Fully-Automated Verification of Linear Systems Using Reachability Analysis with Support Functions

Mark Wetzlinger
Technische Universität München, Germany
m.wetzlinger@tum.de

Niklas Kochdumper
Stony Brook University, NY, USA
niklas.kochdumper@stonybrook.edu

Stanley Bak
Stony Brook University, NY, USA
stanley.bak@stonybrook.edu

Matthias Althoff
Technische Universität München, Germany
althoff@tum.de

ABSTRACT

While reachability analysis is one of the major techniques for formal verification of dynamical systems, the requirement to adequately tune algorithm parameters often prevents its widespread use in practical applications. In this work, we fully automate the verification process for linear time-invariant systems: Based on the computation of tight upper and lower bounds for the support function of the reachable set along a given direction, we present a fully-automated verification algorithm, which is based on iterative refinement of the upper and lower bounds and thus always returns the correct result in decidable cases. While this verification algorithm is particularly well suited for cases where the specifications are represented by halfspace constraints, we extend it to arbitrary convex unsafe sets using the Gilbert-Johnson-Keerthi algorithm. In summary, our automated verifier is applicable to arbitrary convex initial sets, input sets, as well as unsafe sets, can handle time-varying inputs, automatically returns a counterexample in case of a safety violation, and scales to previously unanalyzable high-dimensional state spaces. Our evaluation on several challenging benchmarks shows significant improvements in computational efficiency compared to verification using other state-of-the-art reachability tools.

KEYWORDS

Formal verification, set-based computing, high-dimensional systems, iterative refinement, automated parameter tuning, counterexample.

ACM Reference Format:
Mark Wetzlinger, Niklas Kochdumper, Stanley Bak, and Matthias Althoff. 2023. Fully-Automated Verification of Linear Systems Using Reachability Analysis with Support Functions. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3575870.3587121

HSCC '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0033-0/23/05…$15.00
https://doi.org/10.1145/3575870.3587121

1 INTRODUCTION

Formal verification of dynamical systems aims to show that undesired system behavior is avoided in the presence of uncertainty. A popular technique is reachability analysis, where one checks if the reachable set intersects unsafe sets defined by safety specifications. This principle has been applied in numerous use cases, such as aerospace/automotive applications, circuits, power systems, robotics, and biology [5, Tab. 2]. Since exact reachable sets cannot be computed except for a few special system classes [28], reachability algorithms either compute outer- or inner-approximations. Outer-approximations prove safety by showing that no unsafe state is reachable, whereas inner-approximations disprove safety by showing that at least one unsafe state is definitely reachable. The practical success of the verification process heavily depends on the tightness of the outer- and inner-approximation, which in turn depends on the tuning of algorithm parameters, such as the time step size. Due to the difficulty of manual algorithm parameter tuning, we believe that automation is a crucial step to facilitate the broader use of reachability analysis for formal verification. We aim to achieve this with the fully-automated verifier presented in this work.

1.1 Related work

There exist several groups of approaches for formal verification of dynamical systems: Barrier certificates [46] are level sets separating the unsafe region from the reachable states, thereby omitting an explicit computation of the reachable set. They are primarily studied in the context of stochastic systems [34, 47], where it is checked whether the probability of entering an unsafe region can be bounded by a given threshold. Another approach is theorem proving using differential dynamic logic [43, 44]. This is a special type of first-order logic for deductively proving properties of hybrid programs, which encode safety specifications for hybrid systems. A third group of approaches is based reformulating the reachability problem a constraint satisfaction problem [48]. For linear time-invariant (LTI) systems the predominant approach is to explicitly compute the reachable set and check for intersection with unsafe sets to prove or disprove safety [5]. As our approach utilizes reachability analysis, we restrict the remainder of our literature review to this group and focus on reviewing methods for LTI systems.

Since the reachable set is a zero sublevel set solution of a Hamilton-Jacobi-Isaacs partial differential equation [41], it can be approximated by solving the equation on a gridded state space. This is well-known to scale exponentially with the system dimension, which restricts the applicability to low-dimensional systems. Although

the issue can be alleviated using decoupling [15] or decomposition techniques [14], these methods are essentially not used for reachability analysis of linear systems.

Simulations from a sample of initial states within the initial set can be used to construct reachable sets [19]; this technique also extends to uncertain inputs [18]. Outer-approximations are obtained by enlarging the simulations based on sensitivity analysis, whereas inner-approximations can be constructed from the convex hull of the simulated states at each point in time [23]. Moreover, one can also construct an explicit representation of the reachable set using star sets in polynomial runtime [10, 20].

Another group of methods is based on set propagation [5]. These methods either outer- or inner-approximate the homogeneous and particular solution of an LTI system using set-based computing. Initially, griddy polyhedra [8, 16] and ellipsoids [37] were used as a set representation for computing outer-approximations, while current state-of-the-art techniques mainly use zonotopes [30, 32], support functions [39, 40], or a combination of both [4] as a set representation. In addition to the set representation, another difference between the various set propagation methods is the choice of the approximation model, which defines how to outer-approximate the homogeneous and particular solutions composing the reachable set. A recent survey [22] compares a wide variety of approximation models, which heavily differ in tightness and runtime: First-order methods [30, Sec. 3], [40, Eq. (2)] bloat the convex hull of the initial set and its linear transformation by a ball whose radius is computed using norms of the state matrix and the initial set. These methods are fast but the least accurate due to the first-order Taylor series expansion of the propagation matrix. The correction hull method [1, Sec. 3.2] computes a curvature enclosure by multiplication of an interval matrix representing the influence of higher-order terms of the propagation matrix with the initial set. It yields a tighter enclosure at the cost of a slightly increased computation time. The most accurate method is the forward-backward method for support functions [27, Sec. 3.1], [25, Sec. 2.4]. However, it requires the evaluation of n quadratic optimization problems in each step, with n being the state dimension, resulting in a significantly slower computation. Overall, one has to balance the trade-off between tightness and computation time when choosing the approximation model, which also has to fit the used set representation.

In contrast to the above algorithms for outer-approximations, approaches for inner-approximations using set-based computing are more scarce: By subtracting an error from the computed outer-approximation, inner-approximations can be represented by griddy polyhedra [17]. Another method is to use a union of ellipsoids, each of which touches the exact reachable set at exactly one point from the inside [37]. Linear matrix inequalities [33] have also been applied to compute ellipsoidal inner-approximations of the reachable set. Moreover, polytopic inner-approximations can be constructed by sampling vertices from zonotopes [32]. A simulation-based approach [23] aims to steer the trajectory toward edge cases by optimizing for a piecewise constant input trajectory.

For successful verification, the computed outer-approximation of the reachable set has to be tight enough. Since poor algorithm parameter tuning is one of the main sources for spurious counterexamples, a natural extension is to tune the parameters automatically: By using piecewise polynomial approximations in adaptively selected time intervals, the reachable set of an autonomous system can be approximated within a user-defined error bound by iteratively reducing the time step size [45]. Another method [25, 27] tunes the time step size to satisfy a user-defined error bound on the tightness along the given directions for the support function evaluation of the reachable set, but cannot rule out backtracking. A similar approach adaptively tunes all algorithm parameters without backtracking while respecting an error bound related to the Hausdorff distance between the exact reachable set and the computed enclosure [54]. While the desired error bound still has to be manually specified for the aforementioned approaches, automated verification algorithms automatically refine this error bound until the specification can be either proven or disproven: Brute-force approaches [9, 50] simply re-compute the reachable set with improved algorithm parameter values. The framework of counterexample-guided abstraction refinement (CEGAR) automatically refines the model [26, 53] or the set representation [12, 13]. In a real-time setting, the work in [35] refines the tightness constrained by the available computation time to choose between different controllers.

1.2 Contributions

We first introduce the general notation as well as set representations and operations in Sec. 2 and formally define the problem statement in Sec. 3. Afterward, we provide a comprehensive summary of the reachability algorithm in [4] for computing outer-approximations in Sec. 4.1. Our contributions are as follows:

- First, we present a novel reachability algorithm using support functions to compute inner-approximations (Sec. 4.2).
- Next, we design a fully-automated verification algorithm for the special case of unsafe sets given as halfspaces (Sec. 5.1).
- Moreover, we propose a fully-automated verification algorithm for arbitrary convex unsafe sets (Sec. 5.2).
- In case of a safety violation, our verification algorithms return a counterexample, which provides valuable insights to system engineers (Sec. 5.1-5.2).

Overall, our paper provides a complete description of support function reachability, combining outer- and inner-approximation with automated verification in a self-contained presentation. In contrast to previous work on reachability analysis using support functions, we provide the first approach that automatically verifies a given problem in decidable cases. Finally, the practical benefits of our novel algorithms are demonstrated on several challenging benchmark problems in Sec. 6.

2 PRELIMINARIES

We first define the notation and introduce all required set representations and operations.

2.1 Notation

We denote scalars and vectors by lowercase letters and matrices by uppercase letters. Given a vector $s \in \mathbb{R}^n$, $s_{(i)}$ represents the i-th entry; given a matrix $M \in \mathbb{R}^{m \times n}$, $M_{(i,\cdot)}$ and $M_{(\cdot,j)}$ refer to the i-th row and the j-th column, respectively. We use $\mathbf{0}$ and $\mathbf{1}$ for vectors and matrices of proper dimension containing only zeros or ones, as well as I_n to denote the identity matrix of dimension n. The concatenation of two matrices M_1, M_2 is written as $[M_1\ M_2]$ and $\text{diag}(s)$ returns a

square matrix with the vector s on its main diagonal and zeros otherwise. Exact sets are denoted by standard calligraphic letters $\mathcal{S}$, outer-approximations by $\widehat{\mathcal{S}}$, and inner-approximations by $\check{\mathcal{S}}$. Moreover, we overload the vector notation s to also denote the set $\{s\}$ consisting only of the point s and we abbreviate $-I_n\,\mathcal{S}$ to $-\mathcal{S}$. Intervals are represented by $[a, b]$, $a, b \in \mathbb{R}^n$, where $a \leq b$ holds element-wise. Interval matrices extend intervals by using matrices for the lower and upper bounds: $\boldsymbol{\mathcal{M}} = [\underline{M}, \overline{M}] = \{M \in \mathbb{R}^{m\times n} \mid \underline{M} \leq M \leq \overline{M}\}$, where the inequality is again evaluated element-wise. The operations $\text{center}(\mathcal{S})$ and $\text{box}(\mathcal{S})$ return the volumetric center and the tightest enclosing interval of $\mathcal{S}$, respectively. The sign function $\text{sgn}(x)$ returns $-1, 0$, and 1 for the input ranges $x < 0, x = 0$, and $x > 0$, respectively. We use $O(\cdot)$ to denote the big O notation.

