Ref. Ares(2015)6002490 - 22/12/2015

Unifying Control and Verification of Cyber-Physical Systems (UnCoVerCPS)

WP4 Tool Support (Task 4.1)

D4.1 – Theoretical Foundations for Combining Zonotopes and Support Functions

WP4	**D4.1 – Theoretical Foundations for Combining Zonotopes and Support Functions**
Authors	Goran Frehse - Univ. Grenoble Alpes, Verimag Matthias Althoff - Technische Universität München
Short Description	We present a detailed assessment of the computational efficiency of set representations for reachability, namely zonotopes and support functions. As a result, we propose representing reachable sets as a combination between support functions and zonotopes. This representation can be converted to constraint polyhedra of desired (directional) precision, at a higher precision compared to using only support functions and with more accuracy than only using zonotopes.
Keywords	Reachability analysis, support functions, zonotopes, polyhedra, linear continuous systems.
Deliverable Type	Report
Dissemination level	Public
Delivery Date	31 Dec 2015
Contributions by	—
Internal review by	Simone Schuler, Bastian Schürmann
External review by	—
Internally accepted by	Matthias Althoff
Date of acceptance	17 Dec 2015

Document history:

Version	**Date**	**Author/Reviewer**	**Description**
0.9	26 Oct 2015	Frehse et al.	Internal review version
1.0	26 Nov 2015	Frehse et al.	Final version

Contents

1 Introduction

Dynamical systems are known for producing complex behaviors, e.g., trajectories of continuous variables, that can be difficult to analyze and verify. In this paper, we consider continuous systems defined by a set ordinary differential equations (ODE) with initial and boundary conditions, and address the problem of computing a cover of all the states that are reachable in the system. This is referred to as *set-based reachability analysis* and can be used, e.g., to show that the system does not reach any states that are considered unsafe. It can also provide quantitative information, e.g., for measuring the jitter in an oscillator circuit.

Reachability computation can be seen as a generalization of *numerical simulation*. Just like numerical simulation, reachability computation has to use approximations if the dynamics of the system are complex. Working with sets instead of points, approximate reachability can be conservative in the sense that the computed sets are sure to cover all solutions. Computation costs generally increase sharply in terms of the number of continuous variables. Scalable approximations are available for certain types of dynamics, as discussed later in this section, but this performance comes at a price in accuracy. The trade-off between runtime and accuracy remains a central problem in reachability analysis. The approximation error requires particular attention since it can accumulate rapidly, leading to a coarse cover, prohibitive state explosion, or preventing termination. In this paper, we propose an approach with precise control over the balance between approximation error and scalability, particularly tailored to changing dynamics.

Contributions The main contribution of this paper is the concept of representing the reachable sets themselves as a combination between support functions and zonotopes. This representation can be converted to constraint polyhedra of desired (directional) precision. This work provides the basis for scalable use of support function algorithms on systems with nonlinear dynamics. Existing algorithms for support functions are tailored specifically to affine dynamics (linear dynamics with nondeterministic offsets). Nonlinear dynamics can be approximated with piecewise affine dynamics, but this requires switching the dynamics during the analysis run. Existing support function algorithms are not scalable when the dynamics change, as will be examined in detail in this paper.

In the context of the EU project UnCoVerCPS, this work contributes know-how to Task 3.1 (Faster methods for reachability analysis of nonlinear systems), since it makes fast support function algorithms applicable to nonlinear systems. It is also related to Task

4.3 (Integration of SpaceEx and CORA), since SpaceEx so far is using support function algorithms, and Cora is using zonotopes. Since Task 4.3 will provide both technologies on the same software platform, the algorithms proposed in this report can be implemented and assessed experimentally.

Related Work The content of this paper builds on research going back more than 20 years. Notable break-throughs were the recurrence equations for piecewise affine dynamics in [4], the use of zonotopes for scalability in [14], and the use of support functions for scalability in [19]. To the best of our knowledge, this is the first work on combining zonotopes and support functions. In this paper, we limit the discussion to switched dynamics, which change only at isolated points in time. A more general case of time-varying dynamics was considered in [**?**], where the dynamics are allowed to vary continuously over time. Other related work is cited throughout the paper. For introductory texts on reachability analysis see, e.g., in [22, 24, 7]. Surveys can be found in [3, 21, 6].

The remainder of the paper is structured as follows. In Sect. 2, we briefly present zonotopes and support functions, as well as the main geometric operations on them. In Sect. 3, we present the problem of approximating the flowpipes of systems with affine dynamics, extending known algorithms to variable time steps and switched dynamics. The presentation is largely independent of the set representation used to implement the algorithm. In Sect. 4, we analyze and compare the complexity of implementing the algorithm from Sect. 3 for zonotopes, polyhedra, and support functions. Our observations are then used to devise a combination of support functions and zonotopes, which we present in Sect. 5. We conclude with a summary of our findings and an outlook on future work in Sect. 6.

2 Efficient Set Representations for Reachability

In this section, we briefly introduce two set representations that are known to lead to highly scalable reachability computations. The first one, zonotopes, is a particular type of centrally symmetric polytope. The downside of zonotopes is that they are not closed under certain set operations used in reachability. The second set representation is support functions. Any convex set can be represented by its support function, and all geometric operations have its corresponding operation on the support function. However, the number of evaluations may increase prohibitively if geometric operations are chained. The goal of the report is to isolate the advantages and disadvantages of both in the context of reachability, and to examine how

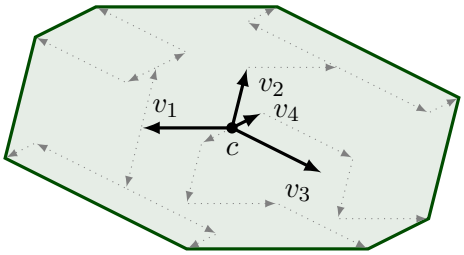

Figure 1: A zonotope is a special form of centrally symmetric polytope, as illustrated here with generators v_1, v_2, v_3, v_4, and center c

both can be combined beneficially.

2.1 Zonotopes

Zonotopes are a compact representation for a special form of polytopes that have been used successfully for reachability analysis due to their computationally attractive features [14, 2]. A *zonotope* $P \subseteq R^n$ is defined by a center $c \in \mathbb{Q}^n$ and a finite number of generators $v_1, \ldots, v_k \in \mathbb{Q}^n$ that span the polytope as bounded linear combinations from the center:

$$P = \Big\{ c + \sum_{i=1}^{k} \alpha_i v_i \;\Big|\; \alpha_i \in [-1, 1] \Big\}.$$

A common denotation for this zonotope is $P = (c, \langle v_1, \ldots, v_k \rangle)$. Zonotopes are central-symmetric convex polytopes, see Fig. 1 for an illustration. An alternative representation of P uses the generators in a matrix. For any p-norm

$$\|x\|_p = \left(\sum\nolimits_{i=1}^{n} |x_i|^p \right)^{1/p},$$

the *k-dim. ball* is

$$\mathcal{B}_p^k = \Big\{ x \in R^k \;\Big|\; \|x\| \leq 1 \Big\}.$$

A zonotope with k generators is an affine transformation of the ball of the infinity-norm (a k-dimensional unit hypercube). Let the generator matrix be $M = [v_1 \ldots v_k]$, then

$$P = c \oplus M\mathcal{B}_\infty^k.$$

.

2.1.1 Geometric Operations

We present how the geometric operations required for reachability are implemented for zonotopes, with particular attention to the computational cost, measured in the number of (binary) arithmetic operations.

Linear and affine transformations can be computed efficiently for zonotopes. For a matrix $A \in \mathbb{Q}^{m \times n}$, the image of the linear transformation can simply be computed component-wise:

$$AP = (Ac, \langle Av_1, \ldots, Av_k \rangle)$$

The number of numeric operations involves is $\mathcal{O}\left(n^2(k+1)\right)$.

The Minkowski sum can be computed efficiently for zonotopes $P = (c, \langle v_1, \ldots, v_k \rangle)$ and $Q = (d, \langle w_1, \ldots, w_m \rangle)$ by a single vector addition and a single list concatenation:

$$P \oplus Q = (c + d, \langle v_1, \ldots, v_k, w_1, \ldots, w_m \rangle).$$

Only the centers of the zonotopes need to be manipulated, for the generators it suffices to join the two lists. The operation therefore involves only $\mathcal{O}(n)$ numeric operations.

Zonotopes are not closed under convex hull, i.e., the convex hull of two zonotopes is not necessarily a zonotope. Zonotopes are not closed under intersection. The lack of accuracy in intersections can make the computation of successors in hybrid automata problematic, since the image computation of discrete transitions requires intersection with the guard set of the transition. In special cases it can be advantageous to use an approach called *continuization* to avoid the intersection operation, see [1]. Instead of intersecting a set of states with the guard set and then applying the dynamics of the successor location to the result, the states suspected to intersect with the guard set (by some approximative measure) are subjected to nondeterministic dynamics that overapproximate the dynamics both before and after the jump. The dynamics of the successor location are used once enough time steps have been carried out to be sure the set no longer intersects with the guard set.

The bounding box and bounding sphere of a zonotope can be constructed efficiently as follows [16]. The *absolute row sums* of a zonotope with generators v_i are

$$r_P = \sum_{i=1}^{k} |v_i| .$$

The bounding box (interval hall) of P is

$$c \oplus \operatorname{diag}(r_P)\mathcal{B}_\infty,$$

where $\mathcal{B}_\infty$ is the unit hypercube. The Euclidian radius of P is

$$\operatorname{rad}(P) = \|r_P\|_2.$$

The bounding sphere of P is

$$c \oplus \operatorname{rad}(P)\mathcal{B}_2,$$

where $\mathcal{B}_2$ is the unit sphere. Both can be computed with at most $n(k+1)$ numerical operations.

For any matrix A with column vectors a_j and zonotope P, let r_P be the vector of absolute row sums of P. The absolute row sums of AP are bounded by

$$r_{AP} = \sum\nolimits_{i=1}^{k} |Av_i| \quad \leq \quad \sum\nolimits_{i=1}^{k} |A|\,|v_i| = |A| \sum\nolimits_{i=1}^{k} |v_i| = |A|\, r_P. \tag{1}$$

and the radius is bounded by

$$\mathrm{rad}(AP) \leq \| |A|\, r_P \|_2 \tag{2}$$

2.1.2 Simplification

Since the number of generators of a zonotope increases with the Minkowski sum and convex hull operations, it can be necessary to overapproximate a zonotope with another one that has fewer generators. The following approach from [14] reduces the number of generators by n. Let $v_1, \ldots, v_{2n}$ be the generators with the lowest values of $\|v_i\|_1 - \|v_i\|_\infty$ (pointing in one of the axis directions). The reduced zonotope is obtained by replacing $v_1, \ldots, v_{2n}$ with $v'_1, \ldots, v'_n$, where the j-th element of v_i is defined by $v'_i[j] = 0$ if $i \neq j$ and

$$v'_i[j] = \sum_{j=1}^{2n} \big| v_i[j] \big|.$$

Note that this approximation is tight in the positive and negative axis directions. The computational cost for k generators is $\mathcal{O}\left(nk \log k\right)$. A big advantage of this simplification is that simplification and Minkowski sum commute. Adding simplified zonotopes gives the same result as simplifying the sum of the original zonotopes. This means that simplification does not lead to an explosion of the approximation error, when applied in an iterative algorithm that involves only Minkowski sum.

2.2 Support Functions

We now introduce our notation, give definitions for polyhedra and support functions, and recall some fundamental properties. A *halfspace* $\mathcal{H} \subseteq R^n$ is the set of points satisfying a linear constraint, $\mathcal{H} = \left\{ x \mid a^\mathsf{T} x \leq b \right\}$, with *normal vector* $a = (a_1 \cdots a_n) \in R^n$ and $b \in R$. A *polyhedron* $\mathcal{P} \subseteq R^n$ is the intersection of a finite number of halfspaces, written as

$$\mathcal{P} = \left\{ \bigwedge\nolimits_{i=1}^{m} a_i^\mathsf{T} x \leq b_i \right\},$$

where $a_i \in R^n$ and $b_i \in R$. A *polytope* is a bounded polyhedron. Any convex set can be represented by its support function. The support function of a compact set $\mathcal{X}$ attributes to a direction $\ell \in R^n$ the scalar value

$$\rho_{\mathcal{X}}(\ell) = \max\left\{ \ell^\mathsf{T} x \;\middle|\; x \in \mathcal{X} \right\}.$$

For a given direction ℓ, it defines the position of a halfspace

$$\mathcal{H}_\ell = \left\{ \ell^\mathsf{T} x \leq \rho_{\mathcal{X}}(\ell) \right\},$$

(a) support function (b) outer approximation

Figure 2: Evaluating the support function in a set of directions gives a polyhedral outer approximation

which touches and contains $\mathcal{X}$. If ℓ is of unit length, then $\rho_{\mathcal{X}}(\ell)$ is the signed distance of $\mathcal{H}_\ell$ to the origin, see Fig. 2(a) for an illustration. Evaluating the support function for a set of directions $L \subseteq R^n$ gives an *outer approximation*

$$\lceil\mathcal{X}\rceil_L = \bigcap_{\ell \in L} \{\ell^{\mathsf{T}} x \leq \rho_{\mathcal{X}}(\ell)\}, \tag{3}$$

i.e., $\mathcal{X} \subseteq \lceil\mathcal{X}\rceil_L$. If $L = R^n$, then $\mathcal{X} = \lceil\mathcal{X}\rceil_L$, so the support function represents $\mathcal{X}$ exactly. If L is a finite set of directions $L = \{\ell_1, \ldots, \ell_m\}$, then $\lceil\mathcal{X}\rceil_L$ is a polyhedron, as shown in Fig. 2(b). This is also referred to as a *template polyhedron* with L being the *template directions*. The difference between using support functions and traditional methods for template polyhedra, e.g., [23], lies in the fact that the outer approximation $\lceil\mathcal{X}\rceil_L$ can be refined at any time, and incrementally, by adding more directions to L. One can interpret evaluating support functions as the lazy, on-demand, construction of a template polyhedron.

