Technische Universität München

Fakultät für Informatik

Extensions of Polynomial Zonotopes and their Application to Verification of Cyber-Physical Systems

Niklas Kochdumper

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Tobias Nipkow, Ph.D.

Prüfende der Dissertation:
1. Prof. Dr.-Ing. Matthias Althoff
2. Prof. Dr. Jan Křetínský

Die Dissertation wurde am 13.09.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05.02.2022 angenommen.

Foreword

This thesis summarizes the results of my research at the Chair of Robotics, Artificial Intelligence and Real-time Systems at the Technical University of Munich. During my time at the chair I was inspired and supported by many people, to all of whom I would like to shortly express my gratitude here.

First of all, I would like to thank my advisor Prof. Dr.-Ing. Matthias Althoff, who offered me a PhD position in his group and always trusted in my abilities. Especially the time he took for discussions about research and providing feedback for papers and articles helped me a lot to improve my knowledge and my writing skills. I am also very thankful to Prof. Dr. Jan Křetínský for acting as the second examiner for my thesis and to Prof. Tobias Nipkow for chairing the defense.

Of course, I would also like to thank all my colleagues at the chair for the great time we had at work and at the numerous after-work events. Thanks to you, I always enjoyed working at the university. In addition, I wish to express my gratitude to the secretaries, coordinators, and system administrators at the chair for helping me at several occasions. I would also like to thank my fellow researchers from the DFG project faveAC for the great collaboration we had. Moreover, I appreciate the great work of all students that I supervised as well as all student assistants that helped to improve the reachability toolbox CORA.

Finally, I would like to thank my family and friends for their support, especially during this challenging time of the Corona pandemic.

Munich, September 2021 Niklas Kochdumper

Abstract

In this thesis, we present the three novel set representations *sparse polynomial zonotopes*, *constrained polynomial zonotopes*, and the *Z-representation* of polytopes. One major application for these novel set representations is the verification of cyber-physical systems using reachability analysis, for which we introduce various improvements as well as novel algorithms and approaches. Besides reachability analysis, there exist several other applications that use set-based computations and consequently profit from our novel set representations, some of which we discuss in detail in this thesis.

We first introduce sparse polynomial zonotopes, a novel non-convex set representation that is closed under linear map, Minkowski sum, Cartesian product, convex hull, and quadratic map. Constrained polynomial zonotopes extend sparse polynomial zonotopes by adding polynomial equality constraints for the dependent factors to obtain a set representation that is additionally closed under intersection and union. For both, sparse polynomial zonotopes and constrained polynomial zonotopes, the computational complexity for all relevant set operations is only polynomial with respect to the dimension, so that these novel set representations are well suited for the analysis and verification of high-dimensional systems. Finally, our novel Z-representation of polytopes is often more compact than the vertex representation and the halfspace representation for polytopes that are similar to zonotopes.

Concerning reachability analysis, we first demonstrate the benefits of sparse polynomial zonotopes for computing outer-approximations of reachable sets for nonlinear continuous systems. Next, we show that with sparse polynomial zonotopes relations between initial states and reachable states are preserved, which results in a very efficient method for the extraction of reachable subsets. Finally, we introduce novel approaches for calculating tight inner-approximations of reachable sets for nonlinear continuous systems and for reachability analysis of hybrid systems with nonlinear guard sets. All reachability algorithms presented only have polynomial complexity with respect to the system dimension and we demonstrate their superior performance compared to other state of the art approaches on several numerical examples.

For constrained polynomial zonotopes, we discuss the two applications set-based observation and program verification using inductive invariants in detail, for both of which the closedness under intersection and union is very advantageous. The novel Z-representation of polytopes can be applied for range bounding on polytopic domains and has a strong relationship to generalized barycentric coordinates.

Zusammenfassung

In dieser Dissertation präsentieren wir die drei neuen Mengendarstellungen *Sparse Polynomial Zonotopes*, *Constrained Polynomial Zonotopes*, und die *Z-Representation* von Polytopen. Eine der Hauptanwendungen dieser neuen Mengendarstellungen ist die Verifikation von cyber-physischen Systemen mittels Erreichbarkeitsanalyse, wofür wir zahlreiche Verbesserungen sowie neue Algorithmen und Methoden einführen. Zusätzlich zu Erreichbarkeitsanalyse gibt es viele andere Anwendungen welche mengenbasierten Berechnungen benutzen und somit von unseren neuen Mengendarstellungen profitieren.

Zuerst führen wir mit Sparse Polynomial Zonotopes eine neue Mengendarstellung ein welche unter linearen Abbildungen, Minkowski Additionen, kartesischen Produkten, konvexen Hüllen, und quadratischen Abbildungen geschlossen ist. Constrained Polynomial Zonotopes erweitern Sparse Polynomial Zonotopes indem sie den Wertebereich der Faktoren durch polynomielle Gleichungen einschränken, was in einer Mengendarstellung resultiert welche zusätzlich unter Schnitten und Vereinigungen geschlossen ist. Sowohl für Sparse Polynomial Zonotopes als auch für Constrained Polynomial Zonotopes wächst die Rechenkomplexität aller gängigen Mengenoperationen nur polynomiell mit der Dimension an. Des Weiteren kann unsere neue Z-Representation Polytope welche Zonotopen ähnlich sind oft kompakter darstellen als die Eckpunkt- und die Halbraum-Darstellung.

Für Erreichbarkeitsanalyse demonstrieren wir zunächst die Verbesserungen durch Sparse Polynomial Zonotopes für die Berechnung von Über-Approximationen der erreichbaren Menge für nichtlineare kontinuierliche Systeme. Anschließend zeigen wir, dass mit Sparse Polynomial Zonotopes die Zuordnung zwischen den initialen Zuständen und den erreichbaren Zuständen erhalten bleibt, was in einer sehr effizienten Methode für die Extraktion von erreichbaren Teilmengen resultiert. Schließlich präsentieren wir neue Ansätze für die Berechnung von engen Unter-Approximationen der erreichbaren Menge für nichtlineare kontinuierliche Systeme sowie für Erreichbarkeitsanalyse von hybriden Systemen mit nichtlinearen Übergängen zwischen den diskreten Moden. Für alle Erreichbarkeitsalgorithmen welche wir präsentieren wächst die Rechenkomplexität nur polynomiell mit der Dimension an. Darüber hinaus demonstrieren wir die Performanz unserer Algorithmen im Vergleich zu anderen Ansätzen anhand zahlreicher numerischer Beispiele.

Für Constrained Polynomial Zonotopes diskutieren wir mengenbasierte Zustandsbeobachter und die Verifikation von Computerprogrammen mittels induktiver Invarianzgebiete im Detail, wobei die Geschlossenheit unter Schnitten und Vereinigungen für beide Anwendungen von großem Vorteil ist. Die neue Z-Representation von Polytopen kann unter anderem dafür genutzt werden Grenzen für die Werte einer nichtlinearen Funktion zu berechnen wenn die Werte für die Variablen der Funktion auf ein Polytop beschränkt sind. Außerdem besteht eine starke Verbindung zwischen der Z-Representation und den generalisierten baryzentrischen Koordinaten eines Polytops.

Contents

Notations

In this thesis, we use the following notations: Vectors are denoted by lowercase letters, matrices by uppercase letters, sets by calligraphic letters, matrix sets (e.g., interval matrices) by bold calligraphic letters, tuples and lists by bold uppercase letters, objects by Ralph Smith's Formal Script Font (e.g., hybrid automaton $\mathscr{H}$), and operators in typewriter font (e.g., `center`).

Number Sets

The set of natural numbers is denoted by $\mathbb{N} = \{1, 2, \dots\}$, the set of natural numbers including zero is denoted by $\mathbb{N}_0 = \{0, 1, 2, \dots\}$, the set of real numbers is denoted by $\mathbb{R}$, and the set of imaginary numbers is denoted by $\mathbb{I}$.

Discrete Sets and Tuples

Given a discrete set $\mathcal{H} \in \{h_1, \dots, h_n\}$, $|\mathcal{H}| = n$ denotes the cardinality of the set and the empty set is denoted by $\emptyset$. Similarly, given a n-tuple $\mathbf{H} = (h_1, \dots, h_n)$, $|\mathbf{H}| = n$ denotes the cardinality of the tuple and $\mathbf{H}_{(i)} = h_i$ refers to the i-th entry of tuple $\mathbf{H}$. Moreover, given a tuple $\mathbf{K} = (k_1, \dots, k_n)$ with $\forall i \in \{1, \dots, n\} : \ k_i \in \mathbb{R}^{m_i}$, notation $\mathbf{K}_{(i,j)} = k_{i(j)}$ refers to the j-th entry in the i-th element of $\mathbf{K}$. The empty tuple is denoted by $\emptyset$. Given two tuples $\mathbf{H} = (h_1, \dots, h_n)$ and $\mathbf{K} = (k_1, \dots, k_m)$, $(\mathbf{H}, \mathbf{K}) = (h_1, \dots, h_n, k_1, \dots, k_m)$ denotes the concatenation of the tuples.

Sets in Euclidean Space

Given two set operations `A`, `B` and a set $\mathcal{S} \subset \mathbb{R}^n$, the composition of the set operations is denoted by $\texttt{A}(\texttt{B}(\mathcal{S})) = (\texttt{A} \circ \texttt{B})(\mathcal{S})$. Moreover, $2^{\mathcal{S}}$ denotes the power set and $\partial\mathcal{S}$ denotes the boundary. In addition, operation $\texttt{center}(\mathcal{S})$ returns the center, operation $\texttt{volume}(\mathcal{S})$ returns the volume, and operation $\texttt{vertices}(\mathcal{S})$ returns the vertices of the set $\mathcal{S}$. Indices for sets are passed on to variables describing quantities, e.g., the number of zonotope generators l_i belongs to the zonotope $\mathcal{Z}_i$.

Matrices and Vectors

Given a vector $b \in \mathbb{R}^n$, $b_{(i)}$ refers to the i-th entry. Similarly, given a matrix $A \in \mathbb{R}^{n \times m}$, $A_{(i,\cdot)}$ represents the i-th matrix row, $A_{(\cdot,j)}$ the j-th column, and $A_{(i,j)}$ the j-th entry of matrix row i. Given a discrete set of positive integer indices $\mathcal{H} = \{h_1, \dots, h_{|\mathcal{H}|}\}$ with $\forall i \in \{1, \dots, |\mathcal{H}|\} : \ 1 \leq h_i \leq m$, notation $A_{(\cdot,\mathcal{H})}$ is used for $[A_{(\cdot,h_1)} \ \dots \ A_{(\cdot,h_{|\mathcal{H}|})}]$, where $[C \ D]$ denotes the concatenation of two matrices C and D. The symbols $\mathbf{0}$ and $\mathbf{1}$ represent

matrices and vectors of zeros and ones of proper dimension and the empty matrix is denoted by $[\,]$. The identity matrix of dimension $n \times n$ is denoted by $I_n \in \mathbb{R}^{n \times n}$. Given two vectors $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$, we use the notation $a \prec b$ as a shorthand for $a_{(1)} \prec b_{(1)} \wedge \ldots \wedge a_{(n)} \prec b_{(n)}$, where $\prec \in \{\leq, <, =, >, \geq\}$. Moreover, $|a| = [|a_{(1)}| \ \ldots \ |a_{(n)}|]^T$ is interpreted elementwise. Given a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $a \in \mathbb{R}^n$, operation `diag` is defined as

$$\texttt{diag}(A) = \begin{bmatrix} A_{(1,1)} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & A_{(n,n)} \end{bmatrix}, \ \texttt{diag}(a) = \begin{bmatrix} a_{(1)} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & a_{(n)} \end{bmatrix},$$

operation `offdiag` is defined as $\texttt{offdiag}(A) = A - \texttt{diag}(A)$, and operation $\texttt{det}(A)$ returns the determinant of the matrix. For matrix sets such as interval matrices we use the same notation as for numerical matrices.

Mathematical Operators

The ceil operator $\lceil x \rceil$ and the floor operator $\lfloor x \rfloor$ round a scalar number $x \in \mathbb{R}$ to the next higher and lower integer, respectively. Given two integers $p \in \mathbb{N}_0$ and $q \in \mathbb{N}$, the modulo operation $p \mod q$ returns the remainder of the division p/q. The binomial coefficient is denoted by $\binom{r}{z}$ with $r, z \in \mathbb{N}$, $r \geq z$. Given a matrix $A \in \mathbb{R}^{n \times m}$, a vector $a \in \mathbb{R}^n$, and scalars $x_1, \ldots, x_n \in \mathbb{R}$, $\max(A)$ returns the maximum entry in the matrix A, $\max(a)$ returns the maximum entry in the vector a, and $\max(x_1, \ldots, x_n)$ returns the maximum scalar. Similarly, $\min(A)$, $\min(a)$, and $\min(x_1, \ldots, x_n)$ returns the minimum entries. Moreover, given a vector $a \in \mathbb{R}^n$, $||a||_1$ denotes the 1-norm, $||a||_2$ denotes the Euclidean norm, and $||a||_\infty$ denotes the infinity norm.

Functions

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, $f(x)|_a = f(a)$ denotes the evaluation of function $f(x)$ at a point $x = a \in \mathbb{R}^n$. Moreover, the Nabla operator is defined as

$$\nabla f(x) = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \ldots & \frac{\partial f(x)}{\partial x_n} \end{bmatrix}^T,$$

where $x = [x_1 \ \ldots \ x_n]^T \in \mathbb{R}^n$. For vector fields $g : \mathbb{R}^n \to \mathbb{R}^m$ we write $\nabla g(x) = [\nabla g_{(1)}(x) \ \ldots \ \nabla g_{(m)}(x)]^T$ and we use the shorthands $\nabla^2 g(x)$, $\nabla^3 g(x)$, etc., for higher order derivatives. In addition, $g_{(i)}$ dentoes the i-th subfunction $g_{(i)} : \mathbb{R}^n \to \mathbb{R}$ of the vector field $g(x)$.

Computational Complexity

The computational complexity is denoted using the Big O notation $\mathcal{O}$. For operations whose complexity depends on the used method we denote by $\mathcal{O}(\texttt{reduce})$ the complexity of zonotope order reduction (see Tab. 2.3), by $\mathcal{O}(\texttt{bound})$ the complexity of range bounding (see Tab. 2.4), and by $\mathcal{O}(\texttt{contract})$ the complexity of contractors (see Tab. 2.5). Moreover, for the derivation of computational complexity we consider all binary operations except concatenations and initializations are explicitly not considered.

Computing Platform and Implementation

All computations in this thesis are carried out in MATLAB on a 2.9GHz quad-core i7 processor with 32GB memory. Moreover, we integrated all approaches presented in this thesis into the reachability toolbox CORA [1] published at `https://cora.in.tum.de`, so that the corresponding code is already publicly available or will be made publicly available with the next release of the CORA toolbox. For Chapter 4, we additionally published our implementation in CodeOcean compute capsules, which make it possible to conveniently reproduce the corresponding results:

- **Section 4.1:** Outer-Approximations of Reachable Sets for Nonlinear Continuous Systems (`https://codeocean.com/capsule/3393653/tree/v1`)
- **Section 4.2:** Reachable Subsets (`https://codeocean.com/capsule/8904078/tree/v1`)
- **Section 4.3:** Inner-Approximations of Reachable Sets for Nonlinear Continuous Systems (`https://codeocean.com/capsule/5233492/tree/v2`)
- **Section 4.4:** Reachability Analysis for Hybrid Systems with Nonlinear Guard Sets (`https://codeocean.com/capsule/4124536/tree/v1`)

Chapter 1

Introduction

1.1 Motivation

During the past three-hundred years, humanity witnessed three major industrial revolutions: First, steam power and mechanization completely transformed the industry during the 18th century. Next, the introduction of mass production and electric energy in the 19th century defined the second revolution. Finally, the third revolution during the 20th century realized computers and automation in industry. Currently, we are in the middle the forth industrial revolution, which is characterized by modern smart technology and artificial intelligence. A major component of the resulting *Industry 4.0* are cyber-physical systems, where computer algorithms control mechanical or electrical components. Many of these systems, like for example autonomous cars, robot surgery, and robots collaborating with humans, are safety-critical, meaning that human lives get endangered every time these systems fail. Consequently, one key aspect for accelerating the forth industrial revolution is to formally verify that cyber-physical systems function correctly. However, since the complexity of these systems increases permanently, their formal verification becomes more and more complicated, so that novel verification approaches and algorithms with improved performance are required. The development of such improved verification techniques is the topic of this thesis.

In particular, we focus on the formal verification of complex cyber-physical systems using reachability analysis. Given a cyber-physical system whose dynamic behavior is described by an ordinary differential equation or a hybrid automaton, the reachable set of the system consists of all states that are reachable from a given initial set $\mathcal{X}_0$ under consideration of bounded disturbances that can change arbitrarily over time. As visualized in Fig. 1.1, this implies that the reachable set contains all possible system trajectories that start in $\mathcal{X}_0$. Since the exact reachable set cannot be computed in general, our goal is to compute a tight outer-approximation instead. Safety requirements and desired system behaviors are expressed by specifications, which define sets of unsafe or forbidden states the system should not enter (see Fig. 1.1). Typical specifications are, for example, that the temperature in a chemical reactor should stay below a certain threshold, that the altitude of a quadrocopter reaches the desired value after a certain time, or that an autonomous car should not collide with other traffic participants. To verify with reachability analysis that the specifications are satisfied, we simply check if the outer-approximation of the reachable set intersects any of the unsafe sets defined by the specifications. If no intersection occurs, then there

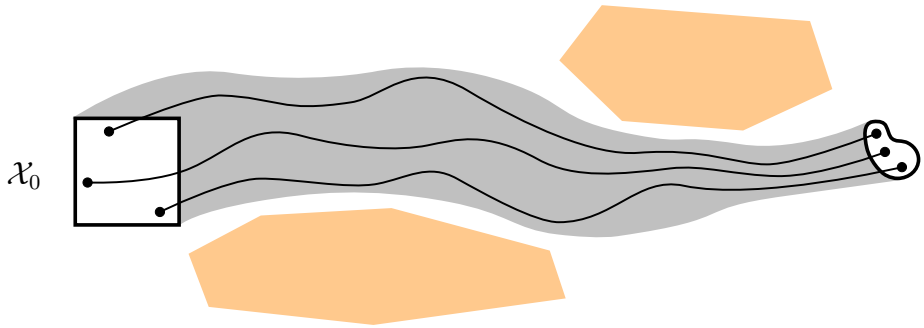

Figure 1.1: Schematic visualization of the reachable set (gray) for the initial set $\mathcal{X}_0$, where single trajectories are depicted by black lines and the unsafe sets defined by the specifications are shown in orange.

exists no trajectory starting in the initial set which enters an unsafe set. Consequently, the specifications are satisfied and the system is proven to be safe. However, if the outer-approximation intersects an unsafe set, we do not know if the system really is unsafe or if the intersection only occurs due to the over-approximation, since we cannot compute the exact reachable set. To solve this issue, an inner-approximation of the reachable set can be used. If the inner-approximation intersects an unsafe set, it is guaranteed that the specification is violated. In this thesis, we present novel approaches for computing tight outer-approximations and inner-approximations for cyber-physical system, which can be used to efficiently prove as well as disprove specifications.

All reachability algorithms we consider in this thesis use set-based computing. One major requirement to obtain tight outer-approximations and inner-approximations of reachable sets is therefore a powerful set representation. The four main performance evaluation criteria for set-representations are their generality, their representation size, their closedness under set operations, and their computational complexity:

- **Generality:** If all sets of a set representation A can be equivalently represented by set representation B, we have that B is a generalization of A. For example, every interval can be equivalently represented as a polytope, so that polytopes are a generalization of intervals.

- **Representation size:** The representation size of a set representation refers to the number of scalar values that have to be stored for each set of this set representation. Intervals, for example, have a very small representation size since an n-dimensional interval can be represented with $2n$ scalar numbers only.

- **Closedness under set operations:** A set representation is closed under a set operation if the set which results from applying the set operation can be represented exactly by the same set representation. For example, the intersection of two intervals yields an interval, so that intervals are closed under intersections. On the other hand, the linear map of an interval is not an interval, so that intervals are not closed under linear maps.

- **Computational complexity:** The computational complexity of a set operation for a specific set representation indicates how the computational effort increases with re-

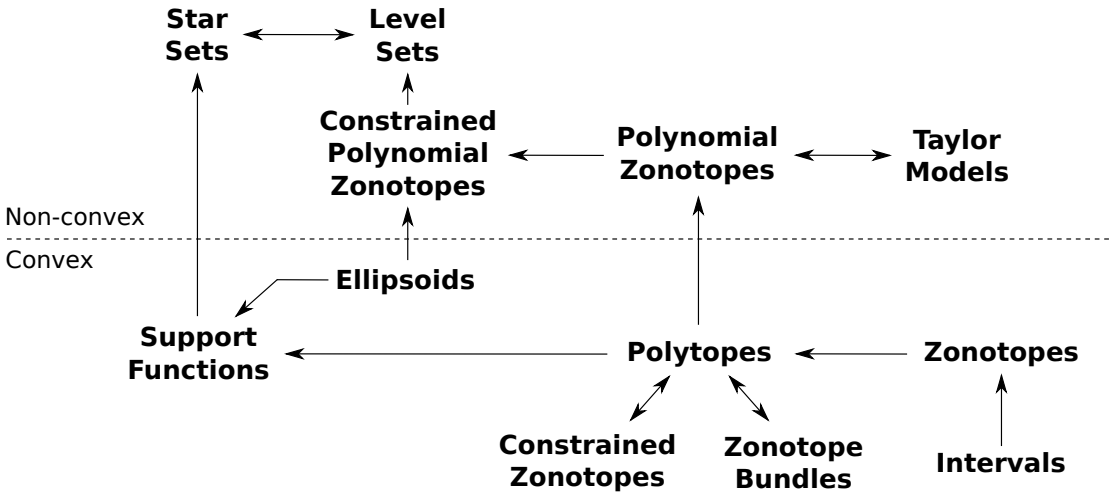

Figure 1.2: Visualization of the relations between different set representations, where A $\rightarrow$ B denotes that B is a generalization of A.

spect to the dimension. For example, the Minkowski sum for intervals has complexity $\mathcal{O}(n)$, so that the computation time increases linearly with the dimension n.

In this thesis we introduce novel set representations that can equivalently represent most common set representations, have quite small representation size due to the usage of a sparse representation, are closed under all relevant set operations, and realize those with algorithms that only have polynomial complexity with respect to the dimension. Due to these advantageous properties, our novel set representations are an excellent fit for reachability analysis of cyber-physical systems. Moreover, there exist many other applications, such as set-based observers or range bounding on polytopic domains, that benefit from our novel set representations; we present some of them in detail in this thesis.

1.2 State of the Art

We now provide an overview of the current state of the art for set-based computing in general, where we focus on different set representations and their respective properties in particular. Literature reviews for specific applications, such as reachability analysis or set-based observers, are provided separately in the corresponding sections later on to improve readability. Over the past years, many different set representations have been used in or developed for set-based computations. Relations between typical set representations are illustrated in Fig. 1.2. Moreover, Tab. 1.1 shows which set representations are closed under relevant set operations, and lists the corresponding computational complexities.

All convex sets can be represented by support functions [2, Ch. C.2]. Moreover, linear map, Minkowski sum, Cartesian product and convex hull are trivial to compute for support functions [3, Prop. 2], which makes them a good fit for reachability analysis of linear continuous systems [3–5] and hybrid systems [6, 7]. Even though support functions are closed under intersection, there exists no closed-form expression for the computation of this operation, and support functions are not closed under union and quadratic maps. One

special case of support functions are ellipsoids, which are closed under linear map only. However, ellipsoids are very compact since an n-dimensional ellipsoid can be represented with only $\frac{n}{2}(n+3)$ scalar numbers [8, Eq. 2.3]. Due to this advantageous property ellipsoids have been widely used for various applications including reachability analysis [9], set-based observers [10], and controlled invariant set computation [11, 12].

Another special case of support functions are polytopes, which are closed under linear map, Minkowski sum, Cartesian product, convex hull, and intersection [13, Ch. 3.1]. The computational complexity of the set operations for polytopes depends on the used representation [14], where the two main representations are the halfspace representation (H-representation) and the vertex representation (V-representation): For the H-representation, linear maps with full-rank square matrices, Cartesian products, and intersections are cheap to compute, while Minkowski sums and convex hulls are computationally expensive [14]. For singular matrices the linear map defines a projection, whose computation is NP-hard in H-representation [15, Tab. 1]. If redundant points are not removed, computation of the Cartesian product and the convex hull are trivial for the V-representation. However, since the number of vertices usually grows exponentially with the dimension, linear map and Minkowski sum have exponential complexity for the V-representation, and calculating the intersection is NP-hard [14]. The conversion from V-representation to H-representation is known as the *facet enumeration problem*, and the inverse problem of finding the V-representation given the H-representation is known as the *vertex enumeration problem*. Due to the duality of H-representation and V-representation [13, Ch. 3.4], both problems are actually equivalent and can be computed in polynomial time with respect to the number of polytope vertices [16], which, however, is exponential with respect to the dimension. While polytopes have been applied for various applications including reachability analysis [17, 18], set-based observers [19], and controlled invariant set computation [20, 21], they are in general restricted to low-dimensional systems due to their large representation size and high computational complexity for some set operations.

An important subclass of polytopes are zonotopes [23, Ch. 7.3], which can be compactly represented by so-called generators: A n-dimensional zonotope with l generators might have up to $2\binom{l}{n-1}$ halfspaces [24, Ch. 2.2.1]. In addition, linear map, Minkowski sum, and Cartesian product can be computed exactly and efficiently for zonotopes [5, Tab. 1], which makes them an excellent fit for reachability analysis of linear systems [5, 25]. Moreover, zonotopes have been successfully applied for set-based observers [26, 27], controlled invariant set computation [28], and program verification [29]. Zonotopes are closely related to affine arithmetic [30] with the zonotope factors being identical to the noise symbols in affine arithmetic. Two extensions of zonotopes are zonotope bundles [31] and constrained zonotopes [32], which are both able to represent any bounded polytope. Constrained zonotopes additionally consider linear equality constraints for the zonotope factors, whereas zonotope bundles represent the set implicitly by the intersection of several zonotopes. Both representations make use of lazy computations and thus suffer much less from the curse of dimensionality as it is the case for the V-representation and H-representation of polytopes. Two other set representations related to zonotopes are complex zonotopes [33] and zonotopes with sub-polyhedric domains [34]. Complex zonotopes are defined by complex valued vectors and are well suited to verify global exponential stability for systems with complex valued eigenvectors [33]. Zonotopes with sub-polyhedric domains use zones, oc-

Table 1.1: Computational complexity of set operations with respect to the dimension n for different set representations, where symbol $\times$ indicates that the set representation is not closed under the operation, symbol $-$ indicates that no closed-form expression exists, and the shorthand s. p. denotes super-polynomial complexity. For zonotopes, polytopes, constrained zonotopes, zonotope bundles, support functions, and star sets the complexity for a set spanned by n generators is specified, and it is assumed that potential redundancies in the set representation are not removed. Moreover, w denotes the resulting dimension for the linear map and the quadratic map, and symbol † indicates that the complexity for linear maps with full-rank square matrices is specified. The computational complexities for sparse polynomial zonotopes and constrained polynomial zonotopes slightly differ from those derived in this thesis since the removal of redundancies and the generation of unique identifiers is omitted for a fair comparison. References for all set operations are provided in [22, Tab. 1].

Set Representation	**Lin. Map**	**Mink. Sum**	**Cart. Prod.**	**Conv. Hull**	**Quad. Map**	**Inter-section**	**Union**
Intervals	$\times$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\times$	$\times$	$\mathcal{O}(n)$	$\times$
Zonotopes	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\times$	$\times$	$\times$	$\times$
Polytopes (H-Rep.)	$\mathcal{O}(n^3)^\dagger$	$\mathcal{O}(2^n)$	$\mathcal{O}(1)$	s. p.	$\times$	$\mathcal{O}(1)$	$\times$
Polytopes (V-Rep.)	$\mathcal{O}(wn2^n)$	$\mathcal{O}(n2^{2n})$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\times$	s. p.	$\times$
Polytopes (Z-Rep.)	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\times$	s. p.	$\times$
Con. Zonotopes	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\times$	$\mathcal{O}(n)$	$\times$
Zonotope Bundles	$\mathcal{O}(n^3)^\dagger$	-	$\mathcal{O}(1)$	-	$\times$	$\mathcal{O}(1)$	$\times$
Ellipsoids	$\mathcal{O}(n^3)^\dagger$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Support Functions	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\times$	$-$	$\times$
Taylor Models	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(wn^3)$	$\times$	$\times$
Level Sets	$\mathcal{O}(n^3)^\dagger$	$-$	$\mathcal{O}(1)$	$-$	$-$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Star Sets	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$-$	$-$	$-$	$-$
Sparse Poly. Zono.	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(wn^3)$	$\times$	$\times$
Con. Poly. Zono.	$\mathcal{O}(wn^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(wn^3)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

tagons, and polyhedra instead of intervals to represent the domain for the zonotope factors, which enables the efficient computation of intersections and unions of sets by exploiting lazy computations [34]. Finally, two special cases of zonotopes are parallelotopes, which are zonotopes with linearly independent generators, and multi-dimensional intervals. Due to their computational efficiency, parallelotopes are a popular set representation for set-based observers [35, 36]. Intervals are especially useful for range bounding of nonlinear functions using interval arithmetic [37, Ch. 2.3], but have also been used for reachability analysis [38,39] and controlled invariant set computation [40]. Since intervals are not closed under linear map, algorithms computing with intervals often split them to obtain a desired accuracy, which yields a so-called subpaving [37, Ch. 3].

All set representations discussed so far are convex. However, most cyber-physical systems are characterized by nonlinear or hybrid dynamics, and reachable sets as well as controlled invariant sets for nonlinear and hybrid systems are in general non-convex. Consequently, non-convex set representations are required for the computation of tight outer-approximations and inner-approximations if the computational expensive splitting of sets should be avoided. Common non-convex set representations are Taylor models, polynomial zonotopes, star sets, and level sets. Taylor models [41] consist of a polynomial and an interval remainder part, and are typically used for range bounding [41,42] and reachability analysis [43–46]. Polynomial zonotopes, which are introduced in [47] for reachability analysis of strongly nonlinear systems, can equivalently represent the set defined by a Taylor model. Quadratic zonotopes [48] are a special case of polynomial zonotopes, which have been used for program verification [48]. The concept of star sets [49] is similar to the one of constrained zonotopes, but logical predicates instead of linear equality constraints are used to constrain the values of the zonotope factors, which increases the expressiveness. Star sets are especially useful for simulation-based reachability analysis [49–51]. Finally, level sets of nonlinear functions are applied for reachability analysis [52, 53], controlled invariant regions computation [54, 55], and program verification [56, 57]. One advantage of level sets is that containment checks are straightforward and computationally efficient. While star sets and level sets are very expressive, it is yet unclear how some operations, such as nonlinear mapping, are computed.

In this thesis, we introduce the novel set representations sparse polynomial zonotopes, constrained polynomial zonotopes, and the Z-representation of polytopes. As shown in Tab. 1.1, contrary to existing set representations, these novel set representations only have polynomial complexity for all relevant set operations. Moreover, constrained polynomial zonotopes are the only set representation for which closed-form expressions for all set operations including intersection and union exist. In addition, all common set representations except support functions, star sets and level sets can be equivalently represented as constrained polynomial zonotopes, which is illustrated in Fig. 1.2. Due to this advantageous properties, our novel set representations are well suited for applications that use set-based computing, such as reachability analysis, set-based observers, or program verification using inductive invariants.

1.3 Outline of the Thesis

In this thesis, we introduce novel set representations with several advantageous properties, present new approaches for reachability analysis of nonlinear continuous and hybrid systems, and discuss several other applications for our novel set representations in detail.

Chapter 2: Background and Preliminaries

We begin by introducing some concepts and operations that are required throughout this thesis in Chapter 2. In particular, we first formally define all set operations that we consider in Sec. 2.1, before we introduce common set representations in Sec. 2.2. Next, in Sec. 2.3, we present different classes of dynamical systems and formally define reachable sets. Afterward, we recapitulate some standard operations, such as linear programming and

singular value decomposition in Sec. 2.4, and introduce some useful auxiliary operations in Sec. 2.5. Finally, we explain and discuss some commonly used methods for zonotope order reduction, range bounding, and contraction in Sec. 2.6, Sec. 2.7, and Sec. 2.8.

Chapter 3: Extensions of Polynomial Zonotopes

One of the main parts of this thesis is Chapter 3, where we introduce the three novel set representations sparse polynomial zonotopes, constrained polynomial zonotopes, and the Z-representation of polytopes. We first present our sparse representation of polynomial zonotopes in Sec. 3.1, which stores polynomial zonotopes very compactly and consequently only has polynomial complexity with respect to the system dimension for all relevant set operations. Next, we introduce constrained polynomial zonotopes in Sec. 3.2, which extend sparse polynomial zonotopes by adding polynomial equality constraints on the dependent factors, resulting in a set representation that is closed under all relevant set operations including intersection and union. Finally, we present the novel Z-representation of polytopes in Sec. 3.3, which stores polynomial zonotopes that represent polytopes very efficiently and is consequently often more compact than the H-representation or V-representation for polytopes that are close to zonotopes. For each set representations we derive closed-from expressions for all basic set operations, for the conversion from and to other set representations, and for useful auxiliary operations, such as order reduction. In addition, we present results for the tight enclosure with simpler set representations and for intersection and containment checks.

Chapter 4: Reachability Analysis

A major application for our novel set representations is the verification of cyber-physical systems using reachability analysis, which we consider in Chapter 4. We begin by demonstrating the advantages resulting from the usage of sparse polynomial zonotopes for computing outer-approximations of reachable sets for nonlinear continuous systems in Sec. 4.1. Next, we show in Sec. 4.2 that sparse polynomial zonotopes preserve the relations between initial states and reachable states, which results in a very efficient method for the extraction of reachable subsets. Afterward, we present a novel algorithm for efficiently computing tight inner-approximations of reachable sets for nonlinear continuous systems in Sec. 4.3. Finally, in Sec. 4.4, we introduce a novel approach for reachability analysis of hybrid systems with nonlinear guard sets. For all reachability approaches we demonstrate the superior performance compared to existing approaches on several numerical examples.

Chapter 5: Selected Applications

Besides reachability analysis, there exist many other applications for the novel set representations presented in this thesis, some of which we discuss in detail in Chapter 5. In particular, we present a novel set-based observer that uses constrained polynomial zonotopes in Sec. 5.1.1 and demonstrate in Sec. 5.1.2 how constrained polynomial zonotopes can be applied for program verification using inductive invariants. Moreover, we illustrate the advantages resulting form the usage of the Z-representation for range bounding on polytopic domains in Sec. 5.2.1 and discuss the strong connection of the Z-representation to generalized barycentric coordinates in Sec. 5.2.2.

Chapter 2

Background and Preliminaries

In this chapter, we introduce some general concepts and operations that are required throughout the thesis.

2.1 Set Operations

Just as there are some standard operations like addition and multiplication for scalar numbers, there also exist standard operations for sets. In this thesis we consider the basic set operations linear map, Minkowski sum, Minkowski difference, Cartesian product, convex hull, quadratic map, intersection, union, and set difference. Given sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, and $\mathcal{S}_3 \subset \mathbb{R}^w$, these set operations are defined as follows:

$$-\text{Linear map}: \quad M \otimes \mathcal{S}_1 := \left\{ Ms \;\middle|\; s \in \mathcal{S}_1 \right\} \tag{2.1}$$

$$-\text{Minkowski sum}: \quad \mathcal{S}_1 \oplus \mathcal{S}_2 := \left\{ s_1 + s_2 \;\middle|\; s_1 \in \mathcal{S}_1,\ s_2 \in \mathcal{S}_2 \right\} \tag{2.2}$$

$$-\text{Minkowski diff.}: \quad \mathcal{S}_1 \ominus \mathcal{S}_2 := \left\{ s \;\middle|\; s \oplus \mathcal{S}_2 \subseteq \mathcal{S}_1 \right\} \tag{2.3}$$

$$-\text{Cartesian prod.}: \quad \mathcal{S}_1 \times \mathcal{S}_3 := \left\{ [s_1^T \ s_3^T]^T \;\middle|\; s_1 \in \mathcal{S}_1,\ s_3 \in \mathcal{S}_3 \right\} \tag{2.4}$$

$$-\text{Convex hull}: \quad conv(\mathcal{S}_1, \mathcal{S}_2) := \left\{ \sum_{i=1}^{n+1} \lambda_i \, s_i \;\middle|\; s_i \in \mathcal{S}_1 \cup \mathcal{S}_2,\ \lambda_i \geq 0,\ \sum_{i=1}^{n+1} \lambda_i = 1 \right\} \tag{2.5}$$

$$-\text{Quadratic map}: \quad sq(\mathcal{Q}, \mathcal{S}_1, \mathcal{S}_2) := \left\{ x \,\middle|\, x_{(i)} = s_1^T Q_i s_2, s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2, i = 1 \dots w \right\} \tag{2.6}$$

$$-\text{Intersection}: \quad \mathcal{S}_1 \cap \mathcal{S}_2 := \left\{ s \;\middle|\; s \in \mathcal{S}_1 \wedge s \in \mathcal{S}_2 \right\} \tag{2.7}$$

$$-\text{Union}: \quad \mathcal{S}_1 \cup \mathcal{S}_2 := \left\{ s \;\middle|\; s \in \mathcal{S}_1 \vee s \in \mathcal{S}_2 \right\} \tag{2.8}$$

$$-\text{Set difference}: \quad \mathcal{S}_1 \setminus \mathcal{S}_2 := \left\{ s \;\middle|\; s \in \mathcal{S}_1 \wedge s \notin \mathcal{S}_2 \right\}, \tag{2.9}$$

where $M \in \mathbb{R}^{w \times n}$ is a matrix, and $\mathcal{Q} = \{Q_1, \dots, Q_w\}$ with $Q_i \in \mathbb{R}^{n \times n}$, $i = 1, \dots, w$ is a discrete set of matrices. We use the shorthands $sq(\mathcal{Q}, \mathcal{S}) = sq(\mathcal{Q}, \mathcal{S}, \mathcal{S})$ and $conv(\mathcal{S}) = conv(\mathcal{S}, \mathcal{S})$ for quadratic maps and convex hulls involving a single set $\mathcal{S} \subset \mathbb{R}^n$.

