(0)) \oplus \hat{\mathcal{W}}(k)}_{\hat{\mathcal{X}}(k)}, \underbrace{\Phi^k\mathcal{X}(0) \oplus \mathcal{W}(k)}_{\mathcal{X}(k)}\big) \\ &\leq d_H^p\big(\widehat{\mathtt{dcp}}(\Phi^k\mathcal{X}(0)), \Phi^k\mathcal{X}(0)\big) + d_H^p\big(\hat{\mathcal{W}}(k), \mathcal{W}(k)\big). \end{aligned}$$

Applying Proposition 3 with $\alpha_j \leq K_\Phi\alpha_\Phi^k$, we get the bound

$$\begin{aligned} &d_H^p\big(\widehat{\mathtt{dcp}}(\Phi^k\mathcal{X}(0)), \Phi^k\mathcal{X}(0)\big) \\ &\leq (b-1)\sum_j \alpha_j\Delta_j^x + \|\Phi^k\|_p d_H^p\big(\hat{\mathcal{X}}(0)), \mathcal{X}(0)\big) \\ &\leq K_\Phi\alpha_\Phi^k(b-1)\Delta_{\text{sum}}^x + K_\Phi\alpha_\Phi^k\varepsilon^x. \end{aligned}$$

Similarly, we get

$$\begin{aligned} d_H^p\big(\hat{\mathcal{W}}(k), \mathcal{W}(k)\big) &\leq \varepsilon^v + K_\Phi\big((b-1)\Delta_{\text{sum}}^v + \varepsilon^v\big)\sum_{s=1}^{k-1}\alpha_\Phi^s \\ &= \varepsilon^v + K_\Phi\big((b-1)\Delta_{\text{sum}}^v + \varepsilon^v\big)\alpha_\Phi\frac{1-\alpha_\Phi^{k-1}}{1-\alpha_\Phi}. \end{aligned}$$

The claim follows from combining both bounds. □

As a summary, 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., in 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 (18), with suitable sets $\mathcal{X}(0)$ and $\mathcal{V}$. The error bound follows from Proposition 4 and the decomposition errors for $\mathcal{X}(0)$ and $\mathcal{V}$.

In the discrete-time case (13), the initial states $\mathcal{X}(0)$ of the affine recurrence (18) are identical to the initial states $\mathcal{X}_0$ of the model, so their decomposition error is

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

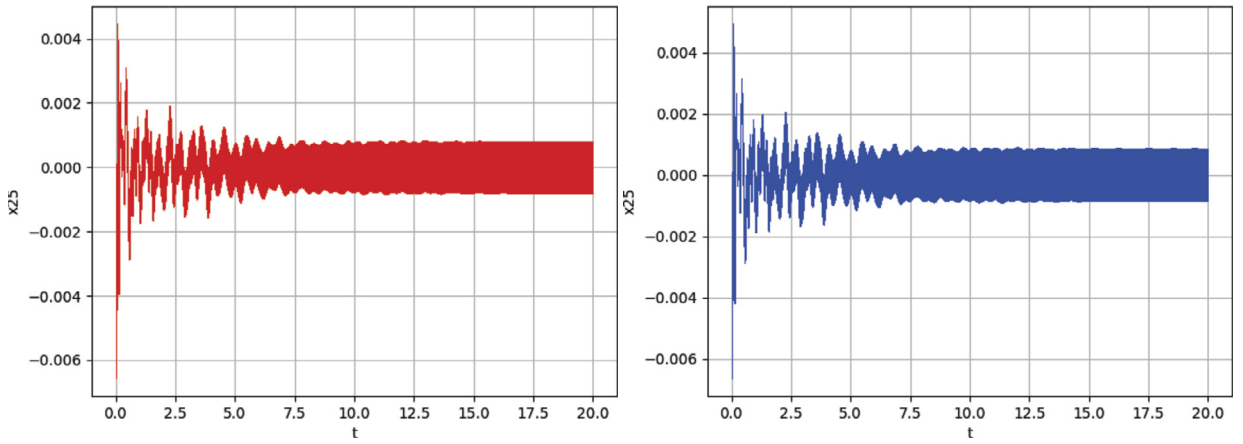
However, $\mathcal{V} = \Phi_1(A,\delta)\mathcal{U}$. Let $\hat{\mathcal{U}} = \mathtt{dcp}(\mathcal{U})$. By Lemma 3 we get

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

In the dense-time case (12), the initial states of the affine recurrence (18) 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 (14) that decomposition distributes over Minkowski sum. We get

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

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



(a) SpaceEx. (b) Our approach.

Fig. 5. Reach tubes for the "Building" example.

the decomposition error due to the inputs does not go to zero in Proposition 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 Proposition 4 gives a nonzero upper bound

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

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

In practice, the approximation error is much more modest, as we illustrate in Fig. 5.

4.4. Empirical evaluation

We investigate the practical approximation error for the "Motor" model, which we first introduce below for the purpose of discussion.

Motor model. The "Motor" model has eight state dimensions and two nondeterministic input dimensions. The dynamics follow the differential equation $x'(t) = Ax(t) + Bu(t)$ where A is

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.0865 & 8487.2 & 0 & 0 & 0 & 0 & 0 \\ -2592.1 & -21.119 & -698.91 & -141399 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.0865 & 8487.2 & 0 \\ 0 & 0 & 0 & 0 & -2592.1 & -21.119 & -698.91 & -141399 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}^{\mathrm{T}},$$

the input domain $\mathcal{U}$ is $[0.16, 0.3] \times [0.2, 0.4]$, and the initial states $\mathcal{X}_0$ are defined as $x_1 \in [0.002, 0.0025]$, $x_5 \in [0.001, 0.0015]$, and $x_i = 0$ for all other dimensions.

Observe that the blocks of dimensions $1-4$ and $5-8$ are completely decoupled from each other (both in $\mathcal{X}_0$ – at least in a set view – and in the dynamics). The pattern of the particular block matrix A is also preserved under matrix exponentiation, such that these blocks stay decoupled in the discretized matrix Φ. Thus we would consider decomposing into these two blocks the natural choice.

Experimental setup. In the following we analyze the difference in precision for various ways to decompose this eight-dimensional system. We fix the time step $\delta = 0.001$ and use a completely lazy set approximation; this means that both the result of the discretization following (12) and the iteration according to (18) are computed symbolically without intermediate approximations. Since $\hat{\mathcal{W}}$ quickly grows with the number of steps, we terminate the analysis after 50 steps.

The main parameter of the decomposition algorithm is the choice of the blocks, which is represented by a partition of the numbers 1 to $n = 8$; we discuss the choice of the partitions later.