2.2 Set representations and operations

Our reachability algorithms are based on support functions [31, Sec. 2], which can represent any convex set:

Definition 1. (Support function) Given a compact convex set $\mathcal{S} \subset \mathbb{R}^n$ and a direction $\ell \in \mathbb{R}^n$, the *support function* $\rho(\mathcal{S}, \ell) : \mathbb{R}^n \to \mathbb{R}$ and the *support vector* $v(\mathcal{S}, \ell) \in \mathbb{R}^n$ are defined as

$$\rho(\mathcal{S}, \ell) := \max_{x\in\mathcal{S}} \ell^\top x, \quad v(\mathcal{S}, \ell) := \arg\max_{x\in\mathcal{S}} \ell^\top x.$$

Note that the support vector is not necessarily unique. □

Our reachability and verification algorithms support arbitrary convex sets, where we only require that the support function can be evaluated. While the set $\mathcal{S}$ can also be defined by a symbolic equation returning its support function, the uncertain sets in verification tasks are often defined using common convex set representations such as intervals, zonotopes, polytopes, zonotope bundles [6], constrained zonotopes [52], ellipsoids, ellipsotopes [36], or capsules [49]. Hence, we now provide the support functions and support vectors for some of these set representations, where we focus on the most commonly-used ones. We begin with zonotopes [30, Def. 1]:

Definition 2. (Zonotope) Given a center $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n\times\gamma}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^{\gamma} G_{(\cdot,i)}\,\alpha_i \;\middle|\; \alpha_i \in [-1, 1] \right\}.$$

For the support function and support vector, we have [31, Sec. 2]

$$\rho(\mathcal{Z}, \ell) = \ell^\top c + \sum_{i=1}^{\gamma} |\ell^\top G_{(\cdot,i)}|,$$

$$v(\mathcal{Z}, \ell) = c + \sum_{i=1}^{\gamma} \text{sgn}(\ell^\top G_{(\cdot,i)})\, G_{(\cdot,i)}.$$

We use the shorthand $\mathcal{Z} = \langle c, G\rangle_Z$. □

Polytopes can be represented in halfspace or vertex representation:

Definition 3. (Polytope) The halfspace representation of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is given by the intersection of w halfspaces, which corresponds to a set of inequality constraints defined by the matrix $H \in \mathbb{R}^{w\times n}$ and the offset vector $f \in \mathbb{R}^w$:

$$\mathcal{P} := \{x \in \mathbb{R}^n \mid Hx \leq f\}.$$

The vertex representation is given by the convex hull of the polytope vertices $v_1, \ldots, v_s \in \mathbb{R}^n$:

$$\mathcal{P} := \left\{ \sum_{i=1}^{s} \beta_i\, v_i \;\middle|\; \sum_{i=1}^{s} \beta_i = 1,\ \beta_i \geq 0 \right\}.$$

For the vertex representation, the support function is given as $\max_{i\in\{1,\ldots,s\}} \ell^\top v_i$ and the support vector is the corresponding maximizing vertex. For the halfspace representation, the support function and the support vector can be obtained by linear programming. We use the shorthands $\mathcal{P} = \langle H, f\rangle_H$ and $\mathcal{P} = \langle [v_1 \ldots v_s]\rangle_V$. □

Another common convex set representation are ellipsoids:

Definition 4. (Ellipsoid) Given a center $c \in \mathbb{R}^n$ and a positive semi-definite shape matrix $Q \in \mathbb{R}^{n\times n}$, an ellipsoid $\mathcal{E} \subset \mathbb{R}^n$ is

$$\mathcal{E} := \left\{x \;\middle|\; (x - c)^\top Q^{-1} (x - c) \leq 1\right\}.$$

The support function and the support vector are computed as [38]

$$\rho(\mathcal{E}, \ell) = \ell^\top c + \sqrt{\ell^\top Q \ell}, \quad v(\mathcal{E}, \ell) = c + \frac{Q\,\ell}{\sqrt{\ell^\top Q \ell}}.$$

We use the shorthand $\mathcal{E} = \langle c, Q\rangle_E$. □

Given the sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{m\times n}$, we require the set operations linear map $M\,\mathcal{S}_1$, Minkowski sum $\mathcal{S}_1 \oplus \mathcal{S}_2$, and convex hull $\text{conv}(\mathcal{S}_1, \mathcal{S}_2)$, which are defined as

$$M\,\mathcal{S}_1 := \{Ms_1 \mid s_1 \in \mathcal{S}_1\},$$
$$\mathcal{S}_1 \oplus \mathcal{S}_2 := \{s_1 + s_2 \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\},$$
$$\text{conv}(\mathcal{S}_1, \mathcal{S}_2) := \left\{\lambda s_1 + (1 - \lambda)s_2 \mid s_1 \in \mathcal{S}_1,\ s_2 \in \mathcal{S}_2,\ \lambda \in [0, 1]\right\}.$$

For support functions, these operations are evaluated by [40, Prop. 2]

$$\rho(M\,\mathcal{S}_1, \ell) = \rho(\mathcal{S}_1, M^\top \ell), \tag{1}$$
$$\rho(\mathcal{S}_1 \oplus \mathcal{S}_2, \ell) = \rho(\mathcal{S}_1, \ell) + \rho(\mathcal{S}_2, \ell), \tag{2}$$
$$\rho(\text{conv}(\mathcal{S}_1, \mathcal{S}_2), \ell) = \max\{\rho(\mathcal{S}_1, \ell), \rho(\mathcal{S}_2, \ell)\}. \tag{3}$$

For zonotopes $\mathcal{Z}_1 = \langle c_1, G_1\rangle_Z, \mathcal{Z}_2 = \langle c_2, G_2\rangle_Z \subset \mathbb{R}^n$, linear map and Minkowski sum are computed by [1, Eq. (2.1)]

$$M\,\mathcal{Z}_1 = \langle Mc_1, MG_1\rangle_Z,$$
$$\mathcal{Z}_1 \oplus \mathcal{Z}_2 = \langle c_1 + c_2, [G_1\ G_2]\rangle_Z,$$

and the linear map $\boldsymbol{\mathcal{M}}\,\mathcal{Z}_1$ with an interval matrix $\boldsymbol{\mathcal{M}} = [\underline{M}, \overline{M}]$ can be tightly enclosed according to [7, Thm. 4]:

$$\boldsymbol{\mathcal{M}}\,\mathcal{Z}_1 \subseteq \left\langle M_c c_1, \left[M_c G_1\ \ \text{diag}\left(M_r(|c_1| + \textstyle\sum_{i=1}^{\gamma} |G_{1(\cdot,i)}|)\right)\right]\right\rangle_Z, \tag{4}$$

where $M_c = 0.5(\overline{M} + \underline{M})$ and $M_r = 0.5(\overline{M} - \underline{M})$.

3 PROBLEM STATEMENT

We consider linear time-invariant systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + p, \tag{5a}$$
$$y(t) = Cx(t) + Wv(t) + q, \tag{5b}$$

where $x(t) \in \mathbb{R}^n$ is the state, $y(t) \in \mathbb{R}^r$ is the output, $u(t) \in \mathbb{R}^m$ is the input, and $v(t) \in \mathbb{R}^o$ represents additional uncertainty on the output. Moreover, we have $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $p \in \mathbb{R}^n$, $C \in \mathbb{R}^{r\times n}$, $W \in \mathbb{R}^{r\times o}$, and $q \in \mathbb{R}^r$. The initial state $x(t_0)$, the input $u(t)$, and $v(t)$ are uncertain within the initial set $\mathcal{X}^0 \subset \mathbb{R}^n$, the input set $\mathcal{U} \subset \mathbb{R}^m$, and the output uncertainty set $\mathcal{V} \subset \mathbb{R}^o$, respectively. Using the geometric center c_u of the input set $\mathcal{U}$, let us define the

vector $\tilde{u} = Bc_u + p \in \mathbb{R}^n$ and the set $\mathcal{U}_0 = B(\mathcal{U} - c_u) \subset \mathbb{R}^n$ for later derivations. Note that if the geometric center of the set cannot be computed, one can use the center of the enclosing interval instead, which can be calculated using support function evaluations only. For a concise presentation, we will generally assume a constant vector $\tilde{u}$ and address the extension to time-varying inputs $\tilde{u}(t)$ where appropriate. Without loss of generality, we set the initial time to $t_0 = 0$ and define the time horizon as $[0, t_{\text{end}}]$, which is divided into an integer number of time intervals $\tau_k = [t_k, t_{k+1}]$ using the time step size Δt. The reachable set is defined as follows:

Definition 5. (Reachable set) For a given initial state $x(0)$ and an input trajectory $u(\cdot)$, let us denote the solution to (5a) at time t by $\xi(t, x(0), u(\cdot))$. The reachable set at time $t \geq 0$ for all initial states $x(0) \in \mathcal{X}^0$ and all input trajectories $u(\cdot) \in \mathcal{U}$ is

$$\mathcal{R}(t) := \left\{ \xi(t, x(0), u(\cdot)) \;\middle|\; x(0) \in \mathcal{X}^0, \, \forall \theta \in [0, t] : u(\theta) \in \mathcal{U} \right\}.$$

We write $\mathcal{R}(t_k)$ for the time-point reachable set at time $t = t_k$ and $\mathcal{R}(\tau_k)$ for the time-interval reachable set over $t \in \tau_k$. □

As mentioned in the introduction, the exact reachable set as defined above cannot be computed for general linear systems [28]. Hence, we aim to compute tight outer-approximations $\widehat{\mathcal{R}}(t) \supseteq \mathcal{R}(t)$ and inner-approximations $\check{\mathcal{R}}(t) \subseteq \mathcal{R}(t)$ instead, which extends to the output sets $\check{\mathcal{Y}}(t) \subseteq \mathcal{Y}(t) \subseteq \widehat{\mathcal{Y}}(t)$ obtained from a set-based evaluation of (5b).

Our goal in this work is to automatically prove or disprove safety of LTI systems using reachability analysis. We distinguish between two types of safety specifications:

(1) First, we examine the common case with a polytope $\mathcal{K} = \langle H, f \rangle_H$ as a safe set, which is equivalent to multiple halfspaces as unsafe sets.
(2) More generally, we consider an arbitrary number of convex unsafe sets $\mathcal{L}_1, ..., \mathcal{L}_b$.