A generalization of the quadratic map are higher-order polynomial maps. Given sets $\mathcal{S}_1, \dots, \mathcal{S}_o \subset \mathbb{R}^n$ and a set of coefficients $\mathcal{A} = \{a^{(i)}_{j_1,\dots,j_o} \in \mathbb{R} \mid i \in \{1, \dots, w\},\ j_1, \dots, j_o \in \{1, \dots, n\}\}$, the polynomial map of order $o \in \mathbb{N}$ defined by the multiplication of the sets

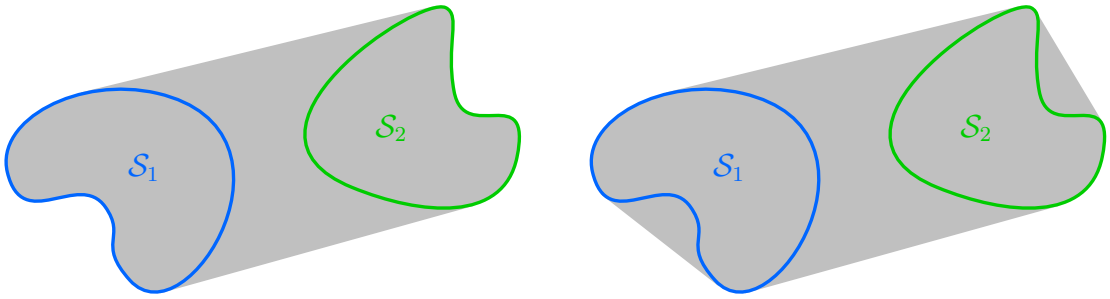

Figure 2.1: Linear combination (left) and convex hull (right) of two non-convex sets $\mathcal{S}_1$ and $\mathcal{S}_2$.

$\mathcal{S}_1, \ldots, \mathcal{S}_o$ with the coefficients in $\mathcal{A}$ is

$$poly(\mathcal{A}, \mathcal{S}_1, \ldots, \mathcal{S}_o) := \Bigg\{ x \;\Bigg|\; x_{(i)} = \sum_{j_1=1}^{n} \ldots \sum_{j_o=1}^{n} a^{(i)}_{j_1,\ldots,j_o} \cdot s_{1(j_1)} \cdot \ldots \cdot s_{o(j_o)}, \; s_1 \in \mathcal{S}_1, \ldots, s_o \in \mathcal{S}_o, \; i = 1, \ldots, w \Bigg\}. \tag{2.10}$$

As for the quadratic map we use the shorthand $poly(\mathcal{A}, \mathcal{S}) = poly(\mathcal{A}, \mathcal{S}, \ldots, \mathcal{S})$ for polynomial maps involving a single set $\mathcal{S}$. The linear map as defined in (2.1) and the quadratic map as defined in (2.6) represent special cases of polynomial maps with order $o = 1$ and $o = 2$, respectively. As we show in Appendix A, all polynomial maps with order $o > 2$ can be computed using a sequence of quadratic maps.

Besides the standard set operations, we consider an additional set operation which we refer to as the *linear combination*. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their linear combination is

$$comb(\mathcal{S}_1, \mathcal{S}_2) := \left\{ \frac{1}{2}(1+\lambda)s_1 + \frac{1}{2}(1-\lambda)s_2 \;\middle|\; s_1 \in \mathcal{S}_1, \; s_2 \in \mathcal{S}_2, \; \lambda \in [-1, 1] \right\}. \tag{2.11}$$

For convex sets, the convex hull and the linear combination are identical. However, for non-convex sets the two operations differ, as it is visualized in Fig. 2.1. We consider both operations since for many algorithms, such as reachability analysis [24, Eq. (3.4)], it is sufficient to compute the linear combination instead of the convex hull, which is often easier and computationally more efficient. Moreover, for connected sets the convex hull can be computed using the linear combination:

Lemma 2.1.1. *Given two connected sets $\mathcal{S}_1 \subset \mathbb{R}^n$ and $\mathcal{S}_2 \subset \mathbb{R}^n$, the relation*

$$conv(\mathcal{S}_1, \mathcal{S}_2) = comb\big(comb(\mathcal{S}_1, \mathcal{S}_1), comb(\mathcal{S}_2, \mathcal{S}_2)\big)$$

holds between the convex hull and the linear combination.

Proof. The definition of the convex hull in (2.5) is based on Carathéodory's theorem [58], which states that even if the set $\mathcal{S}_1 \cup \mathcal{S}_2$ consists of more than $n+1$ disjoint regions, every point in the convex hull can be represented as a linear combination of at most $n+1$ points from $\mathcal{S}_1 \cup \mathcal{S}_2$. For a single connected set $\mathcal{S} \subset \mathbb{R}^n$, which per definition of connectedness

does not have any disjoint regions, it consequently holds that every point in the convex hull can be represented as a linear combination of at most two points from $\mathcal{S}$, yielding

$$\begin{aligned} conv(\mathcal{S}) &= \left\{ \sum_{i=1}^{2} \lambda_i \, s_i \;\middle|\; s_i \in \mathcal{S},\ \lambda_i \geq 0,\ \sum_{i=1}^{2} \lambda_i = 1 \right\} \\ &= \big\{ \lambda_1 s_1 + (1-\lambda_1) s_2 \;\big|\; s_1, s_2 \in \mathcal{S},\ \lambda_1 \in [0,1] \big\}. \end{aligned} \tag{2.12}$$

In addition, it trivially holds that

$$conv(\mathcal{S}_1, \mathcal{S}_2) = conv\big(conv(\mathcal{S}_1), conv(\mathcal{S}_2)\big) \tag{2.13}$$

and

$$\big\{ \lambda_1 \;\big|\; \lambda_1 \in [0,1] \big\} = \big\{ 0.5 + 0.5\lambda \;\big|\; \lambda \in [-1,1] \big\}. \tag{2.14}$$

Moreover, given a set $\mathcal{S} \subset \mathbb{R}^n$ we have according to (2.12) and the definition of the linear combination in (2.11) that

$$\begin{aligned} conv(\mathcal{S}) &\overset{(2.12)}{=} \big\{ \lambda_1 s_1 + (1-\lambda_1) s_2 \;\big|\; s_1, s_2 \in \mathcal{S},\ \lambda_1 \in [0,1] \big\} \overset{(2.14)}{=} \\ &\big\{ 0.5(1+\lambda) s_1 + 0.5(1-\lambda) s_2 \;\big|\; s_1, s_2 \in \mathcal{S},\ \lambda \in [-1,1] \big\} \overset{(2.11)}{=} comb(\mathcal{S}, \mathcal{S}). \end{aligned} \tag{2.15}$$

Using (2.13) and (2.15), we can therefore equivalently formulate the convex hull as

$$conv(\mathcal{S}_1, \mathcal{S}_2) \overset{(2.13)}{=} conv\big(conv(\mathcal{S}_1), conv(\mathcal{S}_2)\big) \overset{(2.15)}{=} conv\big(comb(\mathcal{S}_1, \mathcal{S}_1), comb(\mathcal{S}_2, \mathcal{S}_2)\big),$$

which is identical to $comb(comb(\mathcal{S}_1, \mathcal{S}_1), comb(\mathcal{S}_2, \mathcal{S}_2))$ since the sets $comb(\mathcal{S}_1, \mathcal{S}_1)$ and $comb(\mathcal{S}_2, \mathcal{S}_2)$ are convex according to (2.15). □

Lemma 2.1.1 comes in handy for several derivations and proofs throughout this thesis.

2.2 Set Representations

There exist many different possibilities to represent sets. The most common set representations are introduced in this section. We begin with multi-dimensional intervals:

Definition 2.2.1. *(Interval) [37, Eq. (2.59)] Given a lower bound $l \in \mathbb{R}^n$ and an upper bound $u \in \mathbb{R}^n$ with $l \leq u$, a multi-dimensional interval $\mathcal{I} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{I} := \{ x \in \mathbb{R}^n \mid l \leq x \leq u \}.$$

For a concise notation we use the shorthand $\mathcal{I} = [l, u]$.

Similar to Def. 2.2.1, an interval matrix $\boldsymbol{\mathcal{I}} = [L, U] \subset \mathbb{R}^{n \times n}$, $\forall i, j = 1, \ldots, n:\ L_{(i,j)} \leq U_{(i,j)}$ is a matrix where each matrix entry is an interval. For polytopes, the two most common representations are the halfspace representation and the vertex representation. The halfspace representation expresses a polytope as the intersection of halfspaces:

Definition 2.2.2. *(Polytope H-Representation) [24, Def. 2.1] Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, the halfspace representation of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is defined as*

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

If $\mathcal{P}$ is unbounded, the set is called a polyhedron. The H-representation is redundant if $\mathcal{P}$ can be equivalently represented with less inequality constraints. For a concise notation we use the shorthand $\mathcal{P} = \langle A, b \rangle_H$ for the H-representation.

On the other hand, the vertex representation expresses a polytope as the convex hull of its vertices:

Definition 2.2.3. *(Polytope V-Representation) [24, Def. 2.2] Given the s polytope vertices $v_i \in \mathbb{R}^n$, the vertex representation of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is defined as*

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

The V-representation is redundant if the matrix $[v_1 \ \dots \ v_s]$ contains points that are not vertices of $\mathcal{P}$. For a concise notation we use the shorthand $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V$ for the V-representation.

Zonotopes are a special class of polytopes, which can be represented very compactly using so-called generators:

Definition 2.2.4. *(Zonotope) [25, Def. 1] Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times l}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is defined as*

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

The scalars α_i are called factors, the vectors $G_{(\cdot,i)}$ are called generators, and the zonotope order is defined as $\rho = \frac{l}{n}$. For a concise notation we use the shorthand $\mathcal{Z} = \langle c, G \rangle_Z$.

Constrained zonotopes, which were introduced in [32] and can equivalently represent any bounded polytope, extend zonotopes by restricting the values for the zonotope factors by equality constraints:

Definition 2.2.5. *(Constrained Zonotope) [32, Def. 3] Given a constant offset $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times l}$, a constraint matrix $A \in \mathbb{R}^{m \times l}$, and a constraint vector $b \in \mathbb{R}^m$, a constrained zonotope $\mathcal{CZ} \subset \mathbb{R}^n$ is defined as*

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

For a concise notation we use the shorthand $\mathcal{CZ} = \langle c, G, A, b \rangle_{CZ}$.

An ellipsoid can be interpreted as the linear map of a sphere:

Definition 2.2.6. *(Ellipsoid) [8, Eq. 2.3] Given a center vector $c \in \mathbb{R}^n$ and a symmetric and positive definite matrix $Q \in \mathbb{R}^{n \times n}$, an ellipsoid $\mathcal{E} \subset \mathbb{R}^n$ is defined as*

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

For a concise notation, we use the shorthand $\mathcal{E} = \langle c, Q \rangle_E$.

Support functions can equivalently represent any convex set:

Definition 2.2.7. *(Support Function) [4, Def. 1] Given a set $\mathcal{S} \subset \mathbb{R}^n$ and a direction $d \in \mathbb{R}^n$, the support function $s_{\mathcal{S}} : \mathbb{R}^n \to \mathbb{R}$ of $\mathcal{S}$ is defined as*

$$s_{\mathcal{S}}(d) = \max_{x \in \mathcal{S}} \; d^T x.$$

The set $\mathcal{S}$ can therefore be equivalently defined by specifying a closed-form expression for its support function $s_{\mathcal{S}}(d)$.

While all set representations considered so far are convex, Taylor models can represent non-convex sets:

Definition 2.2.8. *(Taylor Model) [43, Def. 2.1] Given a vector field $w : \mathbb{R}^r \to \mathbb{R}^n$, where each subfunction $w_{(i)} : \mathbb{R}^r \to \mathbb{R}$ is a polynomial function*

$$w_{(i)}(x) = \sum_{j=1}^{m_i} b_{i,j} \prod_{k=1}^{r} x_{(k)}^{E_{i(k,j)}}, \quad i = 1, \ldots, n,$$

an interval domain $\mathcal{I} \subset \mathbb{R}^r$, and an interval remainder $\mathcal{Y} \subset \mathbb{R}^n$, a Taylor model $\mathcal{T}(x) \subset \mathbb{R}^n$ is defined as

$$\forall x \in \mathcal{I}: \quad \mathcal{T}(x) := \left\{ \begin{bmatrix} w_{(1)}(x) \\ \vdots \\ w_{(n)}(x) \end{bmatrix} + \begin{bmatrix} y_{(1)} \\ \vdots \\ y_{(n)} \end{bmatrix} \;\middle|\; y \in \mathcal{Y} \right\},$$

where $E_i \in \mathbb{N}_0^{r \times m_i}$ is an exponent matrix and $b_{i,j} \in \mathbb{R}$ are the polynomial coefficients. While we consider the more general case with arbitrary interval domains $\mathcal{I}$, Taylor models are also often defined for normalized domains $\mathcal{I} = [-\mathbf{1}, \mathbf{1}]$. The set defined by a Taylor model is $\mathcal{S} = \{\mathcal{T}(x) \mid x \in \mathcal{I}\}$. For a concise notation we use the shorthand $\mathcal{T}(x) = \langle w(x), \mathcal{Y}, \mathcal{I} \rangle_{TM}$.

Also level sets can represent non-convex sets:

Definition 2.2.9. *(Level Set) [22, Def. 10] Given a vector field $w : \mathbb{R}^n \to \mathbb{R}^o$, a level set $\mathcal{LS} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{LS} := \big\{ x \;\big|\; w(x) \prec \mathbf{0} \big\},$$

where $\prec \in \{<, \leq, =\}$. For a concise notation, we use the shorthand $\mathcal{LS} = \langle w(x), \prec \rangle_{LS}$.

Relations between the different set representations are illustrated in Fig. 1.2.

2.3 Dynamic Systems and Reachable Sets

In order to verify the correct functionality of cyber-physical systems, a mathematical description of the system behavior is required. In this section, we present different types of system models and introduce reachable sets. The dynamic behavior of a system can often be represented with an ordinary differential equation.

In the simplest case, this differential equation is linear:

Definition 2.3.1. *(Linear System) Given a system matrix $A \in \mathbb{R}^{n\times n}$ and an input matrix $B \in \mathbb{R}^{n\times m}$, a linear system is defined by the differential equation*

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{2.16}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the vector of system inputs, and $t \in \mathbb{R}_{\geq 0}$ is the time.

While linear systems are often easy to analyze and verify due to their simplicity, the dynamics for most systems in the real world is actually nonlinear:

Definition 2.3.2. *(Nonlinear System) Given a nonlinear Lipschitz continuous function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, a nonlinear system is defined by the differential equation*

$$\dot{x}(t) = f\big(x(t), u(t)\big), \tag{2.17}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the vector of system inputs, and $t \in \mathbb{R}_{\geq 0}$ is the time.

Linear systems as defined in Def. 2.3.1 represent a special case of nonlinear systems. A further generalization of nonlinear systems are hybrid systems, which divide the state-space into multiple discrete regions with different continuous dynamics. In this thesis, we represent hybrid systems by hybrid automata:

Definition 2.3.3. *(Hybrid Automaton) A hybrid automaton $\mathscr{H}$ with p discrete modes consists of:*

- *A list $\mathbf{F} = (f_1(x,u), \ldots, f_p(x,u))$ storing the differential equations $\dot{x}(t) = f_i(x(t), u(t))$ describing the continuous dynamics in each mode $i = 1, \ldots, p$.*
- *A list $\mathbf{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_p)$ storing the invariant sets $\mathcal{Y}_i \subset \mathbb{R}^n$ for each mode $i = 1, \ldots, p$, where the invariant set describes the region where the continuous dynamics of a mode is active.*
- *A list $\mathbf{T} = (\mathscr{T}_1, \ldots, \mathscr{T}_q)$ of transitions $\mathscr{T}_i = \langle \mathcal{G}_i, r_i(x), s_i, d_i \rangle_T$, $i = 1, \ldots, q$ between discrete modes, where $\mathcal{G}_i \subset \mathbb{R}^n$ is a guard set, $r_i : \mathbb{R}^n \to \mathbb{R}^n$ is a reset function, and $s_i, d_i \in \{1, \ldots, p\}$ are the indices of the source and target modes.*

The state of a hybrid automaton consists of the continuous state $x(t) \in \mathbb{R}^n$ and the discrete state $\sigma(t) \in \{1, \ldots, p\}$. We use the shorthand $\mathscr{H} = \langle \mathbf{F}, \mathbf{Y}, \mathbf{T} \rangle_{HA}$.

The evolution of a hybrid automaton is described informally as follows: Given an initial continuous state $x_0 = x(0)$ and an initial discrete state $\sigma_0 = \sigma(0)$ with $x_0 \in \mathcal{Y}_{\sigma_0}$, the continuous state $x(t)$ evolves according to the flow function $f_{\sigma_0}(x,u)$ of mode σ_0. If $x(t)$ is within the guard set $\mathcal{G}_i$ of a transition $\mathscr{T}_i = \langle \mathcal{G}_i, r_i(x), s_i, d_i \rangle_T \in \mathbf{T}$ with $s_i = \sigma_0$, the transition to the mode d_i is taken and the continuous state $x(t)$ is updated according to the reset function $r_i(x)$. Afterward, the evolution of the continuous state continues according to the flow function $f_{d_i}(x,u)$ of mode d_i until the next transition is taken. We denote the trajectory of the continuous state for the evolution of the hybrid automaton described above by $\xi(t, x_0, \sigma_0, u(\cdot))$, where $u(\cdot) \in \mathbb{R}^m$ is an input trajectory.

To verify that cyber-physical systems modeled as linear systems, nonlinear systems, or hybrid automata behave as desired, we use reachable sets:

Definition 2.3.4. *(Reachable Set) Given an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ and a set of uncertain inputs $\mathcal{U} \subset \mathbb{R}^m$, the reachable set for a linear or nonlinear system is*

$$\mathcal{R}(t) := \left\{ \xi\big(t, x_0, u(\cdot)\big) \;\middle|\; x_0 \in \mathcal{X}_0,\ \forall \tau \in [0,t] :\ u(\tau) \in \mathcal{U} \right\},$$

and the reachable set for a hybrid automaton is

$$\mathcal{R}(t) := \left\{ \xi\big(t, x_0, \sigma_0, u(\cdot)\big) \;\middle|\; x_0 \in \mathcal{X}_0,\ \forall \tau \in [0,t] :\ u(\tau) \in \mathcal{U} \right\},$$

where $\xi(t, x_0, u(\cdot))$ denotes the solution to (2.16) or (2.17) for an initial state $x(0) = x_0$ and the input trajectory $u(\cdot)$, and $\xi(t, x_0, \sigma_0, u(\cdot))$ denotes the evolution of a hybrid automaton for a continuous initial state $x(0) = x_0$, a discrete initial state $\sigma(0) = \sigma_0$, and the input trajectory $u(\cdot)$.

Sometimes, we use the notation $\mathcal{R}_{\mathcal{X}_0}(t)$ to denote the reachable set starting from the initial set $\mathcal{X}_0$. Moreover, without loss of generality we assume throughout this thesis that the initial time is equal to zero.

2.4 Standard Operations

In this section we present some mathematical standard operations and specify their computational complexity. We begin with the sorting operation:

Definition 2.4.1. *(Sorting) Given a vector $d \in \mathbb{R}^n$, the operation* `sort` *sorts the entries of d in ascending order and returns a vector $o \in \mathbb{N}^n$ that stores the indices of the sorted vector:*

$$\texttt{sort}(d) = o \quad \textit{with} \quad d_{(o_{(1)})} \leq \ldots \leq d_{(o_{(n)})}.$$

The computational complexity of sorting a vector is $\mathcal{O}(n \log(n))$ [59, Ch. 5].

Next, we consider the eigenvalue decomposition:

Definition 2.4.2. *(Eigenvalue Decomposition) Given a square matrix $A \in \mathbb{R}^{n \times n}$, eigenvalue decomposition computes the eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{I}$ and an orthonormal matrix $V \in \mathbb{I}^{n \times n}$ that stores the eigenvectors, such that*

$$A = V \underbrace{\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}}_{D} V^T.$$

The above decomposition is only possible if the matrix A is diagonalizable. For symmetric matrices all eigenvalues and eigenvectors are real, and all symmetric matrices are diagonalizable. The computational complexity is $\mathcal{O}(n^3)$ [60].

Singular value decomposition (SVD) can be interpreted as an extension of eigenvalue decomposition to non-square matrices:

Definition 2.4.3. *(Singular Value Decomposition) Given a matrix $A \in \mathbb{R}^{n\times m}$, singular value decomposition computes an orthonormal matrix $U \in \mathbb{R}^{n\times n}$, a rectangular diagonal matrix $\Sigma \in \mathbb{R}^{n\times m}$, and an orthonormal matrix $V \in \mathbb{R}^{m\times m}$ such that*

$$A = U\Sigma V^T.$$

The computational complexity is $\mathcal{O}(n^2m + n^3)$ [61, Tab. 1].

Principal component analysis (PCA) applies SVD to determine the main directions of a point cloud:

Definition 2.4.4. *(Principal Component Analysis) Given points $x_1, \ldots, x_m \in \mathbb{R}^n$ stored in a matrix $X = [x_1 \ \ldots \ x_m] \in \mathbb{R}^{n\times m}$, principal component analysis computes an orthonormal matrix $U = [u_1 \ \ldots \ u_n] \in \mathbb{R}^{n\times n}$ consisting of the n orthonormal direction vectors $u_1, \ldots, u_n \in \mathbb{R}^n$ that maximize the variance by using the following SVD:*

$$\widehat{X} = U\Sigma V^T \quad \textit{with} \quad \widehat{X} = X - c \cdot \mathbf{1}, \ c = \frac{1}{m}\sum_{i=1}^{m} X_{(\cdot,i)}.$$

For a concise notation, we use the shorthand $U = \texttt{pca}(X)$. The computational complexity is $\mathcal{O}(n^2m + n^3)$ [62, Sec. 2].

The convex hull of a point cloud removes all points that are not extreme points:

Definition 2.4.5. *(Convex Hull) Given points $x_1, \ldots, x_m \in \mathbb{R}^n$ stored in a matrix $X = [x_1 \ \ldots \ x_m] \in \mathbb{R}^{n\times m}$, the* `convHull` *operation returns the extreme points:*

$$\texttt{convHull}(X) = X_{(\cdot,\mathcal{K})},$$

where

$$\mathcal{K} = \left\{ i \;\middle|\; \forall \beta_1, \ldots, \beta_m \geq 0 : \ \left(\left(x_i = \sum_{j=1}^{m} \beta_j x_j \right) \wedge \left(\sum_{j=1}^{m} \beta_j = 1 \right) \right) \Rightarrow (\beta_i = 1) \right\}.$$

The computational complexity for calculation of the convex hull depends on the algorithm. We consider the Beneath-Beyond algorithm [63], which has according to [64, Thm. 3.16] complexity $\mathcal{O}(m^{\lfloor n/2 \rfloor + 1})$.

Linear programs are a special type of optimization problem that can be solved very efficiently:

Definition 2.4.6. *(Linear Program) Given a vector $c \in \mathbb{R}^n$ defining the objective function, a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$ defining the inequality constraints, and a matrix $A_{eq} \in \mathbb{R}^{p\times n}$ and a vector $b_{eq} \in \mathbb{R}^p$ defining the equality constraints, a linear program is defined as*

$$\begin{aligned} x^* = \min_{x\in\mathbb{R}^n} \ & c^T x \\ s.t. \ \ & Ax \leq b \\ & A_{eq}x = b_{eq}. \end{aligned}$$

Table 2.1: Computational complexity of standard operations.

Operation	Description	Complexity	Reference
Matrix multiplication	$A \cdot B,\ A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{m\times p}$	$\mathcal{O}(nmp)$	[67, Ch. 1.1]
Sorting	$\texttt{sort}(d),\ d \in \mathbb{R}^n$	$\mathcal{O}(n\log(n))$	[59, Ch. 5]
Eigenvalue decomp.	$A = VDV^T,\ A \in \mathbb{R}^{n\times n}$	$\mathcal{O}(n^3)$	[60]
Singular value decomp.	$A = U\Sigma V^T,\ A \in \mathbb{R}^{n\times m}$	$\mathcal{O}(n^2m + n^3)$	[61, Tab. 1]
Principal comp. analysis	$\texttt{pca}(X),\ X \in \mathbb{R}^{n\times m}$	$\mathcal{O}(n^2m + n^3)$	[62, Sec. 2]
Convex hull	$\texttt{convHull}(X),\ X \in \mathbb{R}^{n\times m}$	$\mathcal{O}(m^{\lfloor n/2\rfloor+1})$	[64, Thm. 3.16]
Linear program	n variables	$\mathcal{O}(n^{3.5})$	[65]
Convex quad. program	n variables	$\mathcal{O}(n^4)$	[66, Thm. 4.1]

We use the shorthand $x^* = \texttt{linProg}(c, A, b, A_{eq}, b_{eq})$. *The computational complexity for linear programming depends on the algorithm. We consider the algorithm in [65], which has computational complexity* $\mathcal{O}(n^{3.5})$ *with respect to the number of optimization variables* n.

Quadratic programs are a generalization of linear programs. We consider convex quadratic programs, which can be solved exactly:

Definition 2.4.7. *(Convex Quadratic Program) Given a positive semi-definite matrix* $Q \in \mathbb{R}^{n\times n}$ *and a vector* $c \in \mathbb{R}^n$ *defining the objective function, a matrix* $A \in \mathbb{R}^{m\times n}$ *and a vector* $b \in \mathbb{R}^m$ *defining the inequality constraints, and a matrix* $A_{eq} \in \mathbb{R}^{p\times n}$ *and a vector* $b_{eq} \in \mathbb{R}^p$ *defining the equality constraints, a convex quadratic program is defined as*

$$\begin{aligned} x^* = \min_{x\in\mathbb{R}^n}\ & x^TQx + c^Tx \\ s.t.\ \ & Ax \leq b \\ & A_{eq}x = b_{eq}. \end{aligned}$$

We use the shorthand $x^* = \texttt{quadProg}(Q, c, A, b, A_{eq}, b_{eq})$. *The computational complexity for solving convex quadratic programs depends on the algorithm. We consider the algorithm in [66], which has according to [66, Thm. 4.1] computational complexity* $\mathcal{O}(n^4)$ *with respect to the number of optimization variables* n.

A summary of the computational complexities for all standard operations presented in this section is provided in Tab. 2.1.

2.5 Auxiliary Operations

In this section we introduce some useful auxiliary operations and derive their computational complexity. We begin with the operation `sortColumns`, which sorts the columns of a matrix

Algorithm 1 Sort matrix columns

Require: Matrix $A \in \mathbb{R}^{n \times m}$
Ensure: Sorted matrix $A \in \mathbb{R}^{n \times m}$

1: $o \leftarrow \texttt{sort}(A_{(1,\cdot)})$
2: $A \leftarrow [A_{(\cdot,o_{(1)})} \dots A_{(\cdot,o_{(m)})}]$
3: **for** $i \leftarrow 1$ to $n-1$ **do**
4: $\quad \texttt{cnt} \leftarrow 1$
5: $\quad$ **for** $j \leftarrow 1$ to $m+1$ **do**
6: $\quad\quad$ **if** $(j > m) \vee (A_{(1,j)} \neq A_{(1,\texttt{cnt})}) \vee \dots \vee (A_{(i,j)} \neq A_{(i,\texttt{cnt})})$ **then**
7: $\quad\quad\quad$ **if** $j > \texttt{cnt} + 1$ **then**
8: $\quad\quad\quad\quad \mathcal{H} \leftarrow \{\texttt{cnt}, \dots, j-1\}$
9: $\quad\quad\quad\quad \overline{A} \leftarrow A_{(\cdot,\mathcal{H})}$
10: $\quad\quad\quad\quad o \leftarrow \texttt{sort}(\overline{A}_{(i+1,\cdot)})$
11: $\quad\quad\quad\quad A \leftarrow [A_{(\cdot,1)} \ \dots \ A_{(\cdot,\texttt{cnt}-1)} \ \overline{A}_{(\cdot,o_{(1)})} \ \dots \ \overline{A}_{(\cdot,o_{(|\mathcal{H}|)})} \ A_{(\cdot,j)} \ \dots \ A_{(\cdot,m)}]$
12: $\quad\quad\quad$ **end if**
13: $\quad\quad\quad \texttt{cnt} \leftarrow j$
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: **end for**

$A \in \mathbb{R}^{n \times m}$ in ascending order. Decisive for the order are the entries in the first matrix row, where for duplicate entries the order is determined by the entries in the second row, and so on and so forth. For an exemplary matrix A, this yields the result

$$A = \begin{bmatrix} 1 & 2 & 5 & 2 & 1 & 1 \\ 0 & 1 & 3 & 2 & 4 & 4 \\ 4 & 2 & 1 & 0 & 1 & 2 \end{bmatrix}, \quad \texttt{sortColumns}(A) = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 5 \\ 0 & 4 & 4 & 1 & 2 & 3 \\ 4 & 1 & 2 & 2 & 0 & 1 \end{bmatrix}.$$

The implementation of `sortColumns` is shown in Alg. 1. First, the matrix columns are sorted according to the entries in the first row in lines 1-2 of Alg. 1. Afterward, the for-loop in lines 3-16 iterates of the remaining matrix rows and sorts columns with identical entries in the first i rows according to the entry in row $i+1$. The computational complexity of `sortColumns` is as follows:

Proposition 2.5.1. *(Sort Matrix Columns) Given a matrix $A \in \mathbb{R}^{n \times m}$, the computational complexity of sorting the matrix columns with the operation* $\texttt{sortColumns}(A)$ *as implemented in Alg. 1 is $\mathcal{O}(nm\log(m))$.*

Proof. To derive the computational complexity of Alg. 1, we first consider a matrix with only two columns for simplicity. In this case the matrix columns are first sorted according to the entries in the first row in Line 1 of Alg. 1, which has according to Tab. 2.1 computation

Algorithm 2 Remove duplicate matrix columns

Require: Matrix $A \in \mathbb{R}^{n \times m}$
Ensure: Matrix $C \in \mathbb{R}^{n \times k}$ storing the non-redundant columns of A

```
1: A ← sortColumns(A)
2: C ← A_(·,1)
3: cnt ← 1
4: for i ← 2 to m do
5:     if C_(·,cnt) ≠ A_(·,i) then
6:         C ← [C A_(·,i))]
7:         cnt ← i
8:     end if
9: end for
```

complexity $\mathcal{O}(m \log(m))$. Next, in lines 3-16, the second matrix row is split into $K \in \{1, \ldots, m\}$ parts of variable length $m_i \in \{1, \ldots, m\}$ with $\sum_{i=1}^{K} m_i = m$, where each part consists of columns with identical entries in the first matrix row. Finally, each of these parts is sorted according to the values in the second matrix row in Line 11. Since the complexity for sorting each part is $\mathcal{O}(m_i \log(m_i)$, the overall complexity for Alg. 1 is

$$\mathcal{O}(m \log(m)) + \sum_{i=1}^{K} \mathcal{O}(m_i \log(m_i)) = \mathcal{O}(m \log(m)) + \mathcal{O}\Bigg(\sum_{i=1}^{K} m_i \underbrace{\log(m_i)}_{\leq \log(m)} \Bigg)$$

$$\leq \mathcal{O}(m \log(m)) + \mathcal{O}\Bigg(\log(m) \underbrace{\sum_{i=1}^{K} m_i}_{=m} \Bigg) = 2 \cdot \mathcal{O}(m \log(m)).$$

For the general case with n instead of two matrix rows this procedure is repeated n times, resulting in an overall complexity of $n \cdot \mathcal{O}(m \log(m)) = \mathcal{O}(nm \log(m))$. □

Next, we consider the operation `uniqueColumns` which removes all duplicate columns from a matrix $A \in \mathbb{R}^{n \times m}$. The implementation of `uniqueColumns` is specified in Alg. 2. First, the matrix columns are sorted in Line 1 of Alg. 2 using the operation `sortColumns`. Afterward, the for-loop in lines 4-9 iterates over all columns and selects all columns that are not duplicates. The computational complexity is as follows:

Proposition 2.5.2. *(Unique Columns) Given a matrix $A \in \mathbb{R}^{n \times m}$, the computational complexity of removing duplicate matrix columns with the operation* `uniqueColumns`(A) *as implemented in Alg. 2 is $\mathcal{O}(nm \log(m))$.*

Proof. Sorting the matrix columns with the operation `sortColumns` in Line 1 of Alg. 2 has according to Prop. 2.5.1 a computational complexity of $\mathcal{O}(nm \log(m))$. In each iteration of the for-loop in lines 4-9 we check in Line 5 if two vectors are identical, which has complexity $\mathcal{O}(n)$. Since the for-loop consists of m iterations, the complexity of the whole for-loop is

Algorithm 3 Generate unique identifiers

Require: Number of requested unique identifiers $n \in \mathbb{N}$
Ensure: Vector $id \in \mathbb{N}^{1\times n}$ storing the generated identifiers

1: `cnt` $\leftarrow$ load current counter from memory
2: $id \leftarrow [\,]$
3: **for** $i \leftarrow 1$ to n **do**
4: `cnt` $\leftarrow$ `cnt` $+ 1$
5: $id \leftarrow [id$ `cnt`$]$
6: **end for**
7: `cnt` $\rightarrow$ write current counter to memory

Table 2.2: Computational complexity of auxiliary operations.

Operation	Description	Complexity
Sort matrix columns	`sortColumns`$(A),\ A \in \mathbb{R}^{n\times m}$	$\mathcal{O}(nm\log(m))$
Remove duplicate matrix columns	`uniqueColumns`$(A),\ A \in \mathbb{R}^{n\times m}$	$\mathcal{O}(nm\log(m))$
Generate unique identifiers	`uniqueID`$(n),\ n \in \mathbb{N}$	$\mathcal{O}(n)$

$\mathcal{O}(mn)$, which results in an overall complexity of $\mathcal{O}(nm\log(m)) + \mathcal{O}(nm) = \mathcal{O}(nm\log(m))$ for Alg. 2. □

Finally, we consider the operation `uniqueID` which generates a vector of unique integer identifiers. Alg. 3 shows the implementation of `uniqueID`. The counter `cnt` that is stored in memory can be initialized with zero at a beginning of an algorithm that uses unique identifiers. The computational complexity for `uniqueID` is as follows:

Proposition 2.5.3. *(Unique Identifiers) The computational complexity for generating* $n \in \mathbb{N}$ *unique integer identifiers with the operation* `uniqueID` *as implemented in Alg. 3 is* $\mathcal{O}(n)$.

Proof. In the for-loop in lines 3-6 of Alg. 3 n additions are performed, so that the overall complexity is $\mathcal{O}(n)$. □

An overview of the computational complexities for the auxiliary operations introduced in this section is shown in Tab. 2.2.

2.6 Zonotope Order Reduction

For zonotopes, repeated calculation of Minkowski sums as required by many algorithms significantly increased the number of generators, and consequently the representation size. To prevent exploding computation times when computing with zonotopes, one therefore requires efficient methods for representation size reduction. Since the zonotope order ρ

Table 2.3: Computational complexity $\mathcal{O}(\texttt{reduce})$ of zonotope order reduction methods, where $n \in \mathbb{N}$ is the dimension of the zonotope, $l \in \mathbb{N}_0$ is the number of zonotope generators, and $k \in \mathbb{N}_0$ is the number of generators that are reduced.