2.2.1 Geometric Operations

The following set operations are required by our reachability algorithm, and are extremely efficient on support functions. We measure the number of operations per template direction. The final cost will depend on how many directions are evaluated; more on this below.

Consider non-empty compact convex sets $\mathcal{X}, \mathcal{Y} \subseteq R^n$. The *linear map* with a matrix $M \in R^{m \times n}$ is $M\mathcal{X} = \{Mx \mid x \in \mathcal{X}\}$. Using support functions, the linear map simplifies to

$$\rho_{M\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(M^{\mathsf{T}}\ell), \tag{4}$$

which is $\mathcal{O}(mn)$. The *convex hull* of a set $\mathcal{Z} \subseteq R^n$ is

$$\mathrm{CH}(\mathcal{Z}) = \left\{ \sum_{i=1}^{m} \lambda_i v_i \;\middle|\; v_i \in \mathcal{Z}, \lambda_i \in R^{\geq 0}, \sum_{i=1}^{m} \lambda_i = 1 \right\}.$$

Using support functions, the convex hull of $\mathcal{X}$ and $\mathcal{Y}$ is

$$\rho_{\mathrm{CH}(\mathcal{X} \cup \mathcal{Y})}(\ell) = \max\{\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)\},$$

which is $\mathcal{O}(1)$. The *Minkowski sum* is $\mathcal{X} \oplus \mathcal{Y} = \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, which translates to the $\mathcal{O}(1)$ operation

$$\rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell).$$

In the following example, a sequence of linear mapping and Minkowski sums leads to a convex set that is prohibitively complex for polyhedral operations, but that is easy to approximate with support functions. Evidently, support functions do not enable us to avoid the curse of dimensionality in general: An n-dimensional approximation with a distance of ε to the real set requires $\mathcal{O}\left(1/\varepsilon^{n-1}\right)$ evaluations of the support function [20]. But many problems do not require precise knowledge of the whole set; with the support function the approximation can be targeted to where it matters. A simple example shall illustrate this point.

Example 2.1 *Consider the halfspace $\mathcal{H} = \{a^\mathsf{T} x \geq b\}$. Then $\mathcal{X} \cap \mathcal{H} \neq \emptyset$ if and only if $\rho_{\mathcal{X}}(a) \geq b$. Furthermore, $\mathcal{X} \subseteq \mathcal{H}$ if and only if $-\rho_{\mathcal{X}}(-a) \geq b$. Each problem can be decided with a single evaluation of the support function.*

The problem of whether two convex sets intersect is known as the *separation problem*, and it can be addressed by computing the support function in a sequence of directions. Let

$$h(\ell) = \rho_{\mathcal{X}\oplus(-\mathcal{Y})}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(-\ell),$$

which is a convex function. Then $\mathcal{X} \cap \mathcal{H} = \emptyset$ if and only if $\min_{\ell \in R^n} h(\ell) < 0$. Any converging convex minimization algorithm applied to $h(\cdot)$ will produce a sequence of directions that converges to a separating direction if it exists. Separation algorithms adapted to the context of approximate reachability have been proposed in [8].

Two more operations of our reachability algorithm, intersection and containment, turn out to be difficult for support functions. The intersection $\mathcal{X} \cap \mathcal{Y}$ requires solving a convex optimization problem:

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) = \inf_{\nu \in R^n} \rho_{\mathcal{X}}(\ell - \nu) + \rho_{\mathcal{Y}}(\nu).$$

Containment checking with support functions generally involves an uncountable number of directions: $\mathcal{X} \subseteq \mathcal{Y}$ if and only if $\rho_{\mathcal{X}}(\ell) \leq \rho_{\mathcal{Y}}(\ell)$ for all $\ell \in R^n$. We circumvent these problems by switching to polyhedra as set representation at corresponding points in the algorithm. Intersection of polyhedra is cheap since it amounts to taking the union of their constraints. Containment checking is simple if the right hand side is a polyhedron: Let $\mathcal{P} = \left\{\bigwedge_{i=1}^{m} a_i^\mathsf{T} x \leq b_i\right\}$, then $\mathcal{X} \subseteq \mathcal{P}$ if and only if $\rho_{\mathcal{X}}(a_i) \leq b_i$ for $i = 1, \ldots, m$.

2.2.2 The Support Function of Zonotopes

The support function of a zonotope is easily derived from its ball representation. The support function of the k-dimensional unit ball of the p-norm is [17]

$$\rho_{\mathcal{B}_p^k}(d) = \|d\|_{\frac{p}{p-1}},$$

and for $d \neq 0$ it has the set of maximizers

$$\sigma_{\mathcal{B}_p^k}(d) = \left\{ y \in \mathcal{B}_p^k \;\middle|\; d^\mathsf{T} y = \|d\|_{\frac{p}{p-1}} \right\}.$$

To quickly compute one of the maximizers, say y^*, let y be defined element-wise from

$$y_i d_i = |d_i|^{\frac{p}{p-1}}$$

and then norming to $y^* = y/\|y\|_p$. Recall that $\rho_{M\mathcal{X}}(d) = \rho_{\mathcal{X}}(M^\mathsf{T} d)$ and $\sigma_{M\mathcal{X}}(d) = M\sigma_{\mathcal{X}}(M^\mathsf{T} d)$. A zonotope with k generators can be written as $P = M\mathcal{B}_\infty^k \oplus c$, with generator matrix $M = [v_1 \dots v_k]$. The support function of P is

$$\rho_P(d) = \|M^\mathsf{T} d\|_1 + d^\mathsf{T} c,$$

which involves $2n(k+1)$ numeric operations. The (not necessarily unique) maximizers are

$$\sigma_P(d) = \{c + y \mid d^\mathsf{T} y = \|M^\mathsf{T} d\|_1\}.$$

A maximizer y^* can be computed as $y^* = c + My$, where y is defined element-wise as

$$y_i = \operatorname{sign}(M^\mathsf{T} d)_i.$$

3 Flowpipe Approximation

Our goal is to solve the following problem: given an initial set of states $\mathcal{X}_0$ and dynamics in the form of an ODE, compute a finite cover of all possible trajectories starting in $\mathcal{X}_0$. More precisely, let the *affine dynamics* be of the form

$$\dot{x} = Ax + u, \qquad u \in \mathcal{U}, \tag{5}$$

where A is an $n \times n$ matrix and the *input set* $\mathcal{U}$ is compact and convex. To simplify notation, we assume that constants and input mappings are modeled within $\mathcal{U}$, e.g., dynamics of the form $\dot{x} = Ax + b + Bv$, $v \in \mathcal{V}$ are modeled with (5) by setting $\mathcal{U} = \{b\} \oplus B\mathcal{V}$.

The evolution of the nondeterministic input u is described by an *input signal* $\zeta : R^{\geq 0} \to \mathcal{U}$ that attributes to each point in time a value of u. The input signal does not need to be

continuous. A trajectory $\xi(t)$ from a state x_0 is the solution of the differential equation (5) for a given initial condition $\xi(0) = x_0$ and a given input signal ζ. It has the form

$$\xi_{x_0,\zeta}(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}\zeta(s)ds. \tag{6}$$

It consists of the superposition of the solution of the *autonomous* system, obtained for $\zeta(t) = 0$, and the *input convolution*, which is the solution obtained for $x_0 = 0$. Given an initial set $\mathcal{X}_0$, the *reachable states at time* t is the set of values of the solutions of (5) with initial condition $x(0) \in \mathcal{X}_0$. We denote this set with

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \int_0^t e^{A(t-s)}\mathcal{U}ds = e^{At}\mathcal{X}_0 \oplus \int_0^t e^{As}\mathcal{U}ds. \tag{7}$$

Let $\mathcal{Y}_t$ be the input convolution at time t, i.e., the states reachable from $\mathcal{X}_0 = \{0\}$, then (7) can be written as

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \mathcal{Y}_t. \tag{8}$$

The *flowpipe segment* over the time interval $[t_b, t_e]$ is the set

$$\mathcal{X}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \mathcal{X}_t.$$

The goal of our flowpipe approximation is to compute a finite sequence of sets $\Omega_0, \Omega_1, \ldots$ such that

$$\bigcup_{0 \le t \le T} \mathcal{X}_t \subseteq \Omega_0 \cup \Omega_1 \cup \ldots. \tag{9}$$

The goal of this paper is to investigate how the flowpipe approximation can be carried out efficiently when the dynamics and input set change over time. In particular, we will consider piecewise constant A, e.g., $A = A_1$ for $0 \le t < t_1$ and $A = A_2$ for $t \ge t_2$. The above principles are formalized in the next section, which prepares our investigation into handling variable dynamics.

3.1 Fundamental Properties of the Reachable Set

We recall some fundamental properties of the reachable set and examine how these properties hold up when the dynamics change with time. In the following, we consider *time-dependent* dynamics

$$\dot{\xi}(t) = A_t x(t) + \zeta(t), \qquad \zeta(t) \in U_t, \tag{10}$$

where $\xi(t) \in R^n$ is an n-dimensional vector, $A_t : R^{\ge 0} \to R^n \times R^n$ attributes to every time point t a real matrix, and $U_t : R^{\ge 0} \to 2^{R^n}$ attributes to every time point t a convex set. Let

$\text{Reach}_{[t_1,t_2]}(A_t, U_t, X)$ denote the reachable states starting from the set X with input set U_t in the time interval $[t_1, t_2]$,

$$\text{Reach}_{t_1,t_2}(A_t, U_t, X) = \{\xi(\tau) \mid t_1 \leq \tau \leq t_2, \xi(0) \in X, \dot{\xi}(t) = A_t x(t) + \zeta(t), \zeta(t) \in U_t)\}. \quad (11)$$

Our goal is to compute a sequence of continuous sets $\Omega_0, \ldots, \Omega_{N-1}$ that covers the reachable states up to time T (N depends on the chosen time steps):

$$\text{Reach}_{0,T}(A_t, U_t, \mathcal{X}_0) \subseteq \bigcup_{k=0}^{N-1} \Omega_k. \quad (12)$$

Recall that for constant A and given $\xi(t_0) = x_0$ and $\zeta(t)$, the solution of (10) is given by

$$\xi(t) = e^{A(t-t_0)}\xi(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}\zeta(\tau)d\tau. \quad (13)$$

The reachable set can therefore be expressed as

$$\text{Reach}_{t_1,t_2}(A, U_t, X) = \left\{ e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-\tau)}\zeta(\tau)d\tau \;\middle|\; t_1 \leq t \leq t_2, x_0 \in X, \zeta(t) \in U_t \right\}. \quad (14)$$

In the following, we will frequently use the superposition principle to separate the autonomous evolution of the system (i.e., without inputs) from the influence of the inputs. To simplify notation, we denote with 0 the number zero, a vector with all elements zero, and the set $\{0\}$.

Lemma 3.1 (superposition)

$$\text{Reach}_{t_1,t_1}(A_t, U_t, X) = \text{Reach}_{t_1,t_1}(A_t, 0, X) \oplus \text{Reach}_{t_1,t_1}(A_t, U_t, 0).$$

Let the *input convolution* be defined as

$$\mathcal{Y}_{t_1,t_2}(A_t, U_t) = \{\xi(t_2) \mid \xi(t_1) = 0, \dot{\xi}(t) = A_t x(t) + \zeta(t), \zeta(t) \in U_t\}. \quad (15)$$

Lemma 3.2

$$\text{Reach}_{t_1,t_1}(A_t, U_t, 0) = \mathcal{Y}_{0,t_1}(A_t, U_t).$$

The input convolution for constant $A_t = A$ simplifies with (13) to

Lemma 3.3 *If $A_t = A = \text{const}$,*

$$\mathcal{Y}_{t_1,t_2}(A, U_t) = \left\{ \int_{t_1}^{t_2} e^{A(t_2-\tau)}\zeta(\tau)d\tau \;\middle|\; \forall t_1 \leq t \leq t_2 : \zeta(t) \in U_t \right\}. \quad (16)$$

Proof 1 *The result is obtained by applying (13) to (15).*

The following lemma relates the input convolution to the reachable states.

Lemma 3.4 *If $A_t = A = \text{const}$ and $U_t = U = \text{const}$,*

$$\text{Reach}_{t_1,t_2}(A, U, X) = \left\{ e^{At} X \oplus \mathcal{Y}_{0,t}(A, U) \;\middle|\; t_1 \le t \le t_2 \right\}.$$

We now derive some useful properties of the input convolution. The input convolution can be shifted in time if the dynamic matrix A is constant and U_t repeats after the shift.

Lemma 3.5 (input convolution shift) *If $A_t = A = \text{const}$ and $U_t = U_{t+\Delta}$ for all $t_1 \le t \le t_2$, then $\mathcal{Y}_{t_1,t_2} = \mathcal{Y}_{t_1+\Delta,t_2+\Delta}$.*

Proof 2 *Consider (16) for $\mathcal{Y}_{t_1+\Delta,t_2+\Delta}$. Applying the substitution $z = \tau - \Delta$ immediately leads to the conclusion.*

The input convolution can be split if the dynamic matrix A is constant over some time interval. Note that U_t can be time-varying.