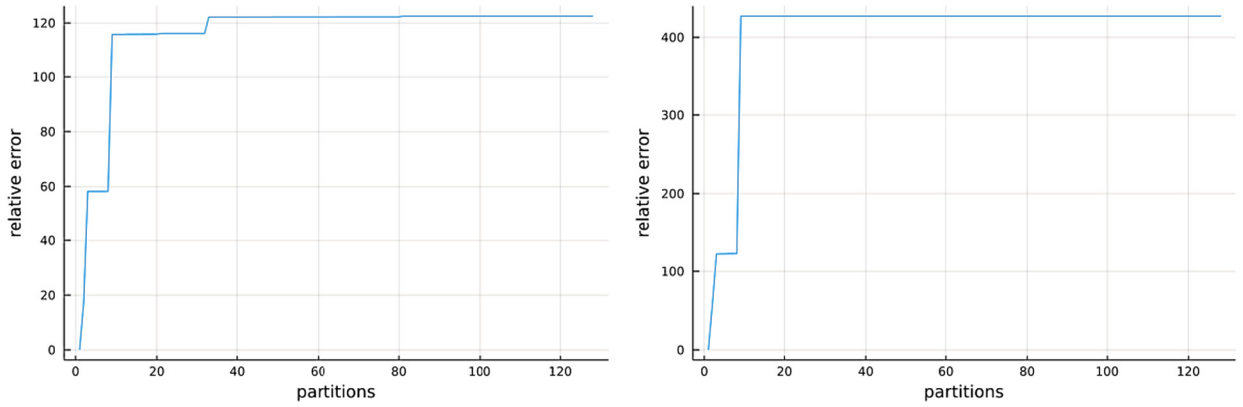


Fig. 6. Maximum relative error for 128 different partitions, sorted in ascending order. Left: initial states $\mathcal{X}_0$. Right: initial states $\mathcal{X}_0^L$.

In this model the initial set $\mathcal{X}_0$ is a hyperrectangle, i.e., the set is perfectly decomposed for all choices of the partition. But we recall that a hyperrectangular shape is generally not preserved by the discretized initial states $\mathcal{X}(0)$, which is the input to the decomposition algorithm; hence the decomposition still induces an approximation error in the initial states here. To investigate the approximation error for a non-hyperrectangular initial set, we also choose an alternative initial set $\mathcal{X}_0^L$ as the line segment between the lower-most and the upper-most vertex of $\mathcal{X}_0$, which can be considered the worst-case scenario for the decomposition algorithm.

We consider full-dimensional decompositions, i.e., we compute (a Cartesian product of) reach tubes for all eight dimensions. Recall again that we compute these reach tubes lazily here. As the reference we take the reach tube obtained for the trivial partition with a single block (i.e., effectively no decomposition). We compare the reach tubes obtained for different choices of the partition to this reference reach tube as follows. We fix a set of directions $D \subseteq \mathbb{R}^n$ and evaluate the support function $\rho(d, \mathcal{X}(k))$ for each direction $d \in D$ and $k = 0, \ldots, 49$. Also evaluating in the negative direction, we thus compute the intervals $(-\rho(-d, \mathcal{X}(k)), \rho(d, \mathcal{X}(k)))$. We then compare the approximation error in each direction by comparing the corresponding intervals $[l, u]$ and $[l_{\text{ref}}, u_{\text{ref}}]$ using the following (relative) formula:

$$100 * \left[\frac{l_{\text{ref}} - l}{u_{\text{ref}} - l_{\text{ref}}}, \quad \frac{u - u_{\text{ref}}}{u_{\text{ref}} - l_{\text{ref}}} \right]$$

In the comparison we use octagon directions for D:

$$D = \{(d_1, \ldots, d_n) \in \mathbb{R}^n \mid \exists i, j : d_i, d_j \in \{-1, 1\} \wedge \forall k : k = i \vee k = j \vee d_k = 0\}$$

Results. We begin with partitions that do not reorder dimensions. For example, in three dimensions these partitions are $[[1, 2, 3]]$, $[[1, 2], [3]]$, $[[1], [2, 3]]$, and $[[1], [2], [3]]$. In general there exist 2^{n-1} such partitions in n dimensions, so we consider 128 partitions in our case.

In Fig. 6 we plot the maximum relative error for different choices of the partition (under all $d \in D$ and steps k). For most partitions the maximum error is above 100% with hyperrectangular $\mathcal{X}_0$ and goes up to over 400% with $\mathcal{X}_0^L$. (The partition with no error is the trivial one-block partition.) However, this bird-eye view is too coarse to draw conclusions.

Next we have a closer look at the approximation error over time. In Fig. 7 we show that the error can differ significantly for different steps k of the recurrence. Interestingly, there is not much variation among the different partitions. Essentially there are two characteristic patterns. The first pattern starts with a relatively large error (around 60% with $\mathcal{X}_0$ and over 400% with $\mathcal{X}_0^L$ (not shown in the plot)) but then quickly shrinks to almost no error; a representative of this pattern is the partition into two four-dimensional blocks $[[1, 2, 3, 4], [5, 6, 7, 8]]$ (blue triangles in the figure). As argued before, this is the most natural choice to decompose the system at hand, as it does not involve errors in the dynamics. The other pattern evolves to two higher peaks at $k = 20$ and $k = 31$; a representative of this pattern is the partition with one-dimensional blocks $[[1], [2], [3], [4], [5], [6], [7], [8]]$ (green circles in the figure). Note that the one-dimensional partition generally results in the lowest precision.

Some partitions start with almost no approximation error for a hyperrectangular $\mathcal{X}_0$, e.g., the partition $[[1, 2], [3, 4, 5, 6, 7, 8]]$ (yellow stars in the figure), but then follow the second pattern and overall result in a larger approximation error. This illustrates that both the dynamics and the initial states can influence the approximation error independently. For the line segment $\mathcal{X}_0^L$ most partitions induce a high initial error as expected; the exceptions are partitions that keep the relevant dimensions (x_1 and x_5) in the same block. But surprisingly, this error quickly vanishes and the curves in the two plots soon look identical.

We also looked at partitions that reorder the dimensions. In general this can help bring dimensions together that are coupled in the initial states or the dynamics. In this particular model, however, the dimensions are already ordered in

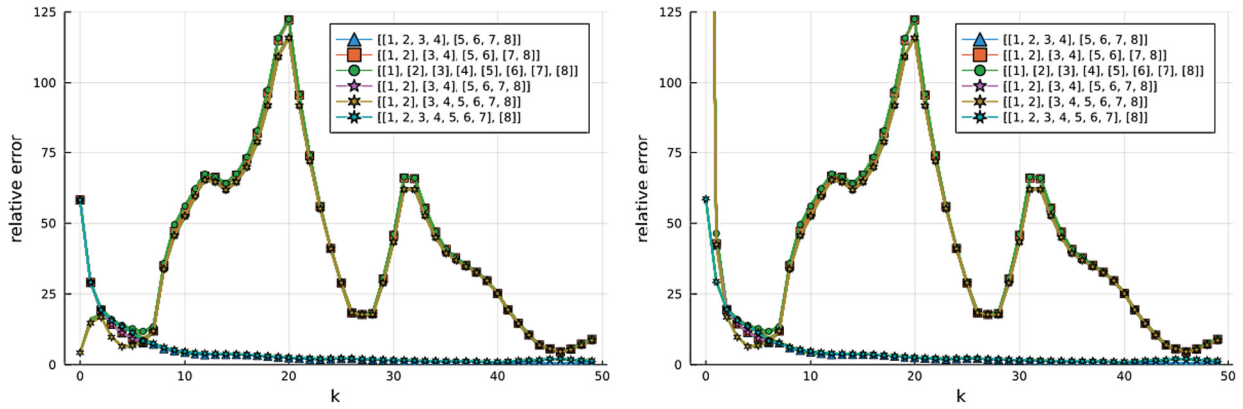


Fig. 7. Maximum relative error for 128 different partitions. Most curves overlay each other. Some selected partitions are drawn with markers on top. Left: initial states $\mathcal{X}_0$. Right: initial states $\mathcal{X}_0^L$. We cut off the y values to the same limit and omitted the trivial one-block partition (which would be a straight line at zero).