Method	Complexity	Complexity using Assumption 2.6.2	Reference
Girard	$\mathcal{O}(n(l+k) + l\log(l))$	$\mathcal{O}(n^2)$	[25, Sec. 3.4]
PCA	$\mathcal{O}(l\log(l) + n^3 + n^2k + nl)$	$\mathcal{O}(n^3)$	[68, Sec. III.A]
Scott	$\mathcal{O}(n^2l + kln)$	$\mathcal{O}(n^3)$	[32, Appendix]
Exh. Search	$\mathcal{O}\left(\binom{k}{n}k\right)$	$\mathcal{O}\left(\binom{k}{n}k\right)$	[69, Sec. 5.3]

represents a dimensionless measure for the complexity of a zonotope, representation size reduction is commonly realized by reducing the zonotope order to a desired maximum value using zonotope order reduction:

Definition 2.6.1. *(Zonotope Order Reduction) Given a zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ and a desired zonotope order $\rho_d \geq 1$, the operation* `reduce` *defined as*

$$\texttt{reduce}(\mathcal{Z}, \rho_d) \supseteq \mathcal{Z}$$

returns a zonotope with order smaller than or equal to ρ_d that encloses $\mathcal{Z}$.

There exist many different approaches for zonotope order reduction. An overview is provided in [68]. We shortly introduce some of the most commonly used methods and derive their computational complexity. For this, we require the following assumption:

Assumption 2.6.2. *Given a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ with $l \in \mathbb{N}_0$ generators, we assume for the derivation of the computational complexity that*

$$l = c_l\, n,$$

with $c_l \in \mathbb{R}_{\geq 0}$.

The assumption is justified by the fact that the zonotope order $\rho = \frac{l}{n}$ is reduced to the desired order ρ_d when computing with zonotopes, such that $l \leq \rho_d n$ holds. We first consider the order reduction approach proposed in [25, Sec. 3.4], which we refer to as Girard's method. Given a zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ this method first sorts the generators according to the following metric:

$$||G_{(\cdot,o_{(1)})}||_1 - ||G_{(\cdot,o_{(1)})}||_\infty \leq \ldots \leq ||G_{(\cdot,o_{(l)})}||_1 - ||G_{(\cdot,o_{(l)})}||_\infty, \tag{2.18}$$

where the vector $o \in \mathbb{N}^l$ stores the indices of the sorted generators. Next, the $k = \lceil \max(0, l - n(\rho_d - 1)) \rceil$ smallest generators according to the matrix in (2.18) are selected and enclosed by an interval, which results in the reduced zonotope:

$$\texttt{reduce}(\mathcal{Z}, \rho_d) = \left\langle c, \left[G_{(\cdot,\mathcal{K})} \;\; \texttt{diag}\left(\sum_{i \in \mathcal{H}} |G_{(\cdot,i)}| \right) \right] \right\rangle_Z, \tag{2.19}$$

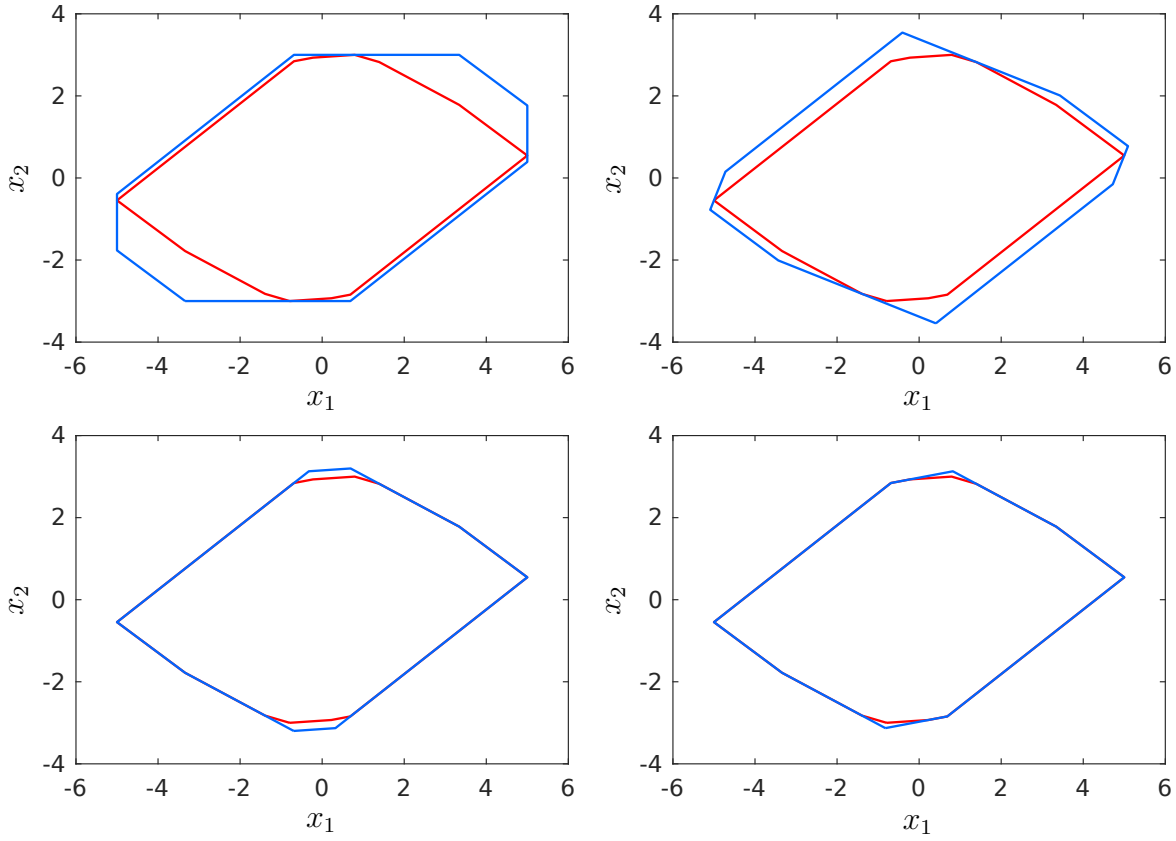

Figure 2.2: Order reduction for an exemplary zonotope with order $\rho = 3$ to the desired order $\rho_d = 2$ using Girard's method (top, left), PCA (top, right), Scott's method (bottom, left), and exhaustive search (bottom, right), where the original zonotope is depicted in red, and the reduced zonotope is depicted in blue.

where $\mathcal{H} = \{o_{(1)}, \dots, o_{(k)}\}$ and $\mathcal{K} = \{o_{(k+1)}, \dots, o_{(l)}\}$. Computation of the norms in (2.18) has complexity $\mathcal{O}(nl)$, and the complexity of sorting is according to Tab. 2.1 $\mathcal{O}(l \log(l))$. Finally, the summation of the generators in (2.19) has complexity $\mathcal{O}(kn)$, which results in an overall complexity of $\mathcal{O}(nl) + \mathcal{O}(l \log(l)) + \mathcal{O}(kn) = \mathcal{O}(n(l+k) + l \log(l))$. Since $k \leq l$ holds, this can be simplified to $\mathcal{O}(n^2)$ using Assumption 2.6.2.

A simple extension to Girard's method is to incorporate principal component analysis [68, Sec. III.A], where prior to the interval enclosure the generators that get reduced are transformed to a different coordinate system that is determined using PCA:

$$\texttt{reduce}(\mathcal{Z}, \rho_d) = \left\langle c, \left[G_{(\cdot,\mathcal{K})} \;\; U \; \texttt{diag}\left(\sum_{i \in \mathcal{H}} U^T |G_{(\cdot,i)}| \right) \right] \right\rangle_Z,$$

where the transformation matrix $U = \texttt{pca}(X)$ is determined by applying the `pca` operation as defined in Def. 2.4.4 to the matrix $X = [-G_{(\cdot,\mathcal{H})} \; G_{(\cdot,\mathcal{H})}]$. Since PCA aligns the coordinate system with the main directions of the reduced generators, the interval enclosure for PCA-based order reduction is expected to be tighter than for Girard's method. PCA has complexity $\mathcal{O}(2n^2k + n^3) = \mathcal{O}(n^2k + n^3)$ according to Tab. 2.1 and the matrix multiplications with U and U^T have complexity $\mathcal{O}(n^2k)$ according to Tab. 2.1. Together with the complexity for Girard's method we therefore obtain an overall complexity of

$\mathcal{O}(n(l+k)+l\log(l))+\mathcal{O}(n^2k+n^3)+\mathcal{O}(n^2k)=\mathcal{O}(l\log(l)+n^3+n^2k+nl)$, which is $\mathcal{O}(n^3)$ using Assumption 2.6.2.

The reduction methods discussed so far are based on heuristics, and consequently often not optimal. For non-degenerate zonotopes with $l = n + 1$ generators, the enclosing parallelotope with minimum volume can be computed efficiently using [36, Thm. 3]. The approach in [32, Appendix], which we refer to as Scott's method, recursively applies [36, Thm. 3] to reduce general zonotopes with $l > n + 1$ generators, where the generators that are reduced are selected with a heuristic. The computational complexity is $\mathcal{O}(n^2l + kln)$ [32, Appendix], which is $\mathcal{O}(n^3)$ using Assumption 2.6.2. While Scott's method uses a heuristic to select suitable generators, the approach in [69, Sec. 5.3] explicitly determines the optimal generators by applying an exhaustive search. This, however, comes at the cost of high computational complexity $\mathcal{O}\big(\binom{k}{n}k\big)$ for $k \geq n$ [69, Prop. 10]. The computational complexities for the different zonotope order reduction methods presented in this section are summarized in Tab. 2.3. A comparison of the different methods for order reduction of an exemplary zonotope is shown in Fig. 2.2.

2.7 Range Bounding

It is usually infeasible to determine the minimum and maximum function value on a certain domain exactly for general nonlinear functions with multiple variables. However, for many applications it is sufficient to compute tight bounds for the function values instead, which can be achieved using range bounding:

Definition 2.7.1. *(Range Bounding) Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a domain $\mathcal{D} \subset \mathbb{R}^n$, the range bounding operation*

$$\texttt{bound}\big(f(x), \mathcal{D}\big) \supseteq \left[\min_{x\in\mathcal{D}} f(x),\ \max_{x\in\mathcal{D}} f(x)\right]$$

returns an enclosure of the exact bounds.

For range bounding of vector fields $g : \mathbb{R}^n \to \mathbb{R}^m$ we use the shorthand notation $\texttt{bound}(g(x), \mathcal{D}) = \texttt{bound}(g_{(1)}(x), \mathcal{D}) \times \ldots \times \texttt{bound}(g_{(m)}(x), \mathcal{D})$. There exist several different approaches for range bounding, the simplest one being *interval arithmetic* [37, Ch. 2.3]. To compute guaranteed bounds, interval arithmetic defines closed-form expressions for the calculation of enclosing intervals for all elementary operations such as $+$, $-$, or $\sin(\cdot)$ [70]. Successive application of these closed-form expressions to all elementary operations in the function $f(x)$ then yields guaranteed bounds for the function values. Let us consider the range bounding problem $\texttt{bound}(f(x), \mathcal{D})$ with

$$f(x) = \sin(x) - x^3 + x, \quad \mathcal{D} = [-1, 2], \tag{2.20}$$

for which interval arithmetic yields the bounds

$$\begin{aligned}\texttt{bound}\big(f(x), \mathcal{D}\big) &= \texttt{bound}\big(\sin(x) - x^3 + x, [-1, 2]\big)\\ &= \sin([-1, 2]) - [-1, 2]^3 + [-1, 2] = [-0.842, 1] - [-1, 8] + [-1, 2] = [-9.842, 4].\end{aligned}$$

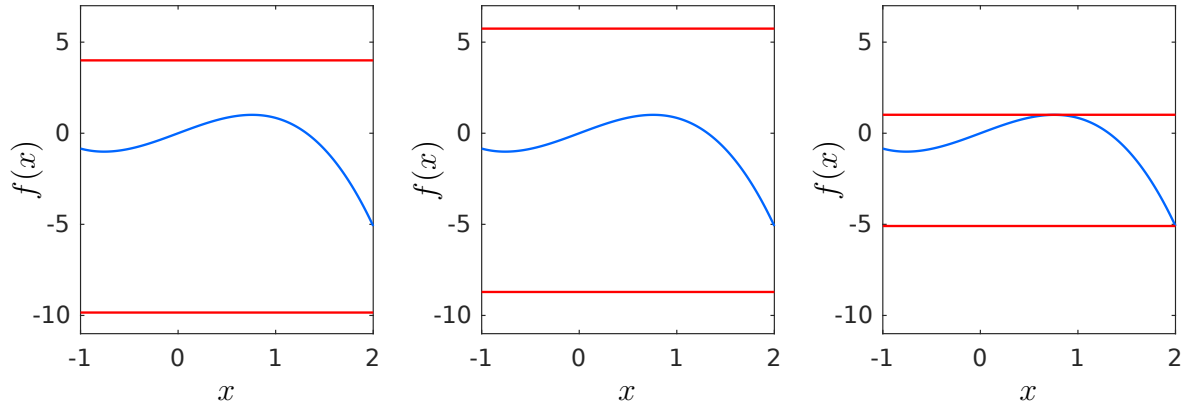

Figure 2.3: Bounds for the range bounding problem in (2.20) computed with interval arithmetic (left), affine arithmetic (middle), and Taylor models (right).

This small example also nicely demonstrates the main disadvantage of interval arithmetic: The expressions $\sin(x)$, $-x^3$, and x are treated as if they were independent from each other, even though they share a common variable x. In particular, this means that interval arithmetic for example computes the upper bound by summation of the maxima for $\sin(x)$, $-x^3$, and x. However, since $\sin(x)$ reaches its maximum value on $\mathcal{D}$ at $x = \pi/2$, $-x^3$ reaches its maximum value on $\mathcal{D}$ at $x = -1$, and x reaches its maximum value on $\mathcal{D}$ at $x = 2$, it is not possible that all expressions reach their maximum value at the same x. This loss of dependency between expressions is called the *dependency problem* [30, Sec. 2] and often results in quite conservative bounds when using interval arithmetic for range bounding. To overcome this issue, *affine arithmetic* [30] explicitly keeps track of dependencies by representing the variables as affine forms. Range bounding using Taylor models [41] further extends this concept by using polynomials instead of affine forms, which allows to tightly enclose the nonlinear function $f(x)$ with a Taylor model $\mathcal{T}(x)$:

$$\forall x \in \mathcal{D} : \quad f(x) \in \mathcal{T}(x).$$

Bounds for the function are then computed by enclosing the set $\mathcal{S} = \{\mathcal{T}(x) \mid x \in \mathcal{D}\}$ defined by the Taylor model with an interval, which can for example be computed by applying interval arithmetic to the polynomial function defining the Taylor model, or by using branch-and-bound methods [71].

While interval arithmetic, affine arithmetic, and Taylor models are the most commonly used methods for range bounding, there also exist other approaches: For polynomial functions $f(x)$ Bernstein polynomials [72] can be used for range bounding. Since Bernstein polynomials have the property that all function values are smaller than the maximum polynomial coefficient and larger than the minimum polynomial coefficient [72, Thm. 1], guaranteed bounds can directly be obtained from the polynomial coefficients after converting $f(x)$ to Bernstein form [73]. While the previous techniques do not provide any guarantees for the tightness of the computed bounds, verified global optimization as introduced in [42] refines the bounds until a user-defined precision is achieved. However, since this approach is based on a subdivision of the domain $\mathcal{D}$, its computational complexity grows exponentially with respect to the number of variables n.

Table 2.4: Computational complexity $\mathcal{O}(\texttt{bound})$ of range bounding methods, where $e \in \mathbb{N}_0$ is the number of elementary operations in the corresponding function $f : \mathbb{R}^n \to \mathbb{R}$.

Method	**Complexity**	**Reference**
Interval arithmetic	$\mathcal{O}(e)$	[37, Ch. 2.3]
Affine arithmetic	$\mathcal{O}(en)$	[30]
Taylor models	$\geq \mathcal{O}(en)$	[41]

For the derivation of the computational complexity we denote by $e \in \mathbb{N}_0$ the number of elementary operations such as $+$, $-$, or $\sin(\cdot)$ in a function $f : \mathbb{R}^n \to \mathbb{R}$. The evaluation of one elementary operation with interval arithmetic has constant complexity $\mathcal{O}(1)$ [37, 70], so that the overall computational complexity for range bounding with interval arithmetic is $e \cdot \mathcal{O}(1) = \mathcal{O}(e)$. For affine arithmetic each elementary operation has to be evaluated on an affine form with n variables [30, Sec. 3], which has complexity $\mathcal{O}(n)$. Consequently, the overall computational complexity for range bounding with affine arithmetic is $e \cdot \mathcal{O}(n) = \mathcal{O}(en)$. The computational complexity for range bounding using Taylor models depends on the representation of Taylor models as well as on the implementation of elementary operations on them. However, since Taylor models are an extension of affine arithmetic, it holds that the computational complexity is at least larger than $\mathcal{O}(en)$. The computational complexities for the range bounding techniques introduced in this section are summarized in Tab. 2.4. Moreover, a comparison of the bounds for an exemplary function computed with different range bounding methods are visualized in Fig. 2.3. An extensive performance comparison of range bounding techniques on multiple benchmarks can be found in [74].

2.8 Contractors

It is well known that characterizing the solution set for a constraint satisfaction problem defined by multiple nonlinear equality constraints is in general NP-hard [37, Ch. 4.1]. Consequently, the goal is often to determine a tight enclose of the solution set instead. Contractors represent an efficient method to downsize interval domains without losing any solutions to the constraint system, and are therefore well suited for the computation of tight solution set enclosures:

Definition 2.8.1. *(Contractor) Given an interval $\mathcal{I} \subset \mathbb{R}^n$ and a vector field $f : \mathbb{R}^n \to \mathbb{R}^m$ which defines the constraint $f(x) = \mathbf{0}$, the operation* `contract` *returns an interval that satisfies*

$$\texttt{contract}\big(f(x), \mathcal{I}\big) \subseteq \mathcal{I}$$

and

$$\forall x \in \mathcal{I}: \quad f(x) = \mathbf{0} \Rightarrow x \in \texttt{contract}\big(f(x), \mathcal{I}\big),$$

so that it is guaranteed that all solutions for $f(x) = \mathbf{0}$ contained in $\mathcal{I}$ are also contained in the contracted interval.

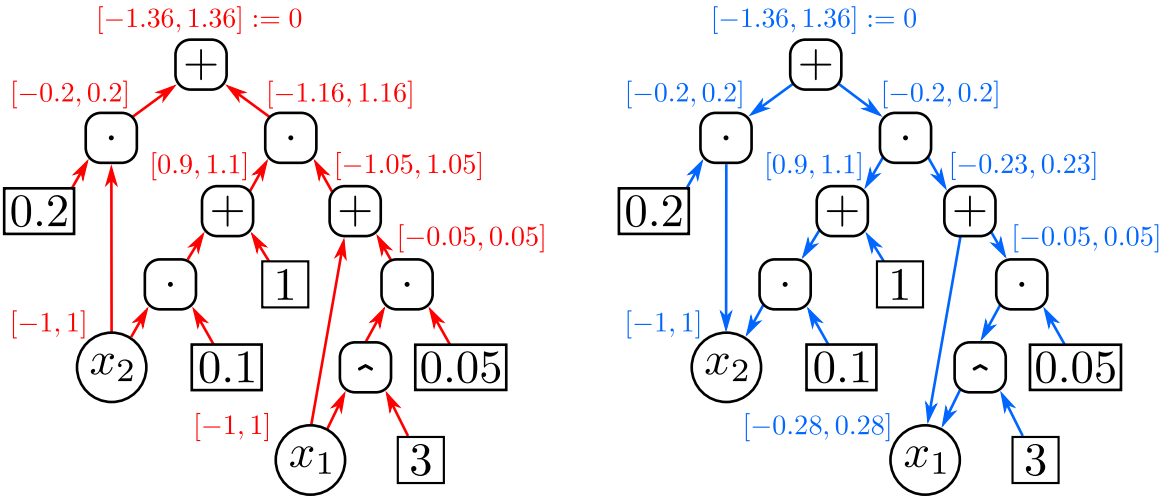

Figure 2.4: Forward propagation (left) and backward propagation (right) of the forward-backward contractor for the contraction problem in (2.21).

Note that the constraint $f(x) = \mathbf{0}$ in Def. 2.8.1 corresponds to the intersection of m single constraints $f_{(1)}(x) = 0 \wedge \ldots \wedge f_{(m)}(x) = 0$. There exist many different contractors, an overview of which is provided in [37, Ch. 4]. In this section we shortly introduce three commonly used contractors and derive their computational complexity. As a running example throughout this section we consider the contraction problem $\texttt{contract}(f(x), \mathcal{I})$ with

$$f(x) = \big(1 + 0.1x_2\big)\big(x_1 + 0.05x_1^3\big) + 0.2x_2, \quad \mathcal{I} = [-1, 1] \times [-1, 1], \tag{2.21}$$

where $x = [x_1 \ x_2]^T$. As for range bounding in Sec. 2.7, we again denote by $e \in \mathbb{N}_0$ the number of elementary operations required to evaluate each subfunction $f_{(i)} : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ of the vector field $f : \mathbb{R}^n \to \mathbb{R}^m$.

We begin with the forward-backward contractor [37, Ch. 4.2.4], which considers the case $m = 1$ corresponding to one single constraint. This contractor applies a forward-backward-propagation scheme to the syntax tree of the function $f(x)$ that defines the constraint. This is visualized in Fig. 2.4 for the contraction problem in (2.21). During forward propagation, bounds for the function $f(x)$ on the domain $\mathcal{I}$ are computed using interval arithmetic, where intermediate results at the nodes of the syntax tree are stored. Since the constraint is defined as $f(x) = 0$, the resulting interval has to be equal to zero. Consequently, during backward propagation, the value 0 is back-propagated through the syntax tree and the bounds at all nodes are updated by intersection with the back-propagated value. This procedure then yields updated domains for all variables. To extend the forward-backward contractor to the general case involving multiple constraints, one can apply a sequential evaluation scheme, where the interval domain is first contracted using the first constraint, and the resulting contracted interval is then used as the initial domain for contraction with the remaining constraints. Usually, the forward-backward contractor performs well for large interval domains $\mathcal{I}$, where other contractors often fail to contract the set. For both, forward propagation using interval arithmetics and backward propagation, the computational complexity for one elementary operation is $\mathcal{O}(1)$. Since the contractor

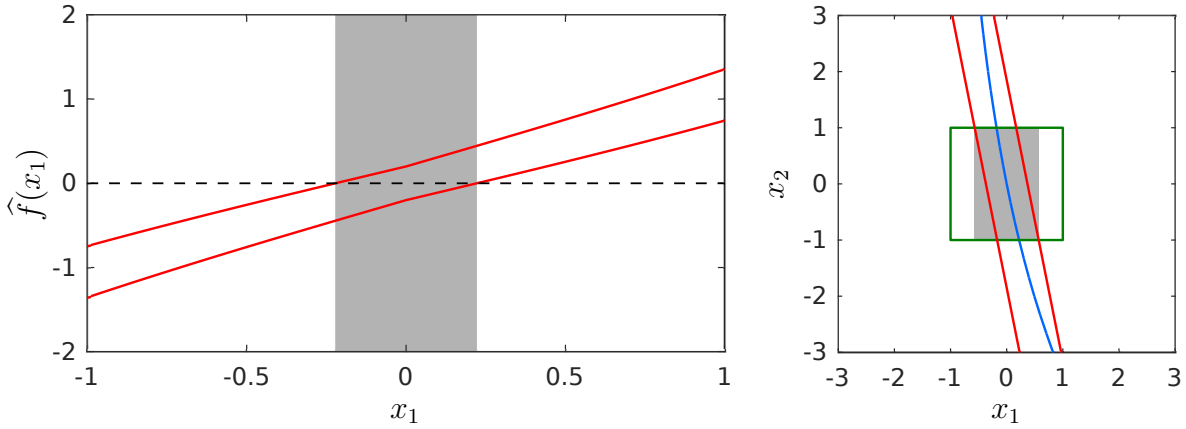

Figure 2.5: Visualization of the extremal function based contractor (left) and the parallel linearization contractor (right) for the contraction problem in (2.21). The original interval domain $\mathcal{I}$ is depicted in green, the contracted domain in gray, the values satisfying the constraint $f(x) = 0$ in blue, and the enclosing extremal functions (left) as well as the enclosing strip (right) are depicted in red.

is sequentially applied to all m constraints, the overall complexity of the forward-backward contractor is therefore $m \cdot e \cdot \mathcal{O}(1) = \mathcal{O}(me)$.

The contractor in [75] considers the case where the constraints are defined by a polynomial function $f(x)$. Like for the forward-backward contractor, the case $m = 1$ corresponding to one single constraint is considered. To contract the interval domain, the contractor encloses $f(x)$ by extremal functions. For this, all variables except for one variable x_i are substituted by their corresponding interval domains. This yields a function $\widehat{f}(x_i)$ with one variable that is a polynomial with interval coefficients. For the contraction problem in (2.21), for example, we obtain the function

$$\widehat{f}(x_1) = [-0.2, 0.2] + [0.9, 1.1] \cdot x_1 + [0.045, 0.055] \cdot x_1^3.$$

by substituting variable x_2 by the interval $[-1, 1]$, which is visualized in Fig. 2.5. Next, this function is enclosed by two extremal functions $\underline{g}(x_i) \leq \widehat{f}(x_i) \leq \overline{g}(x_i)$ that are constructed using the bounds for the interval coefficients. To contract the domain for variable x_i based on the extremal function enclosure, the zero crossings of the extremal functions have to be computed. If the extremal functions are constant, linear, quadratic, or cubic, the solutions for $\underline{g}(x_i) = 0$ and $\overline{g}(x_i) = 0$ can be calculated analytically. For polynomials with higher order branch-and-bound techniques [71] can be used. The whole contraction process is applied for all n variables x_i. To extend the contractor to the general case involving multiple constraints one can apply the same sequential evaluation scheme as for the forward-backward contractor. Computation of the interval coefficients with interval arithmetic has a worst-case complexity of $\mathcal{O}(e)$ according to Tab. 2.4. Because both $\underline{g} : \mathbb{R} \to \mathbb{R}$ and $\overline{g} : \mathbb{R} \to \mathbb{R}$ are function in a single variable, contraction of the corresponding interval domain using the analytical solution or branch-and-bound techniques has worst-case complexity $\mathcal{O}(e)$. Since the contraction is executed for all n variables and all m constraints, the overall worst-case complexity is $n \cdot m \cdot (\mathcal{O}(e) + \mathcal{O}(e)) = \mathcal{O}(nme)$.

Table 2.5: Computational complexity $\mathcal{O}(\texttt{contract})$ of contractors, where $e \in \mathbb{N}_0$ is the number of elementary operations in each subfunction $f_{(i)} : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$ of the vector field $f : \mathbb{R}^n \to \mathbb{R}^m$ that defines the constraint $f(x) = \mathbf{0}$.

Method	**Complexity**	**Reference**
Forward-backward	$\mathcal{O}(me)$	[37, Ch. 4.2.4]
Extremal functions	$\mathcal{O}(nme)$	[75]
Parallel linearization	$\mathcal{O}(nme + n^{4.5})$	[37, Ch. 4.3.4]

The idea of the parallel linearization contractor [37, Ch. 4.3.4] is to enclose each constraint with a strip defined by the intersection of two parallel halfspaces, which yields:

$$\forall x \in \mathcal{I} : \quad Ax + l \leq f(x) \leq Ax + u, \tag{2.22}$$

where

$$\begin{aligned} &A = \nabla f(x)\big|_c, \quad c = \texttt{center}(\mathcal{I}) \\ &[l, u] = (f(c) - Ac) \oplus \big(\texttt{bound}(\nabla f(x) - A, \mathcal{I}) \otimes (\mathcal{I} \oplus (-c))\big). \end{aligned}$$

For the contraction problem in (2.21) the strip enclosure is visualized in Fig. 2.5. The tightest interval enclosing the intersection of the strips for the m constraints can be determined with linear programming. While the previous contractors consider each constraint separately, parallel linearization directly considers the intersection of all constraints, which often results in a bigger contraction. However, for large interval domains $\mathcal{I}$ parallel linearization might not be able to contract the set at all due to potentially large linearization errors. To compute the lower bound $l \in \mathbb{R}^m$ and the upper bound $u \in \mathbb{R}^m$ of the linearization error for the strip enclosure in (2.22), we have to apply range bounding for all mn entries of the Jacobian matrix $\nabla f(x)$. Under the assumption that taking the derivative of a function only changes the number of elementary operations by a constant factor, this has computational complexity $\mathcal{O}(nme)$ according to Tab. 2.4 if interval arithmetic is used. Moreover, to compute the tightest interval that encloses the intersection of the m strips we have to solve $2n$ linear programs in n variables, which has according to Tab. 2.1 complexity $2n \cdot \mathcal{O}(n^{3.5}) = \mathcal{O}(n^{4.5})$. The overall computational complexity of the parallel linearization contractor is therefore $\mathcal{O}(nme) + \mathcal{O}(n^{4.5}) = \mathcal{O}(nme + n^{4.5})$. A summary of the computational complexities for the different contractors presented in this section is shown in Tab. 2.5. As the final results for the contraction problem in (2.21) we obtain $\texttt{contract}(f(x), \mathcal{I}) = [-0.28, 0.28] \times [-1, 1]$ for the forward-backward contractor, $\texttt{contract}(f(x), \mathcal{I}) = [-0.23, 0.23] \times [-1, 1]$ for the extremal function based contractor, and $\texttt{contract}(f(x), \mathcal{I}) = [-0.57, 0.57] \times [-1, 1]$ for the parallel linearization contractor, so that in this case the extremal function based contractor achieved the largest contraction.

Since all contractors presented in this chapter apply range bounding, the conservatism in the contraction heavily depends on the size of the initial interval domain $\mathcal{I}$. It is therefore often meaningful to repeat contraction multiple times since the smaller interval domain $\mathcal{I}$

after each contraction results in tighter bounds for range bounding, which then potentially enables to contract the domain even further. If the number of repetitions is constant, the repetitive application of the contractor does not change the computational complexity. Due to the dependence on the size of the interval domain $\mathcal{I}$, the results can also significantly be improved by dividing $\mathcal{I}$ into smaller intervals. This, however, has exponential complexity with respect to the dimension n. Often, the accuracy can also be improved by running multiple different contractors in parallel and then intersecting the results. Yet another way to improve contraction is to compute a contracted domain $[l, u] = \texttt{contract}(f(x), \mathcal{I})$ by solving the optimization problems

$$l_{(i)} = \min_{x \in \mathcal{I}} \; x_{(i)} \;\; \text{s.t. } f(x) = \mathbf{0} \quad \text{and} \quad u_{(i)} = \max_{x \in \mathcal{I}} \; x_{(i)} \;\; \text{s.t. } f(x) = \mathbf{0} \tag{2.23}$$

for all dimensions $i = 1, \ldots, n$ using nonlinear programming. Since it is not guaranteed that nonlinear programming actually finds the global optimum, we additionally have to verify the correctness of the obtained result, which can be done with a contractor: It holds that the result from (2.23) is correct if the remaining domain $\mathcal{I} \setminus [l, u]$ can be contracted to the empty set $\texttt{contract}(f(x), \mathcal{I} \setminus [l, u]) = \emptyset$.

Chapter 3

Extensions of Polynomial Zonotopes

In this chapter we present three novel set representations that are extensions of polynomial zonotopes as introduced in [47]. We begin with the sparse representation of polynomial zonotopes in Sec. 3.1, which enables to represent polynomial zonotopes very compactly. Constrained polynomial zonotopes as introduced in Sec. 3.2 extend polynomial zonotopes by adding equality constraints to the zonotope factors, resulting in a set representation that is additionally closed under intersection and union. Finally, we introduce the Z-representation of polytopes in Sec. 3.3, which represents polytopes as polynomial zonotopes.

3.1 Sparse Polynomial Zonotopes

We introduce *sparse polynomial zonotopes* (SPZs) in this section[1] and derive closed-form expressions for all relevant set operations on this novel non-convex set representation. The structure of the section is as follows: We first define SPZs in Sec. 3.1.1 and introduce some preliminaries in Sec. 3.1.2. Next, we show in Sec. 3.1.3 how to convert other set representations to SPZs and provide algorithms for tightly enclosing SPZs with simpler set representations in Sec. 3.1.4. Closed-form expressions for all basic set operations on SPZs are derived in Sec. 3.1.5. Finally, we present some results on containment and intersection checks for SPZs in Sec. 3.1.6 and specify several useful auxiliary operations, such as order reduction, in Sec. 3.1.7. An overview showing all operations on SPZs considered in this section is provided in Tab. 3.1.

3.1.1 Definition

Let us first define sparse polynomial zonotopes:

Definition 3.1.1. *(Sparse Polynomial Zonotope) Given a constant offset $c \in \mathbb{R}^n$, a generator matrix of dependent generators $G \in \mathbb{R}^{n \times h}$, a generator matrix of independent generators $G_I \in \mathbb{R}^{n \times q}$, and an exponent matrix $E \in \mathbb{N}_0^{p \times h}$, a sparse polynomial zonotope $\mathcal{PZ} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{PZ} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}.$$

[1] This section is based on [76].

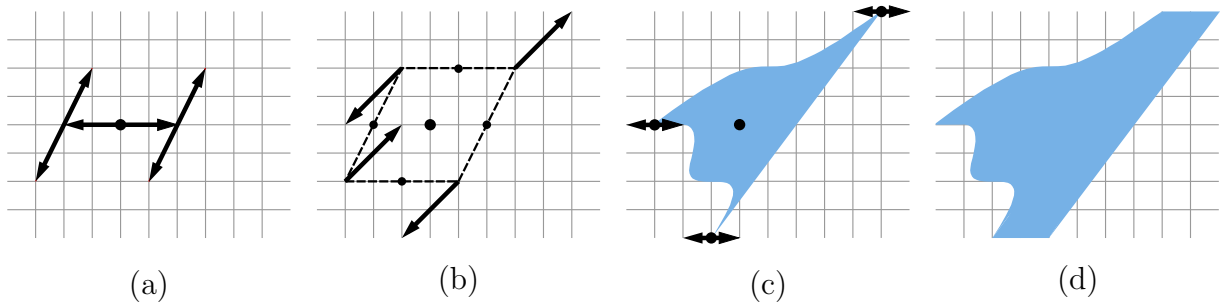

Figure 3.1: Step-by-step construction of the SPZ in Example 3.1.2: (a) shows the set spanned by the constant offset vector and the second and third dependent generator, (b) shows the addition of the dependent generator with the mixed term $\alpha_1^3\alpha_2$, (c) shows the addition of the independent generator, and (d) visualizes the final set.

The scalars α_k are called dependent factors since a change in their value affects multiplication with multiple generators. Consequently, the scalars β_j are called independent factors because they only affect multiplication with one generator. Moreover, the expression $\alpha_1^{E_{(1,i)}} \cdot \ldots \cdot \alpha_p^{E_{(p,i)}} \cdot G_{(\cdot,i)}$ is called a monomial, and $\alpha_1^{E_{(1,i)}} \cdot \ldots \cdot \alpha_p^{E_{(p,i)}}$ the variable part of the monomial. The number of dependent factors is p, the number of independent factors is q, and the number of dependent generators is h. The degree-of-freedom order $\rho_f = \frac{p+q}{n}$ of a SPZ is a measure for the complexity of the set, and the order $\rho = \frac{h+q}{n}$ of a SPZ estimates the representation size. The SPZ is regular if the exponent matrix E does not contain duplicate columns or all-zero columns:

$$\forall i, j \in \{1, \ldots, h\} : \ (i \neq j) \Rightarrow \left(E_{(\cdot,i)} \neq E_{(\cdot,j)}\right) \ \text{ and } \ \forall i \in \{1, \ldots, h\} : \ E_{(\cdot,i)} \neq \mathbf{0}.$$

To keep track of identical dependent factors in different SPZs, an unambiguous integer identifier is assigned to each dependent factor α_k, and the identifiers for all dependent factors are stored in a row vector $id \in \mathbb{N}^{1\times p}$. For a concise notation we use the shorthand $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ}$.

Let us demonstrate SPZs with the following example:

Example 3.1.2. *The SPZ*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}$$

defines the set

$$\mathcal{PZ} = \left\{ \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \alpha_2 + \begin{bmatrix} 2 \\ 2 \end{bmatrix} \alpha_1^3\alpha_2 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \beta_1 \ \middle| \ \alpha_1, \alpha_2, \beta_1 \in [-1, 1] \right\},$$

where the identifier vector $[1\ 2]$ stores the identifier 1 for the dependent factor α_1 and the identifier 2 for the dependent factor α_2. The construction of this SPZ is visualized in Fig. 3.1.

Table 3.1: Overview showing all set operations on SPZs presented in this thesis.