Lemma 3.6 (input convolution split) *If $A_t = A$ for $t_1 \le t < t_1+t_2$, then $\mathcal{Y}_{0,t_1+t_2}(A_t, U_t) = e^{At_2}\mathcal{Y}_{0,t_1}(A_t, U_t) \oplus \mathcal{Y}_{t_1,t_1+t_2}(A, U_t)$.*

Proof 3 *Consider Def. 15. The trajectories $\xi(t)$ that define $\mathcal{Y}_{0,t_1+t_2}(A_t, U_t)$ must satisfy $\xi(0) = 0$, (10) and $A_t = A$ for $t_1 \le t < t_2$. We can separate the part of $\xi(t)$ before and after t_1, and apply (13) to the part after t_1. Then the trajectories $\xi(t)$ are characterized by*

$$\begin{aligned} \forall 0 \le \tau \le t_1 : \quad & \dot{\xi}(\tau) = A_\tau \xi(\tau) + \zeta(\tau), \\ \forall t_1 \le \tau \le t_1 + t_2 : \quad & \xi(\tau) = e^{A(\tau - t_1)}\xi(t_1) + \int_{t_1}^{\tau} e^{A(\tau-\nu)}\zeta(\nu)d\nu. \end{aligned}$$

For $\mathcal{Y}_{0,t_1+t_2}(A_t, U_t)$, we only need the value $\xi(t_1 + t_2)$, defined in the second line. The value depends on $\xi(t_1)$, whose characterization is given by the first line and exactly corresponds to the definition of $\mathcal{Y}_{0,t_1}(A_t, U_t)$. We obtain

$$\mathcal{Y}_{0,t_1+t_2}(A_t, U_t) = \left\{ \xi(t_1 + t_2) \;\middle|\; \xi(t_1) \in \mathcal{Y}_{0,t_1}, \; \xi(t_1 + t_2) = e^{A(t_2+t_1-t_1)}\xi(t_1) + \int_{t_1}^{t_1+t_2} e^{A(t_1+t_2-\nu)}\zeta(\nu)d\nu \right\}.$$

The first summand of $\xi(t_1 + t_2)$ can be factored out to yield $e^{At_2}\mathcal{Y}_{0,t_1}(A_t, U_t)$. The remaining integral term corresponds to the definition of $\mathcal{Y}_{t_1,t_1+t_2}(A, U_t)$. This leaves

$$\mathcal{Y}_{0,t_1+t_2}(A_t, U_t) \;=\; e^{At_2}\mathcal{Y}_{0,t_1}(A_t, U_t) \oplus \mathcal{Y}_{t_1,t_1+t_2}(A, U_t).$$

Note that the Minkowski sum is not an overapproximation, since the choice of $\zeta(\tau)$ for $\tau \le t_1$ is independent of the choice for $\tau \ge t_1$ (a zero measure difference at $\tau = t_1$ does not change the value of the integral).

With the above results, the input convolution allows us to succinctly express a time shift of the reachable states if the dynamics are time-invariant.

Lemma 3.7 (time-invariant shift) *If $A_t = A = \text{const}$ and $U_t = U = \text{const}$,*

$$\text{Reach}_{t_1+\Delta,t_2+\Delta}(A, U, X) = e^{A\Delta}\text{Reach}_{t_1,t_2}(A, U, X) \oplus \mathcal{Y}_{0,\Delta}.$$

Proof 4 *With Lemmas 3.4 and 3.5,*

$$\begin{aligned}\text{Reach}_{t_1+\Delta,t_2+\Delta}(A, U, X) &= \{e^{A\Delta}e^{A(t-\Delta)}X \oplus \mathcal{Y}_{-\Delta,t-\Delta} \mid t_1 \le t - \Delta \le t_2\} \\ &= \{e^{A\Delta}e^{Az}X \oplus \mathcal{Y}_{-\Delta,z} \mid t_1 \le z \le t_2\}.\end{aligned}$$

Using Lemma 3.6 and Lemma 3.5 we obtain

$$\begin{aligned}\mathcal{Y}_{-\Delta,z} &= \mathcal{Y}_{-\Delta,0} \oplus e^{A\Delta}\mathcal{Y}_{0,z} = \mathcal{Y}_{0,\Delta} \oplus e^{A\Delta}\mathcal{Y}_{0,z}, \\ \text{Reach}_{t_1+\Delta,t_2+\Delta}(A, U, X) &= \{e^{A\Delta}e^{Az}X \oplus \mathcal{Y}_{0,\Delta} \oplus e^{A\Delta}\mathcal{Y}_{0,z} \mid t_1 \le z \le t_2\}.\end{aligned}$$

We obtain the conclusion by factoring out the term of the Minkowski sum that is independent of z, and then applying Lemma 3.4.

Finally, we recall a classic property of the reachable set, which can be found, e.g., in [25]:

Lemma 3.8 (time-variant union)

$$\text{Reach}_{0,t_1+t_2}(A_t, U_t, X) = \text{Reach}_{0,t_1}(A_t, U_t, X) \cup \text{Reach}_{t_1,t_1+t_2}(A_t, U_t, X).$$

3.2 A Variable Time Step Algorithm

We now present the construction of a flowpipe approximation in the form of a sequence $\Omega_k := \Omega_{t_k,t_{k+1}}$, where $t_k = \sum_{i=0}^{k-1} \delta_i$ for arbitrary time steps $\delta_0, \delta_1, \ldots$. As we will show, each set $\Omega_{t_k,t_{k+1}}$ covers the reachable states in the time interval $[t_k, t_{k+1}]$. We consider only constant dynamics $A_t = A$ and inputs $U_t = U$.

Recall that $\text{Reach}_{0,\delta}$ denotes the states reachable in the time interval $[0, \delta]$, and $\text{Reach}_{\delta,\delta}$ the states at exactly the time point δ. Let $\Omega_{0,\delta}(X_0, U)$ and $V_{0,\delta}(U)$ be defined as a function of δ such that they satisfy

$$\text{Reach}_{0,\delta}(A, U, \mathcal{X}_0) \subseteq \Omega_{0,\delta}(X_0, U), \qquad \text{Reach}_{\delta,\delta}(A, U, 0) \subseteq V_{0,\delta}(U). \tag{17}$$

Several ways to compute $\Omega_{0,\delta}$ and $V_{0,\delta}$ have been proposed in [17] (other authors have proposed different but related solutions). They are first-order approximations of the form

$$\Omega_{0,\delta}(X_0, U) = \text{CH}\left(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta U \oplus \mathcal{E}_\Omega(\mathcal{X}_0, U, \delta)\right), \tag{18}$$

$$V_{0,\delta}(U) = \delta U \oplus \mathcal{E}_V(U, \delta), \tag{19}$$

where $\mathrm{CH}(X, Y)$ denotes the convex hull of sets X and Y, and $\mathcal{E}_\Omega$ and $\mathcal{E}_V$ are convex sets overapproximating the error. Error terms of different complexities are known. We recall the simplest form

$$\mathcal{E}_\Omega(\mathcal{X}_0, U, \delta) = \alpha(\mathcal{X}_0, U)\mathcal{B}, \tag{20}$$

$$\mathcal{E}_V(U, \delta) = \beta(U)\mathcal{B}, \tag{21}$$

where $\mathcal{B}$ denotes the unit ball of the norm, $\mathcal{B} = \{x \mid \|\mathcal{X}\| \leq 1\}$. Let $\|A\|$ be the norm of a matrix A, and $\|\mathcal{X}\| = \sup_{x \in X} \|\mathcal{X}\|$ be the norm of a set X. Note that any norms can be used, as long as the matrix norm is consistent with the vector norm used to define the norm of the set. If $\|A\| = 0$, then $\alpha(\mathcal{X}_0, U) = 0$ and $\beta(U) = 0$, otherwise

$$\alpha(\mathcal{X}_0, U) = (e^{\|A\|\delta} - 1 - \|A\|\delta)(\|\mathcal{X}_0\| + \|U\|/\|A\|), \tag{22}$$

$$\beta(U) = (e^{\|A\|\delta} - 1 - \|A\|\delta)(\|U\|/\|A\|). \tag{23}$$

Given the approximations $\Omega_{0,\delta}(X_0, U)$ and $V_{0,\delta}(U)$ as defined above, we construct the flowpipe approximation using the sequence

$$\begin{aligned} \Psi_{0,t_{k+1}} &= \Psi_{0,t_k} \oplus e^{At_k} V_{0,\delta_k}(U), \\ \Omega_{t_k,t_{k+1}} &= e^{At_k} \Omega_{0,\delta_k}(X_0, U) \oplus \Psi_{0,t_k}, \end{aligned} \tag{24}$$

with $\Psi_{0,0} = 0$. Before showing that this sequence indeed covers the reachable set, we first show that the Ψ_{0,t_k} overapproximates the input convolution $\mathcal{Y}_{0,t_k}$.

Lemma 3.9 $\mathcal{Y}_{0,t_k} \subseteq \Psi_{0,t_k}$

Proof 5 *With $t_{k+1} = t_k + \delta_k$ and Lemma 3.6,*

$$\mathcal{Y}_{0,t_{k+1}} = \mathcal{Y}_{0,t_k} \oplus e^{At_k} \mathcal{Y}_{0,\delta_k}.$$

With Lemma 3.4 and (17),

$$\mathcal{Y}_{0,\delta_k} = \mathrm{Reach}_{\delta_k,\delta_k}(A, U, 0) \subseteq V_{0,\delta_k}(U).$$

Combining the above two equations, we obtain

$$\mathcal{Y}_{0,t_{k+1}} \subseteq \mathcal{Y}_{0,t_k} \oplus e^{A\delta_k} V_{0,\delta_k}(U).$$

The conclusion follows by induction.

Now we establish the main result of this section: The sequence $\Omega_k := \Omega_{t_k,t_{k+1}}$ covers the reachable set.

Proposition 3.10 *Given a sequence of time steps $\delta_0, \delta_1, \ldots, \delta_{N-1}$ with $t_N = T$, where $t_k = \sum_{i=0}^{k-1} \delta_i$, the sequence $\Omega_k := \Omega_{t_k,t_{k+1}}$ defined by (24) satisfies (12).*

Proof 6 *With Lemma 3.7, we have*

$$\text{Reach}_{t_k,t_{k+1}}(A, U, X_0) = e^{At_k}\text{Reach}_{0,\delta_k}(A, U, X_0) \oplus S_{0,t_k},$$

and with Lemma 3.9,

$$\text{Reach}_{t_k,t_{k+1}}(A, U, X_0) \subseteq e^{At_k}\text{Reach}_{0,\delta_k}(A, U, X_0) \oplus \Psi_{0,t_k}.$$

Then by induction with (17) as base case,

$$\text{Reach}_{t_k,t_{k+1}}(A, U, X_0) \quad \subseteq \quad \Omega_{t_k,t_{k+1}}. \tag{25}$$

The conclusion follows, again by induction, with Lemma 3.8.

3.3 Time-Varying Dynamics

We now consider a dynamic matrix A_t and an input set U_t that change over time. We restrict ourselves to piecewise constant changes: Let A_i be a sequence of matrices, let U_i be a sequence of sets, and let $s_0 = 0, s_1, s_2, \ldots$ be an increasing sequence of *switching times*, such that

$$A_t = A_i \text{ and } U_t = U_i \text{ for } s_i \leq t < s_{i+1}.$$

For simplicity, we assume that every s_i coincides with one of the time points t_k of our variable step algorithm, where $t_k = \sum_{j=0}^{k-1} \delta_j$. This is easily enforced, since the δ_k, and therefore t_k, can be chosen arbitrarily. For a given time point t_k, let I_k be the index i of the last switch in dynamics that comes before t_k:

$$I_k = \underset{i}{\text{argmax}}\{s_i \mid s_i \leq t_k\}.$$

Let $\Omega_{0,\delta}(A, U, X)$ and $V_{0,\delta}(A, U)$ be functions of δ as defined in (17). Let $\Psi_{0,t}(A, U)$ be defined such that $\mathcal{Y}_{0,t}(A, U) \subseteq \Psi_{0,t}(A, U)$. With Lemma 3.9 we know that $\Psi_{0,t}(A, U)$ can be computed by choosing an arbitray increasing sequence $p_0 = 0, p_1, p_2, \ldots, p_m = t$ and

$$\Psi_{0,p_{k+1}}(A, U) \quad = \quad \Psi_{0,p_k}(A, U) \oplus e^{Ap_k}V_{0,p_{k+1}-p_k}(A, U). \tag{26}$$

We propose the following variable time step, variable dynamic algorithm. On the basis, it proceeds like the variable time step algorithm. When a switch in dynamics occurs, say at time s_i, the algorithm restarts with a new initial set of states Ξ_i. This new set is the set of states reachable at exactly time s_i. Note that this is quite different from restarting the algorithm

with some Ω, since Ω overapproximates the reachable states over a time interval. While Ω is a convex set that overapproximates a nonconvex set, the reachable states at exactly time s_i is always a convex set. If $U_t = 0$, Ξ_{s_i} can even be computed exactly. In addition, we are free to vary the time step at any stage in the algorithm.