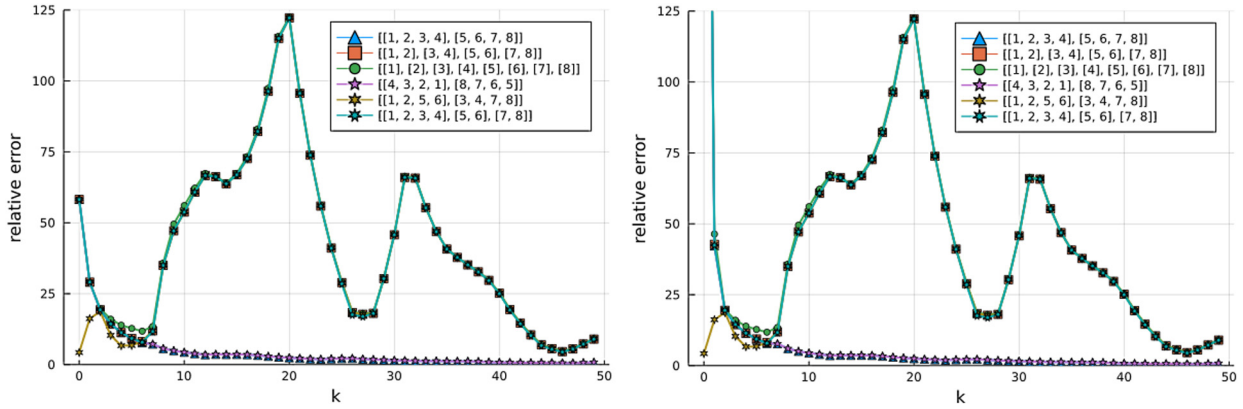


Fig. 8. Maximum relative error for six different partitions. Left: initial states $\mathcal{X}_0$. Right: initial states $\mathcal{X}_0^L$. We cut off the y values to the same limit.

alignment with the coupling in the dynamics, so a reordering is not helpful. In Fig. 8 we show three selected reordering partitions in comparison with three canonical ordered partitions. We observe that the precision stays the same. The notable exception is the partition that puts dimension x_5 into the first block (yellow stars in the figure), which induces almost no initial error; however, with time this partition follows the second (less precise) pattern because it separates some of the dynamically coupled variables.

In conclusion we can say that the decomposition error in the initial states becomes negligible compared to the decomposition error in the dynamics rather quickly. Of course whether a large initial error is acceptable also depends on the system properties one is interested in.

5. Algorithm & implementation

We now describe the decomposition method from Sect. 3 from an algorithmic point of view and discuss some crucial details of our implementation in *Julia* [50]. Given an LTI system of the form (2)-(3), we first apply a suitable approximation model from Sect. 2.4. Then we execute the corresponding decomposed recurrence from Sect. 3.3 to compute the reach tube or to check a safety property.

5.1. Lazy set representation

It is well known that the computational complexity of common set operations such as Minkowski sum, linear map, and Cartesian product crucially depend on two factors: the dimension and the set representation used (see, e.g., [14, Table 1]). Hence, to address scalability, our implementation exploits the principle of lazy (i.e., symbolic) evaluation. Common sets such as hypercubes in different norms or general polytopes each are represented by specific types. Each type has to provide a procedure to compute the support function in a given direction. The operations can be nested symbolically without actually evaluating them. Based on Lemma 1, this allows to compute the support function of the (nested) lazy set on demand. More details on our implementation of lazy operations can be found in [51]. 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 and concrete sets, i.e., the nesting depth is fixed (depending on the model dimension n; but see also Section 5.5). The alternative to using lazy data structures is to make the representation explicit after each operation, typically involving an overapproximation.

5.2. Discretization and approximation in the Krylov subspace

Recall from Sect. 2.4 that in the case of a dense-time model, we transform the system $(A, \mathcal{U}(\cdot), \mathcal{X}_0)$ into its discrete counterpart $(\Phi, \mathcal{V}(\cdot), \mathcal{X}(0))$. As said above, our implementation uses lazy set computations. However, operations of the form $e^{A\delta}\mathcal{X}_0$ require to compute the matrix exponential, which can be prohibitively expensive if the dimension of the system is large. Krylov subspace methods [52,53] are widely used to compute such operations lazily, i.e., by computing the action of the matrix exponential acting on vectors, the explicit computation of $e^{A\delta}$ is avoided. Such methods are particularly effective when applied to large and sparse matrices (we apply them for $n > 500$ in this work). While previous works have used Krylov subspace methods for discrete-time reachability [30] or dense-time reachability with zonotopes [54], we are not aware of a method using the support function. We introduce such a method below.

Let us recall the definition of E^+ in (11).

$$E^+(\mathcal{X}_0, \delta) := \boxdot(\overbrace{\Phi_2(|A|, \delta)}^{M_2} \overbrace{\boxdot(\underbrace{A^2}_{M_1} \underbrace{\mathcal{X}_0}_{\mathcal{H}_1})}^{\mathcal{H}_2}) \tag{24}$$

We assume here the common case that $\mathcal{X}_0$ is a hyperrectangle, respectively with center and radius $c_{\mathcal{X}_0}, r_{\mathcal{X}_0} \in \mathbb{R}^n$. First we observe that Eq. (24) has two common substructures of the form $\boxdot(M\mathcal{H})$, where $M \in \mathbb{R}^{n\times n}$ and $\mathcal{H} \subseteq \mathbb{R}^n$ is a hyperrectangle, respectively with center and radius $c_{\mathcal{H}}, r_{\mathcal{H}} \in \mathbb{R}^n$. The hyperrectangle $\boxdot(M\mathcal{H})$ has center $\mathbf{0}_n$ (the origin) and radius $|Mc_{\mathcal{H}}| + |M|r_{\mathcal{H}}$, where $|\cdot|$ takes the entry-wise absolute value. Thus $\boxdot(A^2\mathcal{X}_0)$ is the hyperrectangle with center $c_{\text{inner}} = \mathbf{0}_n$ and radius $r_{\text{inner}} = |A^2c_{\mathcal{X}_0}| + |A^2|r_{\mathcal{X}_0}$. We can apply this idea a second time (to M_2 and $\mathcal{H}_2$ in (24)) to compute E^+. But we can do even better: We know that $\Phi_2(|A|, \delta)c_{\text{inner}} = \mathbf{0}_n$, and we can compute $|\Phi_2(|A|, \delta)|r_{\text{inner}} = \Phi_2(|A|, \delta)r_{\text{inner}}$ using Krylov methods. The evaluation in Section 6 shows that this technique has a major impact on runtime performance and memory cost.[3] The procedure to compute E_ψ in (10) is almost identical, except that the matrix is not squared. In our implementation we use the Krylov methods implemented in the library `ExponentialUtilities.jl` [55].

It remains to compute $\mathcal{X}(0)$ and $\mathcal{V}(k)$ from (12). The non-scalable part is the computation of $\Phi\mathcal{X}_0$. We can combine Krylov methods with the support function to evaluate the expression for a set of a direction vectors: $\rho_{e^M\mathcal{X}_0}(d) = \rho_{\mathcal{X}_0}((e^M)^T d) = \rho_{\mathcal{X}_0}(e^{(M^T)}d)$ for any direction $d \in \mathbb{R}^n$ (cf. Lemma 1). We use axis-aligned ("box") directions in the evaluation.