Set Operation	**Reference**	**Page**
Merge of identifier vectors (`mergeID`)	Prop. 3.1.5	35
Redundancy removal (`compact`)	Prop. 3.1.7	36
Conversion zonotope to SPZ	Prop. 3.1.9	37
Conversion interval to SPZ	Prop. 3.1.10	37
Conversion polytope to SPZ	Prop. 3.1.11	38
Conversion Taylor model to SPZ	Prop. 3.1.12	39
Conversion SPZ to Taylor model	Prop. 3.1.13	40
Zonotope enclosure of SPZ (`zonotope`)	Prop. 3.1.14	41
Polytope enclosure of SPZ (`polytope`)	Prop. 3.1.15	42
Support function enclosure of SPZ	Prop. 3.1.16	44
Ellipsoid enclosure of SPZ (`ellipsoid`)	Prop. 3.1.17	45
Linear map	Prop. 3.1.18	47
Minkowski sum	Prop. 3.1.19	48
Exact addition	Prop. 3.1.20	49
Cartesian product	Prop. 3.1.22	50
Linear combination	Prop. 3.1.26	53
Convex hull	Prop. 3.1.28	55
Quadratic map	Prop. 3.1.31	59
Containment check SPZ in SPZ	Prop. 3.1.34	63
Containment check interval in SPZ	Prop. 3.1.36	64
Intersection check	Prop. 3.1.38	66
Order reduction (`reduce`)	Prop. 3.1.39	68
Restructuring (`restructure`)	Prop. 3.1.41	70
Subset extraction (`getSubset`)	Prop. 3.1.43	72
Splitting (`split`)	Prop. 3.1.44	73

For the derivation of the computational complexity of operations on SPZs we make the following assumption:

Assumption 3.1.3. *Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ *with* $p \in \mathbb{N}_0$ *dependent factors,* $h \in \mathbb{N}_0$ *dependent generators,* $q \in \mathbb{N}_0$ *independent factors, and a maximum exponent matrix entry* $\epsilon = \max(E)$*, we assume for the derivation of the computational complexity that*

$$p = c_p n,\ h = c_h n,\ q = c_q n,\ \epsilon = c_\epsilon n,$$

with $c_p, c_h, c_q, c_\epsilon \in \mathbb{R}_{\geq 0}$.

The assumption is justified by the fact that the order $\rho = \frac{h+q}{n}$ is reduced to the desired order ρ_d when computing with SPZs, such that $h + q \leq \rho_d n$ holds. Every SPZ can be equivalently represented as a SPZ without independent generators:

Proposition 3.1.4. *Given a polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ}$, $\mathcal{PZ}$ can be equivalently represented without independent generators:*

$$\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} = \left\langle c, [G\ G_I], [\,], \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix}, [id\ \texttt{uniqueID}(q)] \right\rangle_{PZ},$$

where the operation `uniqueID` *introduced in Sec. 2.5 generates new unique identifiers.*

Proof. The result follows directly from the substitution of the independent factors β_j in Def. 3.1.1 with additional dependent factors $\alpha_{h+1} = \beta_1, \ldots, \alpha_{h+q} = \beta_q$. □

However, the independent generators are required for computational reasons: while computations on the dependent generators are exact but computational expensive, computations on the independent generators are often over-approximative but fast. When computing with SPZs we therefore usually define large generators as dependent and small generators as independent, which often yields a good trade-off between computational cost and accuracy.

SPZ are a more compact representation of polynomial zonotopes [47], resulting in completely different algorithms for operations on them. In [47, Def. 1], the generators $g^{([o],i,k,\ldots,m)}$ for all possible combinations of dependent factors up to a certain polynomial degree μ are stored:

$$\begin{aligned}\mathcal{PZ} = \Bigg\{ c + \sum_{i=1}^{p} \alpha_i\ g^{([1],i)} + \sum_{i=1}^{p}\sum_{k=i}^{p} \alpha_i\alpha_k\ g^{([2],i,k)} + \ldots + \\ \sum_{i=1}^{p}\sum_{k=i}^{p}\cdots\sum_{m=l}^{p} \alpha_i\alpha_k\ldots\alpha_m\ g^{([\mu],i,k,\ldots,m)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \ \Bigg| \\ \alpha_i, \alpha_k, \ldots, \alpha_m, \beta_j \in [-1,1] \Bigg\},\end{aligned}$$

with $g^{([o],i,k,\ldots,m)} \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $G_I \in \mathbb{R}^{n \times q}$. This results in $h = \binom{\mu+p}{p}$ generators [77, Eq. (3.8)]. For the one-dimensional polynomial zonotope $\mathcal{PZ} = \{\alpha_1 + \ldots + \alpha_{19} + \alpha_{20}^{10} \mid \alpha_1, \ldots, \alpha_{20} \in [-1,1]\}$ with $p = 20$ dependent factors and with a polynomial degree of $\mu = 10$, the number of dependent generators is $h = 30045015$. This demonstrates that the number of generators that have to be stored can become very large if the polynomial degree and the number of dependent factors are high, which makes computations on the previous set representation very inefficient. SPZs on the other hand employ a sparse representation storing required generators only. Even in comparison with quadratic zonotopes, which correspond to a polynomial order of $\mu = 2$, SPZs consequently have lower or equal complexity for all set operations considered in [47], as shown in Tab. 3.2. Moreover, SPZ do not require limiting the polynomial degree of the polynomial zonotope in advance, which often significantly reduces the over-approximation.

Table 3.2: Comparison of the computational complexity of set operations on SPZs and the quadratic zonotopes in [47], where $n \in \mathbb{N}$ is the dimension of the set.

Set Operation	**SPZ**	**Quad. Zono.**
Linear map with matrix $M \in \mathbb{R}^{w \times n}$	$\mathcal{O}(n^2 w)$	$\mathcal{O}(n^2 w)$
Minkowski addition with zonotope	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Enclosure by zonotope	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Quadratic map with $\mathcal{Q} = \{Q_1, \ldots, Q_w\}$	$\mathcal{O}(n^3(w + \log(n)))$	$\mathcal{O}(n^4 w)$

3.1.2 Preliminaries

We first derive some preliminary operations on SPZs that are required for many other operations. In order to fully exploit the dependencies between identical dependent factors from different SPZs, it is necessary to build a common representation of exponent matrices, which is done with the operator `mergeID`:

Proposition 3.1.5. *(Merge ID) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ}$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ}$, `mergeID` returns two adjusted SPZs $\overline{\mathcal{PZ}}_1$ and $\overline{\mathcal{PZ}}_2$ with identical identifier vectors that are equivalent to $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$:*

$$\texttt{mergeID}(\mathcal{PZ}_1, \mathcal{PZ}_2) = \big(\underbrace{\langle c_1, G_1, G_{I,1}, \overline{E}_1, \overline{id} \rangle_{PZ}}_{\overline{\mathcal{PZ}}_1}, \underbrace{\langle c_2, G_2, G_{I,2}, \overline{E}_2, \overline{id} \rangle_{PZ}}_{\overline{\mathcal{PZ}}_2} \big),$$

where

$$\overline{id} = \begin{bmatrix} id_1 & id_{2(\mathcal{H})} \end{bmatrix}, \quad \mathcal{H} = \left\{ i \mid id_{2(i)} \notin id_1 \right\}, \quad \overline{E}_1 = \begin{bmatrix} E_1 \\ \mathbf{0} \end{bmatrix} \in \mathbb{N}_0^{(p_1 + |\mathcal{H}|) \times h_1},$$

$$\overline{E}_{2(i,\cdot)} = \begin{cases} E_{2(j,\cdot)}, & \exists j \in \{1, \ldots, p_2\} : \ \overline{id}_{(i)} = id_{2(j)} \\ \mathbf{0}, & \textit{otherwise} \end{cases}, \quad i = 1, \ldots, p_1 + |\mathcal{H}|.$$

The computational complexity is $\mathcal{O}(p_1 p_2)$, where p_1 is the number of dependent factors of $\mathcal{PZ}_1$ and p_2 is the number of dependent factors of $\mathcal{PZ}_2$.

Proof. The extension of the exponent matrices with all-zero rows only changes the representation of the set, but not the set itself.

Complexity: The only operation with super-linear complexity is the construction of the set $\mathcal{H}$ with worst-case complexity $\mathcal{O}(p_1 p_2)$. □

We demonstrate operation `mergeID` with an example:

Example 3.1.6. *Given the two SPZs*

$$\mathcal{PZ}_1 = \left\langle \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}, [\,], [1\ 3], 1 \right\rangle_{PZ}, \quad \mathcal{PZ}_2 = \left\langle \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix}, [\,], \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ},$$

`mergeID` *as defined in Prop. 3.1.5 returns the adjusted SPZs*

$$\overline{\mathcal{PZ}}_1 = \left\langle \begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}1 & 2\\1 & 0\end{bmatrix}, [\,], \begin{bmatrix}1 & 3\\0 & 0\end{bmatrix}, [1\ 2] \right\rangle_{PZ}, \ \overline{\mathcal{PZ}}_2 = \left\langle \begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}1 & 1\\0 & 3\end{bmatrix}, [\,], \begin{bmatrix}1 & 2\\0 & 1\end{bmatrix}, [1\ 2] \right\rangle_{PZ},$$

which have identical identifier vectors.

Many operations result in a SPZ that is not regular. We therefore introduce the operation `compact` which converts a non-regular SPZ to a regular one by removing duplicate columns from the exponent matrix:

Proposition 3.1.7. *(Compact) Given a non-regular SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, the operation* `compact` *returns the corresponding regular SPZ:*

$$\texttt{compact}(\mathcal{PZ}) = \left\langle \underbrace{c + \sum_{i \in \mathcal{K}} G_{(\cdot,i)}}_{\overline{c}}, \underbrace{\left[\sum_{i \in \mathcal{H}_1} G_{(\cdot,i)} \ \ldots \ \sum_{i \in \mathcal{H}_w} G_{(\cdot,i)} \right]}_{\overline{G}}, G_I, \overline{E}, id \right\rangle_{PZ},$$

where

$$\mathcal{K} = \left\{ i \mid \forall k \in \{1, \ldots, p\} : \ E_{(k,i)} = 0 \right\}, \quad \overline{E} = \texttt{uniqueColumns}(E_{(\cdot,\mathcal{N})}) \in \mathbb{N}_0^{p \times w},$$

$$\mathcal{N} = \{1, \ldots, h\} \setminus \mathcal{K}, \quad \mathcal{H}_j = \left\{ i \mid \forall k \in \{1, \ldots, p\} : \ \overline{E}_{(k,j)} = E_{(k,i)} \right\}, \ j = 1, \ldots, w,$$

and the operation `uniqueColumns` *introduced in Sec. 2.5 removes identical matrix columns until all columns are unique. The computational complexity is* $\mathcal{O}(h(n + p\log(h)))$*, where* n *is the dimension,* p *is the number of dependent factors, and* h *is the number of dependent generators.*

Proof. For a SPZ where the exponent matrix $E = [e\ e]$ consists of 2 identical columns $e \in \mathbb{N}_0^p$, it holds that

$$\left\{ \left(\prod_{k=1}^{p} \alpha_k^{e(k)} \right) G_{(\cdot,1)} + \left(\prod_{k=1}^{p} \alpha_k^{e(k)} \right) G_{(\cdot,2)} \ \middle| \ \alpha_k \in [-1, 1] \right\}$$
$$= \left\{ \left(\prod_{k=1}^{p} \alpha_k^{e(k)} \right) \left(G_{(\cdot,1)} + G_{(\cdot,2)} \right) \ \middle| \ \alpha_k \in [-1, 1] \right\}.$$

Summation of the generators for monomials with identical exponents therefore does not change the set, which proves that $\texttt{compact}(\mathcal{PZ})$ equivalently represents $\mathcal{PZ}$. Moreover, since all duplicate and all-zero columns are removed, it holds that the resulting SPZ is regular.

Complexity: The construction of the vector $\overline{c}$ and the matrix $\overline{G}$ has worst-case complexity $\mathcal{O}(nh)$, and the construction of the set $\mathcal{K}$ has worst-case complexity $\mathcal{O}(ph)$. Applying the operator `uniqueColumns` to the matrix $E_{(\cdot,\mathcal{N})} \in \mathbb{N}_0^{p \times |\mathcal{N}|}$ has complexity $\mathcal{O}(p|\mathcal{N}|\log(|\mathcal{N}|))$ according to Tab. 2.2, which is $\mathcal{O}(ph\log(h))$ since $|\mathcal{N}| \leq h$. The sets $\mathcal{H}_j$ can be constructed during the removal of duplicate columns with `uniqueColumns` and therefore do not require any additional computations. The overall worst-case complexity is therefore $\mathcal{O}(nh) + \mathcal{O}(ph) + \mathcal{O}(ph\log(h)) = \mathcal{O}(h(n + p\log(h)))$. □

Let us demonstrate the operation `compact` with an example:

Example 3.1.8. *Given the non-regular SPZ*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 1 & 2 & -1 \end{bmatrix}, [\,], \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ},$$

`compact` *as defined in Prop. 3.1.7 returns the equivalent SPZ*

$$\texttt{compact}(\mathcal{PZ}) = \left\langle \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}, [\,], \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ},$$

which is regular.

Both operations, `mergeID` and `compact`, are very useful for the implementation of many other operations on SPZs.

3.1.3 Conversion from other Set Representations

In this section we show how to convert other set representations to SPZs. We begin with zonotopes:

Proposition 3.1.9. *(Conversion Zonotope) A zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ can be equivalently represented by a SPZ*

$$\mathcal{Z} = \big\langle c, G, [\,], I_l, \texttt{uniqueID}(l) \big\rangle_{PZ},$$

where $l \in \mathbb{N}_0$ is the number of zonotope generators. The computational complexity of the conversion is $\mathcal{O}(n)$ with respect to the dimension n.

Proof. If we insert $E = I_l$ and $G_I = [\,]$ into the definition of SPZs in Def. 3.1.1, we obtain a zonotope as defined in Def. 2.2.4.

Complexity: The only operation that is required is the generation of l unique identifiers using `uniqueID`, which has complexity $\mathcal{O}(l)$ according to Tab. 2.2. With Assumption 2.6.2 we therefore obtain an overall complexity of $\mathcal{O}(l) = \mathcal{O}(n)$. □

Next, we consider the conversion of intervals, for which we can reuse the previous result for zonotopes since intervals are a special case of zonotopes:

Proposition 3.1.10. *(Conversion Interval) An interval $\mathcal{I} = [l, u] \subset \mathbb{R}^n$ can be equivalently represented by a SPZ*

$$\mathcal{I} = \big\langle 0.5(l+u), \texttt{diag}\big(0.5(u-l)\big), [\,], I_n, \texttt{uniqueID}(n) \big\rangle_{PZ}.$$

The computational complexity of the conversion is $\mathcal{O}(n)$ with respect to the dimension n.

Proof. Since according to [24, Prop. 2.1] an interval $\mathcal{I} = [l, u]$ can be equivalently represented as a zonotope $\mathcal{I} = \langle 0.5(l+u), \texttt{diag}(0.5(u-l)) \rangle_Z$, the result follows directly from Prop. 3.1.9.

Complexity: Computation of the vectors $0.5(l+u)$ and $0.5(u-l)$ has complexity $\mathcal{O}(n)$, and the generation of n unique identifiers with `uniqueID` has complexity $\mathcal{O}(n)$ according to Tab. 2.2, which results in an overall complexity of $\mathcal{O}(n)$. □

The conversion of polytopes is based on the Z-representation, which we introduce later in Sec. 3.3:

Proposition 3.1.11. *(Conversion Polytope) Given a bounded polytope in V-representation $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V \subset \mathbb{R}^n$, a SPZ equivalently representing $\mathcal{P}$ can be computed by first converting $\mathcal{P}$ to a set in Z-representation using Alg. 4, and then converting the Z-representation to a SPZ using Prop. 3.3.12. The computational complexity of the conversion is $\mathcal{O}(s^2(n + \log(s)))$ with respect to the number of zonotope vertices s, where n is the dimension.*

Proof. Alg. 4 converts from polytope V-representation to polytope Z-representation. Since the Z-representation, which is later introduced in Sec. 3.3, is just a more compact way of storing SPZs that represent polytopes, every polytope in Z-representation can be easily converted to a SPZ using Prop. 3.3.12.

Complexity: The conversion from V-representation to Z-representation using Alg. 4 has complexity $\mathcal{O}(s^2(n + \log(s)))$ according to Prop. 3.3.6 and conversion of a set in Z-representation to a SPZ using Prop. 3.3.12 has complexity $\mathcal{O}(\mu + p)$, where p and μ are the number of factors and number of tuple entries of the Z-representation (see Def. 3.3.1). According to (3.96) and (3.108), upper bounds for the number of factors and tuple entries are given by

$$\begin{aligned} p &\overset{(3.96)}{\leq} 2s - 1, \\ \mu &\overset{(3.108)}{\leq} 4^{\lceil \log(s) \rceil} \left(\frac{1}{6} \lceil \log(s) \rceil + \frac{1}{9} \right) - \frac{1}{9} \\ &\leq 4^{\log(s)+1} \left(\frac{1}{6} \big(\log(s) + 1 \big) + \frac{1}{9} \right) - \frac{1}{9} = 4s^2 \left(\frac{1}{6} \log(s) + \frac{5}{18} \right) - \frac{1}{9}, \end{aligned}$$

so that $\mathcal{O}(\mu + p) = \mathcal{O}(s^2 \log(s))$. The overall complexity is therefore $\mathcal{O}(s^2(n + \log(s))) + \mathcal{O}(s^2 \log(s)) = \mathcal{O}(s^2(n + \log(s)))$. □

Since any bounded polytope in H-representation and any constrained zonotope can be equivalently represented as a polytope in V-representation, Prop. 3.1.11 can also be used to represent polytopes in H-representation and constrained zonotopes as SPZs. Finally, we consider the conversion of Taylor models:

Proposition 3.1.12. *(Conversion Taylor Model) The set $\mathcal{S} = \{\mathcal{T}(x) \mid x \in \mathcal{I}\} \subset \mathbb{R}^n$ defined by a Taylor model $\mathcal{T}(x) = \langle w(x), \mathcal{Y}, \mathcal{I} \rangle_{TM}$ with $w : \mathbb{R}^r \to \mathbb{R}^n$, $\mathcal{Y} = [l_R, u_R] \subset \mathbb{R}^n$, and $\mathcal{I} = [l_D, u_D] \subset \mathbb{R}^r$ can be equivalently represented by a SPZ*

$$\mathcal{S} = \left\langle \frac{l_R + u_R}{2}, \begin{bmatrix} \begin{bmatrix} \overline{b}_{1,1} & \dots & \overline{b}_{1,m_1} \end{bmatrix} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \begin{bmatrix} \overline{b}_{n,1} & \dots & \overline{b}_{n,m_n} \end{bmatrix} \end{bmatrix}, \right.$$
$$\left. \texttt{diag}\big(0.5(u_R - l_R)\big), \begin{bmatrix} \overline{E}_1 & \dots & \overline{E}_n \end{bmatrix}, \texttt{uniqueID}(r) \right\rangle_{PZ},$$

where the coefficients $\overline{b}_{i,j}$ and the matrices $\overline{E}_i$ result from the definition

$$w_{(i)}\left([\delta_1(\alpha_1),\ldots,\delta_r(\alpha_r)]^T\right) := \sum_{j=1}^{\overline{m}_i} \overline{b}_{i,j} \prod_{k=1}^{r} \alpha_k^{\overline{E}_{i(k,j)}}, \quad i = 1,\ldots,n, \tag{3.1}$$

with

$$\delta_k(\alpha_k) = \frac{l_{D(k)} + u_{D(k)}}{2} + \frac{u_{D(k)} - l_{D(k)}}{2}\ \alpha_k, \qquad \alpha_k \in [-1,1],\ k = 1,\ldots,r. \tag{3.2}$$

The `compact` *operation as defined in Prop. 3.1.7 is applied to make the resulting SPZ regular. The computational complexity of the conversion is $\mathcal{O}(n^{4+n}\log(n))$ with respect to the dimension n.*

Proof. The auxiliary variables $\delta_k(\alpha_k)$ in (3.2) equivalently represent the domain $\mathcal{I}$ with dependent factors $\alpha_k \in [-1,1]$:

$$\mathcal{I} = [l_D, u_D] = \left\{ \begin{bmatrix}\delta_1(\alpha_1) & \ldots & \delta_r(\alpha_r)\end{bmatrix}^T \;\middle|\; \alpha_1,\ldots,\alpha_r \in [-1,1] \right\}. \tag{3.3}$$

Moreover, the interval $\mathcal{Y} = [l_R, u_R]$ can be equivalently represented as a zonotope $\mathcal{Y} = \langle 0.5(l_R + u_R), \mathtt{diag}(0.5(u_R - l_R))\rangle_Z$ according to [24, Prop. 2.1]. The set $\mathcal{S}$ defined by the Taylor model $\mathcal{T}(x)$ can therefore be equivalently expressed as

$$\begin{aligned}
\mathcal{S} =& \left\{\mathcal{T}(x) \mid x \in \mathcal{I}\right\} \overset{(3.3)}{=} \left\{\mathcal{T}(\delta(\alpha)) \mid \alpha \in [-\mathbf{1},\mathbf{1}]\right\} \overset{\text{Def. 2.2.8}}{=} \\
& \left\{ \begin{bmatrix} w_{(1)}(\delta(\alpha)) \\ \vdots \\ w_{(n)}(\delta(\alpha)) \end{bmatrix} + \begin{bmatrix} y_{(1)} \\ \vdots \\ y_{(n)} \end{bmatrix} \;\middle|\; y \in \mathcal{Y} \right\} \overset{\substack{(3.1),\\ \mathcal{I}=[l_R,u_R]}}{=} \\
& \left\{ \frac{l_R + u_R}{2} + \sum_{j=1}^{\overline{m}_1} \begin{bmatrix} \overline{b}_{1,j} \\ \mathbf{0} \end{bmatrix} \prod_{k=1}^{r} \alpha_k^{\overline{E}_{1(k,j)}} + \ldots + \sum_{j=1}^{\overline{m}_n} \begin{bmatrix} \mathbf{0} \\ \overline{b}_{n,j} \end{bmatrix} \prod_{k=1}^{r} \alpha_k^{\overline{E}_{n(k,j)}} \right. \\
& \left. + \frac{1}{2}\begin{bmatrix} u_{R(1)} - l_{R(1)} \\ \mathbf{0} \end{bmatrix} \beta_1 + \ldots + \frac{1}{2}\begin{bmatrix} \mathbf{0} \\ u_{R(n)} - l_{R(n)} \end{bmatrix} \beta_n \;\middle|\; \alpha_k, \beta_1,\ldots,\beta_n \in [-1,1] \right\} \\
& = \left\langle \frac{l_R + u_R}{2}, \begin{bmatrix} \begin{bmatrix}\overline{b}_{1,1} & \ldots & \overline{b}_{1,m_1}\end{bmatrix} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \begin{bmatrix}\overline{b}_{n,1} & \ldots & \overline{b}_{n,m_n}\end{bmatrix} \end{bmatrix}, \right. \\
& \qquad \left. \mathtt{diag}\big(0.5(u_R - l_R)\big), \begin{bmatrix}\overline{E}_1 & \ldots & \overline{E}_n\end{bmatrix}, \mathtt{uniqueID}(r) \right\rangle_{PZ},
\end{aligned}$$

where $\alpha = [\alpha_1\ \ldots\ \alpha_r]^T$ and $\delta(\alpha) = [\delta_1(\alpha_1)\ \ldots\ \delta_r(\alpha_r)]^T$.

Complexity: Construction of the constant offset $0.5(l_R + u_R)$ and the matrix of independent generators $\mathtt{diag}(0.5(u_R - l_R))$ of the resulting SPZ has complexity $\mathcal{O}(n)$ and the

generation of r unique identifiers with `uniqueID` has complexity $\mathcal{O}(r)$ according to Tab. 2.2. It remains to consider the complexity of the subsequent application of the `compact` operation. Let $m = \max(m_1, \dots, m_n)$ and $\epsilon = \max(\max(E_1), \dots, \max(E_n))$, where according to Def. 2.2.8 the variables m_i denote the number of monomials and the variables E_i represent the exponent matrices of the polynomial functions $w_{(i)}(x)$ defining the Taylor model. Since the auxiliary variables $\delta_k(\alpha_k)$ in (3.2) represent linear functions $\delta_k(\alpha_k) = c_0 + c_1\alpha_k$ with $c_0, c_1 \in \mathbb{R}$, we have that $\delta_k(\alpha_k)^\epsilon$ is a polynomial in α_k with $\epsilon + 1$ monomials. Naive evaluation without intermediate simplification of the function $w_{(i)}(\delta(\alpha))$ with $w_{(i)}(x)$ defined as

$$w_{(i)}(x) \overset{\text{Def. 2.2.8}}{=} \sum_{j=1}^{m_i} b_{i,j} \prod_{k=1}^{r} x_{(k)}^{E_{i(k,j)}}, \quad i = 1, \dots, n,$$

therefore results in a multivariate polynomial with $\overline{m} = m(\epsilon+1)^r$ monomials in the worst case. The exponent matrix $[\overline{E}_1 \ \dots \ \overline{E}_n]$ with $\overline{E}_i \in \mathbb{R}^{r \times \overline{m}_i}$ of the resulting SPZ consequently consists of at most $h = n\overline{m} = nm(\epsilon+1)^r$ columns since $\forall i \in \{1, \dots, n\} : \ \overline{m}_i \leq \overline{m}$. Because the complexity of `compact` is $\mathcal{O}(h(n+p\log(h)))$ according to Prop. 3.1.7 and the number of dependent factors of the resulting SPZs is $p = r$, the subsequent application of the `compact` operation has complexity $\mathcal{O}(nm(\epsilon+1)^r(n + r\log(nm(\epsilon+1)^r)))$. The overall complexity is therefore

$$\begin{aligned}\mathcal{O}(n)+\mathcal{O}(r) + \mathcal{O}\big(nm(\epsilon+1)^r(n + r\log(nm(\epsilon+1)^r))\big) = \\ \mathcal{O}\big(nm(\epsilon+1)^r(n + r(\log(n) + \log(m) + r\log(\epsilon+1)))\big),\end{aligned}$$

which is $\mathcal{O}(n^{4+n}\log(n))$ since similar to Assumption 3.1.3 for SPZs it holds for Taylor models that $r = c_r n$, $m = c_m n$, and $\epsilon = c_\epsilon n$ with $c_r, c_m, c_\epsilon \in \mathbb{R}_{\geq 0}$. □

SPZs and Taylor models are equivalent, so that also any SPZ can be represented by a Taylor model:

Proposition 3.1.13. *A SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$ can be equivalently represented by the set defined by a Taylor model*

$$\mathcal{PZ} = \big\{\mathcal{T}(x) \ \big| \ x \in [-\mathbf{1}, \mathbf{1}]\big\}, \qquad \mathcal{T}(x) = \big\langle w(x), \emptyset, [-\mathbf{1}, \mathbf{1}]\big\rangle_{TM},$$

where

$$w(x) = c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} x_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} x_{(p+j)} \ G_{I(\cdot,j)}.$$

The computational complexity of the conversion is $\mathcal{O}(1)$.

Proof. Substitution of the dependent factors α_k and the independent factors β_j in the definition of a SPZ in Def. 3.1.1 with variables $x_{(k)}$ and $x_{(p+j)}$ yields a Taylor model as defined in Def. 2.2.8.

Complexity: The conversion does not require any computations, so that the computational complexity is constant. □

While SPZs and Taylor models are equivalent in regard to the sets they can represent, there exist some key differences: For SPZs the independent generators can be viewed

as a zonotope remainder. Since computations on this zonotope remainder are less over-approximative than computations on the interval remainder of Taylor models while still being computationally efficient, SPZs often offer a better trade-off between accuracy and computation costs. Moreover, the generator-based representation used by SPZs significantly simplifies the task of reducing the representation size without adding much over-approximation, as we demonstrate later on in Sec. 3.1.7. For Taylor models, on the other hand, reduction of the representation size [43, Sec. II] is often more conservative.

3.1.4 Enclosure by other Set Representations

Many algorithms that compute with sets require to enclose sets by simpler set representations for computational reasons. Therefore, we show in this section how SPZs can be enclosed by other set representations. As a running example throughout this section we consider the SPZ

$$\mathcal{PZ} = \left\langle \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 1 \end{bmatrix}, [\,], \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}. \tag{3.4}$$

We first show how a SPZ can be enclosed by a zonotope:

Proposition 3.1.14. *(Zonotope Enclosure) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, the operation* `zonotope` returns a zonotope that encloses $\mathcal{PZ}$:

$$\mathcal{PZ} \subseteq \texttt{zonotope}\,(\mathcal{PZ}) = \left\langle c + 0.5 \sum_{i \in \mathcal{H}} G_{(\cdot,i)}, \begin{bmatrix} 0.5 \cdot G_{(\cdot,\mathcal{H})} & G_{(\cdot,\mathcal{K})} & G_I \end{bmatrix} \right\rangle_Z,$$

where

$$\mathcal{H} = \left\{ i \;\middle|\; \prod_{j=1}^{p} \big(1 - (E_{(j,i)} \bmod 2)\big) = 1 \right\}, \quad \mathcal{K} = \{1, \ldots, h\} \setminus \mathcal{H}.$$

The computational complexity with respect to the dimension n is $\mathcal{O}(n^2)$.

Proof. We over-approximate the variable parts of all monomials in the definition of SPZs in 3.1.1 with additional independent factors β_{q+i}, which yields a zonotope as defined in Def. 2.2.4:

$$\mathcal{PZ} = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}$$

$$= \left\{ c + \sum_{i \in \mathcal{H}} \underbrace{\left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right)}_{\in [0,1]} G_{(\cdot,i)} + \sum_{i \in \mathcal{K}} \underbrace{\left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right)}_{\in [-1,1]} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}$$

$$= \left\{ c + 0.5 \sum_{i \in \mathcal{H}} G_{(\cdot,i)} + 0.5 \sum_{i \in \mathcal{H}} \beta_{q+i}\ G_{(\cdot,i)} + \sum_{i \in \mathcal{K}} \beta_{q+i}\ G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j, \beta_{q+i} \in [-1, 1] \right\}$$

$$\overset{\text{Def. 2.2.4}}{=} \left\langle c + 0.5 \sum_{i \in \mathcal{H}} G_{(\cdot,i)}, \begin{bmatrix} 0.5 \cdot G_{(\cdot,\mathcal{H})} & G_{(\cdot,\mathcal{K})} & G_I \end{bmatrix} \right\rangle_Z,$$

where we exploit that monomials with exclusively even exponents ($i \in \mathcal{H}$) are strictly positive so that we can enclose them tighter using

$$\forall i \in \mathcal{H}: \quad \left(\prod_{k=1}^{p} [-1,1]^{E_{(k,i)}} \right) G_{(\cdot,i)} = [0,1] G_{(\cdot,i)} = 0.5\ G_{(\cdot,i)} + [-1,1]\ 0.5\ G_{(\cdot,i)}.$$

For all other monomials ($i \in \mathcal{K}$), evaluation of the monomial variable part directly results in the interval $[-1,1]$. A dependent factor affects all monomials that contain the dependent factor. Since the over-approximation of the monomial variable parts with new independent factors destroys this dependence between different monomials (e.g., $\{\alpha_1\alpha_2^2 + \alpha_1^3\alpha_2 \mid \alpha_1, \alpha_2 \in [-1,1]\} \subseteq \{\beta_1 + \beta_2 \mid \beta_1, \beta_2 \in [-1,1]\}$), the resulting zonotope encloses $\mathcal{PZ}$ because removing dependence results in an over-approximation [30, Sec. 2].

Complexity: The calculation of the set $\mathcal{H}$ has complexity $\mathcal{O}(ph)$, and the summations and multiplications required for the construction of the enclosing zonotope have complexity $\mathcal{O}(nh)$ in the worst-case where all exponents are exclusively even, resulting in an overall complexity of

$$\mathcal{O}(ph) + \mathcal{O}(nh) = \mathcal{O}(h(p+n)), \tag{3.5}$$

which is $\mathcal{O}(n^2)$ using Assumption 3.1.3. □

The enclosing zonotope for the SPZ in (3.4) calculated with Prop. 3.1.14 is visualized in Fig. 3.2. Next, we show how to enclose a SPZ by a polytope:

Proposition 3.1.15. *(Polytope Enclosure) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$, *the operation* `polytope` returns a polytope that encloses $\mathcal{PZ}$:

$$\mathcal{PZ} \subseteq \texttt{polytope}(\mathcal{PZ}) = \langle [v_1\ \ldots\ v_s] \rangle_V,$$

where the polytope vertex-representation $\langle [v_1\ \ldots\ v_s] \rangle_V$ is computed by applying Alg. 5 to the SPZ

$$\overline{\mathcal{PZ}} = \left\langle c_z, G_{(\cdot,\mathcal{K})}, [G_I\ G_z], E_{(\cdot,\mathcal{K})}, id \right\rangle_{PZ}$$

with

$$\mathcal{H} = \left\{ i \;\middle|\; \exists j \in \{1,\ldots,p\}:\ E_{(j,i)} > 1 \right\}, \quad \mathcal{K} = \{1,\ldots,h\} \setminus \mathcal{H},$$

$$\langle c_z, G_z \rangle_Z = \texttt{zonotope}\big(\langle c, G_{(\cdot,\mathcal{H})}, [\,], E_{(\cdot,\mathcal{H})}, id \rangle_{PZ}\big),$$

and the zonotope enclosure is computed using Prop. 3.1.14. The computational complexity with respect to the dimension n is $\mathcal{O}(4^n n)$.

Proof. Alg. 5 computes an enclosing polytope for the Z-representation introduced later in Sec. 3.3, which is a special case of a SPZ where the exponent matrix is restricted to having only zeros or ones as entries. We therefore first split $\mathcal{PZ}$ into one part $\langle \mathbf{0}, G_{(\cdot,\mathcal{K})}, G_I, E_{(\cdot,\mathcal{K})}, id \rangle_{PZ}$ with only zeros or ones in the exponent matrix, and one remainder part $\langle c, G_{(\cdot,\mathcal{H})}, [\,], E_{(\cdot,\mathcal{H})}, id \rangle_{PZ}$. In order to remove exponents that are greater

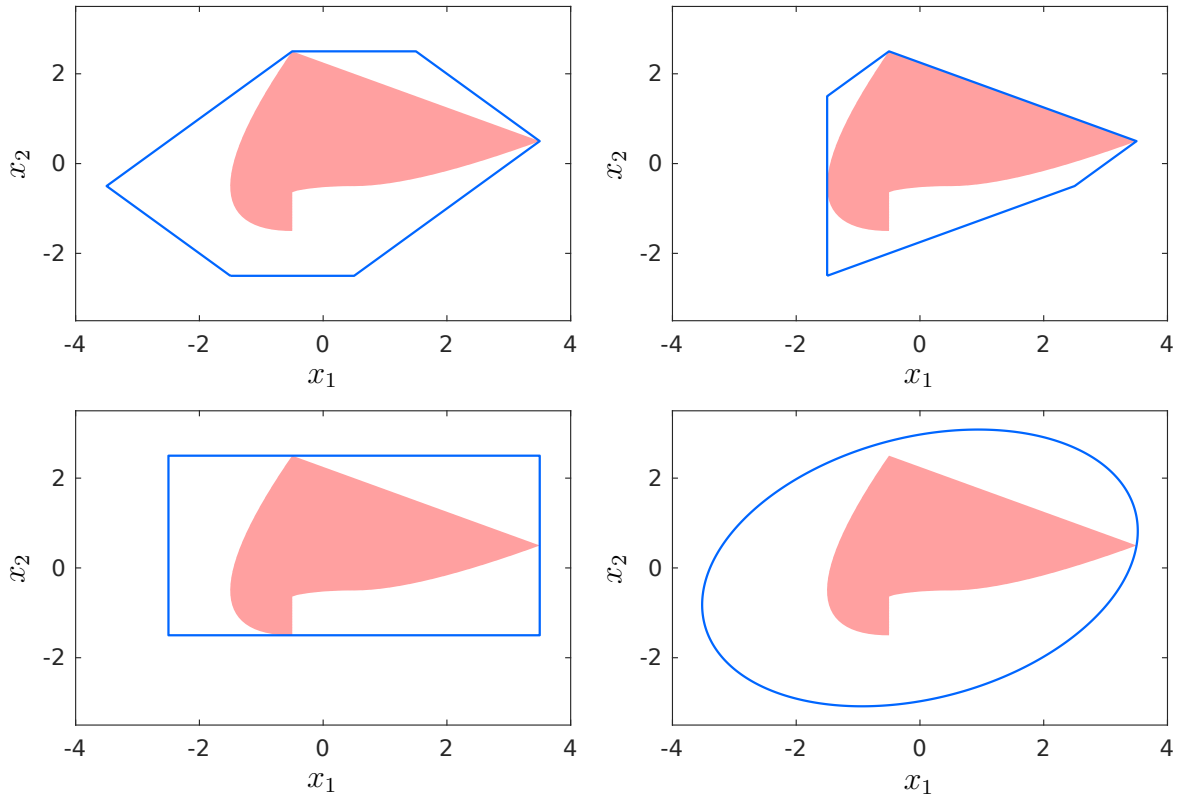

Figure 3.2: Enclosure of the SPZ in (3.4) with a zonotope (top, left), a polytope (top, right), an interval (bottom, left), and an ellipsoid (bottom, right).

than one the remainder part is enclosed by a zonotope using Prop. 3.1.14. Combination of the two parts finally yields the SPZ $\overline{\mathcal{PZ}}$ which satisfies

$$\mathcal{PZ} \subseteq \overline{\mathcal{PZ}} \overset{\text{Alg. 5}}{\subseteq} \langle [v_1 \ \dots \ v_s] \rangle_V.$$

Complexity: The calculation of the sets $\mathcal{H}$ and $\mathcal{K}$ has complexity $\mathcal{O}(ph)$ and the computation of an enclosing zonotope using Prop. 3.1.14 has complexity $\mathcal{O}(h(p+n))$ according to (3.5). Moreover, the complexity of Alg. 5 is $\mathcal{O}(2^{p\lfloor n/2 \rfloor} + 4^p(p+n))$ according to Prop. 3.3.9. The overall complexity is therefore

$$\mathcal{O}(ph) + \mathcal{O}(h(p+n)) + \mathcal{O}(2^{p\lfloor n/2 \rfloor} + 4^p(p+n)),$$

which is $\mathcal{O}(4^n n)$ using Assumption 3.1.3. □

Since every polytope in V-representation can be converted to an equivalent H-representation, Prop. 3.1.15 can also be used to enclose a SPZ by a polytope in H-representation. The enclosing polytope for the SPZ in (3.4) calculated with Prop. 3.1.15 is visualized in Fig. 3.2. We now demonstrate how to compute an over-approximation of the support function of a SPZ, which forms the basis for enclosing a SPZ by an interval or a template polyhedron:

Proposition 3.1.16. *(Support Function Enclosure) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, an over-approximation of its support function* $s_{\mathcal{PZ}}(d)$ *for a given direction* $d \in \mathbb{R}^n$ *can be computed as*

$$s_{\mathcal{PZ}}(d) \leq d^T c + u + \sum_{j=1}^{q} \left| d^T \; G_{I(\cdot,j)} \right|,$$

where $u \in \mathbb{R}$ *is computed using range bounding*

$$[l, u] = \texttt{bound}\big(w(\alpha), [-\mathbf{1}, \mathbf{1}]\big), \quad w(\alpha) = \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) d^T G_{(\cdot,i)}.$$

The computational complexity with respect to the dimension n *is* $\mathcal{O}(n^2) + \mathcal{O}(\texttt{bound})$.