Let Φ_i, $\hat{\Psi}_{0,s_k}$, $\Xi_{s_{i+1}}$ and $\Omega_{t_k,t_{k+1}}$ be defined by

$$\begin{aligned} \Phi_{i+1} &= e^{A_i(s_{i+1}-s_i)}\Phi_i, \\ \hat{\Psi}_{0,s_{i+1}} &= e^{A_i(s_{i+1}-s_i)}\hat{\Psi}_{0,s_i} \oplus \Psi_{0,s_{i+1}-s_i}(A_i, U_i), \\ \Xi_i &= \Phi_i X_0 \oplus \hat{\Psi}_{0,s_i}, \\ \Omega_{t_k,t_{k+1}} &= e^{A_{I_k}(t_k-s_{I_k})}\Omega_{0,\delta_k}(A_{I_k}, U_{I_k}, \Xi_{I_k}) \oplus \Psi_{0,t_k-s_{I_k}}(A_{I_k}, U_{I_k}), \end{aligned} \tag{27}$$

with $\Phi_0 = I$, $\hat{\Psi}_{0,0} = 0$. Intuitively, we can interpret the sequence (27) as follows. It constructs an overapproximation of the reachable states as a sequence of sets $\Omega_{t_k,t_{k+1}}$, each covering the reachable states over the time interval $[t_k, t_{k+1}]$. $\Omega_{t_k,t_{k+1}}$ is obtained by superimposing two components. Both are defined with reference to the last switching time, s_{I_k}. The first term is obtained from Ξ_{I_k}, which are the reachable states at the last switch, by letting time elapse over an interval of $\delta_k = t_{k+1} - t_k$ time units, and finally mapping the result according to the autonomous dynamics (a matrix exponential). The second term is the input convolution since the last switch, i.e., after $t_k - s_{I_k}$ time units. Note that the next switch in dynamics takes place when $t_k = s_{i+1}$, so the term $\Psi_{0,t_k-s_{I_k}} = \Psi_{0,s_{i+1}-s_{I_k}}$ can be used to compute the next element of the sequence $\hat{\Psi}_{0,s_{i+1}}$. Similarly, $e^{A_{I_k}(t_k-s_{I_k})} = e^{A_{I_k}(s_{i+1}-s_{I_k})}$ constitutes the next element of the sequence Φ_{i+1}.

We now show that the sequence $\Omega_{t_k,t_{k+1}}$ in (27) indeed overapproximates the reachable states. We start by showing that $\hat{\Psi}_{0,s_{i+1}}$ is an overapproximation of the input convolution.

Lemma 3.11 $\mathcal{Y}_{0,s_i}(A_t, U_t) \subseteq \hat{\Psi}_{0,s_i}$.

Proof 7 *With Lemma 3.6 and Lemma 3.5,*

$$\mathcal{Y}_{0,s_{i+1}}(A_t, U_t) = e^{A_i(s_{i+1}-s_i)}\mathcal{Y}_{0,s_i}(A_t, U_t) \oplus \mathcal{Y}_{0,s_{i+1}-s_i}(A_i, U_i). \tag{28}$$

The conclusion follows by induction with the base case $\mathcal{Y}_{0,0} = 0 = \hat{\Psi}_{0,0}$.

With the above approximation of the input convolution, we can show that Ξ_i is an approximation of the reachable set of states at the time point s_i. Recall that $\mathrm{Reach}_{t,t}$ denotes the states reachable at exactly time t.

Lemma 3.12 $\mathrm{Reach}_{s_i,s_i}(A_t, U_t, X_0) \subseteq \Xi_i$.

Proof 8 *It is easy to show with induction that*

$$\text{Reach}_{s_{i+1},s_{i+1}}(A_t, 0, X_0) = e^{A_i(s_{i+1}-s_i)}\text{Reach}_{s_i,s_i}(A_t, 0, X_0) = \Phi_{i+1}X_0. \tag{29}$$

Applying superposition followed by Lemma 3.11 we get

$$\begin{aligned}\text{Reach}_{s_{i+1},s_{i+1}}(A_t, U_t, X_0) &= \text{Reach}_{s_{i+1},s_{i+1}}(A_t, 0, X_0) \oplus \text{Reach}_{s_{i+1},s_{i+1}}(A_t, U_t, 0)\\ &= \Phi_{i+1}X_0 \oplus \mathcal{Y}_{0,s_{i+1}}(A_t, U_t)\\ &\subseteq \Phi_{i+1}X \oplus \hat{\Psi}_{0,s_{i+1}} = \Xi_{i+1}\end{aligned}$$

Having approximations for the reachable states at the time points s_i, we can use the split and shift properties to obtain approximations over the time intervals between those points. We require two more auxiliary properties before we can show correctness of (27). The first auxiliary property is a simple consequence of the so-called semi-group property of the reachable set: We can exchange the durations of two nested time elapse operations. Note that Reach_{0,t_1} describes states reachable over an interval of time and Reach_{t_2,t_2} the states reachable at one time point t_2.

Lemma 3.13 $\text{Reach}_{0,t_1}(A, U, \text{Reach}_{t_2,t_2}(A, U, X)) = \text{Reach}_{t_2,t_2}(A, U, \text{Reach}_{0,t_1}(A, U, X))$.

The second auxiliary property allows us to split the time elapse operator into two parts that are easier to compute. It assumes that the dynamics are constant in the time interval $[t_1, t_3]$, but makes no assumptions about the dynamics before t_1.

Lemma 3.14 (split) *Given $t_1 \le t_2 \le t_3$, with $A_t = A$ for $t_1 \le t < t_3$,*

$$\begin{aligned}\text{Reach}_{t_2,t_3}(A_t, U_t, X) = e^{A(t_2-t_1)}\text{Reach}_{0,t_3-t_2}(A, U, \text{Reach}_{t_1,t_1}(A_t, U_t, X))\\ \oplus \mathcal{Y}_{0,t_2-t_1}(A, U).\end{aligned} \tag{30}$$

Proof 9 *To save space, we shall denote* Reach *with R in this proof.*

$$\begin{aligned}R_{t_2,t_3}(A_t, U_t, X) &= R_{0,t_3-t_2}(A_{t+t_2}, U_{t+t_2}, R_{t_2,t_2}(A_t, U_t, X))\\ &= R_{0,t_3-t_2}(A, U, R_{t_2,t_2}(A_t, U_t, X)).\end{aligned}$$

$$\begin{aligned}R_{t_2,t_2}(A_t, U_t, X) &= R_{t_2,t_2}(A_t, 0, X) \oplus \mathcal{Y}_{0,t_2}(A_t, U_t)\\ &= e^{A(t_2-t_1)}R_{t_1,t_1}(A_t, 0, X) \oplus e^{A(t_2-t_1)}\mathcal{Y}_{0,t_1}(A_t, U_t) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U)\\ &= e^{A(t_2-t_1)}(R_{t_1,t_1}(A_t, 0, X) \oplus \mathcal{Y}_{0,t_1}(A_t, U_t)) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U))\\ &= e^{A(t_2-t_1)}R_{t_1,t_1}(A_t, U_t, X) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U).\end{aligned}$$

Inserting this into the first equation yields

$$\begin{aligned} R_{t_2,t_3}(A_t, U_t, X) &= R_{0,t_3-t_2}(A, U, e^{A(t_2-t_1)} R_{t_1,t_1}(A_t, U_t, X) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U)) \\ &= e^{A(t_2-t_1)} R_{0,t_3-t_2}(A, 0, R_{t_1,t_1}(A_t, U_t, X)) \oplus R_{0,t_3-t_2}(A, U, \mathcal{Y}_{0,t_2-t_1}(A, U)) \end{aligned}$$

The last term involves constant A and U, so we can apply Lemma 3.13 to it, then expand again using superposition:

$$\begin{aligned} R_{0,t_3-t_2}(A, U, \mathcal{Y}_{0,t_2-t_1}(A, U)) &= R_{0,t_3-t_2}(A, U, R_{t_2-t_1,t_2-t_1}(A, U, 0)) \\ &= R_{t_2-t_1,t_2-t_1}(A, U, R_{0,t_3-t_2}(A, U, 0)) \\ &= R_{t_2-t_1,t_2-t_1}(A, 0, R_{0,t_3-t_2}(A, U, 0)) \oplus R_{t_2-t_1,t_2-t_1}(A, U, 0) \\ &= e^{A(t_2-t_1)} R_{0,t_3-t_2}(A, U, 0)) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U, 0). \end{aligned}$$

Inserting this in the previous equation allows us to collect the terms involving $e^{A(t_2-t_1)}$, and then combine them using superposition:

$$\begin{aligned} R_{t_2,t_3}(A_t, U_t, X) &= e^{A(t_2-t_1)} R_{0,t_3-t_2}(A, 0, R_{t_1,t_1}(A_t, U_t, X)) \\ &\quad \oplus e^{A(t_2-t_1)} R_{0,t_3-t_2}(A, U, 0)) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U, 0) \\ &= e^{A(t_2-t_1)} (R_{0,t_3-t_2}(A, 0, R_{t_1,t_1}(A_t, U_t, X)) \oplus R_{0,t_3-t_2}(A, U, 0)) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U, 0) \\ &= e^{A(t_2-t_1)} (R_{0,t_3-t_2}(A, U, R_{t_1,t_1}(A_t, U_t, X)) \oplus \mathcal{Y}_{0,t_2-t_1}(A, U, 0). \end{aligned}$$

We now have all the tools to show the main result of this section, which is that the sequence $\Omega_{t_k,t_{k+1}}$ in (27) indeed overapproximates the reachable states.

Proposition 3.15 *The sequence $\Omega_k := \Omega_{t_k,t_{k+1}}$ defined by (27) satisfies (12).*

Proof 10 *According to Lemma 3.14,*

$$\begin{aligned} \mathrm{Reach}_{t_k,t_{k+1}}(A_t, U_t, X_0) = e^{A_{I_k}(t_k-s_{I_k})} \mathrm{Reach}_{0,\delta_k}(A_{I_k}, U_{I_k}, \mathrm{Reach}_{s_{I_k},s_{I_k}}(A_t, U_t, X_0)) \\ \oplus \mathcal{Y}_{0,t_k-s_{I_k}}(A_{I_k}, U_{I_k}). \end{aligned}$$

With Lemma 27, Lemma 3.9, and (17) we get

$$\mathrm{Reach}_{t_k,t_{k+1}}(A_t, U_t, X_0) \subseteq e^{A_{I_k}(t_k-s_{I_k})} \Omega_{0,\delta_k}(A_{I_k}, U_{I_k}, \Xi_{I_k}) \oplus \Psi_{0,t_k-s_{I_k}}(A_{I_k}, U_{I_k}).$$

The conclusion follows by induction with Lemma 3.8.

The downside of the algorithm in (27) is that at every iteration, the input convolution has to be recomputed starting from zero up to the last switching time. This cost is much larger than that of the variable time step algorithm. An efficient approximation of the computation of the input convolution is necessary. The propagation of the initial set with autonomous dynamics, $\Phi_i X_0$, can be computed efficiently and does not need to be approximated. It suffices therefore to approximate $\hat{\Psi}_{0,s_i}$. Note that, if one accepts the resulting error, the input accumulation can be computed in a single step for the interval between each switching time. A sliding window calculation might be a good compromise. In a stable system, the influence of the old inputs decreases with time.

In the next section we will outline how to efficiently compute a flowpipe cover using support functions.

3.4 Flowpipe Approximations using Support Functions

We extend the results from Sect. 2.2 to flowpipes by applying them pointwise over time. For affine dynamics, $\mathcal{X}_t$ is convex for any given t, so $\mathcal{X}_t$ can be represented by its support function. The entire flowpipe can be described by a support function that is parameterized over time. Formally, let the support function over time be $s_\ell(t) = \rho_{\mathcal{X}_t}(\ell)$. As a straightforward consequence of (3), the functions $s_\ell(t)$ describe the flowpipe exactly, i.e.,

$$\mathcal{X}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \bigcap_{\ell \in R^n} \{\ell^{\mathsf{T}} x \le s_\ell(t)\}.$$

We now describe how approximations of $s_\ell(t)$ can be used to derive an approximation of $\mathcal{X}_{t_b,t_e}$. Given an interval $[t_b, t_e]$, a direction ℓ, and an accuracy bound $\varepsilon > 0$, we construct a piecewise linear function $s^+_{\ell,\varepsilon} : [t_b, t_e] \to R$ with

$$s^+_{\ell,\varepsilon}(t) - \varepsilon \le \rho_{\mathcal{X}_t}(\ell) \le s^+_{\ell,\varepsilon}(t) \quad \text{for all} \quad t \in [t_b, t_e].$$

A method to effectively compute $s^+_{\ell,\varepsilon}(t)$ is described in [10]. Briefly summarized, the support function is computed at discrete time points. The continuous bound is then obtained from a linear interpolation between the discrete points, augmenting (bloating) it enough to be conservative between the sampling points. The details of the construction are omitted since they are somewhat technical and the remainder of this paper applies to any $s^+_{\ell,\varepsilon}(t)$ as long as it is piecewise linear.