Obviously, cases where both types of specifications occur are analyzed by combining the corresponding verification algorithms. The overall verification task is formally defined as follows:

Problem 1. *(Verification) Given an LTI system* (5) *with initial set* $\mathcal{X}^0 \subset \mathbb{R}^n$*, input set* $\mathcal{U} \subset \mathbb{R}^m$*, and output uncertainty set* $\mathcal{V} \subset \mathbb{R}^o$*, as well as a safe set* $\mathcal{K} = \langle H, f \rangle_H$ *and/or a number of convex unsafe sets* $\mathcal{L}_1, ..., \mathcal{L}_b$*, decide whether*

$$\forall t \in [0, t_{end}] : \mathcal{Y}(t) \subseteq \mathcal{K} \wedge \mathcal{Y}(t) \cap \bigcup_{i=1}^{b} \mathcal{L}_i = \emptyset,$$

that is, whether the output set $\mathcal{Y}(t)$ *stays within the safe set* $\mathcal{K}$ *and avoids the unsafe sets* $\mathcal{L}_1, ..., \mathcal{L}_b$ *at all times* $t \in [0, t_{end}]$. □

Please note that this problem formulation can be easily adapted to time-varying safe and/or unsafe sets that are only active for a fraction of the time horizon. Also, we do not explicitly consider floating-point errors for simplicity, but restrict our attention to approximation errors only.

4 REACHABILITY ANALYSIS

In this section, we present a self-contained overview of reachability analysis using support functions: In Sec. 4.1, we recall the computation of outer-approximations of reachable sets, which is similar to [4] but explicitly tailored towards computing upper bounds for the support function rather than an explicit representation of the reachable set in form of a template polyhedron. Next in Sec. 4.2, we derive a novel approach for computing lower bounds for the support function of the reachable set using a similar propagation scheme as for the upper bounds. This allows us to unify both computations into a single algorithm for computing upper and lower bounds for the support function of the reachable set in Sec. 4.3. For ease of presentation, we will focus on time-interval solutions as they are required for verification purposes; time-point solutions only serve as intermediate results.

4.1 Outer-approximations of reachable sets

The analytical solution for the linear differential equation (5a) is given as

$$x(t_k) = \underbrace{e^{At_k} x(0)}_{\in \mathcal{H}(t_k)} + \underbrace{\int_0^{t_k} e^{A(t_k - \theta)} \left(B\, u(\theta) + p \right) \mathrm{d}\theta}_{\in \mathcal{P}(t_k)},$$

which consists of the homogeneous solution $\mathcal{H}(t_k)$ resulting from the propagation of the initial state and the particular solution $\mathcal{P}(t_k)$ due to inputs, whose Minkowski addition yields the reachable set $\mathcal{R}(t_k) = \mathcal{H}(t_k) \oplus \mathcal{P}(t_k)$. We use the following wrapping-free propagation formula [27, Eq. (6)]:

$$\mathcal{P}(t_{k+1}) = \mathcal{P}(t_k) \oplus e^{At_k} \mathcal{P}(\Delta t), \tag{6}$$

$$\mathcal{R}(\tau_k) = e^{At_k} \mathcal{H}(\tau_0) \oplus \mathcal{P}(t_{k+1}). \tag{7}$$

For each time interval τ_k, we compute the homogeneous time-interval solution $\mathcal{H}(\tau_k) = e^{At_k} \mathcal{H}(\tau_0)$ and the particular time-interval solution $\mathcal{P}(\tau_k)$, where we exploit that $\mathcal{P}(\tau_k) \subseteq \mathcal{P}(t_{k+1})$ holds if $0 \in \mathcal{U}$ [1, Prop. 3.3]. To guarantee that the origin is contained in the input set, we first split the input set into two parts $\mathcal{U} = \tilde{u} \oplus \mathcal{U}_0$, such that $0 \in \mathcal{U}_0$, where the solution due to $\tilde{u}$ is integrated into the homogeneous solution $\mathcal{H}(\tau_0)$ and the solution due to $\mathcal{U}_0$ constitutes the particular solution $\mathcal{P}(\tau_k)$. The evaluation of (6)-(7) using support functions follows directly from (1)-(2):

$$\rho(\mathcal{P}(t_{k+1}), \ell) = \rho(\mathcal{P}(t_k), \ell) + \rho(\mathcal{P}(\Delta t), (e^{At_k})^\top \ell), \tag{8}$$

$$\rho(\mathcal{R}(\tau_k), \ell) = \rho(\mathcal{H}(\tau_0), (e^{At_k})^\top \ell) + \rho(\mathcal{P}(t_{k+1}), \ell). \tag{9}$$

The back-propagated direction $(e^{At_k})^\top \ell$ can be computed using a sequence of matrix-vector multiplications as $e^{At_k} = e^{A\Delta t} \cdot \ldots \cdot e^{A\Delta t}$.

In some cases, the auxiliary sets $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t)$ can be pre-computed, e.g., when $\mathcal{X}^0$ and $\mathcal{U}$ are zonotopes. This accelerates the computation immensely as one only has to evaluate the support function of the pre-computed sets $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t_k)$ in the direction $(e^{At_k})^\top \ell$ and add the resulting scalar values to compute (8)-(9). For zonotopes, this reduces the computational complexity from $O(n^3)$ for propagating entire zonotopes down to $O(n^2)$ [4]. If the pre-computation is undesirable because either the set representation for $\mathcal{X}^0$ and/or $\mathcal{U}$ is not closed under the required set operations or these operations are not defined for that set representation, one has to evaluate the support functions for $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t)$ based on the support functions for $\mathcal{X}^0$ and $\mathcal{U}$ in each step.

Let us now introduce the approximation models for computing outer-approximations of $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t)$ in order to evaluate (8)-(9). We use the correction hull approximation model [1, Sec. 3.2],

which represents a good trade-off between fast but inaccurate first-order models [30, Sec. 3], [40, Eq. (2)] and the accurate but slow forward-backward method [27, Sec. 3.1], [25, Sec. 2.4]. The affine time-interval solution is computed according to [1, Eq. (3.10)] by

$$\mathcal{H}(\tau_0) \subseteq \text{conv}\big(\mathcal{X}^0, e^{A\Delta t}\mathcal{X}^0 \oplus \mathcal{P}^u(\Delta t)\big) \oplus C, \tag{10}$$

which translates to the support function evaluation

$$\begin{aligned}\rho(\mathcal{H}(\tau_0), \ell) \leq \max\{&\rho(\mathcal{X}^0, \ell), \rho(\mathcal{X}^0, (e^{A\Delta t})^\top \ell) \\ &+ \rho(\mathcal{P}^u(\Delta t), \ell)\} + \rho(C, \ell).\end{aligned} \tag{11}$$

The outer-approximation in (10) is computed using the convex hull of the initial set $\mathcal{X}^0$ and its propagation $e^{A\Delta t}\mathcal{X}^0$, which is first shifted by the particular solution $\mathcal{P}^u(\Delta t)$ due to the constant input $\tilde{u}$ given as [1, Eq. (3.7)]

$$\mathcal{P}^u(\Delta t) = T\tilde{u}, \tag{12}$$

$$\text{where } T = A^{-1}(e^{A\Delta t} - I_n) \tag{13}$$

is the propagation matrix for constant inputs. Alternatively, the term A^{-1} can be integrated into the power series of the exponential matrix $e^{A\Delta t}$ in case A is singular. Finally, we enlarge the resulting set by the curvature enclosure

$$C = \mathcal{F}\mathcal{X}^0 \oplus \mathcal{G}\,\tilde{u}$$

using the interval matrices [1, Sec. 3.2]

$$\mathcal{F} = \bigoplus_{i=2}^{\eta} \left[\left(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}}\right)\Delta t^i, 0\right] \frac{A^i}{i!} \oplus \mathcal{E}, \tag{14}$$

$$\mathcal{G} = \bigoplus_{i=2}^{\eta+1} \left[\left(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}}\right)\Delta t^i, 0\right] \frac{A^{i-1}}{i!} \oplus \mathcal{E}\,\Delta t, \tag{15}$$

where the interval matrix $\mathcal{E}$ represents the remainder of the exponential matrix [1, Eq. (3.2)]:

$$\mathcal{E} = [-E, E], \quad E = e^{|A|\Delta t} - \sum_{i=0}^{\eta} \frac{\left(|A|\Delta t\right)^i}{i!}.$$

While for zonotopes the multiplication of the interval matrix $\mathcal{F}$ with $\mathcal{X}^0$ can be computed according to (4), it is unclear how to implement this set operation for general set convex set representations. In this case, we enclose $\mathcal{X}^0$ by an interval and represent it as a zonotope to compute its product with the interval matrix $\mathcal{F}$, leading to the support function evaluation

$$\rho(C, \ell) = \rho(\mathcal{F}\ \text{box}\big(\mathcal{X}^0\big), \ell) + \rho(\mathcal{G}\,\tilde{u}, \ell).$$

Since the interval matrix $\mathcal{F}$ is small for large enough values for η, the over-approximation induced by the enclosure $\text{box}\big(\mathcal{X}^0\big) \supseteq \mathcal{X}^0$ does not notably impact the tightness of the overall reachable set.

The particular solution due to the time-varying inputs within the set $\mathcal{U}_0$ can be enclosed by [1, Eq. (3.7)]

$$\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t) = \bigoplus_{i=0}^{\eta} \frac{A^i \Delta t^{i+1}}{(i+1)!}\,\mathcal{U}_0 \oplus \mathcal{E}\,\Delta t\,\mathcal{U}_0. \tag{16}$$

Again, in case (16) cannot be computed directly for the given set representation, we enclose the set $\mathcal{U}_0$ by a zonotope representing the box enclosure to evaluate the product with the interval matrix $\mathcal{E}$. From (1)-(2), we obtain the support function evaluation

$$\rho(\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t), \ell) = \sum_{i=0}^{\eta} \rho\left(\mathcal{U}_0, \left(\frac{A^i \Delta t^{i+1}}{(i+1)!}\right)^\top \ell\right) + \rho(\mathcal{E}\,\Delta t\,\text{box}\big(\mathcal{U}_0\big), \ell).$$

An explicit outer-approximation of the reachable set in form of a template polyhedron can be constructed by choosing a set of template directions $\ell_1 \dots \ell_w$:

$$\widehat{\mathcal{R}}(\tau_k) = \langle H, f\rangle_H$$

$$\text{with } H = [\ell_1 \dots \ell_w]^\top,\ f = [\rho(\widehat{\mathcal{R}}(\tau_k), \ell_1) \dots \rho(\widehat{\mathcal{R}}(\tau_k), \ell_w)]^\top.$$

The overall computation of upper bounds $\rho(\widehat{\mathcal{R}}(t), \ell)$ is summarized in Alg. 1 in combination with the computation of lower bounds presented subsequently.