5.3. Reach tube approximation

After we have obtained a discrete 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.

The result, a reach tube for each time interval of index 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 sets representing the low-dimensional 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* consists of all low-dimensional block indices; in particular it contains the blocks from *blocks*. 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. We write $\oplus$ and $\odot$ to denote lazy set representation of Minkowski sum and linear map, respectively.

The main loop starting in line 9 computes the reach tubes for each k. The array $\hat{\mathcal{X}}_{\text{tmp}}$ is filled with low-dimensional reach tubes in the inner loop (lines 11 to 18) for each block in *blocks*. Line 16 computes the term associated with inputs, which is added in line 17. The function `approx` overapproximates its argument (a lazy set) to a concrete set representation, e.g., a polygon or an interval. The function `approx_input` will be explained in Sect. 5.5, and by default it is identical to `approx`.

5.4. 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

[3] In [15] we used lazy matrix exponentiation for the largest benchmark model because the explicit approach would run out of memory. But our implementation was not efficient enough to be used in smaller cases. Our new implementation is several orders of magnitude faster.

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 low-dimensional reach tubes

```
1  X̂(0) ← dcp(X(0));                                                    // see Eq. (14)
2  all_blocks ← get_all_block_indices(dim(Φ));
3  P ← I_dim(Φ);
4  Q ← Φ;
5  V̂_tmp ← [];
6  for b_i ∈ blocks do
7  |  V̂_tmp[b_i] ← {0_dim(b_i)};
8  end
9  for k = 1 to N − 1 do
10 |  X̂_tmp ← [];
11 |  for b_i ∈ blocks do                                               // compute Eq. (16)
12 |  |  X̂_tmp[b_i] ← {0_dim(b_i)};
13 |  |  for b_j ∈ all_blocks do
14 |  |  |  X̂_tmp[b_i] ← X̂_tmp[b_i] ⊕ Q[b_i, b_j] ⊙ X̂(0)[b_j];
15 |  |  end
16 |  |  V̂_tmp[b_i] ← approx_input(V̂_tmp[b_i] ⊕ P[b_i, :] ⊙ V(k − 1));   // see Sect. 5.5
17 |  |  X̂_tmp[b_i] ← approx(X̂_tmp[b_i] ⊕ V̂_tmp[b_i]);                   // see Sect. 5.3
18 |  end
19 |  X̂(k) ← X̂_tmp;
20 |  P ← Q;
21 |  Q ← Q · Φ;
22 end
```

when tracking a constant number of blocks. Consider a six-dimensional model with the property $2x_1 - 3x_5 < 10$, and assume that we use two-dimensional blocks. 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 can compute the support vector directly from a lazy set, and so we replace the `approx` function in line 17 by the identity (i.e., keep the lazy set).

5.5. Lazy set propagation

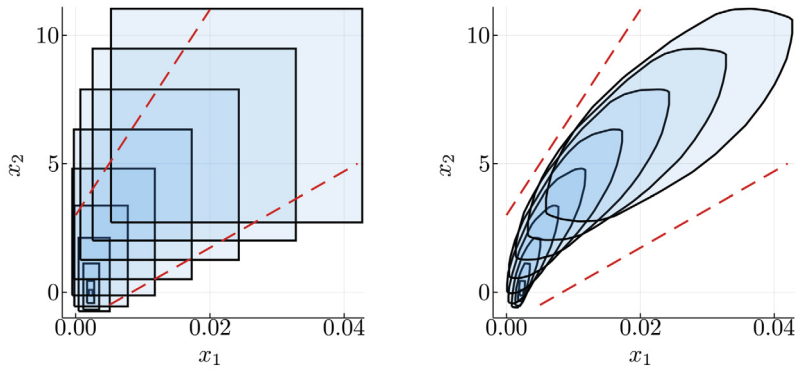
As mentioned in the previous paragraph, we can keep the set representation lazy if we are only interested in verifying a safety property. However, for systems with inputs, the inputs still incur a wrapping effect if we overapproximate them in each iteration. To address this issue, the `approx_input` function in line 16 abstracts from the concrete choice of overapproximation. By default we just call the `approx` function, but we can also instantiate the function with the identity, i.e., keep the elements of $\hat{\mathcal{V}}_{\text{tmp}}$ a lazy set, as it is proposed in the original LGG algorithm [6]. Observe that if we unroll the recurrence for the inputs in (18) resp. (19) then in iteration k we obtain a Minkowski sum of k summands. In general it becomes prohibitely expensive to work with such a deeply nested lazy set as k grows. However, recall from Lemma 1 that the support function of a Minkowski sum is $\rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell)$. Furthermore, assume that we evaluate the sets in the same directions ℓ in each iteration; this assumption is satisfied if we consider approximation with a template polytope or if we want to check a safety property. Then in each iteration we can reuse the result from the previous iterations and just need to evaluate the support function on the new summand. We note that we do not use this lazy approach by default and explicitly mention its use in the evaluation.

5.6. Sparse and dense matrices

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 [56] or a native Julia implementation for sparse matrices following Gustavson [57]. These specializations have a major impact on the runtime (around one order of magnitude).

5.7. Parallelization

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 and assuming uniform blocks of size b, this will give a speedup of n/b. SpaceEx can also be parallelized, for bounding boxes with a theoretical speedup of up to $2n$. Comparing a parallelized



(a) Box approximation. (b) ε-close approximation.

Fig. 9. Refining the "Motor" example in phase space for variables x_1 and x_2. On the right, $\varepsilon = 1e{-}4$.

Algorithm 1 with parallelized SpaceEx, we could theoretically see the speedup in Table 3 reduced from 64× to 32× ("ISS" benchmark). However, in practice, the speedup from parallelizing the LGG algorithm used in SpaceEx turns out to be much more modest [58].

5.8. Quality of the set approximation

Fixing the set $\mathcal{X}(0)$, there are three main factors that control the quality of the reach tube approximation $\hat{\mathcal{X}}(k)$ at step k. The first factor is the block partition, since inter-block dependencies in both $\mathcal{X}(0)$ and $\hat{\mathcal{X}}(k)$ are lost; a finer partition thus results in a more precise result.

The other two factors are the precision of the functions $\widehat{\texttt{dcp}}$ and `approx`, respectively. Assuming that `approx` uses box directions, any block partition and $\widehat{\texttt{dcp}}$ function that together satisfy $\mathcal{X}(0) = \hat{\mathcal{X}}(0)$ will result in the same reach tube (and hence be equivalent to the output of the LGG algorithm with box approximation). Note that $\mathcal{X}(0) = \hat{\mathcal{X}}(0)$ holds if the inter-block dimensions are independent in both $\mathcal{X}_0$ and the dynamics. On the other hand, if using box approximation in both $\widehat{\texttt{dcp}}$ and `approx`, the block partition becomes irrelevant because all dependencies will be lost. In this case the precision cannot improve over one-dimensional blocks.

For more precision, one can use ε-close polytope approximation, possibly with different values of ε for different blocks. In Fig. 9 we plot the reach tube for the "Motor" example in the $x_1 - x_2$ plane. Using ε-close approximation, we can show tight relational properties like non-reachability of states beyond the indicated half-spaces.