Proof. We first project the SPZ onto the direction d using the linear map specified later on in Prop. 3.1.18, and then divide the resulting one-dimensional SPZ into one part with dependent generators and one with independent generators:

$$d^T \otimes \mathcal{PZ} \overset{\text{Prop. 3.1.18}}{=} \left\langle d^T c, d^T G, d^T G_I, E, id \right\rangle_{PZ} =$$

$$d^T c \oplus \underbrace{\left\{ \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) d^T G_{(i)} \;\middle|\; \alpha_k \in [-1, 1] \right\}}_{\subseteq [l,u] \text{ (dependent part)}} \oplus \underbrace{\left\{ \sum_{j=1}^{q} \beta_j d^T G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1, 1] \right\}}_{= [-a,a] \text{ (independent part)}}, \tag{3.6}$$

where $a = \sum_{j=1}^{q} |d^T G_{I(\cdot,j)}|$. The bounds for the independent part calculated by the sum of absolute values correspond to the support function of a zonotope [4, Sec. 2] and are exact. However, the upper bound u of the dependent part is over-approximative since the range bounding operation bound as defined in Def. 2.7.1 returns an over-approximation, so that

$$s_{\mathcal{PZ}}(d) \overset{\text{Def. 2.2.7}}{=} \max_{x \in \mathcal{PZ}} d^T x = \max_{y \in d^T \otimes \mathcal{PZ}} y \overset{(3.6)}{\leq} d^T c + \max_{y \in [l-a, u+a]} y = d^T c + u + \underbrace{\sum_{j=1}^{q} \left| d^T \; G_{I(\cdot,j)} \right|}_{a}.$$

Complexity: The vector matrix multiplications $d^T c$, $d^T G$ and $d^T G_I$ required for the projection $d^T \otimes \mathcal{PZ}$ of $\mathcal{PZ}$ onto direction d have complexity $\mathcal{O}(n) + \mathcal{O}(nh) + \mathcal{O}(nq) = \mathcal{O}(n(h+q))$ according to Tab. 2.1. Moreover, calculation of $\sum_{j=1}^{q} |d^T G_{I(\cdot,j)}|$ has complexity $\mathcal{O}(q)$ and the complexity $\mathcal{O}(\texttt{bound})$ for the calculation of the interval $[l, u]$ using range bounding depends on the applied range bounding technique (see Tab. 2.4). The overall complexity is therefore

$$\mathcal{O}(n(h+q)) + \mathcal{O}(q) + \mathcal{O}(\texttt{bound}) = \mathcal{O}(n(h+q)) + \mathcal{O}(\texttt{bound}), \tag{3.7}$$

which is $\mathcal{O}(n^2) + \mathcal{O}(\texttt{bound})$ using Assumption 3.1.3. The computational complexity of range bounding depends according to Tab. 2.4 on the number of elementary operations in the corresponding function. In our case, evaluation of the function $w(\alpha)$ requires at most

$$e = 2ph + (h - 1) \tag{3.8}$$

elementary operations if the result for $d^T G$ is precomputed, which results in the complexities listed in Tab. 3.3. □

Table 3.3: Computational complexity with respect to the dimension $n \in \mathbb{N}$ for support function, interval, and ellipsoid enclosure of SPZs using the range bounding techniques from Sec. 2.7.

Method	**Support Function**	**Interval**	**Ellipsoid**
Interval arithmetic	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$
Affine arithmetic	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^5)$
Taylor models	$\geq \mathcal{O}(n^3)$	$\geq \mathcal{O}(n^4)$	$\geq \mathcal{O}(n^5)$

Note that the tightness of the support function enclosure solely depends on the tightness of the bounds of the function $w(\alpha)$ obtained by one of the range bounding techniques from Sec. 2.7. The computational complexity for different range bounding techniques is summarized in Tab. 3.3. One approach to improve the tightness is to first split the SPZ multiple times using the `split` operator introduced later in Prop. 3.1.44, and then over-approximate the support functions for the split sets. A template polyhedron enclosing a SPZ can easily be constructed by enclosing the support function $s_{\mathcal{PZ}}(d)$ according to Prop. 3.1.16 for a discrete set of directions $\mathcal{D} = \{d_1, \ldots, d_r\},\ d_i \in \mathbb{R}^n,\ i = 1, \ldots, r$. The enclosure by an interval represents a special case where $\mathcal{D} = \{I_{n(\cdot,1)}, \ldots, I_{n(\cdot,n)}, -I_{n(\cdot,1)}, \ldots, -I_{n(\cdot,n)}\}$. We denote the enclosure of a SPZ $\mathcal{PZ}$ with an interval by $\texttt{interval}(\mathcal{PZ})$. The enclosing interval for the SPZ in (3.4) calculated with Prop. 3.1.16 using Bernstein polynomials for range bounding is visualized in Fig. 3.2. Finally, we demonstrate how to tightly enclose a SPZ by an ellipsoid:

Proposition 3.1.17. *(Ellipsoid Enclosure) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$, *the operation* `ellipsoid` returns an ellipsoid that encloses $\mathcal{PZ}$:

$$\mathcal{PZ} \subseteq \texttt{ellipsoid}(\mathcal{PZ}) = \langle c_e, Q \cdot b \rangle_E,$$

where

$$U = \texttt{pca}\big([-G\ -G_I\ G_I\ G]\big), \quad [l, u] = \texttt{interval}(U^T \otimes \mathcal{PZ}), \quad s_{\widehat{\mathcal{PZ}}}(1) \leq b,$$

$$c_e = 0.5 \cdot U(l + u), \quad \overline{\mathcal{PZ}} = \langle c - c_e, G, G_I, E, id \rangle_{PZ}, \quad \widehat{\mathcal{PZ}} \supseteq sq\big(\{Q^{-1}\}, \overline{\mathcal{PZ}}\big),$$

$$Q = 0.25 \cdot U \underbrace{\begin{bmatrix} \big(u_{(1)} - l_{(1)}\big)^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \big(u_{(n)} - l_{(n)}\big)^2 \end{bmatrix}}_{D} U^T, \quad Q^{-1} = 4 \cdot U D^{-1} U^T,$$

where the operator `pca` as defined in Def. 2.4.4 performs Principal Component Analysis, the linear map $U^T \otimes \mathcal{PZ}$ is computed using Prop. 3.1.18, the enclosure of the quadratic map $sq(\{Q^{-1}\}, \overline{\mathcal{PZ}})$ is computed using Prop. 3.1.31, and the interval enclosure as well as the over-approximation of the support function $s_{\widehat{\mathcal{PZ}}}(1)$ are calculated using Prop. 3.1.16. The computational complexity with respect to the dimension n is $\mathcal{O}(n^3 \log(n)) + n \cdot \mathcal{O}(\texttt{bound})$.

Proof. The main concept of the ellipsoid enclosure is as follows: We first compute the ellipsoid center c_e and the matrix Q that defines the orientation and shape of the ellipsoid based on a heuristic. For this, the principal axes of the ellipsoid represented by the columns of the matrix U are computed by applying PCA to the point cloud $[-G \ -G_I \ G_I \ G]$ defined by the generators of $\mathcal{PZ}$. Next, the extent of the ellipsoid along the principal axes corresponding to the diagonal entries of the matrix D is computed based on an interval enclosure $[l, u]$ of the set in the transformed state-space $U^T \otimes \mathcal{PZ}$. The same interval enclosure is used to heuristically select the ellipsoid center as $c_e = 0.5 \cdot U(l + u)$. Once the shape of the ellipsoid is fixed, the size of the ellipsoid is properly scaled by the scalar $b \in \mathbb{R}_{\geq 0}$ such that the ellipsoid encloses $\mathcal{PZ}$. Since the correctness of the result from the proposition consequently doesn't depend on the matrix Q and the center c_e, it is sufficient to show that the scaling with b always yields an ellipsoid that encloses $\mathcal{PZ}$. Based on the definition of the quadratic map in (2.6) it holds that

$$\forall x \in \mathcal{PZ}: \quad 0 \leq \underbrace{\underbrace{(x-c_e)^T}_{\in \overline{\mathcal{PZ}}} Q^{-1} \underbrace{(x-c_e)}_{\in \overline{\mathcal{PZ}}}}_{\in \widehat{\mathcal{PZ}} \supseteq sq(\{Q^{-1}\}, \overline{\mathcal{PZ}})} \leq s_{\widehat{\mathcal{PZ}}}(1) \leq b, \tag{3.9}$$

where $(x - c_e)^T Q^{-1}(x - c_e) \geq 0$ since Q is positive definite. Using (3.9) we have

$$\mathcal{PZ} = \{x \in \mathcal{PZ}\} \overset{(3.9)}{=}$$

$$\{x \in \mathcal{PZ} \mid (x-c_e)^T Q^{-1}(x-c_e) \leq b\} \subseteq \{x \mid (x-c_e)^T Q^{-1}(x-c_e) \leq b\} =$$

$$\{x \mid (x-c_e)^T (Q \cdot b)^{-1}(x-c_e) \leq 1\} \overset{\text{Def. 2.2.6}}{=} \langle c_e, Q \cdot b \rangle_E,$$

where the equality of the first and second line follows from the fact that the constraint $(x - c_e)^T Q^{-1}(x - c_e) \leq b$ does not further restrict the values for x according to (3.9). Finally, the equation

$$Q^{-1} = (0.25 \cdot UDU^T)^{-1} = 0.25^{-1} \cdot (U^T)^{-1} D^{-1} U^{-1} = 4 \cdot UD^{-1}U^T$$

for computing Q^{-1} is obtained by exploiting that U is an orthonormal matrix for which $U^{-1} = U^T$ holds.

Complexity: Since $[-G \ -G_I \ G_I \ G]$ is a matrix of dimension $n \times 2(h+q)$, PCA has complexity $\mathcal{O}(2n^2(h+q)+n^3) = \mathcal{O}(n^2(h+q)+n^3)$ according to Tab. 2.1. Computation of the linear map $U^T \otimes \mathcal{PZ}$ using Prop. 3.1.18 has complexity $\mathcal{O}(n^2(h+q))$ according to (3.10). Since calculating an interval enclosure requires the over-approximation of $2n$ support functions using Prop. 3.1.16, computing $\texttt{interval}(U^T \otimes \mathcal{PZ})$ has complexity $2n \cdot (\mathcal{O}(n(h+q)) + \mathcal{O}(\texttt{bound})) = \mathcal{O}(n^2(h+q)) + n \cdot \mathcal{O}(\texttt{bound})$ according to (3.7). The matrix multiplications required to compute Q and Q^{-1} have complexity $\mathcal{O}(n^3)$ according to Tab. 2.1. Computation of the quadratic map $sq(\{Q^{-1}\}, \overline{\mathcal{PZ}})$ has complexity $\mathcal{O}(n^3(w + \log(n)))$ according to Prop. 3.1.31, which is $\mathcal{O}(n^3 \log(n))$ in our case since $w = 1$. Since the SPZ $\widehat{\mathcal{PZ}}$ calculated with Prop. 3.1.31 has $\widehat{h} = h^2 + 2h$ dependent and $\widehat{q} = 2q(h+1) + q^2$ independent generators according to Tab. 3.4, over-approximating its support function $s_{\widehat{\mathcal{PZ}}}(1)$ using Prop. 3.1.16

has complexity $\mathcal{O}(\widehat{n}(\widehat{h}+\widehat{q}))+\mathcal{O}(\texttt{bound}) = \mathcal{O}(\widehat{n}(h^2+2h+2q(h+1)+q^2))+\mathcal{O}(\texttt{bound})$ according to (3.7), which is $\mathcal{O}(h^2+hq+q^2)+\mathcal{O}(\texttt{bound})$ in our case since $\widehat{\mathcal{PZ}}$ is one-dimensional so that $\widehat{n}=1$. The overall complexity is therefore

$$\begin{aligned}&\mathcal{O}(n^2(h+q)+n^3)+\mathcal{O}(n^2(h+q))+\mathcal{O}(n^2(h+q))+n\cdot\mathcal{O}(\texttt{bound})\\&\qquad+\mathcal{O}(n^3)+\mathcal{O}(n^3\log(n))+\mathcal{O}(h^2+hq+q^2)+\mathcal{O}(\texttt{bound})\\&=\mathcal{O}(n^2(h+q)+n^3)+\mathcal{O}(n^3\log(n))+\mathcal{O}(h^2+hq+q^2)+n\cdot\mathcal{O}(\texttt{bound}),\end{aligned}$$

which is $\mathcal{O}(n^3\log(n))+n\cdot\mathcal{O}(\texttt{bound})$ using Assumption 3.1.3. Range bounding for computing the interval enclosure $\texttt{interval}(U^T\otimes\mathcal{PZ})$ consists of $e_1 = 2ph+h-1$ elementary operations according to (3.8). Moreover, range bounding for computing the support function enclosure $s_{\widehat{\mathcal{PZ}}}(1)\leq b$ consists of $e_2 = 2p\widehat{h}+\widehat{h}-1$ elementary operations according to (3.8), which is $e_2 = 2ph^2+4ph+h^2+2h-1$ since $\widehat{h}=h^2+2h$. The overall number of required elementary operations is therefore $e_1+e_2 = 2ph^2+6ph+h^2+3h-2$, which results according to Tab. 2.4 in the computational complexities listed in Tab. 3.3. □

The ellipsoid enclosure for the SPZ in (3.4) calculated with Prop. 3.1.17 is visualized in Fig. 3.2, where we used Bernstein polynomials for range bounding. Tab. 3.3 summarizes the computational complexity for calculating an ellipsoid enclosure with different range bounding techniques.

3.1.5 Basic Set Operations

In this section we derive closed-form expressions for the basic set operations in Sec. 2.1 on SPZs. We begin with the linear map:

Proposition 3.1.18. *(Linear Map) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$ and a matrix $M\in\mathbb{R}^{w\times n}$, the linear map is*

$$M\otimes\mathcal{PZ} = \big\langle Mc, MG, MG_I, E, id\big\rangle_{PZ},$$

which has complexity $\mathcal{O}(wn^2)$ with respect to the dimension n, where w is the number of rows of matrix M. The resulting SPZ is regular if $\mathcal{PZ}$ is regular.

Proof. The result follows directly from inserting the definition of SPZs in Def. 3.1.1 into the definition of the linear map in (2.1):

$$\begin{aligned}&M\otimes\mathcal{PZ}\overset{(2.1)}{=}\big\{Ms\;\big|\;s\in\mathcal{PZ}\big\}\overset{\text{Def. }3.1.1}{=}\\&\left\{Mc+\sum_{i=1}^{h}\left(\prod_{k=1}^{p}\alpha_k^{E_{(k,i)}}\right)MG_{(\cdot,i)}+\sum_{j=1}^{q}\beta_j\;MG_{I(\cdot,j)}\;\middle|\;\alpha_k,\beta_j\in[-1,1]\right\}\\&=\big\langle Mc, MG, MG_I, E, id\big\rangle_{PZ}.\end{aligned}$$

Complexity: The three matrix multiplications Mc, MG, and MG_I have complexity

$$\mathcal{O}(wn) + \mathcal{O}(wnh) + \mathcal{O}(wnq) = \mathcal{O}(wn(h+q)) \tag{3.10}$$

according to Tab. 2.1, which is $\mathcal{O}(wn^2)$ using Assumption 3.1.3. □

Even though every zonotope can be represented as a SPZ, we provide a separate definition for the Minkowski sum of a SPZ and a zonotope for computational reasons:

Proposition 3.1.19. *(Minkowski Sum) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^n$, as well as a zonotope $\mathcal{Z} = \langle c_z, G_z \rangle_Z \subset \mathbb{R}^n$, their Minkowski sum is*

$$\mathcal{PZ}_1 \oplus \mathcal{PZ}_2 = \left\langle c_1 + c_2, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} G_{I,1} & G_{I,2} \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle_{PZ} \tag{3.11}$$

and

$$\mathcal{PZ}_1 \oplus \mathcal{Z} = \left\langle c_1 + c_z, G_1, \begin{bmatrix} G_{I,1} & G_z \end{bmatrix}, E_1, id_1 \right\rangle_{PZ}, \tag{3.12}$$

where (3.11) *and* (3.12) *both have complexity $\mathcal{O}(n)$ with respect to the dimension n. The resulting SPZs are regular if $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ are regular.*

Proof. Inserting the definition of SPZs in Def. 3.1.1 into the definition of the Minkowski sum in (2.2) yields

$$\mathcal{PZ}_1 \oplus \mathcal{PZ}_2 \overset{(2.2)}{=} \left\{ s_1 + s_2 \;\middle|\; s_1 \in \mathcal{PZ}_1,\ s_2 \in \mathcal{PZ}_2 \right\} \overset{\text{Def. 3.1.1}}{=}$$

$$\left\{ c_1 + c_2 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} \right.$$
$$\left. + \sum_{j=1}^{q_1} \beta_j \; G_{I,1(\cdot,j)} + \sum_{j=1}^{q_2} \beta_{q_1+j} \; G_{I,2(\cdot,j)} \;\middle|\; \alpha_k, \alpha_{p_1+k}, \beta_j, \beta_{p_1+j} \in [-1, 1] \right\}$$

$$= \left\langle c_1 + c_2, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} G_{I,1} & G_{I,2} \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle_{PZ},$$

where we generate new identifiers for all factors since the Minkowski sum per definition removes all dependencies between the two sets which are added. The Minkowski sum with a zonotope is obtained in the same way by inserting the definition of a zonotope in Def. 2.2.4, where the zonotope generators are added to the independent generators for computational reasons.

Complexity: Addition of the center vectors $c_1 + c_2$ and $c_1 + c_z$ has complexity $\mathcal{O}(n)$ and the generation of $p_1 + p_2$ unique identifiers using operation `uniqueID` has complexity $\mathcal{O}(p_1 + p_2)$ according to Tab. 2.2. The Minkowski sum with a zonotope has therefore complexity $\mathcal{O}(n)$, and Minkowski addition of two SPZs has complexity $\mathcal{O}(n) + \mathcal{O}(p_1 + p_2)$, which is $\mathcal{O}(n)$ using Assumption 3.1.3. □

During calculation of the Minkowski sum of two SPZs $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ using Prop. 3.1.19, possible dependencies between $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ due to common dependent factors get lost. We therefore introduce the exact addition $\mathcal{PZ}_1 \boxplus \mathcal{PZ}_2$ of two SPZs, which explicitly considers dependencies between $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$. To bring the exponent matrices to a common representation, we apply `mergeID` as defined in Prop. 3.1.5 prior to the computation.

Proposition 3.1.20. *(Exact Addition) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id \rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id \rangle_{PZ} \subset \mathbb{R}^n$ with a common identifier vector id, their exact addition is defined as*

$$\mathcal{PZ}_1 \boxplus \mathcal{PZ}_2 = \left\langle c_1 + c_2, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} G_{I,1} & G_{I,2} \end{bmatrix}, \begin{bmatrix} E_1 & E_2 \end{bmatrix}, id \right\rangle_{PZ},$$

which has complexity $\mathcal{O}(n^2 \log(n))$ with respect to the dimension n. The `compact` operation as defined in Prop. 3.1.7 is applied to make the resulting SPZ regular.

Proof. The result is identical to the one for the Minkowski sum of two SPZs in (3.11), with the difference that the common identifier vector *id* is used instead of newly generated unique identifiers.

Complexity: Merging the identifier vectors using `mergeID` has complexity $\mathcal{O}(p_1 p_2)$ according to Prop. 3.1.5 and the addition of the center vectors $c_1 + c_2$ has complexity $\mathcal{O}(n)$. Subsequent application of the `compact` operator has complexity $\mathcal{O}(\overline{h}(n + \overline{p}\log(\overline{h})))$ according to Prop. 3.1.7, where $\overline{h} = h_1 + h_2$ and $\overline{p} = p_1 = p_2$ denote the number of dependent generators and the number of dependent factors of the resulting SPZ. The overall complexity is therefore

$$\begin{aligned} &\mathcal{O}(p_1 p_2) + \mathcal{O}(n) + \mathcal{O}\big(\overline{h}(n + \overline{p}\log(\overline{h}))\big) \\ &\quad \overset{\substack{\overline{h}=h_1+h_2 \\ \overline{p}=p_1=p_2}}{=} \mathcal{O}(p_1 p_2) + \mathcal{O}(n) + \mathcal{O}\big((h_1 + h_2)(n + p_1 \log(h_1 + h_2))\big), \end{aligned}$$

which is $\mathcal{O}(n^2 \log(n))$ using Assumption 3.1.3. □

Let us demonstrate the difference between Minkowski sum and exact addition with an example:

Example 3.1.21. *Consider that we want to compute the set $\mathcal{B} = \{s + Ms \mid s \in \mathcal{PZ}\}$ with*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix}, [\,], \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}, \quad M = \begin{bmatrix} -0.5 & 0.2 \\ -0.1 & 0.6 \end{bmatrix}.$$

As visualized in Fig. 3.3, using the Minkowski sum yields the over-approximation $\mathcal{B} \subseteq \mathcal{PZ} \oplus (M \otimes \mathcal{PZ})$. With the exact addition, on the other hand, we obtain the exact result $\mathcal{B} = \mathcal{PZ} \boxplus (M \otimes \mathcal{PZ})$ since the exact addition explicitly considers the dependencies between the sets $\mathcal{PZ}$ and $M \otimes \mathcal{PZ}$. While for this simple example the exact result can also be obtained without the exact addition by using the simplification $\mathcal{B} = (I_2 + M) \otimes \mathcal{PZ}$, this is in general not possible for more complicated equations and algorithms.

Next, we consider the Cartesian product. Even though every zonotope can be represented as a SPZ, we provide a separate definition for the Cartesian product of a SPZ and a zonotope for computational reasons:

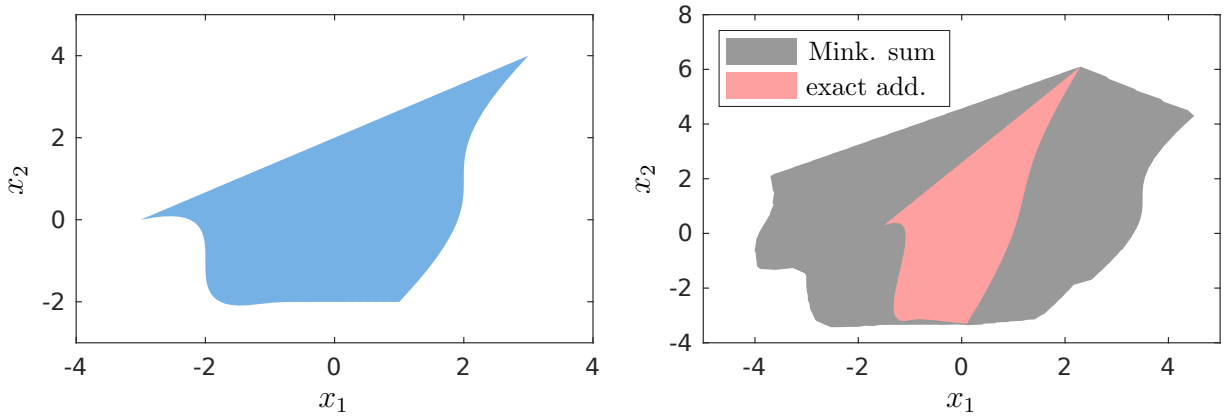

Figure 3.3: Visualization of the results from Example 3.1.21, where the SPZ $\mathcal{PZ}$ is shown on the left and the resulting sets $\mathcal{B}$ calculated with Minkowski sum and exact addition are shown on the right.

Proposition 3.1.22. *(Cartesian Product) Given two SPZs,* $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ *and* $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^w$*, as well as a zonotope* $\mathcal{Z} = \langle c_z, G_z \rangle_Z \subset \mathbb{R}^w$*, their Cartesian product is*

$$\mathcal{PZ}_1 \times \mathcal{PZ}_2 = \left\langle \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \begin{bmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{bmatrix}, \begin{bmatrix} G_{I,1} & \mathbf{0} \\ \mathbf{0} & G_{I,2} \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle_{PZ} \tag{3.13}$$

and

$$\mathcal{PZ}_1 \times \mathcal{Z} = \left\langle \begin{bmatrix} c_1 \\ c_z \end{bmatrix} \begin{bmatrix} G_1 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} G_{I,1} & \mathbf{0} \\ \mathbf{0} & G_z \end{bmatrix}, E_1, id_1 \right\rangle_{PZ}, \tag{3.14}$$

where (3.13) *has complexity* $\mathcal{O}(n)$ *with respect to the dimension* n*, and* (3.14) *has complexity* $\mathcal{O}(1)$*. The resulting SPZs are regular if* $\mathcal{PZ}_1$ *and* $\mathcal{PZ}_2$ *are regular.*

Proof. Inserting the definition of SPZs in Def. 3.1.1 into the definition of the Cartesian product in (2.4) yields

$$\mathcal{PZ}_1 \times \mathcal{PZ}_2 \overset{(2.4)}{=} \left\{ [s_1^T \; s_2^T]^T \;\middle|\; s_1 \in \mathcal{PZ}_1.\; s_2 \in \mathcal{PZ}_2 \right\} \overset{\text{Def. 3.1.1}}{=}$$

$$\left\{ \begin{bmatrix} c_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ c_2 \end{bmatrix} + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) \begin{bmatrix} G_{1(\cdot,i)} \\ \mathbf{0} \end{bmatrix} + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) \begin{bmatrix} \mathbf{0} \\ G_{2(\cdot,i)} \end{bmatrix} \right.$$
$$\left. + \sum_{j=1}^{q_1} \beta_j \begin{bmatrix} G_{I,1(\cdot,j)} \\ \mathbf{0} \end{bmatrix} + \sum_{j=1}^{q_2} \beta_{q_1+j} \begin{bmatrix} \mathbf{0} \\ G_{I,2(\cdot,j)} \end{bmatrix} \;\middle|\; \alpha_k, \alpha_{p_1+k}, \beta_j, \beta_{q_1+j} \in [-1, 1] \right\}$$

$$= \left\langle \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \begin{bmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{bmatrix}, \begin{bmatrix} G_{I,1} & \mathbf{0} \\ \mathbf{0} & G_{I,2} \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle_{PZ}.$$

The Cartesian product with a zonotope is obtained in the same way by inserting the definition of a zonotope in Def. 2.2.4, where the zonotope generators are added to the independent generators for computational reasons.

Complexity: The construction of the resulting SPZs only involves concatenations and therefore has complexity $\mathcal{O}(1)$. For the Cartesian product of two SPZs, $p_1 + p_2$ unique identifiers have to be generated with `uniqueID`, resulting in complexity $\mathcal{O}(p_1+p_2)$ according to Tab. 2.2, which is $\mathcal{O}(n)$ using Assumption 3.1.3. □

For both, the linear combination and the convex hull on SPZs, we require the enclosure of the convex hull of two zonotopes, which we compute as follows:

Proposition 3.1.23. *Given two zonotopes, $\mathcal{Z}_1 = \langle c_1, G_1 \rangle_Z \subset \mathbb{R}^n$ and $\mathcal{Z}_2 = \langle c_2, G_2 \rangle_Z \subset \mathbb{R}^n$, an enclosure of their convex hull can according to [24, Eq. (2.2)] be computed as*

$$conv\big(\mathcal{Z}_1, \mathcal{Z}_2\big) \subseteq \big\langle 0.5(c_1 + c_2), \overline{G} \big\rangle_Z,$$

where

$$\overline{G} = \begin{cases} \begin{bmatrix} 0.5(c_1 - c_2) & \widehat{G}_1 & G_{1(\cdot,\{l_2+1,\dots,l_1\})} \end{bmatrix}, & l_1 \geq l_2 \\ \begin{bmatrix} 0.5(c_1 - c_2) & \widehat{G}_2 & G_{2(\cdot,\{l_1+1,\dots,l_2\})} \end{bmatrix}, & l_1 < l_2 \end{cases},$$

$$\widehat{G}_1 = 0.5 \begin{bmatrix} G_{1(\cdot,\mathcal{K}_2)} + G_2 & G_{1(\cdot,\mathcal{K}_2)} - G_2 \end{bmatrix}, \quad \mathcal{K}_1 = \{1, \dots, l_1\},$$

$$\widehat{G}_2 = 0.5 \begin{bmatrix} G_1 + G_{2(\cdot,\mathcal{K}_1)} & G_1 - G_{2(\cdot,\mathcal{K}_1)} \end{bmatrix}, \quad \mathcal{K}_2 = \{1, \dots, l_2\}.$$

The computational complexity is $\mathcal{O}(n \min(l_1, l_2))$, where l_1 and l_2 are the number of generators of $\mathcal{Z}_1$ and $\mathcal{Z}_2$.

Proof. Since the result is taken from [24, Eq. (2.2)], the corresponding proof can be found in [24, Ch. 2.4].

Complexity: Calculation of the vectors $0.5(c_1 + c_2)$ and $0.5(c_1 - c_2)$ has computational complexity $\mathcal{O}(n)$, and construction of the matrix $\widehat{G}_1$ or $\widehat{G}_2$ has complexity $\mathcal{O}(2n \min(l_1, l_2))$. The overall complexity is therefore $\mathcal{O}(n) + \mathcal{O}(2n \min(l_1, l_2)) = \mathcal{O}(n \min(l_1, l_2))$. □

Moreover, we require the following lemma:

Lemma 3.1.24. *Given sets S_1, S_2, S_3, $S_4 \subset \mathbb{R}^n$, the relation*

$$comb(S_1 \oplus S_2, S_3 \oplus S_4) \subseteq comb(S_1, S_3) \oplus comb(S_2, S_4)$$

holds for the linear combination.

Proof. According to the definition of the linear combination in (2.11) we have

$$comb(S_1 \oplus S_2, S_3 \oplus S_4) \overset{(2.11),(2.2)}{=}$$

$$\big\{ 0.5(1+\lambda)(s_1 + s_2) + 0.5(1-\lambda)(s_3 + s_4) \;\big|\; s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2, s_3 \in \mathcal{S}_3, s_4 \in \mathcal{S}_4, \lambda \in [-1, 1] \big\}$$

$$= \big\{ \underbrace{0.5(1+\lambda)s_1 + 0.5(1-\lambda)s_3}_{comb(\mathcal{S}_1,\mathcal{S}_3)} + \underbrace{0.5(1+\lambda)s_2 + 0.5(1-\lambda)s_4}_{comb(\mathcal{S}_2,\mathcal{S}_4)} \;\big|\; s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2, s_3 \in \mathcal{S}_3,$$

$$s_4 \in \mathcal{S}_4, \lambda \in [-1,1]\big\}$$

$$\overset{(2.2)}{\subseteq} \big\{0.5(1+\lambda)s_1 + 0.5(1-\lambda)s_3 \;\big|\; s_1 \in \mathcal{S}_1, s_3 \in \mathcal{S}_3, \lambda \in [-1,1]\big\}$$
$$\oplus \big\{0.5(1+\lambda)s_2 + 0.5(1-\lambda)s_4 \;\big|\; s_2 \in \mathcal{S}_2, s_4 \in \mathcal{S}_4, \lambda \in [-1,1]\big\},$$

where the last step results in an over-approximation since the dependence between the variable λ in the set $comb(\mathcal{S}_1, \mathcal{S}_3)$ and the variable λ in the set $comb(\mathcal{S}_2, \mathcal{S}_4)$ gets lost, and removing dependence results in an over-approximation [30, Sec. 2]. □

For the linear combination as defined in (2.11) we first consider the case without independent generators and later present the general case:

Proposition 3.1.25. *(Linear Combination) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, [\,], E_1, id_1\rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_1, G_2, [\,], E_2, id_2\rangle_{PZ} \subset \mathbb{R}^n$, their linear combination is*

$$comb(\mathcal{PZ}_1, \mathcal{PZ}_2) = \Bigg\langle 0.5(c_1+c_2), 0.5\begin{bmatrix}(c_1-c_2) & G_1 & G_1 & G_2 & -G_2\end{bmatrix}, [\,],$$
$$\begin{bmatrix}\mathbf{0} & E_1 & E_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & E_2 & E_2\\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}\end{bmatrix}, \texttt{uniqueID}(p_1+p_2+1)\Bigg\rangle_{PZ},$$

which has complexity $\mathcal{O}(n^2)$ with respect to the dimension n. The resulting SPZ is regular if $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ are regular.

Proof. We first generate new unique identifiers for all dependent factors to remove possible dependencies between the two SPZs. Inserting the definition of SPZs in Def. 3.1.1 into the definition of the linear combination is (2.11) yields

$$comb(\mathcal{PZ}_1, \mathcal{PZ}_2) \overset{(2.11)}{=}$$

$$\left\{\frac{1}{2}(1+\lambda)s_1 + \frac{1}{2}(1-\lambda)s_2 \;\middle|\; s_1 \in \mathcal{PZ}_1,\ s_2 \in \mathcal{PZ}_2,\ \lambda \in [-1,1]\right\} \overset{\text{Def. } 3.1.1}{=}$$

$$\Bigg\{\frac{1}{2}(c_1+c_2) + \frac{1}{2}(c_1-c_2)\lambda + \frac{1}{2}\sum_{i=1}^{h_1}\Bigg(\prod_{k=1}^{p_1}\alpha_k^{E_{1(k,i)}}\Bigg)G_{1(\cdot,i)} + \frac{1}{2}\sum_{i=1}^{h_1}\lambda\Bigg(\prod_{k=1}^{p_1}\alpha_k^{E_{1(k,i)}}\Bigg)G_{1(\cdot,i)}$$
$$+ \frac{1}{2}\sum_{i=1}^{h_2}\Bigg(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{E_{2(k,i)}}\Bigg)G_{2(\cdot,i)} - \frac{1}{2}\sum_{i=1}^{h_2}\lambda\Bigg(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{E_{2(k,i)}}\Bigg)G_{2(\cdot,i)} \;\Bigg|\; \alpha_k, \alpha_{p_1+k}, \lambda \in [-1,1]\Bigg\}$$

$$\overset{\alpha_{p_1+p_2+1}:=\lambda}{=} \Bigg\langle \frac{1}{2}(c_1+c_2), \frac{1}{2}\begin{bmatrix}(c_1-c_2) & G_1 & G_1 & G_2 & -G_2\end{bmatrix}, [\,]$$
$$\begin{bmatrix}\mathbf{0} & E_1 & E_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & E_2 & E_2\\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}\end{bmatrix}, \texttt{uniqueID}(p_1+p_2+1)\Bigg\rangle_{PZ},$$

where we substituted the parameter λ with an additional dependent factor $\alpha_{p_1+p_2+1} = \lambda$. Since $\lambda \in [-1, 1]$ and $\alpha_{p_1+p_2+1} \in [-1, 1]$, this substitution does not change the set.