4.2 Inner-approximations of reachable sets

We now present a novel approach for computing an explicit inner-approximation $\check{\mathcal{R}}(t_k)$ for the time-point reachable set and lower bounds $\rho(\check{\mathcal{R}}(\tau_k), \ell)$ for the support function of the time-interval reachable set. To this end, we replace the outer-approximations for the homogeneous solution $\widehat{\mathcal{H}}(\tau_0)$ (10) and the particular solution $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t)$ (16) by inner-approximations. For the homogeneous solution, we can exploit that the time-point solutions are enclosed by the time-interval solution to omit the curvature enclose, which yields the lower bound

$$\rho(\mathcal{H}(\tau_0), \ell) \geq \max\{\rho(\mathcal{X}^0, \ell), \rho(\mathcal{X}^0, (e^{A\Delta t})^\top \ell) + \rho(\mathcal{P}^u(\Delta t), \ell)\}.$$

Since constant inputs over time are a subset of time-varying inputs, the particular solution due to constant inputs represents an inner-approximation of the particular solution due to time-varying inputs. Moreover, we can use (13) to compute the analytical solution

$$\check{\mathcal{P}}^{\mathcal{U}}(\Delta t) = T\,\mathcal{U}_0 \tag{17}$$

for the particular solution due to constant inputs with T as in (13). For the computation of a lower bound for the support function of the time-interval reachable set $\rho(\check{\mathcal{R}}(\tau_k), \ell)$, we use the particular solution $\check{\mathcal{P}}^{\mathcal{U}}(t_k) \subseteq \mathcal{P}^{\mathcal{U}}(\tau_k)$, which represents an inner-approximation of the particular solution for the whole time interval.

An explicit inner-approximation of the time-point reachable set $\check{\mathcal{R}}(t_k)$ can be obtained from the support vectors $v(\check{\mathcal{R}}(t_k), \ell)$ [23], which can be exactly computed via simulation since we have piecewise constant inputs. To this end, we first convert the continuous-time system to the equivalent discrete-time system

$$x(k+1) = e^{A\Delta t}x(k) + Tu(k). \tag{18}$$

The initial state $x(0)$ is given by the support vector of the back-propagated initial set

$$x(0) = v(\mathcal{X}^0, (e^{At_k})^\top \ell) \tag{19}$$

Moreover, the input trajectory consisting of a sequence of piecewise constant inputs is given by the support vectors of the back-propagated input sets:

$$\forall j \in \{0, ..., k\} : u(j) = v(\mathcal{U}_0, (e^{At_j})^\top \ell) + \tilde{u}. \tag{20}$$

For each template direction $\ell_1, \dots, \ell_w$, we compute the initial state by (19), the input trajectory by (20), and evaluate the system (18) to obtain the set of points

$$\forall j \in \{1, \dots, w\} : x_j(k) = v(\check{\mathcal{R}}(t_k), \ell_j),$$

which represent the support vectors of the inner-approximation of the reachable set. Since the time-point solution $\check{\mathcal{R}}(t_k)$ is convex, the convex hull of the support vectors yields an inner-approximation:

$$\langle [v(\check{\mathcal{R}}(t_k), \ell_1), \dots, v(\check{\mathcal{R}}(t_k), \ell_w)]\rangle_V \subseteq \mathcal{R}(t_k).$$

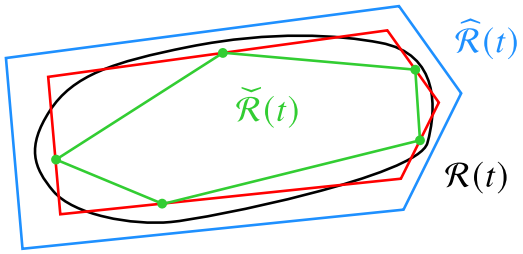


Figure 1: The halfspace polytope $\langle H, f \rangle_H$ (in red) with $H = [\ell_1 \dots \ell_w]^\top$, $f = [\rho(\check{\mathcal{R}}(t), \ell_1) \dots \rho(\check{\mathcal{R}}(t), \ell_w)]^\top$ constructed from lower bounds of the support function is in general *not* an inner-approximation of the exact reachable set $\mathcal{R}(t)$.

In contrast, the time-interval solution $\mathcal{R}(\tau_k)$ is in general non-convex, so that we cannot obtain an explicit inner-approximation analogously to the time-point solution. Moreover, note that a halfspace polytope constructed from the inner-approximations of the support function is in general not an inner-approximation of the reachable set, as illustrated in Fig. 1. The support vectors for the outer-approximation of the reachable set can be computed in the same way as above.

4.3 Reachability algorithm

The overall reachability algorithm computing upper and lower bounds as presented in Sec. 4.1 and Sec. 4.2, respectively, is summarized in Alg. 1: After some instantiations before the main loop, both algorithms share the back-propagation of the direction (Line 8) and the propagations of $\mathcal{P}^u(t_{k+1})$ (Line 9) and $\mathcal{H}(t_{k+1})$ (Line 10). In the same loop, we compute upper bounds $\rho(\widehat{\mathcal{R}}(t), \ell)$ (Lines 11-13) and lower bounds $\rho(\check{\mathcal{R}}(t), \ell)$ (Lines 14-16) of the time-point and time-interval reachable sets in a user-defined direction ℓ. We reiterate that the propagation scales with $O(n^2)$ if the sets $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t_k)$ can be pre-computed, e.g. when using zonotopes, following (1)-(3). Note that one can easily parallelize Alg. 1 for multiple directions of interest $\ell_1, \dots, \ell_w$. For an extension to a time-varying input vector $\tilde{u}(t)$, one simply has to re-compute the particular solution $\mathcal{P}^u(\Delta t)$ (Line 9) and the term $\mathcal{G}\tilde{u}$ to update the curvature enclosure C in Line 13 in each step. If an output equation (5b) is given, we compute the support function of the output set as

$$\rho(\mathcal{Y}(t), \ell) = \rho(\mathcal{R}(t), C^\top \ell) + \rho(\mathcal{V}, W^\top \ell) + \ell^\top q, \tag{21}$$

which induces no additional approximation error. To increase efficiency, one can also pre-process any direction by $\ell \to C^\top \ell$ and pre-compute the second and third term in (21) unless $v(t)$ varies over time. The computed upper and lower bounds become arbitrarily tight for $\Delta t \to 0$ [40, Sec. 3], which we exploit in our automated verification algorithms presented in the next section.

5 VERIFICATION

In this section, we introduce our fully-automated verification algorithms. The first algorithm is tailored to the common case where the unsafe sets are given as halfspaces; the second algorithm is capable of verifying arbitrary convex unsafe sets. Both algorithms also check for falsification and return a counterexample in case of a safety violation. For ease of presentation, we implicitly assume the specifications to be defined in the state space as the algorithms can readily be extended to specifications defined on the outputs.

Algorithm 1 Reachability analysis using support functions

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0$, input set $\mathcal{U}$, direction ℓ, time horizon t_{end}, time step size Δt, truncation order η

Ensure: Sequence of upper and lower bounds for the time-point solutions $\rho(\widehat{\mathcal{R}}(t_k), \ell), \rho(\check{\mathcal{R}}(t_k), \ell)$ and time-interval solutions $\rho(\widehat{\mathcal{R}}(\tau_k), \ell), \rho(\check{\mathcal{R}}(\tau_k), \ell)$ in direction ℓ

1: $t_0 \leftarrow 0$, $d_0 \leftarrow \ell$, $\rho(\mathcal{H}(t_0), \ell) \leftarrow \rho(\mathcal{X}^0, \ell)$
2: $\tilde{u} \leftarrow B\,\text{center}(\mathcal{U}) + p$, $\mathcal{U}_0 \leftarrow B(\mathcal{U} - \text{center}(\mathcal{U}))$
3: $\rho(\mathcal{P}^u(t_0), \ell) \leftarrow 0$, $\rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_0), \ell) \leftarrow 0$, $\rho(\check{\mathcal{P}}^{\mathcal{U}}(t_0), \ell) \leftarrow 0$
4: $\mathcal{P}^u(\Delta t) \leftarrow$ Eq. (12), $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t) \leftarrow$ Eq. (16), $\check{\mathcal{P}}^{\mathcal{U}}(\Delta t) \leftarrow$ Eq. (17)
5: $C \leftarrow \mathcal{F}\mathcal{X}^0 \oplus \mathcal{G}\tilde{u}$ ▷ see Eq. (14)-(15)
6: **for** $k \leftarrow 0$ to $\frac{t_{\text{end}}}{\Delta t} - 1$ **do**
7: $t_{k+1} \leftarrow t_k + \Delta t$, $\tau_k \leftarrow [t_k, t_{k+1}]$
8: $d_{k+1} \leftarrow (e^{A\Delta t})^\top d_k$
9: $\mathcal{P}^u(t_{k+1}) \leftarrow e^{A\Delta t}\mathcal{P}^u(t_k) + \mathcal{P}^u(\Delta t)$
10: $\rho(\mathcal{H}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{X}^0, d_{k+1}) + \rho(\mathcal{P}^u(t_{k+1}), \ell)$
Upper bounds:
11: $\rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell) \leftarrow \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_k), \ell) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t), d_k)$
12: $\rho(\widehat{\mathcal{R}}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{H}(t_{k+1}), \ell) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
13: $\rho(\widehat{\mathcal{R}}(\tau_k), \ell) \leftarrow \max\{\rho(\mathcal{H}(t_k), \ell), \rho(\mathcal{H}(t_{k+1}), \ell)\}$
$+\rho(C, d_k) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
Lower bounds:
14: $\rho(\check{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell) \leftarrow \rho(\check{\mathcal{P}}^{\mathcal{U}}(t_k), \ell) + \rho(\check{\mathcal{P}}^{\mathcal{U}}(\Delta t), d_k)$
15: $\rho(\check{\mathcal{R}}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{H}(t_{k+1}), \ell) + \rho(\check{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
16: $\rho(\check{\mathcal{R}}(\tau_k), \ell) \leftarrow \max\{\rho(\mathcal{H}(t_k), \ell), \rho(\mathcal{H}(t_{k+1}), \ell)\},$
$+\rho(\check{\mathcal{P}}^{\mathcal{U}}(t_k), \ell)$
17: **end for**