Assume we have computed $s^+_{\ell_i,\varepsilon}(t)$ for a set of directions $L = \{\ell_1, \ldots, \ell_m\}$. These functions define a flowpipe approximation as follows. For all t, the outer approximation

$$\Omega_t = \lceil \mathcal{X}_t \rceil^+_L = \bigcap_{\ell_i \in L} \{\ell_i^{\mathsf{T}} x \le s^+_{\ell_i,\varepsilon}(t)\} \tag{31}$$

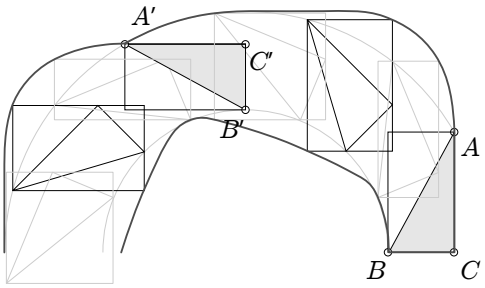

Figure 3: A flowpipe approximation constructed with the axis directions as templates, starting from the initial set $\mathcal{X}_0 = ABC$. At time t', the approximation is the bounding box of the reachable set $\mathcal{X}_{t'} = A'B'C'$

satisfies $\mathcal{X}_t \subseteq \Omega_t$. Taking the union over the time interval $[t_b, t_e]$, an outer approximation of the flowpipe segment is

$$\Omega_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \Omega_t, \text{ with } \mathcal{X}_{t_b,t_e} \subseteq \Omega_{t_b,t_e}.$$

This flowpipe approximation is defined as the union of infinitely many convex polyhedra, see Fig. 3. But for our reachability algorithm the flowpipe approximation needs to produce finitely many convex sets. To transform this infinite union into a finite one, we consider piecewise linear approximations $s^+_{\ell_i,\varepsilon}(t)$. In a time interval where these are linear for all directions ℓ_i, say $s^+_{\ell_i,\varepsilon}(t) = \alpha_i t + \beta_i$, we have

$$\Omega_t = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x \le s^+_{\ell_i,\varepsilon}(t) \right\} = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x \le \alpha_i t + \beta_i \right\} = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \alpha_i t \le \beta_i \right\}. \tag{32}$$

The latter term is a convex polyhedron in x and t. Their union over the time interval $[t_b, t_e]$ is

$$\Omega_{t_b,t_e} = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \alpha_i t \le \beta_i \right\} \cap \{ t_b \le t \le t_e \}. \tag{33}$$

If $s^+_{\ell_i,\varepsilon}(t)$ is piecewise linear concave over $[t_b, t_e]$, then its linear bounds are of the form

$$\bigwedge_{j \in J_i} s^+_{\ell_i,\varepsilon}(t) \le \alpha_i^j t + \beta_i^j,$$

and we also obtain a convex polyhedron

$$\Omega_{t_b,t_e} = \bigcap_{\ell_i \in L, j \in J_i} \left\{ \ell_i^\mathsf{T} x - \alpha_i^j t \le \beta_i^j \right\} \cap \{ t_b \le t \le t_e \}. \tag{34}$$

Support functions can be used to approximate the sequence $\Omega_{t_k,t_{k+1}}$ in (27) arbitrarily well, but there is a catch: Each evaluation of the support function of $\Omega_{t_k,t_{k+1}}$ requires to evaluate the support function of $\hat{\Psi}_{0,s_{I_k}}$, which approximates the input convolution at the last time the dynamics switched before t_k. Evaluating the support function of $\hat{\Psi}_{0,s_{I_k}}$ itself triggers

a sequence of support function computations, which means that the evaluation cost would grow linearly with k. The overall complexity of the algorithm with K time steps would be $\mathcal{O}\left(K^2\right)$, which is problematic since in general K can become arbitrarily large. In the next section we will present a solution that avoids this problem. The idea is to approximate $\hat{\Psi}_{0,s_{I_k}}$ by a zonotope.

4 Complexity Assessment for Flowpipe Approximation

In this section, we compare different set representations, i.e., zonotopes, polyhedra, and support functions, in terms of the runtime cost for approximating flowpipes. Our analysis goes in more detail than the usual $\mathcal{O}(\cdot)$ notation to be sure that no opportunity for constant speed-up is missed. The results will then be used to optimally combine set representations in the following section.

4.1 Approximating the Input Convolution

We consider dynamics of the form (5), i.e., $\dot{x} = Ax + u$, with $u(t) \in \mathcal{U}$ and $x(0) \in \mathcal{X}_0$. At a time instant t, the states reachable with theses dynamics are

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \int_0^t e^{As}\mathcal{U}ds.$$

The problematic term is the input convolution

$$\mathcal{Y}_t = \int_0^t e^{As}\mathcal{U}ds = \lim_{\delta\to 0} \bigoplus_{k=0}^{\lfloor t/\delta \rfloor} e^{A\delta k}\delta\mathcal{U}.$$

The matrix $e^{A\delta k}$ is different for each k, so the Minkowski sum leads to infinitely many vertices and constraints as $\delta \to 0$. Even if $\mathcal{U}$ is a polyhedron, $\mathcal{Y}_t$ can be a smooth set [25]. An outer approximation of $\mathcal{Y}_t$ is obtained by using a finite sum and bloating $\delta\mathcal{U}$ by a sufficient amount. Let $\|\cdot\|_p$ be a norm, $\mathcal{B}_p$ the unit ball of the norm, and $t = N\delta$. We start with a simple version of an error compensation term. More precise compensation terms will be discussed below. An error compensation term

$$\mathcal{E}_\delta = \beta_\delta \|\mathcal{U}\|\mathcal{B}_p \tag{35}$$

with bloating factor

$$\beta_\delta = \sum_{i=1}^{\infty} \frac{\|A\|_p^i \delta^{i+1}}{(i+1)!} = \frac{e^{\|A\|_p\delta} - 1 - \|A\|_p\delta}{\|A\|_p}, \tag{36}$$

leads to an outer approximation of $\mathcal{Y}_t$ [19]

$$\hat{\mathcal{Y}}_t^\delta = \bigoplus_{k=0}^{N-1} e^{A\delta k}\left(\delta\mathcal{U} \oplus \mathcal{E}_\delta\right). \tag{37}$$

Other approximations of $\mathcal{Y}_t$ have been proposed, e.g., in [?, ?]. The approximation in (37) is within distance $\epsilon_\delta = \delta e^{\|A\|_p(t+\delta)}\|\mathcal{U}\|$ according to [19], but this is a very conservative estimate. To illustrate this, consider that it results in the same bound for a stable matrix A as for the unstable matrix $|A|$. However, we can explicitly represent and measure the error by splitting the sequence into two terms

$$\hat{\mathcal{Y}}_t^\delta = \bar{\mathcal{Y}}_t^\delta \oplus \mathcal{E}_t^\delta, \tag{38}$$

with

$$\bar{\mathcal{Y}}_t^\delta = \bigoplus_{k=0}^{N-1} e^{A\delta k}\delta\mathcal{U}, \qquad \mathcal{E}_t^\delta = \bigoplus_{k=0}^{N-1} e^{A\delta k}\mathcal{E}_\delta. \tag{39}$$

The first term, $\bar{\mathcal{Y}}_t^\delta$, is an underapproximation of $\mathcal{Y}_t$, since it corresponds to keeping the input u piecewise constant over intervals of duration δ. The second term, $\mathcal{E}_t^\delta$, compensates such that arbitrary switching of u within the intervals is covered as well.

Lemma 4.1 *Conservative over- and underapproximations of $\mathcal{Y}_t$ are given by $\bar{\mathcal{Y}}_t^\delta \subseteq \mathcal{Y}_t \subseteq \hat{\mathcal{Y}}_t^\delta$.*

The diameter of $\mathcal{E}_t^\delta$ bounds the approximation error.

Lemma 4.2 *The Hausdorff distance between the input convolution and its approximation is*

$$d_H(\mathcal{Y}_t^\delta, \mathcal{Y}_t) \leq d_H(\bar{\mathcal{Y}}_t^\delta, \mathcal{Y}_t) \leq \|\mathcal{E}_t^\delta\|_2.$$

4.1.1 Comparison of Different Set Representations

In the following, we discuss different ways to compute the approximation in (37).

Using Zonotopes Since zonotopes are closed under linear map and Minkowski sum, and both can be computed very efficiently, they are well suited for computing and representing $\hat{\mathcal{Y}}_t^\delta$. In many case studies, both $\mathcal{U}$ and $\mathcal{E}_\delta$ are indeed zonotopes, e.g., hyperboxes ($\mathcal{B}_p = \mathcal{B}_\infty$). If $\mathcal{E}_\delta$ is not a zonotope, it can be overapproximated with one without loss of completeness, since δ can be reduced to compensate for the difference. We briefly show the reduction. For any p-norm and q-norm with $q > p$,

$$\|\mathcal{X}\|_q \leq \|\mathcal{X}\|_p \leq n^{\left(\frac{1}{p}-\frac{1}{q}\right)}\|\mathcal{X}\|_q.$$

As stated above, $\hat{\mathcal{Y}}_t^\delta$ approximates $\mathcal{Y}_t$ within distance ϵ_δ in the p-norm. Using the ∞-norm, $\mathcal{B}_\infty$ has n generators. The error bound is ϵ_δ in the ∞-norm and $\epsilon_\delta n^{\frac{1}{p}}$ in any p-norm. To meet the desired error bound in any p-norm, it therefore suffices to use a smaller time step δ' such that $\beta_{\delta'} n^{\frac{1}{p}} \leq \beta_\delta$. Since β_δ is superlinear in δ, $\delta' \geq n^{-\frac{1}{p}}\delta$. [1]

[1] A different route would be to approximate $\mathcal{E}_\delta$ with a zonotope up to a given error bound ϵ_z. The final error would then be $\epsilon_\delta + \epsilon_z$. However, a zonotope approximating $\mathcal{E}_\delta$ with error ϵ_z can require $\mathcal{O}\left(c(n)\epsilon_z^{-2\frac{n-1}{n+2}}\right)$ generators. [5]

Since $\mathcal{E}_t^\delta$ is a zonotope that is computed anyway to obtain $\hat{\mathcal{Y}}_t^\delta$, we can use it to measure the approximation error ϵ_δ since

$$d_H(\bar{\mathcal{Y}}_t^\delta, \mathcal{Y}_t) \leq \|\mathcal{E}_t^\delta\|_2 = \mathrm{rad}(\mathcal{E}_t^\delta).$$

If $\mathcal{E}_\delta$ is a zonotope with k_B generators, then computing its radius requires $n(Nk_B + 1)$ operations.

Assuming that $\mathcal{U}$ is a zonotope with k_U generators and $\mathcal{E}_\delta$ is a zonotope with k_B generators, then $\hat{\mathcal{Y}}_t^\delta$ is a zonotope with $N \cdot (k_U + k_B)$ generators. The matrices $e^{A\delta k}$ can be computed using the sequence $M_0 = I$, $M_{k+1} = e^{A\delta} M_k$ with $(N-1)\,2n^3$ operations, assuming that $e^{A\delta}$ is known. Each term of the sum can then be computed with $2n^2(k_U + k_B + 2)$ operations. The summation consists of summing the centers and joining the generator lists, i.e., Nn numerical operations. The result of this section can be summarized as follows:

Lemma 4.3 *Assuming $\mathcal{U}$ is a zonotope with k_U generators and $e^{A\delta}$ is known. Computing $\hat{\mathcal{Y}}_t^\delta$ up to an error ϵ_δ in the p-norm is possible with zonotopes using a total of*

$$Nn^{\frac{1}{2p}} \cdot \left(3n^3 + 2n^2 \cdot (k_U + 2) + n\right)$$

numerical operations.

Using Polyhedra Using polyhedra in constraint form, $\hat{\mathcal{Y}}_t^\delta$ is prohibitively expensive to compute, since the number of constraints may grow exponentially during the summation. Recall that if $\mathcal{U}$ and $\mathcal{E}_\delta$ are zonotopes, then $\hat{\mathcal{Y}}_t^\delta$ is a zonotope. The number of facets F ($n-1$-dimensional faces) of a zonotope with k generators in n dimensions is bounded by

$$F \leq 2\binom{k}{n-1} \leq 2\left(\frac{ek}{n-1}\right)^{n-1},$$

where e is Euler's constant. If $\hat{\mathcal{Y}}_t^\delta$ has $N \cdot (k_U + k_B)$ generators, then the number of facets is bounded by

$$\binom{N \cdot (k_U + k_B)}{n-1} \leq 2\left(\frac{eN \cdot (k_U + k_B)}{n-1}\right)^{n-1}.$$

If the desired error bound ϵ_δ is given, (36) provides an upper bound on δ, for which the above inequality gives an upper bound on the number of facets. The number of facets grows quickly with $N = t/\delta$, and prohibitively so in higher dimensions (large n).

Using Support Functions The support function of $\hat{\mathcal{Y}}_t^\delta$ using the bloating factor compensation in (35) is

$$\rho_{\hat{\mathcal{Y}}_t^\delta}(\ell) = \sum_{k=0}^{N-1} \delta \rho_{\mathcal{U}}(e^{A\delta k^\mathsf{T}}\ell) + \beta_\delta \|e^{A\delta k^\mathsf{T}}\ell\|_{\frac{p}{p-1}}. \tag{40}$$

The directions $e^{A\delta k^{\mathsf{T}}}\ell$ can be computed with the sequence $\ell_0 = \ell$, $\ell_{k+1} = e^{A\delta^{\mathsf{T}}}\ell_k$, up to $k = N-1$. This requires $2n^2N$ operations. Let $\mathcal{L}(\mathcal{U})$ be the number of operations for computing the support function of $\mathcal{U}$. Computing the vector norm is $\mathcal{O}(n)$. In total, computing $\rho_{\hat{\mathcal{Y}}_t^\delta}(\ell)$ involves $N \cdot (2n^2 + \mathcal{L}(\mathcal{U}) + n)$ operations [18].

If $\mathcal{U}$ is a zonotope with k_U generators, then $\mathcal{L}(\mathcal{U}) = 2n(k_U + 1)$, so computing the support of $\hat{\mathcal{Y}}_t^\delta$ involves

$$N \cdot (2n^2 + 2n(k_U + 2)) \tag{41}$$

operations. We now assume the infinity norm is used to measure the error ($\mathcal{B}_\infty$ being a zonotope with n generators). Comparing this bound with the one for zonotopes, we note that the zonotope representation involves roughly n times the operations of computing the value of the support function in a single direction. Using the support function to construct a bounding box approximation requires evaluating $2n$ directions. This leads to the following observation:

Observation 1 *A zonotope overapproximation of $\mathcal{Y}_t$ with error ϵ_δ in the ∞-norm can be computed with less operations than a bounding box approximation of $\mathcal{Y}_t$ using support functions.*

Clearly the zonotope approximation is much more precise than the bounding box approximation: The bounding box approximation is only tight (up to error ϵ_δ) in the axis directions, while the zonotope approximation is tight in all directions. The Hausdorff distance (in Euclidean norm) of the bounding box approximation is $\sqrt{n}(d+2\epsilon_\delta)$, where d is the diameter of $\mathcal{Y}_t$. The Hausdorff distance of the zonotope approximation is $\sqrt{n}\epsilon_\delta$, which is independent of the size of $\mathcal{Y}_t$.