5.9. Changes of coordinates

Our block decomposition consists of packing variables into blocks, e.g., of size two, starting from the top to the bottom. As mentioned above, if we output variables from different blocks, e.g., x_1 and x_3, we will always obtain a box-shaped set. An ε-close approximation as described above is only possible between variables in the same block. Consequently, we may increase the precision by reordering variables such that variables that "belong together" stay in the same block.

Reordering can be implemented with a change of coordinates. If $x' = Ax + Bu$, and we let $w = Sx$ with S an $n \times n$ permutation matrix (in particular, S is orthogonal), then we consider the new dynamical system with state matrix $A \rightsquigarrow SAS^{\mathsf{T}}$ and inputs $B \rightsquigarrow SB$, respectively.

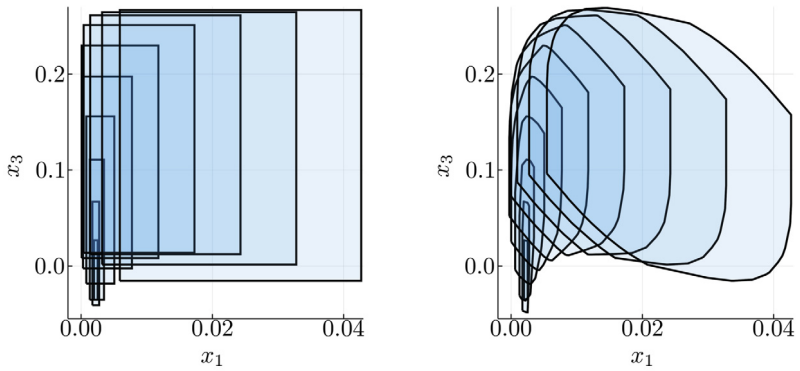
In Fig. 10 we plot the reach tubes of the "Motor" example in the $x_1 - x_3$ dimensions before and after such a transformation, with two-dimensional block decomposition. Here we swap the variables x_2 and x_3, which amounts to mixing blocks 1 and 2 of the original system.

In principle, other similarity transformations, such as a Schur decomposition, can be applied to the system's dynamics. In this case, the number of non-zero blocks may change, having an impact on the accumulated error, performance, or both. Our experiments suggest that it is not beneficial to apply a Schur decomposition to the "Motor" example.

5.10. Efficient 2D approximation

In many reachability tools the reach tube consists of a list of polytopes. For non-decomposed approaches, manipulating polytopes involves using an external linear programming (LP) back-end, possibly in high-dimensional space. When using 2D polygon approximations in Algorithm 1, we can employ the following efficient implementation for evaluating the support vector. This is particularly helpful when using non-template directions (e.g., epsilon-close approximation) (cf. Section 5.5) or if we need to post-process the reach tube (e.g., as part of a hybrid reachability loop).

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



(a) Variables from different blocks. (b) Variables from the same block.

Fig. 10. Phase space plot of the "Motor" example for variables x_1 and x_3 with two-dimensional blocks, before (left) and after (right) a coordinate transformation. In both plots we use ε-close approximation with $\varepsilon = 1e{-}4$.

Table 2
Common benchmark statistics. "Model" stands for the benchmark name. "n" stands for the benchmark dimension. "Variable" stands for the dimension that is analyzed in Table 3. Variables y_i in the safety properties denote outputs, as in (3), consisting of linear combinations of state variables x_i (involving all variables for the models PDE/FOM and half of the variables for the model ISS).

Model	n	Variable	Safety property
Motor	8	x_5	$x_1 \notin [0.35, 0.4] \vee x_5 \notin [0.45, 0.6]$
Building	48	x_{25}	$x_{25} < 6e{-}3$
PDE	84	x_1	$y_1 < 12$
Heat	200	x_{133}	$x_{133} < 0.1$
ISS	270	x_{182}	$y_3 \in [-7, 7] \times 10^{-4}$
Beam	348	x_{89}	$x_{89} < 2100$
MNA1	578	x_1	$x_1 < 0.5$
FOM	1006	x_1	$y_1 < 185$
MNA5	10913	x_1	$x_1 < 0.2 \wedge x_2 < 0.15$

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, e.g., using binary search.

Lemma 4. *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 $i \in \{1, \dots, m\}$, where we identify a_{m+1} with a_1. Let $\ell \in \mathbb{R}^2 \setminus \{\mathbf{0}_2\}$. Then there exists $i \in \{1, \dots, m\}$ such that $a_i \preceq \ell \preceq a_{i+1}$ and an optimal solution $\bar{x}$ of the linear program $\rho_{\mathcal{X}}(\ell) = \max_{x \in \mathcal{X}} \ell^{\mathsf{T}} x$ is given by $\bar{x} \in \{x : a_i^{\mathsf{T}} x \le b_i\} \cap \{x : a_{i+1}^{\mathsf{T}} x \le b_{i+1}\}$.*

The lemma follows from the fact that a linear program has an optimal solution in a vertex.

6. Evaluation

We evaluate our implementation called JuliaReach [59] on a set of SLICOT benchmarks [60,61,49] 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. The basic model statistics are outlined in Table 2. We use a machine with an Intel i5 3.50 GHz CPU and 16 GB RAM, running Linux. For matrix functions of exponential type, ExponentialUtilitiles.jl is used [62].

6.1. Reach tube computation

We compare JuliaReach to the state-of-the-art support function algorithm LGG implemented in SpaceEx. We consider two cases: one dimension, where we only compute the reach tube in one variable, and full dimensions, where we compute the whole reach tube. For our approach we use one-dimensional blocks in both cases. We note that we do not use parallelization or coordinate transformations in the evaluation. The results are given in Table 3.

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, we decompose $\mathcal{X}(0)$ into a Cartesian product, which 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

Table 3
Reach tube computation in dense time. The model statistics are given in Table 2. The number of time steps is 2e4 with step size $\delta = 1e{-}3$ for both JuliaReach (abbreviated JReach in this table) and SpaceEx.

Model	Discretize (sec)	Runtime (sec) one variable			Runtime (sec) all variables			O.A. %
		JReach	SpaceEx	Spdup	JReach	SpaceEx	Spdup	
Motor	3.31e−4	5.12e−1	1.90	3.7	3.85	9.29	2.4	21.53
Bldng	9.99e−3	1.71	9.54	5.6	6.65e1	2.24e2	3.4	6.62
PDE	1.55e−2	3.56	6.17e1	17.3	1.97e2	4.75e3	24.1	81.59
Heat	7.30e−2	8.28	1.02e2	12.3	1.05e3	5.68e3	5.4	0.05
ISS	1.25e−1	9.57e−1	7.91e1	82.7	1.26e2	8.12e3	64.4	14.52
Beam	5.21e−1	2.67e1	3.32e2	12.4	2.96e3	3.80e4	12.8	-30.35
MNA1	1.67e−1⋆	8.75e1	†	n/a	7.58e3	†	n/a	n/a
FOM	5.69e−2⋆	3.40	†	n/a	1.06e3	†	n/a	n/a
MNA5	1.92e−1⋆	8.93e1	†	n/a	T.O.	†	n/a	n/a

"Discretize" stands for the discretization time in JuliaReach. "Runtime" stands for the total runtime. "Spdup" stands for the relative speedup of JuliaReach over SpaceEx. "O.A. %" stands for overapproximation in percent, which is computed as the increase in the bounds computed with JuliaReach for the variable reported in Table 2, measured at the last time step, relative to the SpaceEx bounds. "†" marks a crash and "T.O." marks a timeout (1e5 sec). "⋆" marks the optimized algorithm described in Section 5.2.