5.1 Verifying halfspace specifications

It is well known that safety specifications given as halfspaces and reachability analysis using support functions ideally complement each other: By choosing the directions for the support function evaluation as the normal vectors of the halfspaces, one can efficiently decide whether a given specification is violated. We propose an automated verification algorithm, which refines the upper and lower bounds computed by Alg. 1 automatically by adequately tuning the corresponding algorithm parameters until either the resulting outer-approximation is fully located outside of the unsafe region or the inner-approximation intersects that unsafe region. The accuracy of the reachability algorithm in Sec. 4.3 depends on two parameters, the truncation order η and the time step size Δt. Since it holds that for arbitrary values $\eta > 1$ the computed upper and lower bounds converge to the exact value for $\Delta t \to 0$ [40, Sec. 3], we first determine a suitable value for η before tuning the time step size. Hence, the truncation order η is tuned as in [55, Sec. 5.1] by using the partial sums

$$\mathcal{T}^{(j)} = \bigoplus_{i=2}^{j} \left[\left(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}} \right) \Delta t^i, 0 \right] \frac{A^i}{i!}$$

Algorithm 2 Verification algorithm (halfspace specifications)

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0$, input set $\mathcal{U}$, time horizon t_{end}, safe set $\mathcal{K} = \langle H, f \rangle_H$
Ensure: Safe ($\forall t \in [0, t_{\text{end}}] : \mathcal{R}(t) \subseteq \mathcal{K}$) or unsafe ($\exists t \in [0, t_{\text{end}}] : \mathcal{R}(t) \not\subseteq \mathcal{K}$)

1: **for** $i \leftarrow 1$ to w **do**
2: $\quad \ell_i \leftarrow H_{(i,\cdot)}^\top, \Delta t \leftarrow t_{\text{end}}$
3: $\quad$ **repeat**
4: $\quad\quad$ `verified` $\leftarrow$ `true`
5: $\quad\quad$ **for** $k \leftarrow 0$ to $\frac{t_{\text{end}}}{\Delta t} - 1$ **do**
6: $\quad\quad\quad \rho(\widehat{\mathcal{R}}(\tau_k), \ell_i), \rho(\check{\mathcal{R}}(\tau_k), \ell_i) \leftarrow$ Alg. 1 with Δt
7: $\quad\quad\quad$ **if** $\rho(\widehat{\mathcal{R}}(\tau_k), \ell_i) > f_{(i)}$ **then**
8: $\quad\quad\quad\quad$ `verified` $\leftarrow$ `false`
9: $\quad\quad\quad$ **else if** $\rho(\check{\mathcal{R}}(\tau_k), \ell_i) > f_{(i)}$ **then**
10: $\quad\quad\quad\quad$ **return** unsafe
11: $\quad\quad\quad$ **end if**
12: $\quad\quad$ **end for**
13: $\quad\quad \Delta t \leftarrow 0.5\Delta t$
14: $\quad$ **until** `verified` = `true`
15: **end for**
16: **return** safe

from the computation of $\mathcal{F}$ in (14). We increase η until the relative change between $\mathcal{T}^{(j)}$ and $\mathcal{T}^{(j+1)}$ in the Frobenius norm computed according to [21, Thm. 10] is smaller than 10^{-10}. Since the size of additional terms $\mathcal{T}^{(j)}$ goes to 0 for $\eta \to \infty$ and our bound is relative, we will always find a value for η. The time step size is tuned by halving the last value. This simple tuning strategy is justified by the fact that the reachability algorithm in Sec. 4.3 is so efficient that the computation time of a more sophisticated tuning method for Δt may exceed the computation time for reachability analysis.

Alg. 2 summarizes our verification procedure for a specification represented by the safe set $\mathcal{K} = \langle H, f \rangle_H$, which is equivalent to multiple unsafe sets represented as halfspaces: For each direction $\ell_i = H_{(i,\cdot)}^\top, \forall i \in \{1, \ldots, w\}$, we compute upper bounds for the support function of the reachable set (Line 6) and check whether the corresponding outer-approximation remains within $\mathcal{K}$ at all times (Line 7). At the same time, we compute lower bounds (Line 6) and check whether the corresponding inner-approximation leaves $\mathcal{K}$ in any step (Line 9), which would immediately falsify the specification. If the specification can neither be proven nor disproven by the current results, we re-compute the upper and lower bounds using a smaller time step size (Line 13). As both bounds converge to the support function for the exact reachable set, Alg. 2 is always able to verify or falsify decidable safety specifications in finite time.

In case of a safety violation (Line 10), we return a counterexample: The support vector $v(\check{\mathcal{R}}(t_k), \ell_i)$ for the inner-approximation $\check{\mathcal{R}}(t_k)$ that violates the safety specification is located inside an unsafe set, and thus corresponds to a falsifying trajectory. Hence, we use (19)-(20) to obtain the initial state $x(0)$ and a piecewise constant input trajectory $u(t)$, and then evaluate (18) to compute the support vector $v(\check{\mathcal{R}}(t_k), \ell_i)$, which represents the counterexample.

Substantial runtime improvements can be achieved as follows: Instead of computing the upper and lower bounds for the entire entire time horizon beforehand and checking for safety violation afterward, we already perform the checks during the computation of the reachable set. Moreover, previously verified time intervals do not have to be checked in future iterations. Due to the fixed time step size Δt in each iteration of the main loop, one can pre-compute all directions d_k (Line 8 of Alg. 1). In some cases, this allows one to reformulate the iterative vector-matrix multiplications in the support function evaluation of the sets $\mathcal{P}(\Delta t)$ (Line 11 of Alg. 1) and C (Line 13 of Alg. 1) into a more efficient matrix-matrix multiplication. Lastly, the main loop (Lines 1-15) can be parallelized. For our evaluation in Sec. 6, we initialize the time step size by a hundreth of the time horizon and quarter the time step size in Line 13, which is a heuristic that we observed to work well in practice.

5.2 Verifying arbitrary convex unsafe sets

The Gilbert-Johnson-Keerthi (GJK) algorithm [29] offers an elegant way to check if two convex sets intersect using only their support function evaluation. Consequently, we can utilize this algorithm to extend the verification algorithm from Sec. 5.1 to arbitrary convex unsafe sets. A similar idea was previously presented in [24], where the GJK algorithm is used in combination with support function reachable set computation to eliminate spurious transitions in hybrid system reachability.

Let us briefly recall the GJK algorithm: It is based on the fact that checking if two convex sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ intersect is equivalent to checking if the origin is contained within the set $\mathcal{S} = \mathcal{S}_1 \oplus (-\mathcal{S}_2)$:

$$\big(\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset\big) \Leftrightarrow \big(\mathbf{0} \in \mathcal{S}_1 \oplus (-\mathcal{S}_2)\big).$$

If there exists any direction ℓ for which $\rho(\mathcal{S}, \ell) < 0$, the set $\mathcal{S}$ does not contain the origin, and thus the two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ do not intersect. In contrast, if the polytope $\mathcal{P} = \langle [v(\mathcal{S}, \ell_1) \ldots v(\mathcal{S}, \ell_i)] \rangle_V \subseteq \mathcal{S}$ constructed from the support vectors along multiple directions does contain the origin, the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ intersect. We can replace an explicit computation of $\mathcal{S}$ by computing its support function and corresponding support vectors using (2):

$$\rho(\mathcal{S}, \ell) = \rho(\mathcal{S}_1, \ell) + \rho(\mathcal{S}_2, -\ell), \quad v(\mathcal{S}, \ell) = v(\mathcal{S}_1, \ell) + v(\mathcal{S}_2, -\ell).$$

The GJK algorithm selects new directions ℓ_i until either $\rho(\mathcal{S}, \ell_i) < 0$ or $\mathbf{0} \in \mathcal{P}$ holds, where the next direction is the normal vector of the polytope facet from $\mathcal{P}$ that is closest to the origin. Determining this facet is in general computationally demanding, since the number of polytope facets can be exponential in the number of polytope vertices. To avoid this issue, the GJK algorithm uses simplices, which are polytopes with exactly $n + 1$ vertices and consequently also only $n + 1$ facets. Thus, the algorithm constructs in each iteration a new simplex from the polytope facet closest to the origin and the support vector in the chosen new direction, and disregards all remaining polytope vertices. This procedure is visualized in Fig. 2.

In the verification setting, the set $\mathcal{S}_1$ is the reachable set $\mathcal{R}(t)$ and the set $\mathcal{S}_2$ is a convex unsafe set $\mathcal{L}$, so that $\mathcal{S} = \mathcal{R}(t) \oplus (-\mathcal{L})$. Our verification algorithm is summarized in Alg. 3: In each iteration we first compute upper and lower bounds of the support function in the current direction ℓ_i (Line 3), based on which we try to disprove that the sets intersect by checking if $\rho(\mathcal{S}, \ell_i) \leq \rho(\widehat{\mathcal{R}}(t), \ell_i) + \rho(\mathcal{L}, -\ell_i) < 0$ holds (Line 4). If true, we found a separating hyperplane between

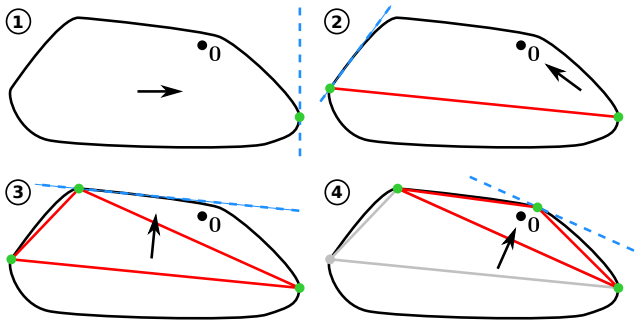


Figure 2: Schematic visualization of the iterations of the GJK algorithm for a set $\mathcal{S} = \mathcal{S}_1 \oplus (-\mathcal{S}_2)$.

$\mathcal{R}(t)$ and $\mathcal{L}$, which proves that the system is safe. Otherwise, we use the corresponding support vector of the lower bound to check if we can prove that the sets intersect (Line 12). In case safety can neither be proven nor disproven based on the current direction ℓ_i, we choose a new direction $\ell_{i+1} = h$. If the polytope $\check{\mathcal{P}}$ constructed from the support vectors is already non-degenerate ($i > n+1$, Line 18), the new direction is chosen as the normal vector of the polytope facet that is closest to the origin (Line 19), which can be determined by solving the quadratic program $\min_{x \in \check{\mathcal{P}}} \|x\|_2^2$ to obtain the point in $\check{\mathcal{P}}$ closest to the origin. Otherwise, we apply Gram-Schmidt orthogonalization to construct a direction orthogonal to the space spanned by the support vectors and pointing towards the origin (Line 23).