4.1.2 Alternative Compensation Terms

More precise version of the error compensation term in (37) are known, e.g., [18, 17, **?**, **?**]. We present a version proposed originally in [**?**] and used in slightly different form in [12]. We need the following notation. The *symmetric interval hull* of a set S, denoted $\boxdot(S)$, is $\boxdot(S) = [-\overline{|x_1|}; \overline{|x_1|}] \times \ldots \times [-\overline{|x_d|}; \overline{|x_d|}]$ where for all i, $\overline{|x_i|} = \sup\{|x_i| \mid x \in S\}$. Let $M = (m_{i,j})$ be a matrix, and $v = (v_i)$ a vector. We define as $|M|$ and $|v|$ the element-wise absolute values of M and v, i.e., $|M| = (|m_{i,j}|)$ and $|v| = (|v_i|)$. We use the following truncated versions of the matrix exponential:

$$\Phi_1(A, \delta) = \sum_{i=0}^{\infty} \frac{\delta^{i+1}}{(i+1)!} A^i, \qquad \Phi_2(A, \delta) = \sum_{i=0}^{\infty} \frac{\delta^{i+2}}{(i+2)!} A^i, \tag{42}$$

which are computed similarly to a matrix exponential. If A is invertible, Φ_1 and Φ_2 can be computed as

$$\Phi_1(A,\delta) = A^{-1}\left(e^{\delta A} - I\right), \qquad \Phi_2(A,\delta) = A^{-2}\left(e^{\delta A} - I - \delta A\right)$$

Otherwise, they can be computed as submatrices of the block matrix

$$\begin{pmatrix} e^{A\delta} & \Phi_1(A,\delta) & \Phi_2(A,\delta) \\ 0 & I & I\delta \\ 0 & 0 & I \end{pmatrix} = \exp\begin{pmatrix} A\delta & I\delta & 0 \\ 0 & 0 & I\delta \\ 0 & 0 & 0 \end{pmatrix}.$$

To bound the approximation error we use the error term

$$\mathcal{E}_\Psi(\mathcal{U},\delta) = \boxdot\left(\Phi_2(|A|,\delta) \boxdot (A\mathcal{U})\right).$$

We have the following bounds on $\rho_{\mathcal{Y}_\delta}(\ell)$.

Lemma 4.4 *[12]* $\delta\rho_{\mathcal{U}}(\ell) - \rho_{-A\Phi_2(A,\delta)\mathcal{U}}(\ell) \le \rho_{\mathcal{Y}_\delta}(\ell) \le \delta\rho_{\mathcal{U}}(\ell) + \rho_{\mathcal{E}_\Psi(\mathcal{U},\delta)}(\ell)$

This leads to the following approximation of $\mathcal{Y}_t^\delta$:

$$\hat{\mathcal{Y}}_t^\delta = \bigoplus_{k=0}^{N-1} e^{A\delta k}\big(\delta\mathcal{U} \oplus \mathcal{E}_\Psi(\mathcal{U},\delta)\big), \tag{43}$$

4.1.3 Adaptive Time Steps

We now consider the problem of computing $\hat{\mathcal{Y}}_t^\delta$ such that a given error bound is met. We will measure the error using the compensation term

$$\mathcal{E}_t^\delta = \bigoplus_{k=0}^{N-1} e^{A\delta k}\mathcal{E}_\delta.$$

Since $0 \in \mathcal{E}_\delta$, the sum is monotonically increasing. Each term in the sum contributes additively to the final set. We propose the following heuristic to guarantee a final error of radius less than a given error ϵ. The goal is to ensure that

$$\|\mathcal{E}_t^\delta\|_2 = \|r_{\mathcal{E}_t^\delta}\|_2 \quad \le \quad \epsilon.$$

Since the absolute row sum is simply the vector sum

$$r_{\mathcal{E}_t^\delta} = \sum\nolimits_{k=0}^{N-1} r_{e^{A\delta k}\mathcal{E}_\delta}, \tag{44}$$

we can cheaply compute its value at the k-th step. We can then compare it to a target value, and decrease the time step δ until the target value is met. We choose as target value the

linear interpolation $kt/(N-1)\epsilon$. Since the radius of the error term (35) is quadratic in δ, we can always find a δ such that $\|\mathcal{E}_\delta\| \leq \delta/t\epsilon$.

A straightforward implementation to compute $\hat{\mathcal{Y}}_t$ with adaptive time steps δ_i would be

$$\hat{\mathcal{Y}}_{t_{i+1}} = e^{A\delta_i}\hat{\mathcal{Y}}_{t_i} \oplus (\delta_i \mathcal{U} \oplus \mathcal{E}_{\delta_i}), \tag{45}$$

with $t_i = \sum_{j=0}^{i-1} \delta_i$. However, this is inefficient if the complexity of the set representation $\hat{\mathcal{Y}}_{t_i}$ is increasing with i. In the case of zonotopes, for example, $\hat{\mathcal{Y}}_{t_i}$ has $i \cdot (k_U + k_B)$ generators. If the zonotope is replaced with a simpler approximation during the computation, the approximation error propagates and possibly augmented by the linear map; this is known as the wrapping effect.

The sequence from (24) is more efficient and free of the wrapping effect [15], and is used with varying time steps in [10]:

$$\Phi_{i+1} = e^{A\delta_i}\Phi_i, \tag{46}$$

$$\hat{\mathcal{Y}}_{t_{i+1}} = \Phi_i(\delta_i \mathcal{U} \oplus \mathcal{E}_{\delta_i}) \oplus \hat{\mathcal{Y}}_{t_i}, \tag{47}$$

with $\Phi_0 = I$, $\hat{\mathcal{Y}}_{t_0} = 0$. In case of zonotopes, the number of generators of $\hat{\mathcal{Y}}_{t_i}$ is $i \cdot (k_U + k_B)$ like in (45), but here they are not subjected to linear mapping in the next iteration. If the set is approximated, the approximation error is accumulated, but not amplified by a linear map. It can therefore increase linearly, but not exponentially with the number of iterations.

Compared to the fixed-step algorithm, the only additional cost is the computation of the matrix exponential $e^{A\delta}$ for different values of δ. This cost can be minimized by scaling the time step in powers of two, i.e., $\delta = 2^j\delta_0$, $j \in \mathbb{Z}$. In conclusion, we can compute $\hat{\mathcal{Y}}_t$ using adaptive time steps without significant penalty in cost or precision.

4.2 Approximating the Autonomous Dynamics

The autonomous dynamics are

$$\dot{x} = Ax,$$

and the solution at a given time point t is

$$\mathcal{X}_t = e^{At}\mathcal{X}_0.$$

This allows us to propagate the approximation of any time interval forward in time. If $\mathcal{X}_{0,\delta} \subseteq \Omega_{0,\delta}$, then

$$\mathcal{X}_{t,t+\delta} \subseteq e^{At}\Omega_{0,\delta},$$

if there are no inputs. This linear map is reversible (e^{At} is nonsingular), so computing this image can be carried out efficiently for all set representations mentioned in this paper. The

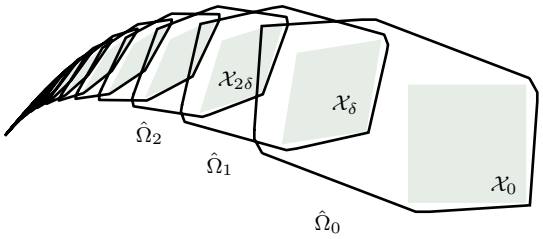

Figure 4: A reach set cover $\hat{\Omega}_0, \hat{\Omega}_1, \ldots$ computed with zonotopes using the implementation in [2] (solid)

difficulty lies in the initial set $\Omega_{0,\delta}$. All approximation models published so far in the literature use the convex hull operation, which is natural since the convex hull between two sets is the set of all points on the linear interpolation between them. Taking the convex hull of $\mathcal{X}_0$ and $\mathcal{X}_\delta = \Phi\mathcal{X}_0$ is thus equivalent to linear interpolation between the sampling times. Consider the approximation model from (18),

$$\Omega_{0,\delta}(X_0, U) = \mathrm{CH}\left(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta U \oplus \mathcal{E}_\Omega\right).$$

Using Zonotopes Zonotopes are not closed under convex hull, i.e., the convex hull of two zonotopes is not necessarily a zonotope. An overapproximation from [14] is

$$\mathrm{CH}(\mathcal{X}_0, e^{A\delta}\mathcal{X}_0) \supseteq \frac{1}{2}(c + e^{A\delta}c, \langle v_1 + e^{A\delta}v_1, \ldots, v_k + e^{A\delta}v_k, v_1 - e^{A\delta}v_1, \ldots, v_k - e^{A\delta}v_k, c - e^{A\delta}c\rangle). \tag{48}$$

Assume that $\mathcal{X}_0$ is a zonotope with k_X generators, $\mathcal{U}$ a zonotope with k_U generators and $\mathcal{E}_\Omega$ a zonotope with n generators. The overapproximation in (48) is very efficient, with $2n^2(k_X + 1) + 2n(k_X + 2)$ operations; note that it doubles the number of generators. Its error is bounded by $\sum_i \|v_i - e^{A\delta}v_i\|$. The accuracy can be improved arbitrarily by taking smaller time step δ, since $\lim_{\delta\to 0} e^{A\delta} = I$. However, this may increase the number of sets prohibitively. To combine the input convolution with the convex hull approximation (48) is slightly more involved. If $0 \in U$, we can use

$$\Omega_{0,\delta}(X_0, U) = \mathrm{CH}\left(\mathcal{X}_0, \Phi\mathcal{X}_0\right) \oplus \delta U \oplus \mathcal{E}_\Omega.$$

The number of operations to compute $\Omega_0, \ldots, \Omega_N$ is $2n^2(k_X + 1) + 2n(k_X + 4)$ for the initial set and $N \cdot 2n^2(2k_X + k_U + n + 2)$ for the propagation.

An illustration of the approximation with zonotopes is shown in Fig. 4. It uses a slightly more sophisticated approximation model than the one presented in this section.

Using Polyhedra Using polyhedra in constraint representation, both the Minkowski sum and the convex hull operation can increase the number of constraints exponentially in n. This

is one of the major obstacles in applying polyhedra to this type of reachability computation.

Using Support Functions The convex hull operation corresponds to taking the maximum of the support functions. For the approximation of the initial set we get

$$\rho_{\Omega_{0,\delta}}(\ell) = \max\left(\rho_{\mathcal{X}_0}(\ell), \rho_{\mathcal{X}_0}(\Phi^{\mathsf{T}}\ell) + \delta\rho_U(\ell) + \rho_{\mathcal{E}_\Omega}(\ell)\right). \tag{49}$$

The support function of the sequence $\Omega_k = e^{Ak\delta}\Omega_{0,\delta}$ we can compute with $\ell_0 = \ell$,

$$\ell_{k+1} = e^{A\delta^{\mathsf{T}}}\ell_k, \tag{50}$$

$$\rho_{\Omega_k}(\ell) = \max\Big(\rho_{\mathcal{X}_0}(\ell_k), \rho_{\mathcal{X}_0}(\ell_{k+1}) + \delta\rho_U(\ell_k) + \rho_{\mathcal{E}_\Omega}(\ell_k)\Big). \tag{51}$$

Note that the value of $\rho_{\mathcal{X}_0}(\ell_{k+1})$ can be reused in the computation of $\rho_{\Omega_{k+1}}(\ell)$. Each iteration therefore requires computing one value of the support of $\mathcal{X}_0$, one of $\mathcal{U}$, and one of $\mathcal{E}_\Omega$. Computing the support of $\Omega_0, \Omega_1, \ldots, \Omega_N$ in a given direction ℓ requires $N(2n^2 + \mathcal{L}(\mathcal{X}_0) + \mathcal{L}(\mathcal{U}) + n)$ operations, where $\mathcal{L}(\mathcal{X}_0)$ be the number of operations for computing the support function of $\mathcal{X}_0$. If $\mathcal{X}_0$ and $\mathcal{U}$ are zonotopes with k_X and k_U generators, then $\mathcal{L}(\mathcal{X}_0) = 2n(k_X+1)$ and $\mathcal{L}(\mathcal{U}) = 2n(k_U + 1)$. Per direction we get

$$N \cdot (2n^2 + 2n(k_X + k_U + 5/2))$$

operations, which is practically the same as for computing $\hat{\mathcal{Y}}_t^\delta$ over N time steps. Comparing the cost to zonotopes, we make similar observation as for the input convolution:

Observation 2 *A zonotope overapproximation of $\Omega_0, \Omega_1, \ldots, \Omega_N$ can be computed with roughly the same number of operations than a bounding box approximation using support functions (assuming $k_X \geq n$). The approximation quality of the zonotopes, however, depends on the quality of the zonotope approximation of $\mathcal{X}_0$, its generators, and the time step.*

While for the input convolution it seem that from a practical point of view, $\mathcal{U}$ can be approximated reasonably well be a zonotope, this need not be the case for the autonomous dynamics. In the case of hybrid systems, $\mathcal{X}_0$ may be the result of an intersection operation. Hence, the approximation quality may be limited and the initial error may be as large as the radius of the initial set.

An illustration of the approximation with support functions with varying degrees of precision is shown in Fig. 4.