Table 4
Verification of safety properties in discrete time. The model statistics are given in Table 2. The number of time steps is 4e3 with step size $\delta = 5e{-}3$ for both JuliaReach and Hylaa.

Model	Runtime (sec)				
	JuliaReach			Hylaa	Speedup
	Discretize	Check	Total		
Motor	2.53e−4	1.29e−1	1.29e−1	1.6	12.4
Building	1.52e−3	3.59e−1	3.61e−1	2.5	6.9
PDE	1.20e−2	2.86e1	2.87e1	3.5	0.1
Heat	9.63e−2	2.03	2.12	1.38e1	6.5
ISS ⊥	1.99e−1	6.68	6.88	1.53e2	22.2
ISS @135D	1.99e−1	2.74	2.93	1.53e2	52.2
Beam	3.14e−1	5.83	6.14	1.69e2	27.5
MNA1	4.06e−2⋆	1.78e1	1.79e1	2.88e2	16.1
FOM ⊥	2.25e−1	8.83e1	8.85e1	3.30e2	3.7
FOM @6D	2.25e−1	1.10e2	1.10e2	3.30e2	3.0
MNA5	1.64e−1⋆	1.87e2	1.87e2	3.44e4	183.9

" ⊥" marks a benchmark for that we could not verify the property for the given time step. For these benchmarks we show a second row with a successful run with different block sizes and lazy inputs (called *@kD* where *k* is the block size). "⋆" marks the optimized algorithm described in Section 5.2.

Table 5
Verification of safety properties in dense time.

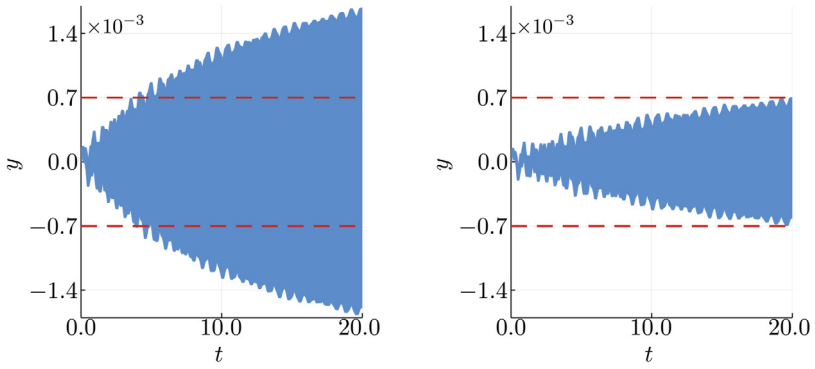
Model	n	Safety property	δ	Runtime (sec)
Motor	8	$x_1 \notin [0.35, 0.4] \vee x_5 \notin [0.45, 0.6]$	1e−3	6.66e−1
Building	48	$x_{25} < 6e{-}3$	2e−3	9.01e−1
PDE	84	$y_1 < 12$	3e−4	8.63e2
Heat	200	$x_{133} < 0.1$	1e−3	1.15e1
ISS @135D	270	$y_3 \in [-7, 7] \times 10^{-4}$	6e−4	2.17e1
Beam	348	$x_{89} < 2100$	5e−5	6.37e2
MNA1	578	$x_1 < 0.5$	4e−4	2.50e2
MNA5	10913	$x_1 < 0.2 \wedge x_2 < 0.15$	3e−1	7.22e2

Step sizes are selected such that the property is satisfied. The time horizon is 20. For ISS we use different block sizes and lazy inputs.

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 most experiments, the precision is moderately below that of SpaceEx. Only 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 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. 5.4, we can check safety properties in the form of (Boolean combinations of) linear inequalities over the state variables. In Table 4 we compare our results to those of Hylaa [30], 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 use the same time step as in the evaluation of [30] and again only look at one-dimensional blocks. With



(a) Overapproximated inputs. (b) Lazy inputs.

Fig. 11. Safety property output (in dense time) for the ISS model with mixed 1D/135D blocks, without and with lazy input representation.

these settings we are able to verify all safety properties except for the models ISS and FOM. HyLAA verifies all benchmarks.[4] For eight out of the nine examples we observe a speedup, which ranges from ×3 up to ×87. As expected, our approach scales best for the models whose properties only involve a few variables; PDE and FOM involve all variables, and for PDE (a dense model) HyLAA is faster; ISS involves half of the variables, and here we still achieve a ×22 speedup.

We also apply JuliaReach to the benchmarks in dense time, which HyLAA does not handle. With one-dimensional blocks we are able to verify seven out of the nine benchmarks. The results are shown in Table 5. In particular, we are able to verify the MNA5 model with 10,913 variables in 12 minutes, where 46% of the time is spent in the discretization.

Improving precision

In both the discrete-time and dense-time cases, JuliaReach fails to verify the same instances with one-dimensional block decomposition, namely ISS and FOM, because it is not precise enough. For both models, the property involves a linear combination of state variables from different blocks. It is not surprising that this leads to a high approximation error.

We thus increase the block size for a more precise analysis: For the ISS model, we wrap the output variables x_{136} to x_{270} in one 135-dimensional block. For FOM we use uniform six-dimensional blocks (plus one 4D block in the end). Additionally, we represent the inputs lazily (see Section 5.5) in order to get rid of the wrapping effect in the inputs. Fig. 11 shows the output function corresponding to the safety property for the ISS model.

With these settings we are able to verify the safety properties in discrete time and in dense time (ISS only). We report the performance in the respective Tables 4 and 5. Note that the analysis of the ISS model with lazy input representation is even faster, while the analysis of the FOM model is slower; we explain this observation with the fact that the ISS model has three inputs, and thus avoiding unnecessary overapproximations pays off, while the FOM model has only one input.

6.3. Sparsity

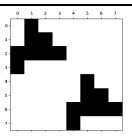
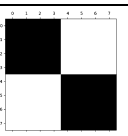
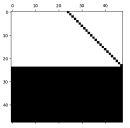
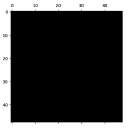
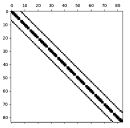
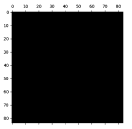
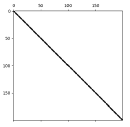
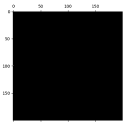
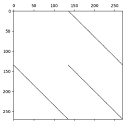
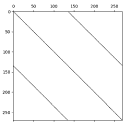
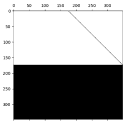
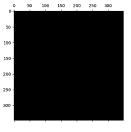
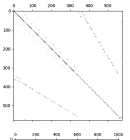
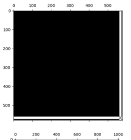
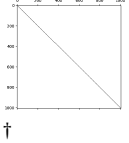
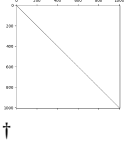
Each LTI system has its own structural properties, describing how each state influences the system dynamics. The SLICOT models have different sparsity patterns, which we effectively exploited in JuliaReach. We measure the sparsity of Φ as the number of low-dimensional blocks with at least one non-zero element, divided by the total number of blocks. Observe that line 14 of Algorithm 1 simplifies to a no-op for each zero block, and thus, as a rule of thumb, for a given row-block the cost increases linearly in the number of occupied blocks. Consider the case of 1×1 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 the sparsity is 99.3% and 99.9%, 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 ×64 for the ISS model.