Complexity: Construction of the constant offset $0.5(c_1 + c_2)$ requires n additions and n multiplications, and construction of the generator matrix $0.5[(c_1 - c_2)\ G_1\ G_1\ G_2\ -G_2]$ requires n subtractions and $n(2h_1 + 2h_2 + 1)$ multiplications. The overall complexity is therefore

$$\mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n(2h_1 + 2h_2 + 1)) = \mathcal{O}(n(h_1 + h_2), \tag{3.15}$$

which is $\mathcal{O}(n^2)$ using Assumption 3.1.3. □

We now extend Prop. 3.1.25 to the general case including independent generators. Since according to Prop. 3.1.4 every SPZ can equivalently be represented as a SPZ without independent generators, we could compute the exact linear combination based on Prop. 3.1.25. This, however, would significantly increase the number of dependent generators and consequently slow down subsequent computations on the SPZ. Instead of the exact result we therefore compute a tight enclosure of the linear combination if the SPZs contain independent generators to achieve a good trade-off between accuracy and computational cost:

Proposition 3.1.26. *(Linear Combination) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1\rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2\rangle_{PZ} \subset \mathbb{R}^n$, it holds that*

$$comb(\mathcal{PZ}_1, \mathcal{PZ}_2) \subseteq \left\langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}, \overline{id}\right\rangle_{PZ},$$

where

$$\left\langle \overline{c}, \overline{G}, [\ ], \overline{E}, \overline{id}\right\rangle_{PZ} = comb\big(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big), \quad \left\langle \mathbf{0}, \overline{G}_I\right\rangle_Z \supseteq conv\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big),$$

$$\overline{\mathcal{PZ}}_1 = \left\langle c_1, G_1, [\ ], E_1, id_1\right\rangle_{PZ}, \quad \overline{\mathcal{PZ}}_2 = \left\langle c_2, G_2, [\ ], E_2, id_2\right\rangle_{PZ},$$

the linear combination $comb(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2)$ is calculated using Prop. 3.1.25, and the enclosure of the convex hull of the two zonotopes $\langle \mathbf{0}, G_{I,1}\rangle_Z$ and $\langle \mathbf{0}, G_{I,2}\rangle_Z$ is computed using Prop. 3.1.23. The computational complexity with respect to the dimension n is $\mathcal{O}(n^2)$, and the resulting SPZ is regular if $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ are regular.

Proof. Each SPZ can equivalently be represented as the Minkowski sum of a SPZ and a zonotope:

$$\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} = \langle c, G, [\ ], E, id\rangle_{PZ} \oplus \langle \mathbf{0}, G_I\rangle_Z. \tag{3.16}$$

According to Lemma 3.1.24 it therefore holds that

$$comb(\mathcal{PZ}_1, \mathcal{PZ}_2) \overset{(3.16)}{=} comb\big(\overline{\mathcal{PZ}}_1 \oplus \langle \mathbf{0}, G_{I,1}\rangle_Z, \overline{\mathcal{PZ}}_2 \oplus \langle \mathbf{0}, G_{I,2}\rangle_Z\big)$$

$$\overset{\text{Lemma 3.1.24}}{\subseteq} comb\big(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big) \oplus comb\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big) =$$

$$= \underbrace{comb\big(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big)}_{=\langle \overline{c}, \overline{G}, [\ ], \overline{E}, \overline{id}\rangle_{PZ}} \oplus \underbrace{conv\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big)}_{\subseteq \langle \mathbf{0}, \overline{G}_I\rangle_Z} \subseteq \left\langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}, \overline{id}\right\rangle_{PZ},$$

where we exploit that linear combination and convex hull are equivalent for zonotopes since zonotopes are convex.

Complexity: According to (3.15), the complexity for computing the linear combination of two SPZs without independent generators is $\mathcal{O}(n(h_1 + h_2))$. Moreover, the complexity for calculating an enclosure of the convex hull of two zonotopes using Prop. 3.1.23 is $\mathcal{O}(n \min(l_1, l_2))$, where in our case the number of zonotope generators is $l_1 = q_1$ and $l_2 = q_2$. Consequently, the overall complexity is

$$\mathcal{O}(n(h_1 + h_2)) + \mathcal{O}(n \min(q_1, q_2)), \tag{3.17}$$

which is $\mathcal{O}(n^2)$ using Assumption 3.1.3. □

For the convex hull of two SPZs we again first consider the case without independent generators, and later present the general case:

Proposition 3.1.27. *(Convex Hull) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, [\,], E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, [\,], E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^n$, their convex hull is*

$$\begin{aligned} conv(\mathcal{PZ}_1, \mathcal{PZ}_2) = \Bigg\langle & 0.5(c_1 + c_2), 0.5 \begin{bmatrix} (c_1 - c_2) & \overline{G}_1 & \overline{G}_1 & \overline{G}_2 & -\overline{G}_2 \end{bmatrix}, [\,], \\ & \begin{bmatrix} \mathbf{0} & \overline{E}_1 & \overline{E}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \overline{E}_2 & \overline{E}_2 \\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \texttt{uniqueID}(2p_1 + 2p_2 + 3) \Bigg\rangle_{PZ}, \end{aligned} \tag{3.18}$$

where

$$\begin{aligned} \overline{G}_1 &= 0.5 \begin{bmatrix} G_1 & G_1 & G_1 & -G_1 \end{bmatrix}, \quad \overline{G}_2 = 0.5 \begin{bmatrix} G_2 & G_2 & G_2 & -G_2 \end{bmatrix}, \\ \overline{E}_1 &= \begin{bmatrix} E_1 & E_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & E_1 & E_1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \overline{E}_2 = \begin{bmatrix} E_2 & E_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & E_2 & E_2 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \end{aligned} \tag{3.19}$$

which has complexity $\mathcal{O}(n^2)$ with respect to the dimension n. The resulting SPZ is regular if $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ are regular.

Proof. According to Lemma 2.1.1, the convex hull can be computed as

$$conv(\mathcal{PZ}_1, \mathcal{PZ}_2) = comb\big(comb(\mathcal{PZ}_1, \mathcal{PZ}_1), comb(\mathcal{PZ}_2, \mathcal{PZ}_2)\big).$$

Using Prop. 3.1.25, we obtain

$$\begin{aligned} comb(\mathcal{PZ}_1, \mathcal{PZ}_1) &= \big\langle c_1, \overline{G}_1, [\,], \overline{E}_1, \underbrace{\texttt{uniqueID}(2p_1 + 1)}_{\overline{id}_1} \big\rangle_{PZ}, \\ comb(\mathcal{PZ}_2, \mathcal{PZ}_2) &= \big\langle c_2, \overline{G}_2, [\,], \overline{E}_2, \underbrace{\texttt{uniqueID}(2p_2 + 1)}_{\overline{id}_2} \big\rangle_{PZ}. \end{aligned}$$

Applying Prop. 3.2.20 again to compute

$$conv(\mathcal{PZ}_1, \mathcal{PZ}_2) = comb\big(\langle c_1, \overline{G}_1, [\,], \overline{E}_1, \overline{id}_1 \rangle_{PZ}, \langle c_2, \overline{G}_2, [\,], \overline{E}_2, \overline{id}_2 \rangle_{PZ}\big)$$

then yields the result in (3.18).

Complexity: Computation of the matrices $\overline{G}_1$ and $\overline{G}_2$ in (3.19) requires $4(h_1 + h_2)$ multiplications, and construction of the constant offset $0.5(c_1 + c_2)$ in (3.18) requires n additions and n multiplications. Additionally, construction of the generator matrix $0.5[(c_1 - c_2)\ \overline{G}_1\ \overline{G}_1\ \overline{G}_2\ -\overline{G}_2]$ in (3.18) requires n subtractions and $n(2\overline{h}_1 + 2\overline{h}_2 + 1)$ multiplications, where $\overline{h}_1 = 4h_1$ and $\overline{h}_2 = 4h_2$ denote the number of columns of the matrices $\overline{G}_1$ and $\overline{G}_2$. Finally, generation of $2p_1 + 2p_2 + 3$ unique identifiers using `uniqueID` has complexity $\mathcal{O}(2p_1 + 2p_2 + 3)$ according to Tab. 2.2. The overall complexity is therefore

$$\begin{aligned}
&\mathcal{O}(4(h_1 + h_2)) + \mathcal{O}(4n) + \mathcal{O}(n(2\overline{h}_1 + 2\overline{h}_2 + 1)) + \mathcal{O}(2p_1 + 2p_2 + 3) \overset{\substack{\overline{h}_1=4h_1\\ \overline{h}_2=4h_2}}{=} \\
&\mathcal{O}(4(h_1 + h_2)) + \mathcal{O}(4n) + \mathcal{O}(n(8h_1 + 8h_2 + 1)) + \mathcal{O}(2p_1 + 2p_2 + 3) = \\
&\mathcal{O}(n(h_1 + h_2)) + \mathcal{O}(p_1 + p_2),
\end{aligned} \tag{3.20}$$

which is $\mathcal{O}(n^2)$ using Assumption 3.1.3. □

We now extend Prop. 3.1.27 to the general case including independent generators. As for the linear combination we compute a tight over-approximation instead of the exact result for computational reasons:

Proposition 3.1.28. *(Convex Hull) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1\rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2\rangle_{PZ} \subset \mathbb{R}^n$, it holds that*

$$conv(\mathcal{PZ}_1, \mathcal{PZ}_2) \subseteq \left\langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}, \overline{id}\right\rangle_{PZ},$$

where

$$\left\langle \overline{c}, \overline{G}, [\,], \overline{E}, \overline{id}\right\rangle_{PZ} = conv\big(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big), \quad \left\langle \mathbf{0}, \overline{G}_I\right\rangle_Z \supseteq conv\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big),$$

$$\overline{\mathcal{PZ}}_1 = \langle c_1, G_1, [\,], E_1, id_1\rangle_{PZ}, \quad \overline{\mathcal{PZ}}_2 = \langle c_2, G_2, [\,], E_2, id_2\rangle_{PZ},$$

the linear combination $comb(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2)$ is calculated using Prop. 3.1.27, and the enclosure of the convex hull of the two zonotopes $\langle \mathbf{0}, G_{I,1}\rangle_Z$ and $\langle \mathbf{0}, G_{I,2}\rangle_Z$ is computed using Prop. 3.1.23. The computational complexity with respect to the dimension n is $\mathcal{O}(n^2)$, and the resulting SPZ is regular if $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ are regular.

Proof. Since every SPZ can equivalently be represented as the Minkowski sum of a SPZ and a zonotope

$$\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} = \underbrace{\langle c, G, [\,], E, id\rangle_{PZ}}_{\overline{\mathcal{PZ}}} \oplus \langle \mathbf{0}, G_I\rangle_Z, \tag{3.21}$$

we have

$$\begin{aligned}
&comb(\mathcal{PZ}, \mathcal{PZ}) \overset{(3.21)}{=} comb\big(\overline{\mathcal{PZ}} \oplus \langle \mathbf{0}, G_I\rangle_Z, \overline{\mathcal{PZ}} \oplus \langle \mathbf{0}, G_I\rangle_Z\big) \overset{\text{Lemma 3.1.24}}{\subseteq} \\
&comb\big(\overline{\mathcal{PZ}}, \overline{\mathcal{PZ}}\big) \oplus \underbrace{comb\big(\langle \mathbf{0}, G_I\rangle_Z, \langle \mathbf{0}, G_I\rangle_Z\big)}_{\langle \mathbf{0}, G_I\rangle_Z} = comb\big(\overline{\mathcal{PZ}}, \overline{\mathcal{PZ}}\big) \oplus \langle \mathbf{0}, G_I\rangle_Z,
\end{aligned} \tag{3.22}$$

where we utilize that the linear combination of two identical zonotopes is equivalent to the zonotope itself since zonotopes are convex. Since every SPZ is connected, we can apply Lemma 2.1.1 to obtain

$$conv(\mathcal{PZ}_1, \mathcal{PZ}_2) \overset{\text{Lemma 2.1.1}}{=} comb\big(comb(\mathcal{PZ}_1, \mathcal{PZ}_1), comb(\mathcal{PZ}_2, \mathcal{PZ}_2)\big) \overset{(3.22)}{=}$$

$$comb\big(comb(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_1) \oplus \langle \mathbf{0}, G_{I,1}\rangle_Z, comb(\overline{\mathcal{PZ}}_2, \overline{\mathcal{PZ}}_2) \oplus \langle \mathbf{0}, G_{I,2}\rangle_Z\big) \overset{\text{Lemma 3.1.24}}{\subseteq}$$

$$\underbrace{comb\big(comb(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_1), comb(\overline{\mathcal{PZ}}_2, \overline{\mathcal{PZ}}_2)\big)}_{\overset{\text{Lemma 2.1.1}}{=} conv(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2))} \oplus comb\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big) =$$

$$\underbrace{conv\big(\overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big)}_{=\langle \overline{c}, \overline{G}, [\ ], \overline{E}, \overline{id}\rangle_{PZ}} \oplus \underbrace{conv\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big)}_{\subseteq \langle \mathbf{0}, \overline{G}_I\rangle_Z} \subseteq \langle \overline{c}, \overline{G}, \overline{G}_I, \overline{E}, \overline{id}\rangle_{PZ},$$

where we exploit that linear combination and convex hull are identical for zonotopes since zonotopes are convex.

Complexity: Computing the convex hull of two SPZs without independent generators using Prop. 3.1.27 has complexity $\mathcal{O}(n(h_1+h_2))+\mathcal{O}(p_1+p_2)$ according to (3.20). Moreover, the computation of an enclosure of the convex hull of two zonotopes using Prop. 3.1.23 has complexity $\mathcal{O}(n \min(l_1, l_2))$, where in our case the number of zonotope generators is $l_1 = q_1$ and $l_2 = q_2$. The overall complexity is therefore

$$\mathcal{O}(n(h_1 + h_2)) + \mathcal{O}(p_1 + p_2) + \mathcal{O}(n \min(q_1, q_2)),$$

which is $\mathcal{O}(n^2)$ using Assumption 3.1.3. □

Let us demonstrate the tightness of the enclosures for the linear combination and the convex hull with an example:

Example 3.1.29. *We consider the SPZs*

$$\mathcal{PZ}_1 = \left\langle \begin{bmatrix} -5 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & 2 \end{bmatrix}, [\ ], \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 \end{bmatrix} \right\rangle_{PZ}$$

and

$$\mathcal{PZ}_2 = \left\langle \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 & -2 & 2 \\ 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 \end{bmatrix} \right\rangle_{PZ}.$$

A comparison of the exact linear combination and the exact convex hull with the enclosures computed using Prop. 3.1.26 and Prop. 3.1.28 is shown in Fig. 3.4.

For the quadratic map, we first consider the special case without independent generators, and later present the general case. To bring the exponent matrices to a common representation, we apply `mergeID` as defined in Prop. 3.1.5 prior to the computation.

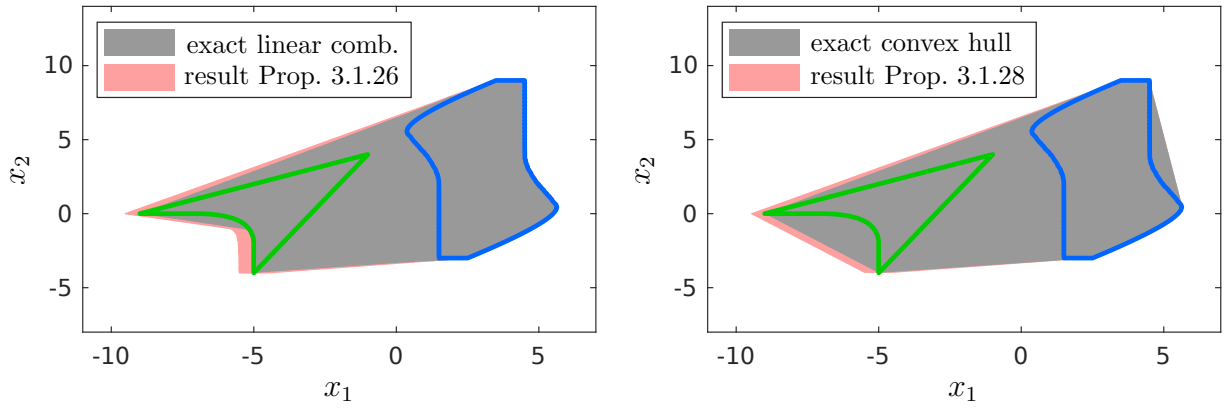

Figure 3.4: Visualization of the results from Example 3.1.29, where the SPZ $\mathcal{PZ}_1$ is depicted in green and the SPZ $\mathcal{PZ}_2$ is depicted in blue.

Proposition 3.1.30. *(Quadratic Map) Given two SPZs* $\mathcal{PZ}_1 = \langle c_1, G_1, [\,], E_1, id \rangle_{PZ} \subset \mathbb{R}^n$ *and* $\mathcal{PZ}_2 = \langle c_2, G_2, [\,], E_2, id \rangle_{PZ} \subset \mathbb{R}^n$ *with a common identifier vector* id *and a discrete set of matrices* $\mathcal{Q} = \{Q_1, \ldots, Q_w\}$ *with* $Q_i \in \mathbb{R}^{n \times n}$, $i = 1, \ldots, w$, *the result of the quadratic map is*

$$sq(\mathcal{Q}, \mathcal{PZ}_1, \mathcal{PZ}_2) = \left\langle c, \begin{bmatrix} \widehat{G}_1 & \widehat{G}_2 & \overline{G}_1 & \ldots & \overline{G}_{h_1} \end{bmatrix}, [\,], \begin{bmatrix} E_1 & E_2 & \overline{E}_1 & \ldots & \overline{E}_{h_1} \end{bmatrix}, id \right\rangle_{PZ},$$

where

$$\begin{aligned} c &= \begin{bmatrix} c_1^T Q_1 c_2 \\ \vdots \\ c_1^T Q_w c_2 \end{bmatrix}, \quad \widehat{G}_1 = \begin{bmatrix} c_2^T Q_1^T G_1 \\ \vdots \\ c_2^T Q_w^T G_1 \end{bmatrix}, \quad \widehat{G}_2 = \begin{bmatrix} c_1^T Q_1 G_2 \\ \vdots \\ c_1^T Q_w G_2 \end{bmatrix}, \\ \overline{E}_j &= E_2 + E_{1(\cdot,j)} \cdot \mathbf{1}, \quad \overline{G}_j = \begin{bmatrix} G_{1(\cdot,j)}^T Q_1 G_2 \\ \vdots \\ G_{1(\cdot,j)}^T Q_w G_2 \end{bmatrix}, \; j = 1, \ldots, h_1. \end{aligned} \tag{3.23}$$

The `compact` *operation as defined in Prop. 3.1.7 is applied to make the resulting SPZ regular. The computational complexity with respect to the dimension* n *is* $\mathcal{O}(n^3(w + \log(n)))$, *where* w *is the dimension of the resulting SPZ.*

Proof. Inserting the definition of SPZs in Def. 3.1.1 into the definition of the quadratic map in (2.6) yields:

$$sq(\mathcal{Q}, \mathcal{PZ}_1, \mathcal{PZ}_2) \overset{(2.6)}{=} \left\{ x \;\middle|\; x_{(i)} = s_1^T Q_i s_2, \; s_1 \in \mathcal{PZ}_1, \; s_2 \in \mathcal{PZ}_2, \; i = 1, \ldots, w \right\}$$

$$\overset{\text{Def. 3.1.1}}{=} \left\{ x \;\middle|\; x_{(i)} = \left(c_1 + \sum_{j=1}^{h_1} \prod_{k=1}^{p_1} \alpha_k^{E_{1(k,j)}} G_{1(\cdot,j)} \right)^T Q_i \left(c_2 + \sum_{l=1}^{h_2} \prod_{k=1}^{p_2} \alpha_k^{E_{2(k,l)}} G_{2(\cdot,l)} \right), \right.$$
$$\left. i = 1, \ldots, w, \; \alpha_k \in [-1, 1] \right\}$$

$$= \Bigg\{ x \;\Bigg|\; x_{(i)} = \underbrace{c_1^T Q_i c_2}_{\overset{(3.23)}{=} c_{(i)}} + \sum_{j=1}^{h_1} \prod_{k=1}^{p} \alpha_k^{E_{1(k,j)}} \underbrace{G_{1(\cdot,j)}^T Q_i c_2}_{\overset{(3.23)}{=} \widehat{G}_{1(i,j)}} + \sum_{l=1}^{h_2} \prod_{k=1}^{p} \alpha_k^{E_{2(k,l)}} \underbrace{c_1^T Q_i G_{2(\cdot,l)}}_{\overset{(3.23)}{=} \widehat{G}_{2(i,l)}}$$
$$+ \sum_{j=1}^{h_1} \sum_{l=1}^{h_2} \left(\prod_{k=1}^{p} \underbrace{\alpha_k^{E_{1(k,j)}+E_{2(k,l)}}}_{\overset{(3.23)}{=} \alpha_k^{\overline{E}_{j(k,l)}}} \right) \underbrace{G_{1(\cdot,j)}^T Q_i G_{2(\cdot,l)}}_{\overset{(3.23)}{=} \overline{G}_{j(i,l)}},\ i = 1, \ldots, w,\ \alpha_k \in [-1, 1] \Bigg\}$$

$$= \left\langle c, \begin{bmatrix} \widehat{G}_1 & \widehat{G}_2 & \overline{G}_1 & \ldots & \overline{G}_{h_1} \end{bmatrix}, [\,], \begin{bmatrix} E_1 & E_2 & \overline{E}_1 & \ldots & \overline{E}_{h_1} \end{bmatrix}, id \right\rangle_{PZ},$$

where $p = p_1 = p_2$ holds since both SPZs have an identical identifier vector id. Note that only the generator matrix, but not the exponent matrix, is different for each dimension $x_{(i)}$.

Complexity: According to Prop. 3.1.5, the complexity of `mergeID` is $\mathcal{O}(\widehat{p}_1 \widehat{p}_2)$, where $\widehat{p}_1$ and $\widehat{p}_2$ represent the number of dependent factors before applying `mergeID`. Since $p = p_1 = p_2 \leq \widehat{p}_1 + \widehat{p}_2$, this is identical to $\mathcal{O}(p^2)$. Using the complexity of matrix multiplication in Tab. 2.1 we can derive the complexity for constructing the constant offset c in (3.23) as $\mathcal{O}(n^2 w)$ and the complexity for constructing $\widehat{G}_1$ and $\widehat{G}_2$ in (3.23) as $\mathcal{O}(n^2 w(h_1 + h_2))$. Moreover, construction of the exponent matrices $\overline{E}_j$ in (3.23) has complexity $\mathcal{O}(h_1 h_2 p)$ and construction of the generator matrices $\overline{G}_j$ in (3.23) has complexity $\mathcal{O}(n^2 h_2 w) + \mathcal{O}(n h_1 h_2 w)$ if the results for $Q_i G_2$ are stored and reused. Subsequent application of the `compact` operation has according to Prop. 3.1.7 complexity $\mathcal{O}(\overline{h}(\overline{n} + \overline{p}\log(\overline{h})))$, where $\overline{n} = w$, $\overline{p} = p$, and $\overline{h} = h_1 h_2 + h_1 + h_2$ denote the dimension, the number of dependent factors, and the number of dependent generators of the resulting SPZ. The resulting overall complexity is therefore

$$\begin{aligned} &\mathcal{O}(p^2) + \mathcal{O}(n^2 w) + \mathcal{O}(n^2 w(h_1 + h_2)) + \mathcal{O}(h_1 h_2 p) + \mathcal{O}(n^2 h_2 w) \\ &\qquad + \mathcal{O}(n h_1 h_2 w) + \mathcal{O}(\overline{h}(n + \overline{p}\log(\overline{h}))) \\ &\overset{\substack{\overline{h}=h_1h_2+h_1+h_2 \\ \overline{n}=w,\ \overline{p}=p}}{=} \mathcal{O}(p^2) + \mathcal{O}(n^2 w(h_1 + h_2)) + \mathcal{O}(h_1 h_2 p) + \mathcal{O}(n h_1 h_2 w) \\ &\qquad + \mathcal{O}((h_1 h_2 + h_1 + h_2)(w + p\log(h_1 h_2 + h_1 + h_2))) \end{aligned} \tag{3.24}$$

$$= \mathcal{O}(p^2) + \mathcal{O}(n^2 w(h_1 + h_2)) + \mathcal{O}(n h_1 h_2 w) + \mathcal{O}(h_1 h_2 (w + p\log(h_1 h_2))),$$

which is $\mathcal{O}(n^3(w + \log(n))$ using Assumption 3.1.3. □

We now extend Prop. 3.1.30 to the general case including independent generators. As for the linear combination and the convex hull we compute a tight over-approximation instead of the exact result for computational reasons. Again we apply `mergeID` as defined in Prop. 3.1.5 prior to the computation to bring the exponent matrices to a common representation.

Proposition 3.1.31. *(Quadratic Map) Given two SPZs* $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id\rangle_{PZ} \subset \mathbb{R}^n$ *and* $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id\rangle_{PZ} \subset \mathbb{R}^n$ *with a common identifier vector* id *and a discrete set of matrices* $\mathcal{Q} = \{Q_1, \dots, Q_w\}$ *with* $Q_i \in \mathbb{R}^{n\times n}$, $i = 1, \dots, w$, *it holds that*

$$sq(\mathcal{Q}, \mathcal{PZ}_1, \mathcal{PZ}_2) \subseteq \left\langle c_z, \widehat{G}_{(\cdot,\mathcal{H})}, G_z, \widehat{E}_{(\{1,\dots,p_1\},\mathcal{H})}, id\right\rangle_{PZ},$$

where

$$\begin{aligned}
&\left\langle \widehat{c}, \widehat{G}, [\,], \widehat{E}, \overline{id}\right\rangle_{PZ} = sq\big(\mathcal{Q}, \overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big), \quad \overline{\mathcal{PZ}}_1 = \langle c_1, \overline{G}_1, [\,], \overline{E}_1, \overline{id}\rangle_{PZ},\\
&\overline{\mathcal{PZ}}_2 = \langle c_2, \overline{G}_2, [\,], \overline{E}_2, \overline{id}\rangle_{PZ}, \quad \langle c_z, G_z\rangle_Z = \texttt{zonotope}\Big(\left\langle \widehat{c}, \widehat{G}_{(\cdot,\mathcal{K})}, [\,], \widehat{E}_{(\cdot,\mathcal{K})}, \overline{id}\right\rangle_{PZ}\Big), \quad (3.25)\\
&\mathcal{K} = \big\{i \mid \exists j > p_1 :\ \widehat{E}_{(j,i)} \neq 0\big\}, \quad \mathcal{H} = \big\{i \mid \forall j > p_1 :\ \widehat{E}_{(j,i)} = 0\big\}.
\end{aligned}$$

The quadratic map $sq(\mathcal{Q}, \overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2)$ *is calculated using Prop. 3.1.30, the zonotope enclosure is computed using Prop. 3.1.14, and the extended generator and exponent matrices* $\overline{G}_1, \overline{G}_2, \overline{E}_1, \overline{E}_2$, *as well as the extended identifier vector* $\overline{id}$ *are defined as*

$$\begin{aligned}
&\overline{E}_1 = \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix}, \ \overline{E}_2 = \begin{bmatrix} E_2 & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix}, \ \overline{id} = \begin{bmatrix} id & \texttt{uniqueID}(q) \end{bmatrix},\\
&\overline{G}_1 = \begin{bmatrix} G_1 & G_{I,1} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n\times(h_1+q)}, \quad \overline{G}_2 = \begin{bmatrix} G_2 & \mathbf{0} & G_{I,2} \end{bmatrix} \in \mathbb{R}^{n\times(h_2+q)},
\end{aligned} \quad (3.26)$$

where $q = q_1 + q_2$. *The resulting SPZ is regular, and the computational complexity with respect to the dimension* n *is* $\mathcal{O}(n^3(w + \log(n)))$, *where* w *is the dimension of the resulting SPZ.*

Proof. According to Prop. 3.1.4 the extended generator and exponent matrices $\overline{G}_1$,$\overline{G}_2$,$\overline{E}_1$,$\overline{E}_2$ as well as the extended identifier vector $\overline{id}$ as defined in (3.26) equivalently represent $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$ as SPZs $\overline{\mathcal{PZ}}_1$ and $\overline{\mathcal{PZ}}_2$ without independent generators, which enables the computation of the quadratic map according to Prop. 3.1.30. For computational reasons, the resulting matrices $\widehat{G}$ and $\widehat{E}$ from Prop. 3.1.30 are divided into a dependent part that contains the dependent factors $\alpha_1, \dots, \alpha_p$ only ($i \in \mathcal{H}$), and a remainder part that contains all remaining monomials ($i \in \mathcal{K}$):

$$sq\big(\mathcal{Q}, \overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2\big) = \left\{ \widehat{c} + \underbrace{\sum_{i\in\mathcal{H}} \left(\prod_{k=1}^{p} \alpha_k^{\widehat{E}_{(k,i)}} \right) \widehat{G}_{(\cdot,i)}}_{\text{dependent part}} + \underbrace{\sum_{i\in\mathcal{K}} \left(\prod_{k=1}^{p+q} \alpha_k^{\widehat{E}_{(k,i)}} \right) \widehat{G}_{(\cdot,i)}}_{\text{remainder part}} \;\middle|\; \alpha_k \in [-1, 1] \right\},$$

where $p = p_1 = p_2$ holds since both SPZs have an identical identifier vector. Since the part containing the remaining monomials is enclosed by a zonotope, it holds that the resulting SPZ encloses the quadratic map.

Complexity: The computational complexity of `mergeID` is $\mathcal{O}(\widehat{p}_1\widehat{p}_2)$ according to Prop. 3.1.5, where $\widehat{p}_1$ and $\widehat{p}_2$ represent the number of dependent factors before applying `mergeID`. Since $p = p_1 = p_2 \leq \widehat{p}_1 + \widehat{p}_2$, this is identical to $\mathcal{O}(p^2)$. According to (3.24), calculation of the quadratic map $sq(\mathcal{Q}, \overline{\mathcal{PZ}}_1, \overline{\mathcal{PZ}}_2)$ using Prop. 3.1.30 has complexity

$$\mathcal{O}(\overline{p}^2) + \mathcal{O}(n^2 w(\overline{h}_1 + \overline{h}_2)) + \mathcal{O}(n\overline{h}_1\overline{h}_2 w) + \mathcal{O}(\overline{h}_1\overline{h}_2(w + \overline{p}\log(\overline{h}_1\overline{h}_2))),$$

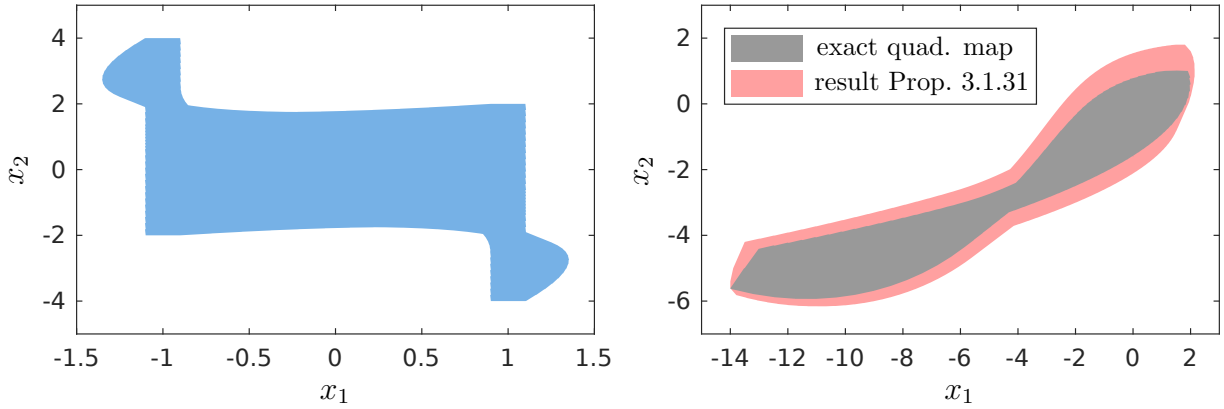

Figure 3.5: Visualization of the results from Example 3.1.32, where the SPZ $\mathcal{PZ}$ is depicted on the left and its quadratic map on the right.

where $\overline{p} = p+q_1+q_2$, $\overline{h}_1 = h_1+q_1+q_2$, and $\overline{h}_2 = h_2+q_1+q_2$ denote the number of factors and the number of dependent generators of the SPZs $\overline{\mathcal{PZ}}_1$ and $\overline{\mathcal{PZ}}_2$. Moreover, construction of the sets $\mathcal{K}$ and $\mathcal{H}$ in (3.25) has worst-case complexity $\mathcal{O}(\widehat{h}\widehat{p})$, where $\widehat{h} = \overline{h}_1\overline{h}_2 + \overline{h}_1 + \overline{h}_2$ and $\widehat{p} = \overline{p}$ denote the number of columns and rows of the exponent matrix $\widehat{E}$, respectively. Finally, the zonotope enclosure in (3.25) computed with Prop. 3.1.14 has in the worst case complexity $\mathcal{O}(\widehat{h}(\widehat{p}+\widehat{n}))$ according to (3.5), where $\widehat{n} = w$ denotes the dimension of the resulting SPZ. The overall complexity is therefore

$$
\begin{aligned}
&\mathcal{O}(p^2) + \mathcal{O}(\overline{p}^2) + \mathcal{O}(n^2 w(\overline{h}_1 + \overline{h}_2)) + \mathcal{O}(n\overline{h}_1\overline{h}_2 w) \\
&\qquad + \mathcal{O}(\overline{h}_1\overline{h}_2(w + \overline{p}\log(\overline{h}_1\overline{h}_2))) + \mathcal{O}(\widehat{h}\widehat{p}) + \mathcal{O}(\widehat{h}(\widehat{p}+\widehat{n})) \\
&\overset{\substack{\widehat{h}=\overline{h}_1\overline{h}_2+\overline{h}_1+\overline{h}_2 \\ \widehat{p}=\overline{p},\ \widehat{n}=w}}{=} \mathcal{O}(p^2) + \mathcal{O}(\overline{p}^2) + \mathcal{O}(n^2 w(\overline{h}_1 + \overline{h}_2)) + \mathcal{O}(n\overline{h}_1\overline{h}_2 w) \\
&\qquad + \mathcal{O}(\overline{h}_1\overline{h}_2(w + \overline{p}\log(\overline{h}_1\overline{h}_2))) + \mathcal{O}((\overline{h}_1\overline{h}_2 + \overline{h}_1 + \overline{h}_2)(\overline{p}+w)) \\
&\overset{\substack{\overline{p}=p+q_1+q_2 \\ \overline{h}_1=h_1+q_1+q_2 \\ \overline{h}_2=h_2+q_1+q_2}}{=} \mathcal{O}((p+q_1+q_2)^2) + \mathcal{O}(n^2 w(h_1+h_2+2q_1+2q_2)) + \\
&\mathcal{O}(n(h_1+q_1+q_2)(h_2+q_1+q_2)w) + \\
&\mathcal{O}((h_1+q_1+q_2)(h_2+q_1+q_2)(w+(p+q_1+q_2)\log((h_1+q_1+q_2)(h_2+q_1+q_2)))),
\end{aligned}
$$

which is $\mathcal{O}(n^3(w+\log(n))$ using Assumption 3.1.3. □

We use the shorthand $sq(\mathcal{Q}, \mathcal{PZ}) = sq(\mathcal{Q}, \mathcal{PZ}, \mathcal{PZ})$ to denote the quadratic map of a single polynomial zonotope. As we show in Appendix A, all higher-order polynomial maps can be reformulated as a sequence of quadratic maps, so that we can compute maps on SPZs of arbitrary order based on Prop. 3.1.31. Let us demonstrate the tightness of the quadratic map enclosure according to Prop. 3.1.31 with an example:

Example 3.1.32. *We consider the SPZ*

$$
\mathcal{PZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ -1 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}
$$

Table 3.4: Growth of the number of dependent factors p, the number of dependent generators h, and the number of independent generators q for basic set operations on SPZs, where we use the shorthand $a = q_1q_2 + q_1(h_2 + 1) + q_2(h_1 + 1)$.

Set Operation	**Dependent Factors**	**Dependent Generators**	**Independent Generators**
Linear map	p	h	q
Minkowski sum	$p_1 + p_2$	$h_1 + h_2$	$q_1 + q_2$
Exact addition	$p = p_1 = p_2$	$h_1 + h_2$	$q_1 + q_2$
Cartesian product	$p_1 + p_2$	$h_1 + h_2$	$q_1 + q_2$
Linear combination	$p_1 + p_2 + 1$	$2h_1 + 2h_2 + 1$	$q_1 + q_2$
Convex hull	$2p_1 + 2p_2 + 3$	$8h_1 + 8h_2 + 1$	$q_1 + q_2$
Quadratic map	$p = p_1 = p_2$	$h_1h_2 + h_1 + h_2$	a

and the matrices discrete set of matrices $\mathcal{Q} = \{Q_1, Q_2\}$ *with*

$$Q_1 = \begin{bmatrix} 0.5 & 0.5 \\ 1 & -0.5 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}.$$

Computing an enclosure of the quadratic map using Prop. 3.1.31 yields

$$sq(\mathcal{Q}, \mathcal{PZ}) \subseteq \left\langle \begin{bmatrix} 0.0025 \\ -0.005 \end{bmatrix}, \begin{bmatrix} -4.5 & 5.5 & -1.5 & -1.5 & 2 & 1.5 \\ -3 & 5 & -2 & 3 & -2 & 0 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 0.0025 & -0.05 & 0.15 & 0.15 & 0 & 0.05 & 0.1 \\ -0.005 & -0.2 & 0.3 & 0 & -0.1 & 0.1 & -0.1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 \\ 2 & 1 & 0 & 2 & 1 & 2 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}.$$

A comparison of the exact quadratic map and the enclosure computed with Prop. 3.1.31 is shown in Fig. 3.5.