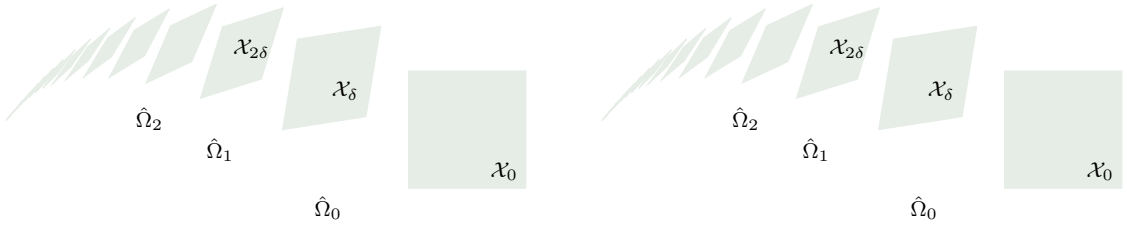

Figure 5: A reach set cover $\hat{\Omega}_0, \hat{\Omega}_1, \ldots$ computed with support functions using the implementation in [12] (solid), evaluating the support function in the axis directions (left) and in 256 uniformly distributed directions (right)

4.3 Approximating the Nonautonomous Dynamics

We now turn to the general case, where both $\mathcal{X}_0$ and $\mathcal{U}$ may be nonempty. According to the superposition principle, the reachable set approximation is the Minkowski sum of the autonomous approximation Ω_i and the input convolution,

$$\hat{\Omega}_{t_k,t_{k+1}} = e^{At_k}\Omega_{0,\delta_k} \oplus \hat{\mathcal{Y}}_{t_k}.$$

Using Zonotopes Since the Minkowski sum essentially amounts to joining the generators, computing the sequence $\hat{\Omega}_{t_k,t_{k+1}}$ poses no problem if both Ω_{0,δ_k} and $\hat{\mathcal{Y}}_{t_k}$ are zonotopes.

Using Polyhedra Using polyhedra in constraint representation, the Minkowski sum can increase the number of constraints exponentially in n. This can be seen from the fact that each non-redundant constraint of a polyhedron defines a facet, i.e., an $n-1$ dimensional face. The facets of $P \oplus Q$ are generated by combinations of a-dimensional faces of P and b-dimensional faces of Q such that $a + b = n - 1$. Let $f_P(i)$ be the number of i-dimensional faces of P. If P has m_P linear inequalities, an upper bound on $f_P(i)$ is $\binom{m_P}{n-i}$. There may be as many as $\sum_{i=0}^{n-1} f_P(i) f_Q(n-1-i)$ facets in $P \oplus Q$.

Using Support Functions Since the Minkowski sum reduces to the scalar sum, computing the support function of the sequence $\hat{\Omega}_{t_k,t_{k+1}}$ is efficient for all previously mentioned set representations of Ω_{0,δ_k} and $\hat{\mathcal{Y}}_{t_k}$.

5 Combining Support Functions and Zonotopes

In this section we use the observations of the previous sections to devise algorithms that combines support functions and zonotopes. In Sect. 4.1, we saw that zonotopes are efficient

for computing the input convolution. This lead to an ϵ-close approximation for the cost of a bounding box approximation using support functions, which is much coarser. In many case studies, the inputs are indeed zonotopes. In Sect. 4.2, we saw that zonotopes are similarly efficient, but that the approximation quality depends very much on the time step, and whether the initial set is a zonotope. Contrary to the input set, the initial set is frequently not a zonotope, in particular if applied to hybrid systems.

Our conclusion is to use support functions for the autonomous part, zonotopes for the input convolution, and combining the two. In the following, we will examine the costs and benefits compared to the conventional support function approach.

5.1 Flowpipe Approximation Using Support Functions

In the following we show that zonotopes can be used to compute, at equal cost, a support function approximation with higher precision than using support functions directly.

According to Lemma 4.3, the construction a zonotope approximation $Z_{\mathcal{Y}} = \hat{\mathcal{Y}}_{t_N}$ is possible with

$$N \cdot \left(3n^3 + 2n^2 \cdot (k_U + 2) + n\right)$$

operations, with the approximation error being explicitly represented by a subset of the generators. The zonotope has $N(k_U + n)$ generators, where k_U is the number of generators of the input set $\mathcal{U}$. Recall that the support function of a zonotope with center c and generators $v_1, \ldots, v_{k_P}$ is

$$\rho_P(\ell) = c^{\mathsf{T}}\ell + \sum_{i=1}^{k_P} \left| v_i^{\mathsf{T}}\ell \right| .$$

Computing the support of $Z_{\mathcal{Y}}$ therefore requires $2n + 2N(n^2 + nk_U)$ operations. The total cost of computing first $Z_{\mathcal{Y}}$ and then the support in L directions is

$$LN \cdot \left(2n^2 + 2nk_U + \frac{3n^3 + 2n^2 \cdot (k_U + 2) + n}{L}\right) + Ln$$

If $L \geq 2n$, $N \geq n$, and $n \geq 2$, a simplified upper bound is

$$LN \cdot \left(\tfrac{7}{2}n^2 + 3nk_U + 3n\right).$$

According to (41), computing the L values of the support function of $\hat{\mathcal{Y}}_{t_k}$ directly requires

$$LN \cdot (2n^2 + 2nk_U + 4n)$$

operations. For $L \geq 2n$, i.e., at least bounding box directions, using the zonotope uses at most $2\times$ the operations; and as $L \to \infty$, using the zonotope is marginally faster. The downside of using the zonotope is its memory consumption, since it requires to store $1 + N(n + k_U)$

vectors with n elements each. Recall that zonotope and direct support function methods incur the same error if the infinity norm is used.

The operation count above applies to computing the support of $\hat{\mathcal{Y}}_{t_N}$, but covers also intermediate results that are sufficient to compute the support of $\hat{\mathcal{Y}}_{t_0}, \ldots, \hat{\mathcal{Y}}_{t_{N-1}}$. Reducing the number of generators on the fly can reduce the operation count, for the price of an increased approximation error. Let the number of generators be k_R, then the operation count (without the reduction cost) is

$$LN \cdot \left(\frac{2n(k_R+1)}{N} + \frac{3n^3 + 2n^2 \cdot (k_U+2) + n}{L} \right).$$

Note that for $k_R \geq n$, the approximation error can be made ϵ_δ-tight in the axis directions, i.e., no worse than a bounding box. The direct support function computation is $\mathcal{O}\left(LNn^2\right)$, while the zonotope construction with $k_R = c \cdot n$ generators is $\mathcal{O}\left(Lcn^2 + Nn^3\right)$.

Observation 3 *The zonotope construction amortizes with increasing L when $L \geq 2n$. Reducing the number of generators to $k_R = c \cdot n$, the zonotope allows the evaluation of $\frac{N}{c+1}$ times the directions at cost equal to directly computing the support function.*

The combination of the autonomous part (support function) and input convolution (zonotope) is straightforward: it suffices to compute the support of the autonomous part and the zonotope and then add the two values:

$$\rho_{\hat{\Omega}_{t_k,t_{k+1}}}(\ell) = \rho_{\Omega_{0,\delta_k}}(e^{At_k}{}^{\mathsf{T}}\ell) + \rho_{\hat{\mathcal{Y}}_{t_k}}(\ell).$$

Observation 3 indicates that this enables, for a given cost, a much higher precision, which is governed by the ratio between the number of steps N and the degree c (generators divided by n) of the simplified zonotope .

Potential Applications In the context of a support function, it is not trivial how the above increase in precision manifests itself within practical bounds on N and L. The straightforward way would be to approximate $\hat{\Omega}_{t_k,t_{k+1}}$ with its outer polytope in many directions. Algorithms to do this with an optimal number of directions are known, but the number is always high at $\mathcal{O}\left(1/\varepsilon^{n-1}\right)$ for an error of ε [20]. So while the use of zonotopes would be beneficial relative to support functions, any such procedure would not scale to higher dimensions. A more promising use case is the intersection with another set. Recall that the support function of

the intersection of $\hat{\Omega}_{t_k,t_{k+1}}$ with, say, a guard set $\mathcal{G}$ is

$$\rho_{\hat{\Omega}_{t_k,t_{k+1}}\cap\mathcal{G}}(\ell) = \inf_{\nu\in R^n} \rho_{\hat{\Omega}_{t_k,t_{k+1}}}(\ell-\nu) + \rho_{\mathcal{G}}(\nu).$$

A similar problem is to decide whether $\hat{\Omega}_{t_k,t_{k+1}}$ and $\mathcal{G}$ intersect at all, which is the case iff

$$\inf_{\nu\in R^n} \rho_{\hat{\Omega}_{t_k,t_{k+1}}}(-\nu) + \rho_{\mathcal{G}}(\nu) \quad \geq 0.$$

In both cases, the support function of $\hat{\Omega}_{t_k,t_{k+1}}$ doesn't necessarily need to be evaluated in many directions, and in particular not exponentially many. Indeed, our experiments indicate that the number of directions can in practice be low polynomial in n [13, 9].

So far the difference between support functions and zonotopes haven been quantitative rather than qualitative. An application where the difference is more dramatic is discussed in the next section.

5.2 Time-triggered Switching

We now turn the case of switching dynamics, where the switch is triggered at specific points in time. This is a particular case of state-based triggering (one could include time as a state variable), which simplifies reachability computation since only points, not intervals, of time need to be taken into account. For piecewise affine dynamics, this means that the set of states that takes the switch is convex, which is not the case in general.

We use the algorithm for switched dynamics from (27), and discuss its complexity for different set representations. For the input convolution, we use the approximation $\hat{\mathcal{Y}}_t(A_i,U_i)$, from Sect. 4.1. The algorithm computes the states $\Omega_{t_k,t_{k+1}}$ with time points $t_k = \sum_{i=0}^{k-1}\delta_i$, defined by arbitrary time steps $\delta_0,\delta_1,\ldots$. The dynamics A_i,$\mathcal{U}_i$ change at subset of t_k that we denote by s_i. Let the duration between switches be $\Delta_i = s_{i+1} - s_i$. For a given t_k, the dynamics are A_{I_k},$\mathcal{U}_{I_k}$, and they are active since the last switch at time s_{I_k}. Let $\hat{\Phi}_i$, $\hat{\Psi}_{s_k}$, $\Xi_{s_{i+1}}$ and $\Omega_{t_k,t_{k+1}}$ be defined by $\hat{\Phi}_0 = I$, $\hat{\Psi}_0 = 0$,

$$\hat{\Phi}_{i+1} = e^{A_i\Delta_i}\hat{\Phi}_i, \tag{52}$$

$$\hat{\Psi}_{s_{i+1}} = e^{A_i\Delta_i}\hat{\Psi}_{s_i} \oplus \hat{\mathcal{Y}}_{\Delta_i}(A_i,U_i), \tag{53}$$

$$\Xi_i = \hat{\Phi}_i\mathcal{X}_0 \oplus \hat{\Psi}_{s_i}, \tag{54}$$

$$\Omega_{t_k,t_{k+1}} = e^{A_{I_k}(t_k-s_{I_k})}\Omega_{0,\delta_k}(A_{I_k},U_{I_k},\Xi_{I_k}) \oplus \hat{\mathcal{Y}}_{t_k-s_{I_k}}(A_{I_k},U_{I_k}), \tag{55}$$

To simplify the analysis, we now assume that the sets $\hat{\mathcal{Y}}_{\Delta_i}(A_i,U_i)$ are computed with N steps each.

Using Support Functions Directly Using the support function, we have

$$\rho_{\Omega_{t_k,t_{k+1}}}(\ell) = \rho_{\Omega_{0,\delta_k}(A_{I_k},U_{I_k},\Xi_{I_k})}(e^{A_{I_k}(t_k-s_{I_k})^\mathsf{T}}\ell) + \rho_{\hat{\mathcal{Y}}_{t_k-s_{I_k}}(A_{I_k},U_{I_k})}(\ell).$$

Using formula (49) for Ω_{0,δ_k} requires two evaluations of $\rho_{\Xi_{I_k}}(\cdot)$ at each time step. We have

$$\rho_{\Xi_{I_k}}(\ell) = \rho_{\mathcal{X}_0}(\hat{\Phi}_i^\mathsf{T}\ell) + \rho_{\hat{\mathcal{Y}}_{s_{I_k}}}(\ell).$$

The latter term poses a problem. Computing its value in the recursive sequence above requires evaluating the support of $\hat{\mathcal{Y}}_{\Delta_i}$ for each i, which is $\mathcal{O}\left(Nn^2\right)$. So each evaluation of $\rho_{\Xi_{I_k}}(\ell)$ is $\mathcal{O}\left(iNn^2\right)$. Without switching, the cost to compute $\rho_{\Omega_{t_k,t_{k+1}}}(\ell)$ is $\mathcal{O}\left(kn^2\right)$, but it is incremental in the sense that computing the value at t_{k+1} only adds $\mathcal{O}\left(n^2\right)$ operations. The combined cost for all $\Omega_{t_k,t_{k+1}}$ up to $k = K$ is $\mathcal{O}\left(Kn^2\right)$. The switched case is not incremental, since the intermediate results of $\rho_{\hat{\mathcal{Y}}_{\Delta_k}}(\ell)$ can not be reused because of the changed dynamics. The cost up to $k = K$ with i switches is therefore $\mathcal{O}\left(KiNn^2\right)$. Assuming N time steps per switch, we have $K = iN$ at the end of the i-th switch, so the cumulated cost is $\mathcal{O}\left(i^2N^2n^2\right)$.

Observation 4 *The total operation count per direction when using support functions directly with i switchings and N time steps per switch is $\mathcal{O}\left(i^2N^2n^2\right)$. Since N can be an arbitrarily large number (in practice often in the thousands), this makes the direct support function computation unusable even for low-dimensional systems and moderate number of switches.*

Using Support Functions on Zonotopes Step-Wise We now follow the approach in Sect. 5.1 and compute $\hat{\mathcal{Y}}_{\Delta_i}$ as well as $\hat{\Psi}_{s_i}$ using zonotopes. The zonotope construction of $\hat{\mathcal{Y}}_{\Delta_i}$ with $k_R = c \cdot n$ generators is $\mathcal{O}\left(Nn^3\right)$. Since i such sets need to be constructed, we have $\mathcal{O}\left(iNn^3\right)$ as total construction cost, independent from the number of support function evaluations. Each support function evaluation is $\mathcal{O}\left(cn^2\right)$. Evaluating the support of $\hat{\Psi}_{s_i}$ as above has complexity $\mathcal{O}\left(Kicn^2\right)$, with $K = iN$ this gives a cumulative cost of $\mathcal{O}\left(ci^2Nn^2\right)$. As in Sect. 5.1, the generator reduction allows us to replace one factor of N with the order c of the reduced zonotope. For $c = 1$, we obtain a bounding box approximation of $\hat{\mathcal{Y}}_{\Delta_i}$, with the approximation error discussed in Sect. 4.1.1.