For the largest models (MNA1, FOM and MNA5), we have used the optimized discretization method described in Section 5.2. The size of the Krylov subspace is $m = 30$, which is a common default value [63]. The runtimes obtained are up to three orders of magnitude faster than those obtained with the previous approach in [15] (times were 2.62, 3.26e−1 and 3.28e2 respectively). This illustrates the improved scalability of our method. For an overview of the sparsity patterns of the benchmark models used in the evaluation, see Table 6.

[4] We had to modify HyLAA to prevent out-of-memory problems for the FOM model; specifically, we reduced the time horizon chunk size `max_steps_in_mem` from 527 to 400.

Table 6
Sparsity characteristics of the SLICOT benchmarks. The sparsity of A (Φ), denoted as "sp A" ("sp Φ"), is the relative number of zero entries.

Model	n	sp A	sp Φ	Sparsity plot (A)	Sparsity plot (Φ)
Motor	8	75.0%	50.0%		
Building	48	48.9%	0.0%		
PDE	84	94.6%	0.0%		
Heat	200	98.5%	0.0%		
ISS	270	99.4%	99.3%		
Beam	384	49.9%	0.0%		
MNA1	578	99.5%	4.1%		
FOM	1006	99.9%	99.9%		
MNA5	10913	99.9%	99.7%	†	†

For the MNA5 model the plotting engine crashed (†).

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 integrated high-dimensional matrix computations with low-dimensional set computations in a state-of-the-art reachability algorithm. Our 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 analytic 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. With one exception, the overapproximation is within 22% of the non-decomposed solution. In the dense-time case, our approach can handle systems with substantially (almost two orders of magnitude) more variables than the state-of-the-art tool SpaceEx.

Note that in this paper we have only used box directions to represent low-dimensional sets. The accuracy was sufficient because, for our benchmark suite, all the low-dimensional properties are axis-aligned. In general, more accurate low-dimensional projections, arbitrarily close to the exact projection, may be required, and we have illustrated how to efficiently work with two-dimensional approximations.

We have only discussed uniform block structures that are not overlapping. Allowing blocks to overlap may give additional precision (for instance, it leads to relative completeness for software [64]). However, to make use of overlaps, we would have to intersect reach tubes, which is an operation that is difficult when using support functions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

M.F. acknowledges stimulating discussions with Alexandre Rocca and Cesare Molinari. We are thankful to Daniel Freire and Jorge Pérez for discussions with the scalable discretization approach. 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, by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems), by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754411, and by a grant from Toyota Motors North America R&D. 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.

References

[1] A. Girard, C.L. Guernic, O. Maler, Efficient computation of reachable sets of linear time-invariant systems with inputs, in: HSCC, in: LNCS, vol. 3927, Springer, 2006, pp. 257–271.
[2] A.B. Kurzhanski, P. Varaiya, Ellipsoidal techniques for reachability analysis, in: HSCC, in: LNCS, vol. 1790, Springer, 2000, pp. 202–214.
[3] A.A. Kurzhanskiy, P. Varaiya, Ellipsoidal toolbox (ET), in: CDC, 2006, pp. 1498–1503.
[4] A. Girard, Reachability of uncertain linear systems using zonotopes, in: HSCC, in: LNCS, vol. 3414, Springer, 2005, pp. 291–305.
[5] M. Althoff, B.H. Krogh, Reachability analysis of nonlinear differential-algebraic systems, IEEE Trans. Autom. Control 59 (2) (2014) 371–383, https://doi.org/10.1109/TAC.2013.2285751.
[6] C. Le Guernic, A. Girard, Reachability analysis of linear systems using support functions, Nonlinear Anal. Hybrid Syst. 4 (2) (2010) 250–262, https://doi.org/10.1016/j.nahs.2009.03.002, iFAC World Congress 2008.
[7] G. Frehse, C.L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: scalable verification of hybrid systems, in: CAV, in: LNCS, vol. 6806, Springer, 2011, pp. 379–395.
[8] G. Frehse, R. Kateja, C.L. Guernic, Flowpipe approximation and clustering in space-time, in: HSCC, ACM, 2013, pp. 203–212.
[9] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, A. Podelski, Eliminating spurious transitions in reachability with support functions, in: HSCC, ACM, 2015, pp. 149–158.
[10] S. Bogomolov, G. Frehse, M. Giacobbe, T.A. Henzinger, Counterexample-guided refinement of template polyhedra, in: TACAS, in: LNCS, vol. 10205, Springer, 2017, pp. 589–606.
[11] M. Althoff, G. Frehse, Combining zonotopes and support functions for efficient reachability analysis of linear systems, in: CDC, IEEE, 2016, pp. 7439–7446.
[12] E. Asarin, T. Dang, O. Maler, O. Bournez, Approximate reachability analysis of piecewise-linear dynamical systems, in: HSCC, in: LNCS, vol. 1790, Springer, 2000, pp. 20–31.
[13] C.L. Guernic, A. Girard, Reachability analysis of hybrid systems using support functions, in: CAV, in: LNCS, vol. 5643, Springer, 2009, pp. 540–554.
[14] M. Althoff, G. Frehse, A. Girard, Set propagation techniques for reachability analysis, Annu. Rev. Control, Robot. Auton. Syst. 4 (1) (2021) 369–395, https://doi.org/10.1146/annurev-control-071420-081941.
[15] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, C. Schilling, Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices, in: HSCC, ACM, 2018, pp. 41–50.
[16] JuliaReach, https://juliareach.github.io/JuliaReach-website/, 2019.
[17] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, C. Schilling, Reachability analysis of linear hybrid systems via block decomposition, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (11) (2020) 4018–4029, https://doi.org/10.1109/TCAD.2020.3012859.
[18] S. Kaynama, M. Oishi, Overapproximating the reachable sets of LTI systems through a similarity transformation, in: ACC, 2010, pp. 1874–1879.
[19] S. Kaynama, M. Oishi, Complexity reduction through a Schur-based decomposition for reachability analysis of linear time-invariant systems, Int. J. Control 84 (1) (2011) 165–179, https://doi.org/10.1080/00207179.2010.543703.
[20] Z. Han, B.H. Krogh, Reachability analysis of large-scale affine systems using low-dimensional polytopes, in: HSCC, in: LNCS, vol. 3927, Springer, 2006, pp. 287–301.
[21] A.L. Dontchev, Time-scale decomposition of the reachable set of constrained linear systems, Math. Control Signals Syst. 5 (3) (1992) 327–340, https://doi.org/10.1007/BF01211565.
[22] E.V. Goncharova, A.I. Ovseevich, Asymptotics for singularly perturbed reachable sets, in: LSSC, in: LNCS, vol. 5910, Springer, 2009, pp. 280–285.
[23] M.R. Greenstreet, I. Mitchell, Reachability analysis using polygonal projections, in: HSCC, in: LNCS, vol. 1569, Springer, 1999, pp. 103–116.
[24] C. Yan, M.R. Greenstreet, Faster projection based methods for circuit level verification, in: ASP-DAC, IEEE, 2008, pp. 410–415.
[25] Y. Seladji, O. Bouissou, Numerical abstract domain using support functions, in: NASA Formal Methods, in: LNCS, vol. 7871, Springer, 2013, pp. 155–169.
[26] A.C. Antoulas, D.C. Sorensen, S. Gugercin, A survey of model reduction methods for large-scale systems, Contemp. Math. 280 (2001) 193–220.
[27] H. Tran, L.V. Nguyen, W. Xiang, T.T. Johnson, Order-reduction abstractions for safety verification of high-dimensional linear systems, Discrete Event Dyn. Syst. 27 (2) (2017) 443–461, https://doi.org/10.1007/s10626-017-0244-y.
[28] S. Bogomolov, C. Mitrohin, A. Podelski, Composing reachability analyses of hybrid systems for safety and stability, in: ATVA, in: LNCS, vol. 6252, Springer, 2010, pp. 67–81.
[29] S. Bogomolov, C. Herrera, M. Muñiz, B. Westphal, A. Podelski, Quasi-dependent variables in hybrid automata, in: HSCC, ACM, 2014, pp. 93–102.
[30] S. Bak, P.S. Duggirala, Simulation-equivalent reachability of large linear systems with inputs, in: CAV, in: LNCS, vol. 10426, Springer, 2017, pp. 401–420.