A major challenge is that we cannot compute the exact value of the support function for the reachable set, but only tight upper and lower bounds. Hence, we have to decide when to stop generating new directions ℓ_i and instead refine the tightness of the approximations. If $\rho(\check{\mathcal{R}}(t), \ell_i) + \rho(\mathcal{L}, -\ell_i) < 0$ holds for any direction ℓ_i, we cannot prove that an intersection occurs using the current accuracy of the lower bound (Line 7), but we may still be able to prove that no intersection occurs using the upper bound. Only if the polytope $\hat{\mathcal{P}}$ constructed from the support vectors of the upper bound contains the origin, we definitely cannot disprove the intersection using the current accuracy, and thus reduce the time step size (Line 15). The truncation order η is tuned as described in Sec. 5.1. Finally, since the computed upper and lower bounds converge to the exact value for $\Delta t \to 0$ and an increasing number of directions ℓ_i, it is guaranteed that Alg. 3 is always able to either prove or disprove safety in decidable cases.

For simplicity, Alg. 3 only considers the intersection between the reachable set $\mathcal{R}(t)$ at a specific point in time and a single unsafe set $\mathcal{L}$. To extend this to checking multiple unsafe sets $\mathcal{L}_1, ..., \mathcal{L}_b$ for all times $t \in [0, t_{\text{end}}]$ as required by Problem 1, we can simply run Alg. 3 for all possible pairs of time-interval reachable sets $\mathcal{R}(\tau_k)$ and unsafe sets $\mathcal{L}_j$, where we omit certain pairs if the unsafe sets are time-varying. Note that we still use the time-point reachable set $\check{\mathcal{R}}(t_k) \subseteq \check{\mathcal{R}}(\tau_k) \subseteq \mathcal{R}(\tau_k)$ for the inner-approximation since we require that all sets are convex, which is not the case for $\check{\mathcal{R}}(\tau_k)$.

Improvements to accelerate the verification include parallelizing the computations for the different $(\mathcal{R}(\tau_k), \mathcal{L}_j)$-pairs, re-using the support functions computed for the current pair to disprove intersections for other pairs, initializing the first search direction based on the distance between the unsafe set and a simulated trajectory, and selecting the new search direction based on the polytope $\hat{\mathcal{P}}$ instead of $\check{\mathcal{P}}$ once we know that the inner-approximation cannot be used to disprove safety (`falsifiable = false` in Alg. 3).

Algorithm 3 Verification algorithm (arbitrary convex unsafe sets)

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0$, input set $\mathcal{U}$, time t, unsafe set $\mathcal{L}$
Ensure: Safe ($\mathcal{R}(t) \cap \mathcal{L} = \emptyset$) or unsafe ($\mathcal{R}(t) \cap \mathcal{L} \neq \emptyset$)

1: $i \leftarrow 1$, $\ell_1 \leftarrow I_{n(\cdot,1)}$, $\Delta t \leftarrow t$, $\hat{\mathcal{P}}, \check{\mathcal{P}} \leftarrow \emptyset$, `falsifiable` $\leftarrow$ `true`
2: **repeat**
3: $\quad \rho(\hat{\mathcal{R}}(t), \ell_i), \rho(\check{\mathcal{R}}(t), \ell_i) \leftarrow$ Alg. 1 with Δt
4: $\quad$ **if** $\rho(\hat{\mathcal{R}}(t), \ell_i) + \rho(\mathcal{L}, -\ell_i) < 0$ **then**
5: $\quad\quad$ **return** safe
6: $\quad$ **end if**
7: $\quad$ **if** $\rho(\check{\mathcal{R}}(t), \ell_i) + \rho(\mathcal{L}, -\ell_i) < 0$ **then**
8: $\quad\quad$ `falsifiable` $\leftarrow$ `false`
9: $\quad$ **end if**
10: $\quad \hat{v}_i \leftarrow v(\hat{\mathcal{R}}(t), \ell_i) - v(\mathcal{L}, -\ell_i)$, $\check{v}_i \leftarrow v(\check{\mathcal{R}}(t), \ell_i) - v(\mathcal{L}, -\ell_i)$
11: $\quad \hat{\mathcal{P}} \leftarrow \text{conv}(\hat{\mathcal{P}}, \hat{v}_i)$, $\check{\mathcal{P}} \leftarrow \text{conv}(\check{\mathcal{P}}, \check{v}_i)$
12: $\quad$ **if** $\mathbf{0} \in \check{\mathcal{P}}$ **then**
13: $\quad\quad$ **return** unsafe
14: $\quad$ **else if** `falsifiable` = `false` $\wedge\ \mathbf{0} \in \hat{\mathcal{P}}$ **then**
15: $\quad\quad \Delta t \leftarrow 0.5\,\Delta t$, $i \leftarrow 1$, $\hat{\mathcal{P}}, \check{\mathcal{P}} \leftarrow \emptyset$, `falsifiable` $\leftarrow$ `true`
16: $\quad\quad$ **continue**
17: $\quad$ **end if**
18: $\quad$ **if** $i > n+1$ **then**
19: $\quad\quad \mathcal{H} = \langle h, f \rangle_H \leftarrow$ halfspace for facet of $\check{\mathcal{P}}$ closest to $\mathbf{0}$
20: $\quad\quad o_1, \ldots, o_n \leftarrow$ indices of the vertices that lie on facet $\mathcal{H}$
21: $\quad\quad \hat{\mathcal{P}} \leftarrow \langle [\hat{v}_{o_1} \ldots \hat{v}_{o_n}] \rangle_V$, $\check{\mathcal{P}} \leftarrow \langle [\check{v}_{o_1} \ldots \check{v}_{o_n}] \rangle_V$
22: $\quad$ **else**
23: $\quad\quad h \leftarrow$ vector orthogonal to $\check{v}_1, \ldots, \check{v}_i$ using Gram-Schmidt
24: $\quad$ **end if**
25: $\quad \ell_{i+1} \leftarrow h$, $i \leftarrow i+1$

Finally, we show how to construct a counterexample: If the system is unsafe, the simplex $\check{\mathcal{P}} = \langle [v_1 \ldots v_{n+1}] \rangle_V$ in Alg. 3 contains the origin. For each vertex of $\check{\mathcal{P}}$, we determine the weighting factor λ_j from the linear equation system $\sum_{j=1}^{n+1} \lambda_j v_j = \mathbf{0}$, $\sum_{j=1}^{n+1} \lambda_j = 1$, and use them to obtain the initial state $x(0)$ and input trajectory $u(k)$ by interpolation, i.e.,

$$x(0) = \sum_{j=1}^{n+1} \lambda_j x_j(0), \quad u(k) = \sum_{j=1}^{n+1} \lambda_j u_j(k),$$

between the values $x_j(0)$ and $u_j(k)$ for the trajectories resulting in the vertices of $\check{\mathcal{P}}$.

6 NUMERICAL EXAMPLES

We now demonstrate the performance of our verification algorithms on several challenging benchmarks. Our algorithms are implemented in the MATLAB toolbox CORA [2] and a repeatability package is publicly available[1]. All computations are performed on a 2.59GHz quad-core i7 processor with 32GB memory.

[1] https://codeocean.com/capsule/1155014/tree/v2

Table 1: Comparison of computation times for the ARCH benchmarks, where n is the system dimension, m is the number of inputs, r is the output dimension, and w is the number of halfspace specifications. For our approach we additionally specify the number of refinement iterations of Alg. 2. The computation times of the other tools are taken from [3].

Benchmark						Our approach		Time comparison			
Identifier	n	m	r	w	Safe?	Time	Refinements	CORA [2]	HyDRA [51]	JuliaReach [11]	SpaceEx [27]
HEAT01	125	0	1	1	✓	0.17s	2	2.2s	13.2s	**0.13**s	4.2s
HEAT02	1000	0	1	1	✓	**2.2**s	2	9.3s	160s	32s	—
CBC01	201	0	1	1	✓	**0.11**s	2	7.1s	—	1.4s	312.78s
CBC02	1001	0	1	1	✓	**2.2**s	2	—	—	—	—
CBC03	2001	0	1	1	✓	**28**s	3	—	—	—	—
CBF01	200	1	1	1	✓	**0.27**s	2	30s	—	12s	318.88s
CBF02	1000	1	1	1	✓	**3.7**s	2	—	—	—	—
CBF03	2000	1	1	1	✓	**49**s	3	—	—	—	—
BLDC01-BDS01	49	0	1	2	✓	0.07s	2	2.9s	0.426s	**0.0096**s	1.6s
BLDF01-BDS01	48	1	1	2	✓	0.09s	2	3.3s	—	**0.012**s	1.8s
ISSC01-ISS02	273	0	3	1	✓	**0.11**s	1	1.3s	—	1.4s	29s
ISSC01-ISU02	273	0	3	1	✗	0.16s	2	**0.072**s	—	1.4s	29s
ISSF01-ISS01	270	3	3	1	✓	**0.49**s	2	59s	—	10s	49s
ISSF01-ISU01	270	3	3	1	✗	**0.16**s	1	38s	—	10s	48s

6.1 ARCH benchmarks

In the annual ARCH competition, state-of-the-art reachability tools compete to efficiently verify a set of benchmarks. We applied our verification algorithm to all LTI systems from the 2021 edition [3]: The HEAT benchmarks spatially discretize the partial differential heat equation with varying mesh size, the CB benchmarks monitor oscillations along a clamped beam, the BLD benchmarks describe spatial and rotational movement of individual stories of a hospital building, and the ISS benchmarks represent structural models of a service module of the International Space Station.

Since the specifications of all benchmarks are defined by halfspaces, we used the verification algorithm from Sec. 5.1. Table 1 shows that all specifications are correctly verified or falsified using at most three time step size refinements. The computation time is fastest in most cases and often even orders of magnitude faster than for other tools, even though their results are expert-tuned and our algorithm works fully automatically. Additionally, we were able to solve higher-dimensional variations of the CB benchmark where other tools failed, and our algorithms provide a falsifying trajectory as shown in Fig. 3. Part of the reason is that the propagation scales with $O(n^2)$ as discussed in Sec. 4.3. In summary, this shows a clear superiority of Alg. 2 over state-of-the-art reachability tools. Moreover, even non-experts are able to efficiently solve challenging verification tasks since our algorithm is fully automated.