SPZs are not closed under intersection and union. However, we can compute a tight enclosure of the intersection and union of two SPZs based on constrained polynomial zonotopes as introduced later in Sec. 3.2. For this, we first convert the SPZs to constrained polynomial zonotopes using Prop. 3.2.9, calculate the intersection or union for constrained polynomial zonotopes according to Prop. 3.2.23 or Prop. 3.2.25, and then enclose the resulting constrained polynomial zonotope with a SPZ using Prop. 3.2.12.

We conclude this section with an analysis of the growth of the representation size of SPZs. As shown in Tab. 3.4, many basic set operations on SPZs significantly increase the number of factors and generators, and therefore the representation size. For computational reasons, it is therefore crutial to limit this growth when computing with SPZs. This can be realized by repeatedly reducing the order of the SPZ using the operation `reduce` which we present later in Prop. 3.1.39.

3.1.6 Intersection and Containment Checks

In this section we provide necessary and sufficient conditions for intersection and set containment of two SPZs. Moreover, we discuss mixed intersection and containment checks involving a SPZ and another set representation. For our derivations we require the following lemma:

Lemma 3.1.33. *Given two SPZs,* $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ *and* $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^n$*, their intersection is*

$$\mathcal{PZ}_1 \cap \mathcal{PZ}_2 = \left\{ c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_2} \beta_j G_{I,1(\cdot,j)} \;\middle|\; f(y) = \mathbf{0},\ y \in [-\mathbf{1}, \mathbf{1}] \right\},$$

where

$$\begin{aligned} f(y) = c_1 - c_2 &+ \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} y_{(k)}^{E_{1(k,i)}} \right) G_{1(\cdot,i)} - \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} y_{(p_1+k)}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} \\ &+ \sum_{j=1}^{q_1} y_{(p_1+p_2+j)} G_{I,1(\cdot,j)} - \sum_{j=1}^{q_2} y_{(p_1+p_2+q_1+j)} G_{I,2(\cdot,j)} \end{aligned} \tag{3.27}$$

and $y = [\alpha_1 \ \dots \ \alpha_{p_1+p_2} \ \beta_1 \ \dots \ \beta_{q_1+q_2}]^T$.

Proof. The intersection of two SPZs can be computed by restricting the factors α_k, β_j of $\mathcal{PZ}_1$ to values that belong to points that are located inside $\mathcal{PZ}_2$, which is identical to adding the equality constraint

$$\underbrace{c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_1} \beta_j G_{I,1(\cdot,j)}}_{x \in \mathcal{PZ}_1} = \underbrace{c_2 + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} + \sum_{j=1}^{q_2} \beta_{q_1+j} G_{I,2(\cdot,j)}}_{x \in \mathcal{PZ}_2}$$

to $\mathcal{PZ}_1$:

$$\begin{aligned} \mathcal{PZ}_1 \cap \mathcal{PZ}_2 = \Bigg\{ & c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_2} \beta_j G_{I,1(\cdot,j)} \;\Bigg|\; \alpha_k, \alpha_{p_1+k}, \beta_j, \beta_{q_1+j} \in [-1, 1], \\ & \underbrace{c_1 - c_2 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} - \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} + \sum_{j=1}^{q_1} \beta_j G_{I,1(\cdot,j)} - \sum_{j=1}^{q_2} \beta_{q_1+j} G_{I,2(\cdot,j)}}_{f(y)} = \mathbf{0} \Bigg\}, \end{aligned}$$

where $y = [\alpha_1 \ \dots \ \alpha_{p_1+p_2} \ \beta_1 \ \dots \ \beta_{q_1+q_2}]^T$. □

Based on Lemma 3.1.33, we formulate the following criterion for set containment:

Proposition 3.1.34. *(Containment Check) Given two SPZs, $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^n$, it holds that*

$$\big(\mathcal{I} \subset [-\mathbf{1}, \mathbf{1}]\big) \Rightarrow \big(\mathcal{PZ}_1 \nsubseteq \mathcal{PZ}_2\big),$$

where the interval $\mathcal{I}$ is

$$\mathcal{I} = [l_{(1)}, u_{(1)}] \times \ldots \times [l_{(p_1)}, u_{(p_1)}] \times [l_{(p_1+p_2+1)}, u_{(p_1+p_2+1)}] \times \ldots \times [l_{(p_1+p_2+q_1)}, u_{(p_1+p_2+q_1)}],$$

the lower bound $l \in \mathbb{R}^{p_1+p_2+q_1+q_2}$ and the upper bound $u \in \mathbb{R}^{p_1+p_2+q_1+q_2}$ are calculated by contraction

$$[l, u] = \texttt{contract}\big(f(y), [-\mathbf{1}, \mathbf{1}]\big),$$

and the function $f(y)$ is defined as in (3.27).

Proof. If $\mathcal{PZ}_1 \subseteq \mathcal{PZ}_2$, it obviously holds that $\mathcal{PZ}_1 \cap \mathcal{PZ}_2 = \mathcal{PZ}_1$. Using the equation for the intersection of two SPZs from Lemma 3.1.33, this condition is identical to

$$\begin{aligned}\mathcal{PZ}_1 \cap \mathcal{PZ}_2 &= \left\{ c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_2} \beta_j G_{I,1(\cdot,j)} \;\middle|\; y \in [-\mathbf{1}, \mathbf{1}],\ f(y) = \mathbf{0} \right\} \\ &= \left\{ c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_2} \beta_j G_{I,1(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\} = \mathcal{PZ}_1,\end{aligned}$$

where $y = [\alpha_1 \ \ldots \ \alpha_{p_1+p_2} \ \beta_1 \ \ldots \ \beta_{q_1+q_2}]^T$. Note that $\mathcal{PZ}_1 \cap \mathcal{PZ}_2$ is identical to $\mathcal{PZ}_1$, but values for the factors α_k, β_j of $\mathcal{PZ}$ are restricted by the constraint $f(y) = \mathbf{0}$. Consequently, if it is possible to contract any of the intervals $\alpha_k \in [-1, 1]$, $k = 1, \ldots, p_1$ or $\beta_j \in [-1, 1]$, $j = 1, \ldots, q_1$ with regard to the constraint $f(y) = \mathbf{0}$, which is identical to the condition $\mathcal{I} \subset [-\mathbf{1}, \mathbf{1}]$, then there exists a point $x \in \mathcal{PZ}_1$ that is not contained in the intersection $\mathcal{PZ}_1 \cap \mathcal{PZ}_2$, so that $\mathcal{PZ}_1 \nsubseteq \mathcal{PZ}_2$. □

The criterion presented in Prop. 3.1.34 can be used to show that a SPZ is not contained in another SPZ. To derive a criterion for verifying that a SPZ is contained in another SPZ, we first recapitulate the following result from [78], which provides a sufficient condition for an interval to be contained in the image of a nonlinear function:

Lemma 3.1.35. *([78, Corollary 3.1]) Given a function $f : \mathbb{R}^m \to \mathbb{R}^n$ with $m \geq n$, an interval $\mathcal{I} \subset \mathbb{R}^n$, and an interval $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \subset \mathbb{R}^n \times \mathbb{R}^{m-n}$ satisfying*

$$\forall i \in \{1, \ldots, n\} : \ 0 \notin \boldsymbol{\mathcal{J}}_{(i,i)} \ \text{ and } \ \forall J_1 \in \boldsymbol{\mathcal{J}}_1 : \ \texttt{det}(J_1) \neq 0,$$

where the matrix set $\boldsymbol{\mathcal{J}} = [\boldsymbol{\mathcal{J}}_1 \ \boldsymbol{\mathcal{J}}_2] \supseteq \{\nabla f(y) \mid y \in \mathcal{Y}\}$ encloses the Jacobian matrix of the function $f(y)$ on $\mathcal{Y}$, it holds that

$$\Big(\texttt{func}\big(\boldsymbol{\mathcal{J}}, \widetilde{y}, \mathcal{Y}, \mathcal{I}, f(y)\big)\big) \subseteq \mathcal{Y}_1 \setminus \partial\mathcal{Y}_1\Big) \Rightarrow \Big(\mathcal{I} \subseteq \big\{f(y) \mid y \in \mathcal{Y}\big\}\Big),$$

where

$$\texttt{func}\big(\boldsymbol{\mathcal{J}}, \widetilde{y}, \mathcal{Y}, \mathcal{I}, f(y)\big) = \widetilde{y}_1 \oplus \texttt{diag}(\boldsymbol{\mathcal{J}}_1)^{-1} \otimes \Big(\mathcal{I} \oplus \big(-f(\widetilde{y})\big) \oplus \big(-\texttt{offdiag}(\boldsymbol{\mathcal{J}}_1)\big) \otimes \big(\mathcal{Y}_1 \oplus (-\widetilde{y}_1)\big) \oplus (-\boldsymbol{\mathcal{J}}_2) \otimes \big(\mathcal{Y}_2 \oplus (-\widetilde{y}_2)\big)\Big),$$

and $\widetilde{y} = [\widetilde{y}_1^T \ \widetilde{y}_2^T]^T \in \mathcal{Y}_1 \times \mathcal{Y}_2$ *is a point in* $\mathcal{Y}$ *that satisfies* $f(\widetilde{y}) \in \mathcal{I}$.

Proof. A detailed proof is provided in [78]. □

Lemma 3.1.35 is a generalization of the result in [79], where the special case $m = n$ is considered. Based on Lemma 3.1.35, we can formulate the following criterion for interval in SPZ containment:

Proposition 3.1.36. *(Containment Check Interval) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$ *with degree-of-freedom order* $\rho_f = \frac{p+q}{n} \geq 1$ *and an interval* $\mathcal{I} \subset \mathbb{R}^n$*, it holds that*

$$\big(\texttt{func}(\boldsymbol{\mathcal{A}}, \widetilde{y}, \mathcal{Y}, \mathcal{I}, f(y)) \subseteq \mathcal{Y}_1 \setminus \partial\mathcal{Y}_1\big) \Rightarrow \big(\mathcal{I} \subseteq \mathcal{PZ}\big), \tag{3.28}$$

where

$$f(y) = c + \sum_{i=1}^{h} \Big(\prod_{k=1}^{p} y_{(k)}^{E_{(k,i)}} \Big) G_{(\cdot,i)} + \sum_{j=1}^{q} y_{(p+j)} G_{I(\cdot,j)},$$

$$\boldsymbol{\mathcal{A}} = [\boldsymbol{\mathcal{A}}_1 \ \boldsymbol{\mathcal{A}}_2] = \texttt{bound}(\nabla f(y), \mathcal{Y}), \quad \widetilde{y} = \texttt{center}(\mathcal{Y}),$$

the operator `func` *is defined as in Lemma 3.1.35, and* $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_1 \subset \mathbb{R}^n \times \mathbb{R}^{p+q-n}$ *is a suitable interval that guarantees that the four conditions*

$$\begin{aligned} &\text{(a)} \quad \forall i \in \{1, \ldots, n\} : \ 0 \notin \boldsymbol{\mathcal{A}}_{(i,i)} \\ &\text{(b)} \quad \forall A_1 \in \boldsymbol{\mathcal{A}}_1 : \ \texttt{det}(A_1) \neq 0 \\ &\text{(c)} \quad f(\widetilde{y}) \in \mathcal{I} \\ &\text{(d)} \quad \mathcal{Y} \subseteq [-\mathbf{1}, \mathbf{1}] \end{aligned} \tag{3.29}$$

are satisfied.

Proof. With the definition $y = [\alpha_1 \ \ldots \ \alpha_p \ \beta_1 \ \ldots \ \beta_q]^T$ the polynomial zonotope $\mathcal{PZ}$ can be equivalently represented as the image of the nonlinear function $f : \mathbb{R}^{p+q} \to \mathbb{R}^n$

$$\mathcal{PZ} = \left\{ c + \sum_{i=1}^{h} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\} = \big\{ f(y) \mid y \in [-\mathbf{1}, \mathbf{1}] \big\},$$

which enables us to use Lemma 3.1.35. Moreover, since $\widetilde{y} = \texttt{center}(\mathcal{Y}) \in \mathcal{Y}$ and $\boldsymbol{\mathcal{A}} = \texttt{bound}(\nabla f(y), \mathcal{Y}) \supseteq \{\nabla f(y) \mid y \in \mathcal{Y}\}$, it holds that all conditions required to apply Lemma 3.1.35 are satisfied if $\mathcal{Y}$ satisfies conditions (a), (b), and (c) in (3.29). □

The main challenge when applying the criterion in Prop. 3.1.36 is to find a suitable interval $\mathcal{Y}$. A strategy that we observed to work well in practice is to first determine a suitable point $\widetilde{y} = \texttt{center}(\mathcal{Y})$ by solving the following polynomial optimization problem:

$$\min_{y \in [-\mathbf{1}, \mathbf{1}]} ||y||_2 \text{ s.t. } f(y) = \texttt{center}(\mathcal{I}), \tag{3.30}$$

where we chose to minimize the length of x since this later helps to satisfy the constraint $\mathcal{Y} \subseteq [-\mathbf{1}, \mathbf{1}]$. In a second step, we inflate the interval $\mathcal{Y}$ around $\widetilde{y}$ using the strategy described in [79, Sec. 5.2] and [78, Alg. 3], which explicitly aims at finding an $\mathcal{Y}$ that satisfies the condition in (3.28). Another issue with the criterion presented in Prop. 3.1.36 is that the conditions (a) and (b), which require that each sub-matrix $\mathcal{A}_1$ contained in the matrix set $\boldsymbol{\mathcal{A}}$ enclosing the Jacobian has full rank, could be violated. Since the number of factors $p+q$ of a SPZs is often larger than the dimension n, we can try to satisfy conditions (a) and (b) by changing the order of the factors if the conditions are violated for the original order. Different strategies for finding a suitable order to obtain a sub-matrix with full rank are presented in [78, Sec. 4.3]. Finally, since the criterion in Prop. 3.1.36 is based on a linearization of the nonlinear function $f(y)$, it is in general only satisfiable for small intervals $\mathcal{I}$. To use Prop. 3.1.36 for checking if a SPZ $\mathcal{PZ}_1 \subset \mathbb{R}^n$ is contained in a SPZ $\mathcal{PZ}_2 \subset \mathbb{R}^n$, we therefore propose the following strategy: We first recursively split $\mathcal{PZ}_1$ into smaller sets using the `split` operation that we present later in Prop. 3.1.44. Next, we enclose all split SPZs with an interval as described in Sec. 3.1.4, and then apply Prop. 3.1.36 to prove that all intervals are contained in $\mathcal{PZ}_2$, which ensures that $\mathcal{PZ}_1 \subseteq \mathcal{PZ}_2$. Since according to Sec. 3.1.3 all intervals, zonotopes, bounded polytopes, constrained zonotopes, and Taylor models can be represented as SPZs, we can apply the same strategy to check if any of these set representations is contained in a SPZ. However, due to the recursive splitting, the computational complexity is exponential with respect to the system dimension n. Let us demonstrate containment checks for SPZs by an example:

Example 3.1.37. *We consider the SPZs*

$$\mathcal{PZ}_1 = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -0.5 & -1 & 0.1 \\ -1 & 0 & -0.1 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, [1\ 2\ 3] \right\rangle_{PZ}$$

and

$$\mathcal{PZ}_2 = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 & 1 \\ -2 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}.$$

To show that $\mathcal{PZ}_1 \subseteq \mathcal{PZ}_2$ holds, we recursively split $\mathcal{PZ}_1$ using Prop. 3.1.44 until the interval enclosures of the split SPZs are small enough to successfully prove containment using Prop. 3.1.36. The resulting enclosure of $\mathcal{PZ}_1$ with a union of intervals is visualized in Fig. 3.6.

To check if a single point $x \in \mathbb{R}^n$ is contained in a SPZ, one could argue that it is sufficient to show that the optimization problem in (3.30) has a feasible solution. However, nonlinear optimization solvers only guarantee satisfaction of the constraints up to a certain precision in practice, and the computations are subject to rounding errors. To soundly prove that a point is contained in a SPZ, we therefore have to apply the criterion in Prop. 3.1.36.

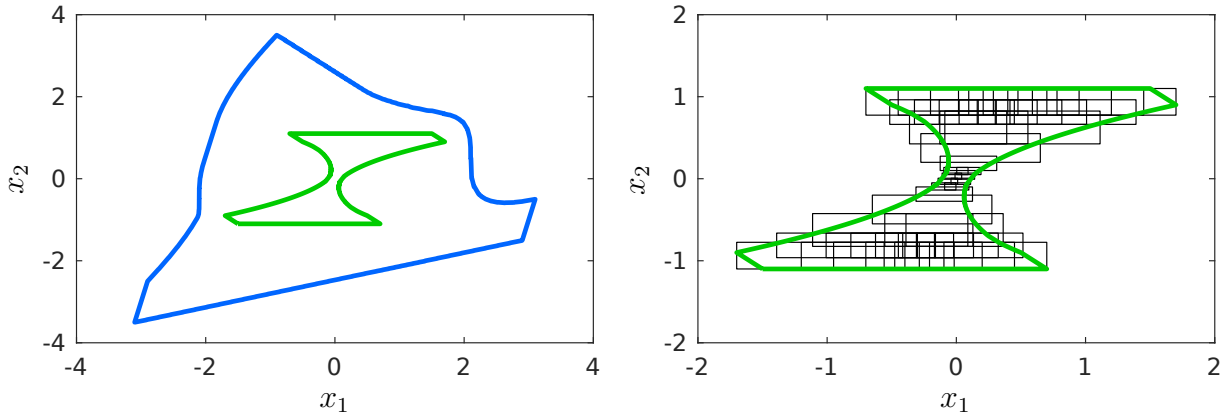

Figure 3.6: Visualization of the SPZs $\mathcal{PZ}_1$ (green) and $\mathcal{PZ}_2$ (blue) from Example 3.1.37. The resulting enclosure of $\mathcal{PZ}_1$ with a union of intervals required to prove the containment $\mathcal{PZ}_1 \subseteq \mathcal{PZ}_2$ using Prop. 3.1.36 is shown on the right side.

While it is hard to show that a set is contained in a SPZ, it is significantly easier to show that a SPZ is contained in another set representation: To check if a SPZ $\mathcal{PZ} \subset \mathbb{R}^n$ is contained in an ellipsoid $\mathcal{E} = \langle c, Q \rangle_E \subset \mathbb{R}^n$, the following criterion can be used:

$$\big(\texttt{interval}\big(sq(\{Q^{-1}\}, -c \oplus \mathcal{PZ})\big) \subseteq [0,1]\big) \Rightarrow \big(\mathcal{PZ} \subseteq \mathcal{E}\big),$$

which follows directly from the definition of an ellipsoid in Def. 2.2.6 and the definition of the quadratic map in (2.6). A similar criterion exists for checking if a SPZ $\mathcal{PZ} \subset \mathbb{R}^n$ is contained in a polytope in H-representation $\mathcal{P} = \langle A, b \rangle_H \subset \mathbb{R}^n$:

$$\big(\forall i : \; s_{\mathcal{PZ}}(A_{(i,\cdot)}) \leq b_{(i)}\big) \Rightarrow \big(\mathcal{PZ} \subseteq \mathcal{P}\big), \tag{3.31}$$

where Prop. 3.1.16 can be used to calculate an enclosure of the support function $s_{\mathcal{PZ}}(A_{(i,\cdot)})$. The criterion in (3.31) follows directly from the definition of the polytope H-representation in Def. 2.2.2. Since all intervals, zonotopes, and constrained zonotopes can be equivalently represented as polytopes in H-representation, the criterion in (3.31) can also be applied to check if a SPZ is contained in an interval, zonotope, or constrained zonotope. Next, we provide a criterion for checking if two SPZs intersect:

Proposition 3.1.38. *(Intersection Check) Given two SPZs,* $\mathcal{PZ}_1 = \langle c_1, G_1, G_{I,1}, E_1, id_1 \rangle_{PZ} \subset \mathbb{R}^n$ *and* $\mathcal{PZ}_2 = \langle c_2, G_2, G_{I,2}, E_2, id_2 \rangle_{PZ} \subset \mathbb{R}^n$*, it holds that*

$$\big(\texttt{contract}(f(y), [-\mathbf{1}, \mathbf{1}]) = \emptyset\big) \Rightarrow \big(\mathcal{PZ}_1 \cap \mathcal{PZ}_2 = \emptyset\big),$$

where the function $f(y)$ *is defined as in* (3.27)*.*

Proof. According to Lemma 3.1.33, the intersection of two SPZs can be computed as

$$\mathcal{PZ}_1 \cap \mathcal{PZ}_2 = \left\{ c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_1(k,i)} \right) G_{1(\cdot,i)} + \sum_{j=1}^{q_1} \beta_j G_{I,1(\cdot,j)} \;\middle|\; f(y) = \mathbf{0},\; y \in [-\mathbf{1}, \mathbf{1}] \right\},$$

where $y = [\alpha_1 \; \dots \; \alpha_{p_1+p_2} \; \beta_1 \; \dots \; \beta_{q_1+q_2}]^T$. Consequently, if the domain $y \in [-\mathbf{1}, \mathbf{1}]$ can be contracted to the empty set under consideration of the nonlinear constraint $f(y) = \mathbf{0}$, then it is guaranteed that the intersection $\mathcal{PZ}_1 \cap \mathcal{PZ}_2$ is empty. □

Note that the criterion $\texttt{contract}(f(y), [-\mathbf{1}, \mathbf{1}]) = \emptyset$ presented in Prop. 3.1.38 is sufficient, but not necessary for the intersection to be empty. The intersection could therefore still be empty, even if $\texttt{contract}(f(y), [-\mathbf{1}, \mathbf{1}]) \neq \emptyset$. To prove that two SPZs intersect, we can solve a polynomial optimization problem similar to (3.30)

$$\min_{y \in [-\mathbf{1}, \mathbf{1}]} ||y||_2 \text{ s.t. } f(y) = \mathbf{0},$$

where $f(y)$ is defined as in (3.27), and then apply Prop. 3.1.36 to show that the corresponding point is contained in $\mathcal{PZ}_1$ and $\mathcal{PZ}_2$.

In set-based computing it is often required to perform intersection checks with sets belonging to a different set representation. Since according to Sec. 3.1.3 any interval, zonotope, bounded polytope, and Taylor model can be equivalently represented as a SPZ, we can first convert these set representations to SPZs and then apply Prop. 3.1.38 to check for intersection. For a SPZ $\mathcal{PZ} \subset \mathbb{R}^n$ and an ellipsoid $\mathcal{E} = \langle c, Q \rangle_E \subset \mathbb{R}^n$, the following criterion can be used to check for intersection

$$\big(\texttt{interval}\big(sq(\{Q^{-1}\}, -c \oplus \mathcal{PZ})\big) \cap [0, 1] = \emptyset\big) \Rightarrow \big(\mathcal{PZ} \cap \mathcal{E} = \emptyset\big),$$

which follows directly from the definition of an ellipsoid in Def. 2.2.6 and the definition of the quadratic map in (2.6). In order to speed up intersection tests, one can also enclose SPZs with simpler set representations as described in Sec. 3.1.4, and then perform the intersection tests on the simpler set representation.

3.1.7 Auxiliary Set Operations

This section derives useful auxiliary operations on SPZs. Many basic set operations on SPZs, such as Minkowski sum or convex hull, increase the number of generators and consequently also the representation size of the SPZ, as shown in Tab. 3.4. Thus, for computational reasons, it is necessary in many algorithms to repeatedly reduce the representation size when computing with SPZs. Since the order ρ of a SPZ is proportional to the representation size, this can be realized by order reduction. We propose an order reduction operation for SPZs that is based on order reduction for zonotopes:

Proposition 3.1.39. *(Order Reduction) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ and a desired order $\rho_d \geq 1$, the operation* `reduce` *returns a SPZ with an order smaller than or equal to ρ_d that encloses $\mathcal{PZ}$:*

$$\mathcal{PZ} \subseteq \texttt{reduce}(\mathcal{PZ}, \rho_d) = \left\langle c_z, G_{(\cdot,\overline{\mathcal{K}})}, \begin{bmatrix} G_{I(\cdot,\overline{\mathcal{H}})} & G_z \end{bmatrix}, \overline{E}_{(\mathcal{N},\cdot)}, id_{(\mathcal{N})} \right\rangle_{PZ}, \tag{3.32}$$

where

$$\begin{aligned} &\langle c_z, G_z \rangle_Z = \texttt{reduce}(\mathcal{Z}, 1), \quad \mathcal{Z} = \texttt{zonotope}\big(\big\langle c, G_{(\cdot,\mathcal{K})}, G_{I(\cdot,\mathcal{H})}, E_{(\cdot,\mathcal{K})}, id \big\rangle_{PZ}\big), \\ &\overline{E} = E_{(\cdot,\overline{\mathcal{K}})}, \quad \mathcal{N} = \big\{ i \mid \exists j \in \{1, \ldots, |\overline{\mathcal{K}}|\} : \ \overline{E}_{(i,j)} \neq 0 \big\}, \end{aligned} \tag{3.33}$$

and the zonotope enclosure is calculated by applying Prop. 3.1.14. For reduction, the

$$a = \max\left(0, \min\left(h + q, \lceil h + q - n(\rho_d - 1) \rceil\right)\right) \tag{3.34}$$

smallest generators are selected:

$$\begin{aligned}\mathcal{K} &= \begin{cases}\emptyset, & a = 0\\ \{i \mid ||G_{(\cdot,i)}||_2 \leq ||\overline{G}_{(\cdot,o_{(a)})}||_2\}, & otherwise\end{cases}, \quad \overline{\mathcal{K}} = \{1,\ldots,h\} \setminus \mathcal{K},\\ \mathcal{H} &= \begin{cases}\emptyset, & a = 0\\ \{i \mid ||G_{I(\cdot,i)}||_2 \leq ||\overline{G}_{(\cdot,o_{(a)})}||_2\}, & otherwise\end{cases}, \quad \overline{\mathcal{H}} = \{1,\ldots,q\} \setminus \mathcal{H},\end{aligned} \tag{3.35}$$

where

$$||\overline{G}_{(\cdot,o_{(1)})}||_2 \leq \ldots \leq ||\overline{G}_{(\cdot,o_{(h+q)})}||_2, \quad \overline{G} = [G\ G_I], \tag{3.36}$$

and the vector $o \in \mathbb{N}^{h+q}$ stores the indices of the sorted generators. The resulting SPZ is regular if $\mathcal{PZ}$ is regular. The computational complexity with respect to the dimension n is $\mathcal{O}(n^2) + \mathcal{O}(\texttt{reduce})$.

Proof. The resulting SPZ has $|\overline{\mathcal{K}}| + |\overline{\mathcal{H}}| + n$ generators since $G_z \in \mathbb{R}^{n\times n}$. Since the order ρ of a SPZ is defined as the number of generators divided by the dimension, we have

$$\rho = \frac{|\overline{\mathcal{K}}| + |\overline{\mathcal{H}}| + n}{n} \overset{(3.35)}{=} \frac{h + q - a + n}{n} \overset{(3.34)}{\leq} \rho_d$$

for $\rho_d \geq 1$, so that the order of the resulting SPZ is smaller than or equal to the desired order ρ_d. Moreover, $\texttt{reduce}(\mathcal{PZ}, \rho_d) \supseteq \mathcal{PZ}$ since the zonotope enclosure `zonotope` and order reduction for zonotopes `reduce` are both over-approximative operations, so that their composition $\texttt{reduce} \circ \texttt{zonotope}$ is over-approximative, too. In addition, the assignment $\overline{E}_{(\mathcal{N},\cdot)}$ in (3.32) removes all-zero rows in the exponent matrix, which does not change the set.

Complexity: Computing the norm $||\overline{G}_{(\cdot,i)}||_2$ in (3.36) for all $i = 1,\ldots,h+q$ generators has complexity $\mathcal{O}(n(h+q))$ and sorting the generators with respect to their norm has complexity $\mathcal{O}((h+q)\log(h+q))$ according to Tab. 2.1. In the worst case where all dependent generators get reduced, the enclosure with a zonotope in (3.33) using Prop. 3.1.14 has complexity $\mathcal{O}(h(p+n))$ according to (3.5). Moreover, the construction of the set $\mathcal{N}$ in (3.33) has complexity $\mathcal{O}(ph)$ in the worst case and we denote the complexity of zonotope order reduction by $\mathcal{O}(\texttt{reduce})$. The overall complexity is therefore

$$\begin{aligned}&\mathcal{O}(n(h+q)) + \mathcal{O}\big((h+q)\log(h+q)\big) + \mathcal{O}(h(p+n)) + \mathcal{O}(ph) + \mathcal{O}(\texttt{reduce})\\ &\qquad = \mathcal{O}\big((h+q)(n+\log(h+q))\big) + \mathcal{O}(ph) + \mathcal{O}(\texttt{reduce}),\end{aligned} \tag{3.37}$$

which is $\mathcal{O}(n^2) + \mathcal{O}(\texttt{reduce})$ using Assumption 3.1.3. □

An overview of the computational complexity for different zonotope order reduction methods is provided in Tab. 3.5. Let us demonstrate the tightness of the reduction strategy proposed in Prop. 3.1.39 by an example:

Example 3.1.40. *We consider the SPZ*

$$\mathcal{PZ} = \left\langle \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1 & -2 & -1 & 2 & 0.01 & 0.4\\ 1 & 0 & -1 & 1 & 0.2 & 0\end{bmatrix}, \begin{bmatrix}0.2 & 0.01\\ 0.02 & -0.4\end{bmatrix}, \begin{bmatrix}1 & 0 & 1 & 2 & 2 & 0\\ 0 & 1 & 1 & 0 & 0 & 2\\ 0 & 0 & 0 & 0 & 1 & 2\end{bmatrix}, [1\ 2\ 3] \right\rangle_{PZ},$$

Table 3.5: Computational complexity with respect to the dimension $n \in \mathbb{N}$ for order reduction and restructuring of SPZs for the zonotope order reduction methods from Sec. 2.6, where $k \in \mathbb{N}_0$ is the number of generators that are reduced.

Method	**Order Reduction**	**Restructuring**
Girard	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
PCA	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Scott	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
Exhaustive Search	$\mathcal{O}(\binom{k}{n}k)$	$\mathcal{O}(\binom{k}{n}k)$

which has order $\rho = 4$. Order reduction with Prop. 3.1.39 to a desired order of $\rho_d = 3$ using Girard's method (see Sec. 2.6) for order reduction of zonotopes yields

$$\texttt{reduce}(\mathcal{PZ}, 3) = \left\langle \begin{bmatrix} 0.2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 & -2 & -1 & 2 \\ 1 & 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 0.42 & 0 \\ 0 & 0.62 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}.$$

The original as well as the reduced SPZ are visualized in Fig. 3.7.

Order reduction as defined in Prop. 3.1.39 as well as other operations, such as Minkowski sum or Cartesian product with a zonotope, increase the volume spanned by the independent generators relative to the volume spanned by the dependent generators. Since computations on the dependent part are exact while computations on the independent part are often over-approximative, this has a negative effect on the accuracy when computing with SPZs. We therefore define the operation `restructure`, which introduces new dependent generators that over-approximate the independent ones. For computational reasons, we use an upper bound p_d for the number of dependent factors for the SPZ after restructuring.

Proposition 3.1.41. *(Restructure) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ and an upper bound for the number of dependent factors after restructuring $p_d \geq n$, the operation* `restructure` *returns a SPZ that encloses $\mathcal{PZ}$ and generates new dependent factors from the independent factors:*

$$\texttt{restructure}(\mathcal{PZ}, p_d) = \begin{cases} \mathcal{PZ}_1, & p \leq p_d - n \\ \mathcal{PZ}_2, & otherwise \end{cases},$$

where $\mathcal{PZ}_1$ is constructed by redefining all independent factors after order reduction as new dependent factors

$$\mathcal{PZ}_1 = \left\langle c_z, \underbrace{[G\ G_z]}_{G_1}, [\,], \underbrace{\begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_n \end{bmatrix}}_{E_1}, \underbrace{[id\ \texttt{uniqueID}(n)]}_{id_1} \right\rangle_{PZ}, \ \langle c_z, G_z \rangle_Z = \texttt{reduce}\big(\langle c, G_I \rangle_Z, 1\big),$$

and $\mathcal{PZ}_2$ by reducing the number of dependent factors of $\mathcal{PZ}_1$ to satisfy the upper bound p_d:

$$\mathcal{PZ}_2 = \big\langle \overline{c}_z, G_{1(\cdot,\overline{\mathcal{H}})}, \overline{G}_z, E_{1(\overline{\mathcal{N}},\overline{\mathcal{H}})}, id_{1(\overline{\mathcal{N}})} \big\rangle_{PZ},$$

with

$$\mathcal{N} = \{i \mid d_i \leq d_{o_{(p_d)}}\}, \quad \overline{\mathcal{N}} = \{1, \ldots, p+n\} \setminus \mathcal{N}, \quad \mathcal{H} = \bigcup_{i \in \mathcal{N}} \mathcal{K}_i, \quad \overline{H} = \{1, \ldots, h+n\} \setminus \mathcal{H},$$

$$\mathcal{K}_i = \{j \mid E_{1(i,j)} \neq 0\}, \quad d_i = \sum_{j \in \mathcal{K}_i} ||G_{1(\cdot,j)}||_2, \qquad i = 1, \ldots, p+n,$$

$$\langle \overline{c}_z, \overline{G}_z \rangle_Z = \texttt{zonotope}\big(\langle c_z, G_{1(\cdot,\mathcal{H})}, [\,], E_{1(\cdot,\mathcal{H})}, id_1 \rangle_{PZ}\big), \qquad d_{o_{(1)}} \geq \ldots \geq d_{o_{(p+n)}},$$

where the vector $o \in \mathbb{N}^{p+n}$ stores the indices of the sorted scalars d_i and the zonotope enclosure is computed using Prop. 3.1.14. The resulting SPZ is regular and the computational complexity with respect to the dimension n is $\mathcal{O}(n^2) + \mathcal{O}(\texttt{reduce})$.

Proof. For the case $p \leq p_d - n$ we have $\texttt{restructure}(\mathcal{PZ}, p_d) \supseteq \mathcal{PZ}$ since `reduce` is over-approximative and according to Prop. 3.1.4 the redefinition of independent generators as new dependent generators just changes the set representation, but not the set itself. For the case $p > p_d - n$ the generators $G_{1(\cdot,i)}$ of $\mathcal{PZ}_1$ are split into generators that belong to dependent factors that are removed ($i \in \mathcal{H}$), and the remaining generators ($i \in \overline{\mathcal{H}}$). The generators that belong to dependent factors that are removed are enclosed by a zonotope. Since the enclosure of a SPZ with a zonotope using Prop. 3.1.14 is over-approximative, it therefore holds that $\mathcal{PZ}_2 \supseteq \mathcal{PZ}_1 \supseteq \mathcal{PZ}$.

Complexity: We first consider the construction of $\mathcal{PZ}_1$: The generation of n new unique identifiers using `uniqueID` has complexity $\mathcal{O}(n)$ according to Tab. 2.2, so that the overall complexity for constructing $\mathcal{PZ}_1$ is $\mathcal{O}(n) + \mathcal{O}(\texttt{reduce})$, where $\mathcal{O}(\texttt{reduce})$ is the complexity of zonotope order reduction. Next, we consider the construction of $\mathcal{PZ}_2$: Computation of the sets $\mathcal{N}, \overline{\mathcal{N}}, \mathcal{H}, \overline{\mathcal{H}}, \mathcal{K}_i$ and the scalars d_i has in the worst case complexity $\mathcal{O}((h+n)(p+n))$. The SPZ that is enclosed by a zonotope has $\overline{h} = h+n$ generators and $\overline{p} = p+n$ dependent factors in the worst case where all dependent factors are reduced. Since the enclosure by a zonotope using Prop. 3.1.14 has complexity $\mathcal{O}(\overline{h}(\overline{p}+n))$ according to (3.5), we therefore obtain a worst-case complexity of $\mathcal{O}((h+n)(p+2n)) = \mathcal{O}((h+n)(p+n))$ for the zonotope enclosure. Finally, sorting the values d_i has complexity $\mathcal{O}((p+n)\log(p+n))$ according to Tab. 2.1. The overall complexity of `restructure` is therefore

$$\underbrace{\mathcal{O}(n) + \mathcal{O}(\texttt{reduce})}_{\mathcal{PZ}_1} + \underbrace{\mathcal{O}((h+n)(p+n)) + \mathcal{O}((h+n)(p+n)) + \mathcal{O}((p+n)\log(p+n))}_{\mathcal{PZ}_2},$$

which is $\mathcal{O}(n^2) + \mathcal{O}(\texttt{reduce})$ using Assumption 3.1.3. □

An overview of the computational complexity for different zonotope order reduction methods is provided in Tab. 3.5. A good heuristic for triggering the restructure process is to check when the volume spanned by the independent generators exceeds a certain threshold, as we demonstrate later in Sec. 4.1. Prop. 3.1.41 is based on a heuristic that we observed to work well in practice. However, there also exist other possible strategies for restructuring that might perform better in some cases. Instead of redefining all n reduced independent factors as new dependent factors, one could for example only define the independent factor corresponding to the longest generator as a new dependent factor. Which strategy for restructuring performs best usually depends on the application. Even though every execution of the `restructure` operation introduces an over-approximation,

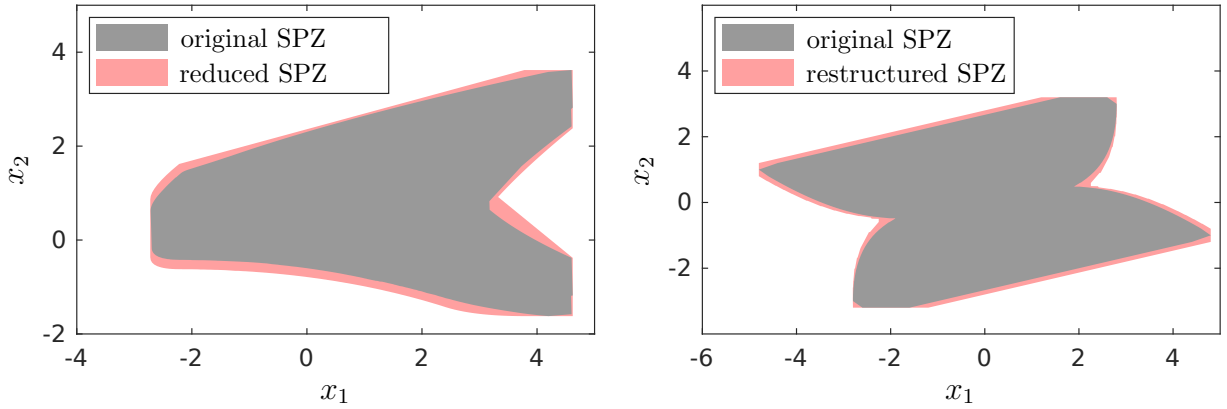

Figure 3.7: Visualization of the original and the reduced SPZ from Example 3.1.40 (left), as well as the original and the restructured SPZ from Example 3.1.42 (right).

using restructuring often leads to tighter results since the over-approximation introduced by restructuring is compensated by the more accurate computations on the dependent part of the SPZ. Let us demonstrate restructuring with the method proposed in Prop. 3.1.41 by an example:

Example 3.1.42. *We consider the SPZ*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 1 \\ 1 & -2 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.2 & 0.1 \\ 0 & 0.1 & -0.1 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 1 & 0 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}.$$

Restructuring with Prop. 3.1.41 using an upper bound of $p_d = 3$ for the number of dependent factor and Girard's method (see Sec. 2.6) for order reduction of zonotopes yields

$$\texttt{restructure}(\mathcal{PZ}, 3) = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 & 1 & 0.8 \\ 1 & -2 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, [1\ 2\ 3] \right\rangle_{PZ}.$$

The original as well as the restructered SPZ are visualized in Fig. 3.7.