[31] S. Bak, H. Tran, T.T. Johnson, Numerical verification of affine systems with up to a billion dimensions, in: HSCC, ACM, 2019, pp. 23–32.
[32] M. Chen, S.L. Herbert, C.J. Tomlin, Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition, in: ICRA, IEEE, 2017, pp. 87–92.
[33] I.M. Mitchell, C. Tomlin, Overapproximating reachable sets by Hamilton-Jacobi projections, J. Sci. Comput. 19 (1–3) (2003) 323–346, https://doi.org/10.1023/A:1025364227563.
[34] E. Asarin, T. Dang, Abstraction by projection and application to multi-affine systems, in: HSCC, in: LNCS, vol. 2993, Springer, 2004, pp. 32–47.
[35] X. Chen, S. Sankaranarayanan, Decomposed reachability analysis for nonlinear systems, in: RTSS, IEEE, 2016, pp. 13–24.
[36] S. Schupp, J. Nellen, E. Ábrahám, Divide and conquer: variable set separation in hybrid systems reachability analysis, in: QAPL@ETAPS, in: EPTCS, vol. 250, 2017, pp. 1–14.
[37] M.J. Cloud, B.C. Drachman, L.P. Lebedev, A Brief Introduction to Interval Analysis, Springer, 2014, pp. 179–193.
[38] P. Collins, A. Goldsztejn, The reach-and-evolve algorithm for reachability analysis of nonlinear dynamical systems, Electron. Notes Theor. Comput. Sci. 223 (2008) 87–102, https://doi.org/10.1016/j.entcs.2008.12.033.
[39] N.S. Nedialkov, Interval tools for ODES and DAES, in: SCAN, 2006.
[40] F. Johansson, Arb: efficient arbitrary-precision midpoint-radius interval arithmetic, IEEE Trans. Comput. 66 (2017) 1281–1292, https://doi.org/10.1109/TC.2017.2690633.
[41] L. Chen, A. Miné, P. Cousot, A sound floating-point polyhedra abstract domain, in: APLAS, in: LNCS, vol. 5356, Springer, 2008, pp. 3–18.
[42] C. Le Guernic, Reachability analysis of hybrid systems with linear continuous dynamics, Ph.D. thesis, Université Grenoble 1 - Joseph Fourier, 2009.
[43] G.K. Kamenev, An algorithm for approximating polyhedra, Comput. Math. Math. Phys. 36 (4) (1996) 533–544.
[44] A.V. Lotov, A.I. Pospelov, The modified method of refined bounds for polyhedral approximation of convex polytopes, Comput. Math. Math. Phys. 48 (6) (2008) 933–941, https://doi.org/10.1134/S0965542508060055.
[45] M. Forets, C. Schilling, Conservative time discretization: a comparative study, in: iFM, in: LNCS, vol. 13274, Springer, 2022, pp. 149–167.
[46] K. Fukuda, From the zonotope construction to the Minkowski addition of convex polytopes, J. Symb. Comput. 38 (4) (2004) 1261–1272, https://doi.org/10.1016/j.jsc.2003.08.007.
[47] D. Monniaux, Quantifier elimination by lazy model enumeration, in: CAV, in: LNCS, vol. 6174, Springer, 2010, pp. 585–599.
[48] A. Girard, C.L. Guernic, Efficient reachability analysis for linear systems using support functions, IFAC Proc. Vol. 41 (2) (2008) 8966–8971, https://doi.org/10.3182/20080706-5-KR-1001.01514.
[49] H. Tran, L.V. Nguyen, T.T. Johnson, Large-scale linear systems from order-reduction, in: ARCH, EasyChair, in: EPiC Series in Computing, vol. 43, 2016, pp. 60–67.
[50] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: a fresh approach to numerical computing, SIAM Rev. 59 (1) (2017) 65–98, https://doi.org/10.1137/141000671.
[51] M. Forets, C. Schilling, Lazysets.jl: scalable symbolic-numeric set computations, Proc. JuliaCon Conf. 1 (1) (2021) 11, https://doi.org/10.21105/jcon.00097.
[52] W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Q. Appl. Math. 9 (1) (1951) 17–29.
[53] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 29 (1) (1992) 209–228.
[54] M. Althoff, Reachability analysis of large linear systems with uncertain inputs in the Krylov subspace, IEEE Trans. Autom. Control 65 (2) (2020) 477–492, https://doi.org/10.1109/TAC.2019.2906432.
[55] ExponentialUtilities.jl, https://github.com/SciML/ExponentialUtilities.jl, 2021.
[56] E. Anderson, Z. Bai, J.J. Dongarra, A. Greenbaum, A. McKenney, J.D. Croz, S. Hammarling, J. Demmel, C.H. Bischof, D.C. Sorensen, LAPACK: a portable linear algebra library for high-performance computers, in: Supercomputing, IEEE Computer Society, 1990, pp. 2–11.
[57] F.G. Gustavson, Two fast algorithms for sparse matrices: multiplication and permuted transposition, ACM Trans. Math. Softw. 4 (3) (1978) 250–269, https://doi.org/10.1145/355791.355796.
[58] R. Ray, A. Gurung, B. Das, E. Bartocci, S. Bogomolov, R. Grosu, XSpeed: accelerating reachability analysis on multi-core processors, in: HVC, Springer, 2015, pp. 3–18.
[59] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, C. Schilling, Juliareach: a toolbox for set-based reachability, in: HSCC, ACM, 2019, pp. 39–44.
[60] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, A. Varga, SLICOT – a subroutine library in systems and control theory, in: Applied and Computational Control, Signals, and Circuits, Springer, 1999, pp. 499–539.
[61] Y. Chahlaoui, P. Van Dooren, Benchmark examples for model reduction of linear time-invariant dynamical systems, in: Dimension Reduction of Large-Scale Systems, Springer, 2005, pp. 379–392.
[62] C. Rackauckas, Q. Nie, Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia, J. Open Res. Softw. 5 (1) (2017), https://doi.org/10.5334/jors.151.
[63] J. Niesen, W.M. Wright, Algorithm 919: a Krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators, ACM Trans. Math. Softw. 38 (3) (2012) 22:1–22:19, https://doi.org/10.1145/2168773.2168781.
[64] J. Hoenicke, R. Majumdar, A. Podelski, Thread modularity at many levels: a pearl in compositional verification, in: POPL, ACM, 2017, pp. 473–485.