6.2 Frequency and voltage control

To demonstrate the benefit of our verification algorithms for real-world applications, we consider the highly relevant use case of power system frequency and voltage control. In particular, we examine the verification task described in [42], where the goal is to show that a given PI controller keeps the grid frequency and voltage of a power systems stable within a certain margin. Here, we analyze various IEEE busses, namely the 9-bus, 14-bus, and 33-bus systems with state dimensions $n = 28$, $n = 46$, and $n = 133$,

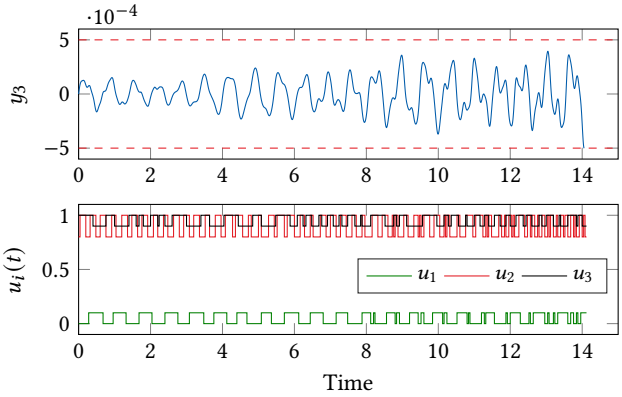


Figure 3: Specifications for ISSF01-ISU01 in red (dashed) and falsifying trajectory in blue (top); input trajectory $u(t) \in \mathbb{R}^3$ generating the falsifying trajectory (bottom).

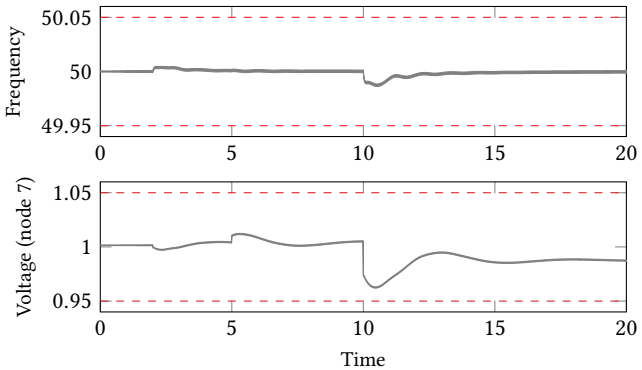


Figure 4: Reachable set for the 9-bus, where the bounds defined by the specification are depicted by the dashed red lines.

respectively, for the closed-loop dynamics. Additionally, the load change is represented by a time-varying constant input vector $\tilde{u}(t)$. For all systems, the frequency has to remain within 50 ± 0.05Hz. In addition, for the 9-bus the voltage drops of four generators have to stay within a margin of ±0.05V of the individual reference values.

In [42], the authors computed explicit reachable sets using the approach from [1, Alg. 3], which requires 15.1s, 15.5s, and 27.9s for the different bus systems (on our machine). Our approach yields significant speed-ups, since Alg. 2 successfully verifies all specifications in 0.49s, 0.14s, and 1.3s, respectively. Fig. 4 visualizes the resulting reachable set enclosure for the 9-bus. In summary, our verification improves the previous results by an order of magnitude, which enables real-time execution of verification during runtime.

6.3 Quadcopter example

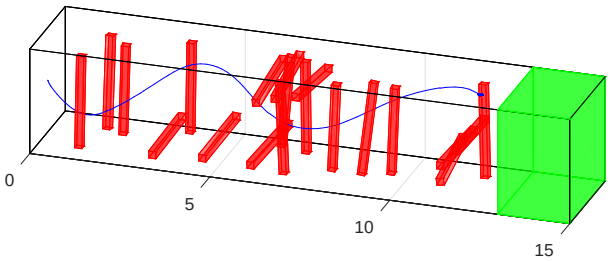
Finally, we demonstrate that our approach can efficiently verify linear systems even in the presence of complex time-varying obstacles. To this end, we consider a quadcopter, where the task is to verify that a planned reference trajectory $x_{\text{ref}}(t)$ tracked by a feedback controller is robustly safe despite disturbances and measurement errors. We describe the dynamics of the quadcopter with a linear point-mass model, which yields the closed-loop system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{\text{ref}}(t) \end{bmatrix} = \begin{bmatrix} A+BK & -BK \\ \mathbf{0} & A \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\text{ref}}(t) \end{bmatrix} + \begin{bmatrix} B & B & BK \\ B & \mathbf{0} & \mathbf{0} \end{bmatrix} u(t) + \begin{bmatrix} p \\ p \end{bmatrix}$$
$$y(t) = \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\text{ref}}(t) \end{bmatrix} + v(t)$$

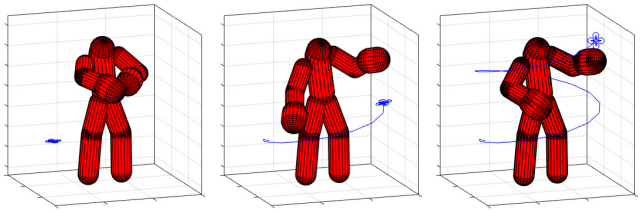
with $A = [\mathbf{0}\ [I_3\ \mathbf{0}]^\top]$, $B = [\mathbf{0}\ I_3]^\top$, $C = [I_3\ \mathbf{0}]$, $p = -9.81 \cdot \mathbf{1} \in \mathbb{R}^6$. The feedback matrix $K \in \mathbb{R}^{3\times 6}$ is determined by applying an LQR control approach with state weighting matrix $Q = I_6$ and input weighting matrix $R = 0.1 \cdot I_3$ to the open-loop system. Given an initial state $x(0) \in \mathbb{R}^6$, a piecewise constant reference input $u_{\text{ref}}(t)$, the set of process noise $\mathcal{W} = 0.1 \cdot [-\mathbf{1}, \mathbf{1}] \subseteq \mathbb{R}^3$, and the set of measurement errors $\mathcal{D} = 0.01 \cdot [-\mathbf{1}, \mathbf{1}] \subseteq \mathbb{R}^6$, the initial set is $\mathcal{X}^0 = (x(0) \oplus \mathcal{D}) \times x(0)$ and the set of uncertain inputs is $\mathcal{U} = u_{\text{ref}}(t) \times \mathcal{W} \times \mathcal{D}$. The output $y(t) \in \mathbb{R}^3$ represents the space occupied by the quadcopter, where the set $\mathcal{V} = \langle \mathbf{0}, 0.07^2 \cdot I_3 \rangle_E$ accounts for the spatial dimension of the quadcopter.

As a first verification task, we consider that the quadcopter has to reach the goal set at the end of a 15m long tunnel filled with static obstacles. For this scenario the initial state is $x(0) = \mathbf{0} \in \mathbb{R}^6$, the reference input $u_{\text{ref}}(t)$ consists of 100 constant pieces, and the final time is $t_{\text{end}} = 10$s. For the setup described above, our approach successfully verifies the given trajectory in only 0.51s. Next, we increase the process noise to $\mathcal{W} = 0.2 \cdot [-\mathbf{1}, \mathbf{1}] \subseteq \mathbb{R}^3$ to obtain an unsafe scenario. Here, our automated verifier disproves safety in only 2.9s and returns the falsifying trajectory visualized in Fig. 5a.

For the second verification task, the quadcopter has to avoid collisions with a human, whose occupancy space is predicted by the tool SaRA [49]. The resulting time-varying obstacles are represented as capsules, the initial state is $x(0) = \mathbf{0} \in \mathbb{R}^6$, the reference input $u_{\text{ref}}(t)$ consists of 30 constant pieces, and the final time is $t_{\text{end}} = 3$s. For the setup with process noise $\mathcal{W} = 0.1 \cdot [-\mathbf{1}, \mathbf{1}] \subseteq \mathbb{R}^3$ the reference trajectory is safe, which our algorithm sucessfully verifies in 0.94s. By increasing the process noise to $\mathcal{W} = 0.2 \cdot [-\mathbf{1}, \mathbf{1}] \subseteq \mathbb{R}^3$ we obtain an unsafe scenario, for which our verifier disproves safety in 0.64s and returns the falsifying trajectory depicted in Fig. 5b. This



(a) Static obstacles (red), goal set (green), and falsifying trajectory (blue).



(b) Time-varying obstacles (red) and falsifying trajectory (blue) at $t = \{0, 1, 2.3\}$s.

Figure 5: Quadcoptor verification task.

demonstrates that our automated verifier can solve highly complex verification tasks with arbitrary convex time-varying obstacles very efficiently, even though both considered tasks represent edge cases with a narrow margin between safe and unsafe.

7 CONCLUSION

We address the verification of safety specifications for linear time-invariant systems. Our proposed algorithm based on reachability analysis with support functions operates fully automatically and refines the tightness of the upper and lower bounds until safety can either be proven or disproven; in the latter case, we additionally return an initial state and an input trajectory yielding a counterexample. For the common case with unsafe sets defined as halfspaces, reducing the complexity of the propagation to quadratic with respect to the state dimension results in significant runtime improvements compared to state-of-the-art reachability tools, which enables us to verify previously unanalyzable high-dimensional benchmarks. Furthermore, an extension to verify arbitrary convex unsafe sets without major computational overhead has been demonstrated on a complex safety-critical scenario, where a quadrotor aims to avoid a human moving in close proximity. The low computation time highlights the real-time capability of our approach.

ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial supports from the research training group ConVeY funded by the German Research Foundation under grant GRK 2428 and the project justIT-SELF funded by the European Research Council (ERC) under grant agreement No 817629. This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156. 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 or the United States Navy.

REFERENCES

[1] M. Althoff. 2010. *Reachability analysis and its application to the safety assessment of autonomous cars.* Dissertation. Technische Universität München.