We now examine the approximation error due to the generator reduction. The approximation is no longer tight due to the switching of dynamics: Computing the support of Ξ_i in the axis directions requires the support of $\hat{\mathcal{Y}}_{\Delta_i}$ in other directions. The error can thus be amplified. The sequence suffers from the wrapping effect if the generator reduction was

carried out tight in the axis directions, as in Sect. 2.1.2. To demonstrate this, we reformulate (53) to

$$\hat{\Psi}_{s_{i+1}} = e^{A_i\Delta_i}\hat{\Psi}_{s_i} \oplus \hat{\mathcal{Y}}_{\Delta_i}(A_i, U_i) \tag{56}$$

$$= e^{A_i\Delta_i} \cdots e^{A_1\Delta_1}\hat{\mathcal{Y}}_{\Delta_0}(A_0, U_0) \oplus \cdots \oplus e^{A_i\Delta_i}\hat{\mathcal{Y}}_{\Delta_{i-1}}(A_{i-1}, U_{i-1}) \oplus \hat{\mathcal{Y}}_{\Delta_i}(A_i, U_i) \tag{57}$$

The matrices before each term, e.g., $\hat{\mathcal{Y}}_{\Delta_0}(A_0, U_0)$, change with i. We can reinstate tightness by reducing the generators so that they are tight with respect to the transformed axis directions, but this requires us to keep the original generators in memory. The reduction and transformation cost is $\mathcal{O}\left(i^2(Nn^2 + N\log N)\right)$, since all i input convolutions need to be remapped and reduced at every step. Note that the memory consumption can be reduced by storing the original zonotopes in compressed form (as a set of transformation matrices plus the original generators).

Observation 5 *Computing $\hat{\mathcal{Y}}_{\Delta_i}$ as zonotopes with cn generators is a factor $N/(c+1)$ faster than using support functions directly. Keeping the original zonotopes and reducing the mapped zonotopes at each step allows one to keep the approximation tight, at a price of considerable increase in memory. The cost is quadratic in i, and therefore limited to a moderate number of switches.*

Using Support Functions on Zonotopes Globally We improve on the complexity by computing $\hat{\Psi}_{s_i}$ as a zonotope with reduced generators. As before, the zonotope construction of $\hat{\mathcal{Y}}_{\Delta_i}$ with $k_R = c \cdot n$ generators requires $\mathcal{O}\left(iNn^3\right)$ operations as total construction cost, independent from the number of support function evaluations. The zonotope construction of $\hat{\Psi}_{s_i}$ with $k_\Psi = d \cdot n$ generators requires essentially the reduction cost, which cumulates to $\mathcal{O}\left(icn^2 \log cn\right)$. As discussed above, the approximation is not tight and the approximation error may be amplified by the wrapping effect. Both c and d can be kept fairly high to improve the accuracy.

Each support function evaluation is $\mathcal{O}\left(dn^2\right)$ with a cumulative cost of $\mathcal{O}\left(Kdn^2\right) = \mathcal{O}\left(idNn^2\right)$ for $K = iN$. The cost is now linear in both the number of switchings as well as the time steps per switch. The key here is that d does not necessarily need to increase with i to keep the approximation error acceptable.

Observation 6 *Computing $\hat{\Psi}_{s_i}$ as zonotopes with dn generators is a factor $iN/(d+1)$ faster than using support functions directly. The approximation error is subject to the wrapping effect, but the runtime scales linearly in both the time steps and the number of switchings.*

Note that the number of reduced generators does not necessarily need to be fixed in advance; the approximation error can be bounded, and this bound can be used to find an appropriate number of generators.

5.3 From Time- to State-Triggered Switching

In state-triggered switching, the switch from dynamics A_i to A_{i+1} takes place not at a specific point in time, but when the state enters a given *guard set* $\mathcal{G}$. Depending on the semantics, the switch takes place as soon as possible, which is called *urgent* or *must* semantics. In *may* semantics, it may be nondeterministically delayed as long as a given *staying condition* $\mathcal{S}$ is satisfied. For the purposes of reachability computation, both cases are similarly difficult, since trajectories starting from the initial states may either way take the switch at different points in time. There is thus a set of intervals on the time axis $\mathbb{I} \subseteq R^{\geq 0}$ such that $t \in \mathbb{I}$ whenever $\mathcal{X}_t \in \mathcal{G}$. In the sequel we will assume that there is only a single such interval; our approach extends to sequences of intervals by enumeration. The main difference with time-triggered switching is that the set of states that takes the switch is nonconvex,

$$\mathcal{X}_{\mathbb{I}} = \bigcup_{t \in \mathbb{I}} \mathcal{X}_t \cap \mathcal{G}.$$

Identifying the time interval $\mathbb{I}$ is the first step in the successor computation. It is necessarily done only approximatively, and the accuracy of the time interval can impact the accuracy of the reachable states. The approach proposed in [8] uses an optimization procedure on the support function. This requires a certain number of support function evaluations and is thus a good candidate for the use of zonotopes, as discussed in Sect. 5.1.

The second step is to construct an approximation of $\mathcal{X}_{\mathbb{I}}$ in the form of one or more convex sets. In the support function algorithm from [12], the template hull (a polyhedron) is constructed. A more refined version in [11] produces a minimal set of semi-template polyhedra in space-time (with time as additional variable). The reason for the polyhedron construction is that this simplifies the intersection with the guard set $\mathcal{G}$, which is typically also a polyhedron.

An alternative to this second step has been proposed in [**?**] and is called *continuization*. It avoids the intersection step altogether. Instead, it replaces the state-based switch by two time triggered switches, one at the beginning and one at the end of the time interval. In the interior of the interval, an auxiliary dynamic $\hat{A}_i$ is used such that the trajectories of both A_i and A_{i+1} are captured. Let $\mathcal{D}$ be some domain that encloses $\mathcal{X}_{\mathbb{I}}$, e.g., computed by a prior coarse reachability run. The continuization procedure is sure to cover all reachable states if for all $x \in \mathcal{D}$,

$$\mathrm{CH}(A_i x \oplus \mathcal{U}_i, A_{i+1} x \oplus \mathcal{U}_{i+1}) \subseteq \hat{A}_i x + \hat{\mathcal{U}}_i.$$

This can be ensured, e.g., by letting $\hat{A}_i = \frac{1}{2}(A_i + A_{i+1})$ and

$$\hat{\mathcal{U}}_i = \mathrm{CH}\Big((A_i - \hat{A}_i)\mathcal{D} \oplus \mathcal{U}_i, (A_{i+1} - \hat{A}_i)\mathcal{D} \oplus \mathcal{U}_{i+1}\Big).$$

We can now apply the time-triggered switching algorithm from Sect. 5.2 to the switching sequence A_i, $\hat{A}_i$, A_{i+1}.

6 Conclusions

In the literature on reachability analysis and, more particular, flowpipe approximation, algorithms on zonotopes and support functions have so far been reported separately. Both set representations scale very well with respect to the geometric operations used in reachability analysis, but they also have their shortcomings. Zonotopes have the disadvantage that they are not closed under intersection,[2] and approximating the convex hull requires small time steps. This limits also the capacity to cluster sets, so that an explosion in the number of sets is more difficult to avoid.[3] Support functions do not scale particularly well for intersection, but by combining them with template polyhedra this problem can be mitigated. However, the support function algorithms loose their efficiency when the dynamics change. One is forced to construct an outer approximation in the form of a template polyhedron whenever the dynamics change. Zonotopes do not have this shortcoming.

In this paper, we compared the complexity incurred by both set representations in detail, for the case of flowpipe approximation with affine dynamics. Our finding is that zonotopes are inherently suited for approximating the input convolution. This is not surprising, since zonotopes are defined as the Minkowski sum of independent line segments, while the input

[2] In [**?**], it has been proposed to represent sets as the intersection of zonotopes. This removes the problem of intersecting zonotopes with guards, but makes the computation of the Minkowski sum trickier.

[3] The problems with intersection and clustering can be avoided by switching from zonotopes to polytopes and back [**?**]. Scalability and precision can be tuned by choosing the number of constraints, but the approach generally scales less well than using just zonotopes.

convolution is the Minkowski sum of independent input sets. We were somewhat surprised, however, that an ϵ-close zonotope approximation of the input convolution can be computed for the same cost as a bounding box approximation using the support function algorithm (Observations 1 and 2). We therefore propose to use a zonotope to approximate the input convolution, and the support function algorithm to compute the autonomous evolution. Both can be combined within the support function framework. The result is therefore an enhanced variation of the support function algorithm, which uses zonotopes as an auxiliary data structure. Reducing the number of generators of the zonotope to cn, where n is the dimension of the state space, allows us to evaluate, at equal cost, $N/(c+1)$ times the directions of the direct support function algorithm, with N being the number of time steps (Observation 3). While zonotopes may not help to accelerate the support function algorithm, they can substantially increase its precision by allowing one to evaluate more directions.

An even more distinct advantage was found for switching dynamics, where the switches take place at fixed points in time. Here, a direct computation with support functions does not scale even for low-dimensional systems and modest numbers of switches (Observation 4). Using separate zonotopes for the input convolution within switching times allows us to accelerate the computation by a factor of $N/(c+1)$. It remains, however, limited to moderate number of switches (Observation 5). Finally, using a single zonotope to compute the input convolution and reducing its number of generators to dn accelerates the computation by a factor of $iN/(d+1)$, where i is the number of switches (Observation 6). The resulting cost of evaluating the support function over i switches, with N time steps per switch, is $\mathcal{O}\left(idNn^2\right)$. This cost is a factor d higher than the support function algorithm without switching, but it scales linearly with the number of switches and time steps. The downside of using a reduced zonotope for switched dynamics is a potential error accumulation known as the wrapping effect. Note, however, that all other published algorithms for high-dimensional switched dynamics suffer from the same effect. A more detailed study of the generator reduction for this particular application may help to alleviate this problem.

Outlook The algorithm proposed in this paper is highly relevant to the analysis of systems with nonlinear dynamics. By linearizing the system at time-triggered intervals, our algorithm can immediately be applied. Note that the switching times need not be fixed in advance, and can themselves be triggered using state-based conditions as shown in Sect. 5.3. This paper thus lays the groundwork for bringing the power of support function algorithms to bear on systems with nonlinear dynamics.

References

[1] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control (HSCC'12)*, pages 45–54. ACM, 2012.

[2] M. Althoff, B. H. Krogh, and O. Stursberg. Analyzing reachability of linear dynamic systems with parametric uncertainties. In A. Rauh and E. Auer, editors, *Modeling, Design, and Simulation of Systems with Uncertainties*. Springer, 2011.

[3] R. Alur. Formal verification of hybrid systems. In S. Chakraborty, A. Jerraya, S. K. Baruah, and S. Fischmeister, editors, *EMSOFT*, pages 273–278. ACM, 2011.

[4] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate reachability analysis of piecewise-linear dynamical systems. In *Proc. HSCC 00: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, pages 20–31. Springer-Verlag, 2000.

[5] J. Bourgain, J. Lindenstrauss, and V. Milman. Approximation of zonoids by zonotopes. *Acta mathematica*, 162(1):73–141, 1989.

[6] L. Doyen, G. Frehse, G. Pappas, and A. Platzer. Verification of hybrid systems. In *Handbook of Model Checking*. Springer, 2016. (to appear).

[7] G. Frehse. An introduction to hybrid automata, numerical simulation and reachability analysis. In *Formal Modeling and Verification of Cyber-Physical Systems, 1st International Summer School on Methods and Tools for the Design of Digital Systems, Bremen, Germany*. Springer, 2015.

[8] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. Eliminating spurious transitions in reachability with support functions. In *Hybrid Systems: Computation and Control (HSCC'15)*, pages 149–158. ACM, 2015.

[9] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. Eliminating spurious transitions in reachability with support functions. In *HSCC'15*, pages 149–158. ACM, 2015.

[10] G. Frehse, R. Kateja, and C. L. Guernic. Flowpipe approximation and clustering in space-time. In C. Belta and F. Ivancic, editors, *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 203–212. ACM, 2013.

[11] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *HSCC'13*, pages 203–212. ACM, 2013.

[12] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV*, pages 379–395, 2011.

[13] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *IFAC ADHS*, pages 94–101, 2012.

[14] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *HSCC*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[15] A. Girard, C. L. Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In J. P. Hespanha and A. Tiwari, editors, *HSCC*, volume 3927 of *LNCS*, pages 257–271. Springer, 2006.

[16] W. Kühn. Towards an optimal control of the wrapping effect. In *Developments in Reliable Computing*, pages 43–51. Springer, 1999.

[17] C. Le Guernic. *Reachability analysis of hybrid systems with linear continuous dynamics*. PhD thesis, Université Grenoble 1 - Joseph Fourier, 2009.

[18] C. Le Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 540–554. Springer, 2009.

[19] C. Le Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250 – 262, 2010. IFAC World Congress 2008.

[20] A. V. Lotov, V. A. Bushenkov, and G. K. Kamenev. *Interactive Decision Maps*, volume 89 of *Applied Optimization*. Kluwer, 2004.

[21] J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, 2009.

[22] O. Maler. Algorithmic verification of continuous and hybrid systems. In *Int. Workshop on Verification of Infinite-State System (Infinity)*, 2013.

[23] S. Sankaranarayanan, T. Dang, and F. Ivancic. Symbolic model checking of hybrid systems using template polyhedra. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2008.

[24] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[25] P. Varaiya. Reach set computation using optimal control. In *Proc. KIT Workshop*, pages 377–383, 1997.