[2] M. Althoff. 2015. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*. 120–151. https://doi.org/10.29007/zbkv

[3] M. Althoff, E. Ábrahám, M. Forets, G. Frehse, D. Freire, C. Schilling, S. Schupp, and M. Wetzlinger. 2021. ARCH-COMP21 category report: continuous and hybrid systems with linear continuous dynamics. In *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*. 1–31. https://doi.org/10.29007/lhbw

[4] M. Althoff and G. Frehse. 2016. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *Proc. of the 55th IEEE Conference on Decision and Control*. 7439–7446. https://doi.org/10.1109/CDC.2016.7799418

[5] M. Althoff, G. Frehse, and A. Girard. 2021. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems* 4, 1 (2021), 369–395. https://doi.org/10.1146/annurev-control-071420-081941

[6] M. Althoff and B. H. Krogh. 2011. Zonotope bundles for the efficient computation of reachable sets. In *Proc. of the 50th IEEE Conference on Decision and Control*. 6814–6821. https://doi.org/10.1109/CDC.2011.6160872

[7] M. Althoff, O. Stursberg, and M. Buss. 2007. Reachability analysis of linear systems with uncertain parameters and inputs. In *Proc. of the 46th IEEE Conference on Decision and Control*. 726–732. https://doi.org/10.1109/CDC.2007.4434084

[8] E. Asarin, O. Bournez, T. Dang, and O. Maler. 2000. Approximate reachability analysis of piecewise-linear dynamical systems. In *3rd International Workshop on Hybrid Systems: Computation and Control*. Springer, 20–31. https://doi.org/10.1007/3-540-46430-1_6

[9] S. Bak, S. Bogomolov, and C. Schilling. 2016. High-level hybrid systems analysis with Hypy. In *Proc. of the Workshop on Applied Verification of Continuous and Hybrid Systems*. 80–90. https://doi.org/10.29007/4f3d

[10] S. Bak and P. S. Duggirala. 2017. Simulation-equivalent reachability of large linear systems with inputs. In *Proc. of International Conference on Computer Aided Verification*. 401–420. https://doi.org/10.1007/978-3-319-63387-9_20

[11] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. 2019. JuliaReach: a toolbox for set-based reachability. In *Proc. of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 39–44. https://doi.org/10.1145/3302504.3311804

[12] S. Bogomolov, G. Frehse, M. Giacobbe, and T. A. Henzinger. 2017. Counterexample-guided refinement of template polyhedra. In *23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 589–606. https://doi.org/10.1007/978-3-662-54577-5_34

[13] S. Bogomolov and et al. 2016. Guided search for hybrid systems based on coarse-grained space abstractions. *International Journal on Software Tools for Technology Transfer* 18 (2016), 449–467. https://doi.org/10.1007/s10009-015-0393-y

[14] M. Chen, S. Herbert, and C. J. Tomlin. 2017. Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition. In *Proc. of the International Conference on Robotics and Automation*. IEEE, 87–92. https://doi.org/10.1109/ICRA.2017.7989015

[15] M. Chen and C. J. Tomlin. 2015. Exact and efficient Hamilton-Jacobi reachability for decoupled systems. In *Proc. of the 54th International Conference on Decision and Control*. IEEE, 1297–1303. https://doi.org/10.1109/CDC.2015.7402390

[16] A. Chutinan and B. H. Krogh. 2003. Computational techniques for hybrid system verification. *IEEE Trans. Automat. Control* 48, 1 (2003), 64–75. https://doi.org/10.1109/TAC.2002.806655

[17] T. Dang. 2000. *Verification and synthesis of hybrid systems.* Dissertation. Institut National Polytechnique de Grenoble - INPG.

[18] T. Dang, A. Donzé, O. Maler, and N. Shalev. 2008. Sensitive state-space exploration. In *Proc. of the 47th IEEE Conference on Decision and Control*. 4049–4054. https://doi.org/10.1109/CDC.2008.4739371

[19] A. Donzé and O. Maler. 2007. Systematic simulation using sensitivity analysis. In *10th International Workshop on Hybrid Systems: Computation and Control*. Springer, 174–189. https://doi.org/10.1007/978-3-540-71493-4_16

[20] P. S. Duggirala and M. Viswanathan. 2016. Parsimonious, simulation based verification of linear systems. In *Proc. of the 28th International Conference on Computer Aided Verification*. Springer, 477–494. https://doi.org/10.1007/978-3-319-41528-4_26

[21] R. Farhadsefat, J. Rohn, and T. Lotfi. 2011. *Norms of interval matrices.* Technical Report. Academy of Sciences of the Czech Republic, Institute of Computer Science.

[22] M. Forets and C. Schilling. 2022. Conservative time discretization: a comparative study. In *International Conference on Integrated Formal Methods*. Springer, 149–167.

[23] G. Frehse. 2015. Computing maximizer trajectories of affine dynamics for reachability. In *Proc. of the 54th Conference on Decision and Control*. 7454–7461. https://doi.org/10.1109/CDC.2015.7403397

[24] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. 2015. Eliminating spurious transitions in reachability with support functions. In *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, New York, NY, USA, 149–158. https://doi.org/10.1145/2728606.2728622

[25] G. Frehse, R. Kateja, and C. Le Guernic. 2013. Flowpipe approximation and clustering in space-time. In *Proc. of the 16th ACM International Conference on Hybrid Systems: Computation and Control*. 203–212. https://doi.org/10.1145/2461328.2461361

[26] G. Frehse, B. H. Krogh, and R. A. Rutenbar. 2006. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *Proc. of the Design Automation & Test in Europe Conference*. IEEE, 257–262. https://doi.org/10.1109/DATE.2006.244113

[27] G. Frehse and et al. 2011. SpaceEx: Scalable verification of hybrid systems. In *Proc. of the 23rd International Conference on Computer Aided Verification (LNCS 6806)*. Springer, 379–395. https://doi.org/10.1007/978-3-642-22110-1_30

[28] T. Gan, M. Chen, Y. Li, B. Xia, and N. Zhan. 2018. Reachability analysis for solvable dynamical systems. *IEEE Trans. Automat. Control* 63, 7 (2018), 2003–2018. https://doi.org/10.1109/TAC.2017.2763785

[29] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation* 4, 2 (1988), 193–203.

[30] A. Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *8th International Workshop on Hybrid Systems: Computation and Control*. Springer, 291–305. https://doi.org/10.1007/978-3-540-31954-2_19

[31] A. Girard and C. Le Guernic. 2008. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes* 41, 2 (2008). https://doi.org/10.3182/20080706-5-KR-1001.01514

[32] A. Girard, C. Le Guernic, and O. Maler. 2006. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *9th International Workshop on Hybrid Systems: Computation and Control*. Springer, 257–271. https://doi.org/10.1007/11730637_21

[33] A. Hamadeh and J. Goncalves. 2008. Reachability analysis of continuous-time piecewise affine systems. *Automatica* 44, 12 (2008), 3189–3194. https://doi.org/10.1016/j.automatica.2008.05.023

[34] C. Huang, X. Chen, W. Lin, Z. Yang, and X. Li. 2017. Probabilistic safety verification of stochastic hybrid systems using barrier certificates. *ACM Transactions on Embedded Computing Systems* 16, 5s, Article 186 (2017). https://doi.org/10.1145/3126508

[35] T. T. Johnson, S. Bak, M. Caccamo, and L. Sha. 2016. Real-time reachability for verified simplex design. *ACM Transactions on Embedded Computing Systems* 15, 2 (2016). https://doi.org/10.1145/2723871

[36] S. Kousik, A. Dai, and G. X. Gao. 2022. Ellipsotopes: Uniting ellipsoids and zonotopes for reachability analysis and fault detection. *IEEE Trans. Automat. Control* Early Access (2022), 1–13. https://doi.org/10.1109/TAC.2022.3191750

[37] A. A. Kurzhanskiy and P. Varaiya. 2000. Ellipsoidal techniques for reachability analysis. In *3rd International Workshop on Hybrid Systems: Computation and Control*. Springer, 202–214. https://doi.org/10.1007/3-540-46430-1_19

[38] A. A. Kurzhanskiy and P. Varaiya. 2006. Ellipsoidal Toolbox (ET). In *Proceedings of the 45th IEEE Conference on Decision and Control*. 1498–1503. https://doi.org/10.1109/CDC.2006.377036

[39] C. Le Guernic. 2009. *Reachability analysis of hybrid systems with linear continuous dynamics.* Dissertation. Université Joseph-Fourier - Grenoble I.

[40] C. Le Guernic and A. Girard. 2010. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems* 4, 2 (2010), 250–262. https://doi.org/10.1016/j.nahs.2009.03.002

[41] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. 2005. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Control* 50, 7 (2005), 947–957. https://doi.org/10.1109/TAC.2005.851439

[42] A. Mohapatra, V. S. Perić, and T. Hamacher. 2021. Formal verification of grid frequency controllers. In *2021 IEEE PES Innovative Smart Grid Technologies Europe*. 1–6. https://doi.org/10.1109/ISGTEurope52324.2021.9640096

[43] A. Platzer. 2008. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning* 41 (2008), 143–189. https://doi.org/10.1007/s10817-008-9103-8

[44] A. Platzer. 2010. *Logical analysis of hybrid systems: Proving theorems for complex dynamics.* Springer. https://doi.org/10.1007/978-3-642-14509-4

[45] P. Prabhakar and M. Viswanathan. 2011. A dynamic algorithm for approximate flow computations. In *Proc. of the 14th ACM International Conference on Hybrid Systems: Computation and Control*. 133–142. https://doi.org/10.1145/1967701.1967722

[46] S. Prajna and A. Jadbabaie. 2004. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 477–492. https://doi.org/10.1007/978-3-540-24743-2_32

[47] S. Prajna, A. Jadbabaie, and G. J. Pappas. 2007. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. Automat. Control* 52, 8 (2007), 1415–1428. https://doi.org/10.1109/TAC.2007.902736

[48] S. Ratschan and Z. She. 2007. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *Transactions on Embedded Computing Systems* 6, 1 (2007), 8–31. https://doi.org/10.1145/1210268.1210276

[49] S. R. Schepp, J. Thumm, S. B. Liu, and M. Althoff. 2022. SaRA: A tool for safe human-robot coexistence and collaboration through reachability analysis. In *Proc. of the International Conference on Robotics and Automation*. 4312–4317.

[50] S. Schupp and E. Ábrahám. 2018. Efficient dynamic error reduction for hybrid systems reachability analysis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 287–302. https://doi.org/10.1007/978-3-319-89963-3_17

[51] S. Schupp and E. Ábrahám. 2018. The HyDRA tool–a playground for the development of hybrid systems reachability analysis methods. In *Proc. of the PhD Symposium at iFM18*. 22–23.

[52] J. K. Scott, D. Raimondo, G. R. Marseglia, and R. D. Braatz. 2016. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica* 69 (2016), 126–136. https://doi.org/10.1016/j.automatica.2016.02.036

[53] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh. 2004. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice* 12, 10 (2004), 1269–1278. https://doi.org/10.1016/j.conengprac.2004.04.002

[54] M. Wetzlinger, N. Kochdumper, and M. Althoff. 2020. Adaptive parameter tuning for reachability analysis of linear systems. In *Proc. of the 59th IEEE Conference on Decision and Control*. 5145–5152. https://doi.org/10.1109/CDC42340.2020.9304431

[55] M. Wetzlinger, A. Kulmburg, A. Le Penven, and M. Althoff. 2022. Adaptive reachability algorithms for nonlinear systems using abstraction error analysis. *Nonlinear Analysis: Hybrid Systems* 46 (2022), 101252. https://doi.org/10.1016/j.nahs.2022.101252