For many applications, and especially for the extraction of reachable subsets presented later in Sec. 4.2, it is required to extract a subset of a SPZ by narrowing the domain for a specific dependent factor. We implement this functionality with the operation `getSubset`:

Proposition 3.1.43. *(Get Subset) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$, the index of one dependent factor $r \in \{1, \dots, p\}$, and an interval $[l, u] \subseteq [-1, 1]$, the operation* `getSubset` *substitutes the domain for the dependent factor α_r with $\alpha_r \in [l, u]$, which yields a SPZ that is a subset of $\mathcal{PZ}$:*

$$\texttt{getSubset}\big(\mathcal{PZ}, r, [l, u]\big)$$

$$= \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1],\ \alpha_r \in [l, u] \right\} \tag{3.38}$$

$$= \big\langle c, \overline{G}, G_I, \overline{E}, \texttt{uniqueID}(p) \big\rangle_{PZ} \subseteq \mathcal{PZ},$$

where the exponent matrix $\overline{E}$ *and the generator matrix* $\overline{G}$ *are defined as*

$$\begin{aligned}\overline{E} &= \begin{bmatrix} E_{(\cdot,\mathcal{H})} & \widehat{E}_{k_1} & \dots & \widehat{E}_{k_{|\mathcal{K}|}} \end{bmatrix}, \quad \overline{G} = \begin{bmatrix} G_{(\cdot,\mathcal{H})} & \widehat{G}_{k_1} & \dots & \widehat{G}_{k_{|\mathcal{K}|}} \end{bmatrix} \\ \mathcal{K} &= \{ i \mid E_{(r,i)} > 0 \} := \{k_1, \dots, k_{|\mathcal{K}|}\}, \quad \mathcal{H} = \{1, \dots, h\} \setminus \mathcal{K},\end{aligned} \tag{3.39}$$

and the exponent matrices $\widehat{E}_i$ *and generator matrices* $\widehat{G}_i$ *are defined as*

$$\begin{aligned}\widehat{E}_i &= \begin{bmatrix} E_{(\{1,\dots,r-1\},i)} & E_{(\{1,\dots,r-1\},i)} & \dots & E_{(\{1,\dots,r-1\},i)} & E_{(\{1,\dots,r-1\},i)} \\ 0 & 1 & \dots & E_{(r,i)} - 1 & E_{(r,i)} \\ E_{(\{r+1,\dots,p\},i)} & E_{(\{r+1,\dots,p\},i)} & \dots & E_{(\{r+1,\dots,p\},i)} & E_{(\{r+1,\dots,p\},i)} \end{bmatrix}, \quad i \in \mathcal{K} \\ \widehat{G}_i &= \begin{bmatrix} b_{i,0} \cdot G_{(\cdot,i)} & \dots & b_{i,E_{(r,i)}} \cdot G_{(\cdot,i)} \end{bmatrix}, \quad i \in \mathcal{K},\end{aligned} \tag{3.40}$$

with the scalars $b_{i,0}, \dots, b_{i,E_{(r,i)}}$ *obtained from the definition*

$$\left(\frac{l+u}{2} + \frac{u-l}{2} \, \widehat{\alpha}_r \right)^{E_{(r,i)}} := b_{i,0} + \ldots + b_{i,E_{(r,i)}} \, \widehat{\alpha}_r^{E_{(r,i)}}, \quad i \in \mathcal{K}. \tag{3.41}$$

The `compact` *operation as defined in Prop. 3.1.7 is applied to make the resulting SPZ regular. The computational complexity with respect to the dimension* n *is* $\mathcal{O}(n^3 \log(n))$.

Proof. To represent the set $\texttt{getSubset}(\mathcal{PZ}, r, [l, u])$ as a SPZ, we substitute $\alpha_r \in [l, u]$ with a new dependent factor $\widehat{\alpha}_r \in [-1, 1]$:

$$\{ \alpha_r \mid \alpha_r \in [l, u] \} = \left\{ \frac{l+u}{2} + \frac{u-l}{2} \, \widehat{\alpha}_r \;\middle|\; \widehat{\alpha}_r \in [-1, 1] \right\}. \tag{3.42}$$

Inserting this substitution into the definition of $\texttt{getSubset}(\mathcal{PZ}, r, [l, u])$ in (3.38) yields

$$\begin{aligned}&\texttt{getSubset}\big(\mathcal{PZ}, r, [l, u]\big) = \\ &\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1], \; \alpha_r \in [l, u] \right\} \\ &\overset{\mathcal{K} \cup \mathcal{H} = \{1,\dots,h\}}{=} \left\{ c + \sum_{i \in \mathcal{K}} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \alpha_r^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{i \in \mathcal{H}} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \right. \\ &\qquad \left. + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1], \; \alpha_r \in [l, u] \right\} \\ &\overset{(3.42)}{=} \left\{ c + \sum_{i \in \mathcal{K}} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \left(\frac{l+u}{2} + \frac{u-l}{2} \, \widehat{\alpha}_r \right)^{E_{(r,i)}} G_{(\cdot,i)} \right.\end{aligned}$$

$$+ \sum_{i \in \mathcal{H}} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \widehat{\alpha}_r, \beta_j \in [-1,1] \Bigg\}$$

$$\overset{(3.41)}{=} \Bigg\{ c + \sum_{i \in \mathcal{K}} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \left(b_{i,0} + \ldots + b_{i,E_{(r,i)}} \widehat{\alpha}_r^{E_{(r,i)}} \right) G_{(\cdot,i)}$$

$$+ \sum_{i \in \mathcal{H}} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \widehat{\alpha}_r, \beta_j \in [-1,1] \Bigg\}$$

$$\overset{(3.39),(3.40)}{=} \left\langle c, \overline{G}, G_I, \overline{E}, \texttt{uniqueID}(p) \right\rangle_{PZ},$$

which concludes the proof.

Complexity: Generation of p unique identifiers using `uniqueID` has complexity $\mathcal{O}(p)$ according to Tab. 2.1 and construction of the set $\mathcal{K}$ in (3.39) has complexity $\mathcal{O}(h)$. Let $\epsilon = \max(E_{(r,\cdot)})$ denote the largest exponent for the dependent factor α_r. Then $(0.5(l+u) + 0.5(u-l)\ \widehat{\alpha}_r)^\epsilon$ as defined in (3.41) is a polynomial in $\widehat{\alpha}_r$ with $\epsilon + 1$ monomials. The number of required elementary operations for `getSubset` is largest if every monomial of the SPZ contains the expression α_r^ϵ. In this case, construction of the matrices $\widehat{G}_i$ in (3.40) has complexity $\mathcal{O}(nh(\epsilon+1))$ and each of the h matrices $\widehat{E}_i$ and $\widehat{G}_i$ constructed according to (3.40) consists of $\epsilon + 1$ columns. Due to the concatenation of the matrices $\widehat{E}_i$ and $\widehat{G}_i$ in (3.39), the resulting SPZ consequently contains at most $\overline{h} = h(\epsilon+1)$ generators. Since the complexity of `compact` is $\mathcal{O}(\overline{h}(n + p\log(\overline{h})))$ according to Prop. 3.1.7, the subsequent application of the `compact` operation has complexity $\mathcal{O}(h(\epsilon+1)(n + p\log(h(\epsilon+1))))$. Summarizing all complexities finally results in an overall complexity of

$$\begin{aligned} \mathcal{O}(p) + \mathcal{O}(h) + \mathcal{O}(nh(\epsilon+1)) + \mathcal{O}\big(h(\epsilon+1)(n + p\log(h(\epsilon+1)))\big) \\ = \mathcal{O}\big(h\epsilon(n + p\log(h\epsilon))\big), \end{aligned} \tag{3.43}$$

which is $\mathcal{O}(n^3 \log(n))$ using Assumption 3.1.3. □

Splitting, amongst other things, is useful for computing an enclosing support function as shown in Sec. 3.1.4. For polynomial zonotopes, we propose the following implementation of the `split` operation:

Proposition 3.1.44. *(Split) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ and the index of one dependent factor $r \in \{1, \ldots, p\}$, $\texttt{split}(\mathcal{PZ}, r) = (\mathcal{PZ}_1, \mathcal{PZ}_2)$ returns two SPZs*

$$\begin{aligned} \mathcal{PZ}_1 &= \texttt{getSubset}\big(\mathcal{PZ}, r, [-1, 0]\big), \\ \mathcal{PZ}_2 &= \texttt{getSubset}\big(\mathcal{PZ}, r, [0, 1]\big) \end{aligned}$$

that satisfy $\mathcal{PZ}_1 \cup \mathcal{PZ}_2 = \mathcal{PZ}$, where `getSubset` *is defined as in Prop. 3.1.43. The computational complexity with respect to the dimension n is $\mathcal{O}(n^3 \log(n))$.*

Proof. The `split` operation for SPZs is based on the substitution of the selected dependent factor α_r with two new dependent factors $\alpha_{r,1}$ and $\alpha_{r,2}$:

$$\begin{aligned}\{\alpha_r \mid \alpha_r \in [-1,1]\} =& \{\alpha_{r,1} + \alpha_{r,2} \mid \alpha_{r,1} \in [-1,0],\ \alpha_{r,2} \in [0,1]\} \\ & \{\alpha_{r,1} \mid \alpha_{r,1} \in [-1,0]\} \cup \{\alpha_{r,2} \mid \alpha_{r,2} \in [0,1]\}.\end{aligned} \tag{3.44}$$

Inserting this substitution into the definition of SPZs in Def. 3.1.1 yields

$$\mathcal{PZ} = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1] \right\} \overset{(3.44)}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) (\alpha_{r,1} + \alpha_{r,2})^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1],\ \alpha_{r,1} \in [-1,0],\ \alpha_{r,2} \in [0,1] \right\}$$

$$\overset{(3.44)}{=} \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \alpha_{r,1}^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1],\ \alpha_{r,1} \in [-1,0] \right\}}_{=\mathcal{PZ}_1 = \texttt{getSubset}(\mathcal{PZ}, r, [-1,0])}$$

$$\cup \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \alpha_{r,2}^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1],\ \alpha_{r,2} \in [0,1] \right\}}_{=\mathcal{PZ}_2 = \texttt{getSubset}(\mathcal{PZ}, r, [0,1])},$$

which concludes the proof.

Complexity: The operation `getSubset` has complexity $\mathcal{O}(n^3 \log(n))$ according to Prop. 3.1.43. For the `split` operation `getSubset` has to be applied two times, resulting in a complexity of $2 \cdot \mathcal{O}(n^3 \log(n)) = \mathcal{O}(n^3 \log(n))$. □

The `split` operation for polynomial zonotopes is not exact, meaning that the resulting sets usually overlap, where the size of the overlapping area mainly depends on the selected dependent factor α_r. One heuristic for minimizing the overlap that we observed to perform well is to select the dependent factor α_r that maximizes the value $\sum_{i \in \mathcal{K}} \|G_{(\cdot,i)}\|_2$, where the set $\mathcal{K}$ is defined as in (3.39). Let us demonstrate the operation `split` by an example:

Example 3.1.45. *We consider the SPZ*

$$\mathcal{PZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ}.$$

Splitting $\mathcal{PZ}$ along the first dependent factor using Prop. 3.1.44 yields the two SPZs

$$\mathcal{PZ}_1 = \left\langle \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1.5 & 1 & 0.5 \\ 0 & 2.5 & 1 & 0.5 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{bmatrix}, [3\ 4] \right\rangle_{PZ}$$

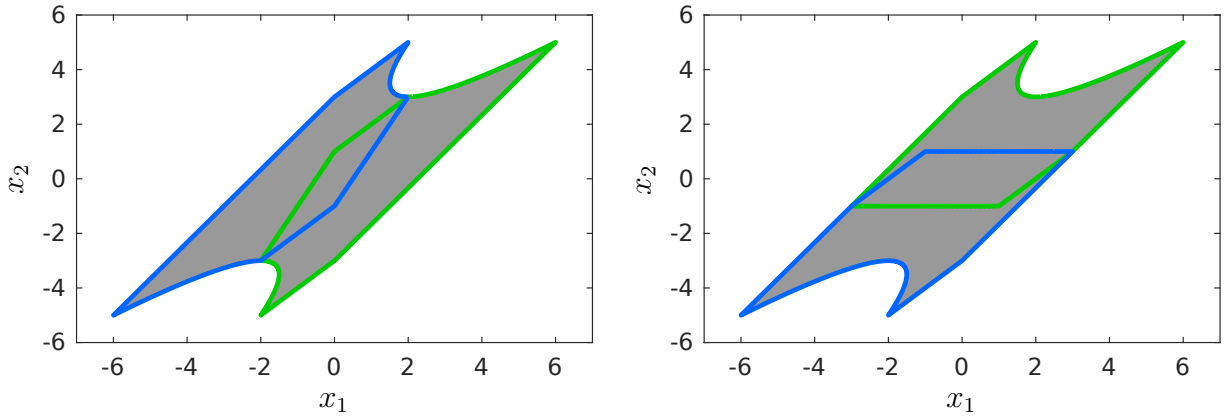

Figure 3.8: Splits of the SPZ $\mathcal{PZ}$ from Example 3.1.45 (gray) computed with Prop. 3.1.44 using the first dependent factor (left) and the second dependent factor (right).

and

$$\mathcal{PZ}_2 = \left\langle \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1.5 & -1 & 0.5 \\ 0 & 2.5 & -1 & 0.5 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{bmatrix}, [5\ 6] \right\rangle_{PZ},$$

which are visualized in Fig. 3.8.

3.2 Constrained Polynomial Zonotopes

In this section[2], we extend SPZs by adding polynomial equality constraints on the dependent factors, which results in a novel non-convex set representation called *constrained polynomial zonotopes* (CPZs). Contrary to the previously presented SPZs, CPZs are also closed under intersection and union. The section is structured as follows: After defining CPZs in Sec. 3.2.1 we introduce some useful preliminaries in Sec. 3.2.2. Next, we demonstrate how to convert other set representations to CPZs in Sec. 3.2.3 and show how to tightly enclose CPZs by simpler set representations in Sec. 3.2.4. In Sec. 3.2.5 we derive closed-form expressions for basic set operations on CPZs. Finally, we present necessary and sufficient conditions for intersection and containment checks for CPZs in Sec. 3.2.6 and introduce useful auxiliary operations on CPZs in Sec. 3.2.7. An overview for all operations on CPZs considered in this section is shown in Tab. 3.6.

3.2.1 Definition

Let us first define constrained polynomial zonotopes. A CPZ is constructed by adding polynomial equality constraints to a sparse polynomial zonotope:

Definition 3.2.1. *(Constrained Polynomial Zonotope) Given a constant offset $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times h}$, an exponent matrix $E \in \mathbb{N}_0^{p \times h}$, a constraint generator matrix $A \in \mathbb{R}^{m \times q}$, a constraint vector $b \in \mathbb{R}^m$, and a constraint exponent matrix $R \in \mathbb{N}_0^{p \times q}$, a constrained polynomial zonotope $\mathcal{CPZ} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{CPZ} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [-1,1] \right\},$$

where the scalars α_k are called factors of the CPZ. The number of factors is p, the number of generators $G_{(\cdot,i)}$ is h, the number of constraints is m, and the number of constraint generators $A_{(\cdot,i)}$ is q. The degree-of-freedom order $\rho_f = \frac{p-m}{n}$ of an CPZ is a measure for the complexity of the set, and the order $\rho = \frac{h+q}{n}$ of an CPZ estimates the representation size. The CPZ is regular if the exponent matrix E and the constraint exponent matrix R do not contain duplicate columns or all-zero columns:

$$\forall i, j \in \{1, \ldots, h\} :\ (i \neq j) \Rightarrow \left(E_{(\cdot,i)} \neq E_{(\cdot,j)}\right) \quad \text{and} \quad \forall i \in \{1, \ldots, h\} :\ E_{(\cdot,i)} \neq \mathbf{0},$$

and

$$\forall i, j \in \{1, \ldots, q\} :\ (i \neq j) \Rightarrow \left(R_{(\cdot,i)} \neq R_{(\cdot,j)}\right) \quad \text{and} \quad \forall i \in \{1, \ldots, q\} :\ R_{(\cdot,i)} \neq \mathbf{0}.$$

For a concise notation we introduce the shorthand $\mathcal{CPZ} = \langle c, G, E, A, b, R \rangle_{CPZ}$.

Similar to SPZs, it would be meaningful to introduce independent generators for CPZs to accelerate the computations and to add unique identifiers for keeping track of dependencies. However, to simplify the derivations of operations on CPZs and to maintain a compact notation we omit independent generators and unique identifiers here and use the definition

[2]This section is based on [80].

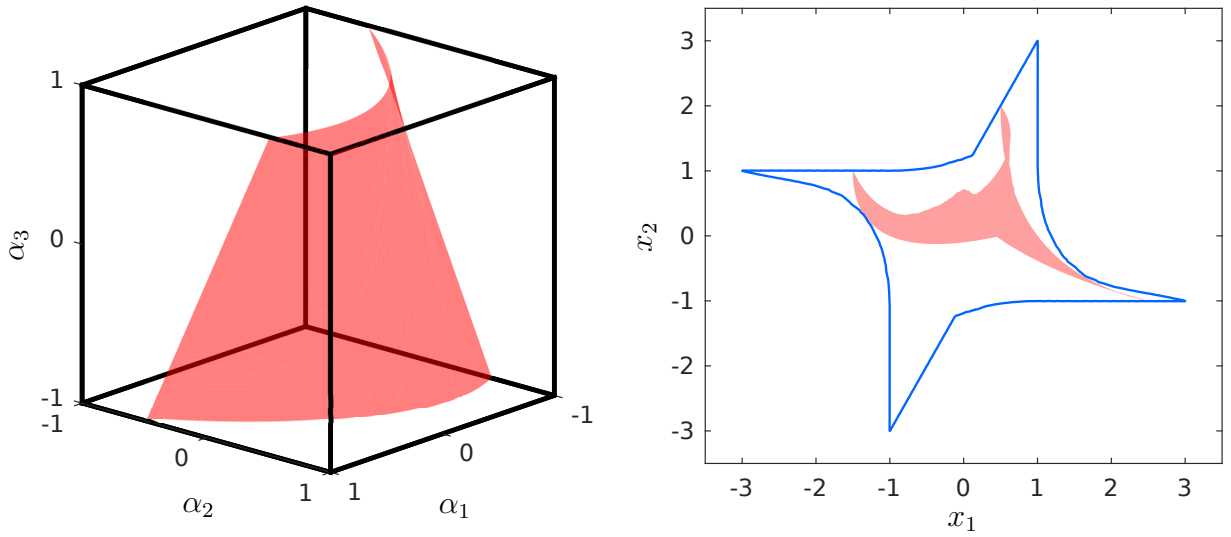

Figure 3.9: Visualization of the polynomial constraint (left), the unconstrained polynomial zonotope (right,blue), and the constrained polynomial zonotope (right,red) for the CPZ in Example 3.2.2.

of CPZs presented in Def. 3.2.1. Nevertheless, the incorporation of unique identifiers into operations on CPZs is straightforward and can be done in the same way as for SPZ presented in Chapter 3.1. In addition, we will shortly discuss how to extend CPZs with independent generators at the end of Sec. 3.2.5. We demonstrate the concept of CPZs by an example:

Example 3.2.2. *The CPZ*

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 & 0.5 \end{bmatrix}, 0.5, \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \right\rangle_{CPZ}$$

defines the set

$$\mathcal{CPZ} = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \alpha_2 + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \alpha_1\alpha_2\alpha_3 + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \alpha_1^2\alpha_3 \;\middle|\; \alpha_2 - 0.5\alpha_1\alpha_3 + 0.5\alpha_1^2 = 0.5,\ \alpha_1, \alpha_2, \alpha_3 \in [-1, 1] \right\},$$

which is visualized in Fig. 3.9.

For the derivation of the computational complexity of operations on CPZs, we make the following assumption:

Assumption 3.2.3. *Given a CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R \rangle_{CPZ} \subset \mathbb{R}^n$ *with* $p \in \mathbb{N}_0$ *factors,* $h \in \mathbb{N}_0$ *generators,* $q \in \mathbb{N}_0$ *constraint generators,* $m \in \mathbb{N}_0$ *constraints, and a maximum exponent matrix entry* $\epsilon = \max([E\ R])$*, we assume for the derivation of the computational complexity that*

$$p = c_p n,\ \ h = c_h n,\ \ q = c_q n,\ \ m = c_m n,\ \ \epsilon = c_\epsilon n,$$

with $c_p, c_h, c_q, c_m, c_\epsilon \in \mathbb{R}_{\geq 0}$.

The assumption is justified by the fact that the order $\rho = \frac{h+q}{n}$ and the number of constraints m are reduced to the desired order ρ_d and desired number of constraints m_d when computing with CPZs, such that $h + q \leq \rho_d n$ and $m \leq m_d$ holds.

Table 3.6: Overview showing all set operations on CPZs presented in this thesis.

Set Operation	**Reference**	**Page**
Redundancy removal for generators (`compactGen`)	Prop. 3.2.5	80
Redundancy removal for constraints (`compactCon`)	Prop. 3.2.6	80
Rescaling (`rescale`)	Prop. 3.2.7	81
Conversion SPZ to CPZ	Prop. 3.2.9	83
Conversion constrained zonotope to CPZ	Prop. 3.2.10	84
Conversion ellipsoid to CPZ	Prop. 3.2.11	85
SPZ enclosure of CPZ (`polyZonotope`)	Prop. 3.2.12	86
Con. zonotope enclosure of CPZ (`conZonotope`)	Prop. 3.2.13	88
Zonotope enclosure of CPZ (`zonotope`)	Prop. 3.2.14	89
Support function enclosure of CPZ	Prop. 3.2.15	91
Linear map	Prop. 3.2.16	93
Minkowski sum	Prop. 3.2.17	94
Exact addition	Prop. 3.2.18	95
Cartesian product	Prop. 3.2.19	95
Linear combination	Prop. 3.2.20	96
Convex hull	Prop. 3.2.21	97
Quadratic map	Prop. 3.2.22	99
Intersection	Prop. 3.2.23	101
Intersection with level set	Prop. 3.2.24	102
Union	Prop. 3.2.25	104
Minkowski difference with polytope	Prop. 3.2.26	105
Containment check interval in CPZ	Prop. 3.2.27	108
Intersection check	Prop. 3.2.28	108
Order reduction (`reduce`)	Prop. 3.2.29	110
Constraint reduction (`reduceCon`)	Prop. 3.2.31	112
Subset extraction (`getSubset`)	Prop. 3.2.34	117
Splitting (`split`)	Prop. 3.2.35	118

3.2.2 Preliminaries

Let us first establish some useful identities that are required in many of the proofs for operations on CPZs. According to the definition of CPZs in Def. 3.2.1, it holds that

$$\begin{aligned}&\Big\{c+\sum_{i=1}^{h_1}\Big(\prod_{k=1}^{p}\alpha_k^{E_{1(k,i)}}\Big)G_{1(\cdot,i)}+\sum_{i=1}^{h_2}\Big(\prod_{k=1}^{p}\alpha_k^{E_{2(k,i)}}\Big)G_{2(\cdot,i)}\ \Big|\\&\sum_{i=1}^{q}\Big(\prod_{k=1}^{p}\alpha_k^{R_{(k,i)}}\Big)A_{(\cdot,i)}=b,\ \alpha_k\in[-1,1]\Big\}=\big\langle c,[G_1\ G_2],[E_1\ E_2],A,b,R\big\rangle_{CPZ}\end{aligned}\tag{3.45}$$

and

$$\begin{aligned}&\Big\{c+\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_k^{E_{(k,i)}}\Big)G_{(\cdot,i)}\ \Big|\ \sum_{i=1}^{q_1}\Big(\prod_{k=1}^{p}\alpha_k^{R_{1(k,i)}}\Big)A_{1(\cdot,i)}=b_1,\\&\qquad\qquad\sum_{i=1}^{q_2}\Big(\prod_{k=1}^{p}\alpha_k^{R_{2(k,i)}}\Big)A_{2(\cdot,i)}=b_2,\ \alpha_k\in[-1,1]\Big\}\end{aligned}\tag{3.46}$$

$$=\left\langle c,G,E,\begin{bmatrix}A_1&\mathbf{0}\\\mathbf{0}&A_2\end{bmatrix},\begin{bmatrix}b_1\\b_2\end{bmatrix},\begin{bmatrix}R_1&R_2\end{bmatrix}\right\rangle_{CPZ}.$$

Moreover, we have

$$\Big\{c+\sum_{k=1}^{p}\alpha_kG_{(\cdot,k)}\ \Big|\ \sum_{k=1}^{p}\alpha_kA_{(\cdot,k)}=b,\ \alpha_k\in[-1,1]\Big\}=$$

$$\Big\{c+\sum_{i=1}^{p}\Big(\prod_{k=1}^{p}\alpha_k^{I_{p(k,i)}}\Big)G_{(\cdot,i)}\ \Big|\ \sum_{i=1}^{p}\Big(\prod_{k=1}^{p}\alpha_k^{I_{p(k,i)}}\Big)A_{(\cdot,i)}=b,\ \alpha_k\in[-1,1]\Big\}\tag{3.47}$$

$$=\langle c,G,I_p,A,b,I_p\rangle_{CPZ}.$$

Next, we introduce the *lifted polynomial zonotope* corresponding to an CPZ in the following lemma, which is inspired by [32, Prop. 3]:

Lemma 3.2.4. *Given a CPZ* $\mathcal{CPZ}=\langle c,G,E,A,b,R\rangle_{CPZ}\subset\mathbb{R}^n$*, the corresponding lifted polynomial zonotope* $\mathcal{PZ}^+\subset\mathbb{R}^{n+m}$ *defined as*

$$\mathcal{PZ}^+=\left\langle\begin{bmatrix}c\\-b\end{bmatrix},\underbrace{\begin{bmatrix}G&\mathbf{0}\\\mathbf{0}&A\end{bmatrix}}_{\overline{G}},[\ ],\underbrace{\begin{bmatrix}E&R\end{bmatrix}}_{\overline{E}},\texttt{uniqueID}(p)\right\rangle_{PZ}\tag{3.48}$$

satisfies

$$\forall x\in\mathbb{R}^n:\quad\big(x\in\mathcal{CPZ}\big)\Leftrightarrow\left(\begin{bmatrix}x\\\mathbf{0}\end{bmatrix}\in\mathcal{PZ}^+\right).$$

Proof. According to the definition of CPZs in Def. 3.2.1 it holds that

$$\big(x \in \mathcal{CPZ}\big) \overset{\text{Def. 3.2.1}}{\Leftrightarrow}$$

$$\left(\exists \alpha \in [-\mathbf{1}, \mathbf{1}] : \left(x = c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}}\right) G_{(\cdot,i)}\right) \wedge \left(\sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}}\right) A_{(\cdot,i)} = b\right)\right)$$

$$\overset{(3.48)}{\Leftrightarrow} \left(\exists \alpha \in [-\mathbf{1}, \mathbf{1}] : \left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} c \\ -b \end{bmatrix} + \sum_{i=1}^{h+q} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}}\right) \overline{G}_{(\cdot,i)}\right) \overset{(3.48)}{\Leftrightarrow} \left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathcal{PZ}^+\right),$$

where $\alpha = [\alpha_1 \ \dots \ \alpha_p]^T$. □

As for SPZs, many operations on CPZs result in CPZs that are not regular. We therefore introduce the operations `compactGen` and `compactCon` to remove redundancies from the generator representation and from the constraints of a CPZs, respectively:

Proposition 3.2.5. *(Compact Generators) Given a non-regular CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$*, the operation* `compactGen` *returns the corresponding CPZ with regular exponent matrix* $\overline{E}$*:*

$$\texttt{compactGen}(\mathcal{CPZ}) = \big\langle \overline{c}, \overline{G}, \overline{E}, A, b, R\big\rangle_{CPZ},$$

where $\overline{c}, \overline{G}, \overline{E}$ *are calculated using the* `compact` *operation for SPZs as defined in Prop. 3.1.7:*

$$\langle \overline{c}, \overline{G}, [\,], \overline{E}, id\rangle_{PZ} = \texttt{compact}\big(\langle c, G, [\,], E, id\rangle_{PZ}\big),$$

with $id = \texttt{uniqueID}(p)$*. The computational complexity is* $\mathcal{O}(h(n + p\log(h)))$*.*

Proof. According to Def. 3.2.1, the exponent matrix of a CPZ is regular if it does not contain duplicate columns or all-zero columns. Since the definition of regularity for CPZs is therefore identical to the definition of regularity for SPZs, we can apply the `compact` operation for SPZs to remove redundancies in the exponent matrix.

Complexity: The `compact` operation for SPZs has complexity $\mathcal{O}(h(n+p\log(h)))$ according to Prop. 3.1.7, which is consequently also the complexity of `compactGen`. □

While `compactGen` removes redundancies in the generator representation, `compactCon` removes redundancies in the constraints:

Proposition 3.2.6. *(Compact Constraints) Given a non-regular CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$*, the operation* `compactCon` *returns the corresponding CPZ with regular constraint exponent matrix* $\overline{R}$*:*

$$\texttt{compactCon}(\mathcal{CPZ}) = \big\langle c, G, E, \overline{A}, -\overline{b}, \overline{R}\big\rangle_{CPZ},$$

where $\overline{A}, \overline{b}, \overline{R}$ *are calculated using the* `compact` *operation for SPZs as defined in Prop. 3.1.7:*

$$\langle \overline{b}, \overline{A}, [\,], \overline{R}, id\rangle_{PZ} = \texttt{compact}\big(\langle -b, A, [\,], R, id\rangle_{PZ}\big),$$

with $id = \texttt{uniqueID}(p)$*. The computational complexity is* $\mathcal{O}(q(m + p\log(q)))$*.*

Proof. Due to the similarity of the constraints of a CPZ

$$\mathbf{0} \in \left\{ -b + \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1,1] \right\}$$

and the definition of SPZs in Def. 3.1.1

$$x \in \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1] \right\},$$

we can apply the `compact` operations for SPZs with the substitutions $c = -b$, $G = A$, $G_I = [\,]$, and $E = R$ to remove redundancies from the constraint exponent matrix R.

Complexity: According to Prop. 3.1.7 the `compact` operation for SPZs has complexity $\mathcal{O}(h(n + p\log(h)))$. Since we apply the `compact` operation for the constraints of the CPZ we have to insert the substitutions $n = m$ and $h = q$, so that the overall complexity of `compactCon` is $\mathcal{O}(q(m + p\log(q)))$. □

For the enclosure of CPZs with other set representations or for the reduction of the number of constraints we often remove constraints from the CPZ. The over-approximation resulting from the removal is mainly determined by the size of the corresponding unconstrained polynomial zonotope. Since the constraints often intersect only part of the factor hypercube $\alpha_1, \ldots, \alpha_p \in [-1,1]$, we can reduce the size of the unconstrained polynomial zonotope prior to constraint removal by applying a contractor. For this, we introduce the operation `rescale`:

Proposition 3.2.7. *(Rescale) Given a CPZ $\mathcal{CPZ} = \langle c, G, E, A, b, R \rangle_{CPZ} \subset \mathbb{R}^n$, the operation* `rescale` *defined as*

$$\texttt{rescale}(\mathcal{CPZ}) = \mathcal{CPZ}_p := \langle \overline{c}, \overline{G}, \overline{E}, \overline{A}, \overline{b}, \overline{R} \rangle_{CPZ}$$

returns a CPZ that is equivalent to $\mathcal{CPZ}$ and satisfies

$$\left\langle \overline{c}, \overline{G}, [\,], \overline{E}, \texttt{uniqueID}(p) \right\rangle_{PZ} \subseteq \left\langle c, G, [\,], E, \texttt{uniqueID}(p) \right\rangle_{PZ},$$

where

$$\mathcal{CPZ}_i = \begin{cases} \texttt{getSubset}\big(\mathcal{CPZ}_{i-1}, i, [l_{(i)}, u_{(i)}]\big), & (l_{(i)} \neq -1) \vee (u_{(i)} \neq 1) \\ \mathcal{CPZ}_{i-1}, & \textit{otherwise} \end{cases}, \quad i = 1, \ldots, p$$

$$\mathcal{CPZ}_0 = \mathcal{CPZ},$$

the operator `getSubset` *is defined as in Prop. 3.2.34, and the new lower and upper bound $l, u \in \mathbb{R}^p$ for the factor domain are computed by applying a contractor to the constraints of the CPZ:*

$$[l, u] = \texttt{contract}\big(f(\alpha), [-\mathbf{1}, \mathbf{1}]\big), \quad f(\alpha) = -b + \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_{(k)}^{R_{(k,i)}} \right) A_{(\cdot,i)}.$$

The computational complexity with respect to the dimension n is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$.

Table 3.7: Computational complexity of the `rescale` operation as defined in Prop. 3.2.7 with respect to the dimension $n \in \mathbb{N}$ for the different contractors presented in Sec. 2.8.

Forward-backward	**Extremal functions**	**Parallel linearization**
$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^{4.5})$

Proof. According to the definition of contractors in Def. 2.8.1, it holds that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \big(f(\alpha) = \mathbf{0}\big) \Rightarrow \Big(\alpha \in \underbrace{\texttt{contract}\big(f(\alpha), [-\mathbf{1}, \mathbf{1}]\big)}_{[l,u]}\Big). \tag{3.49}$$

The substitution of the factor domain $\alpha \in [-\mathbf{1}, \mathbf{1}]$ with $\alpha \in [l, u]$ therefore does not remove any values for the factors α that satisfy the constraints, which proves that the set resulting from $\texttt{rescale}(\mathcal{CPZ})$ is equivalent to the original set $\mathcal{CPZ}$:

$$\mathcal{CPZ} = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \underbrace{\sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b}_{f(\alpha)=\mathbf{0}},\ \alpha_k \in [-1, 1] \right\} \overset{(3.49)}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [l_{(k)}, u_{(k)}] \right\} \overset{\text{Prop. 3.2.34}}{=}$$

$$\texttt{getSubset}\Big(\ldots\ \texttt{getSubset}\big(\mathcal{CPZ}, 1, [l_{(1)}, u_{(1)}]\big)\ \ldots\ , p, [l_{(p)}, u_{(p)}] \Big) = \texttt{rescale}(\mathcal{CPZ}),$$

where $\alpha = [\alpha_1\ \ldots\ \alpha_p]^T$.

Complexity: The computational complexity for the operation `getSubset` is $\mathcal{O}(n^3 \log(n))$ according to Prop. 3.2.34. Since we have to apply `getSubset` p times at most, the overall complexity for the `rescale` operation is

$$p \cdot \mathcal{O}(n^3 \log(n)) + \mathcal{O}(\texttt{contract}) \overset{\text{Assumption 3.2.3}}{=} \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}).$$

The computational complexity $\mathcal{O}(\texttt{contract})$ of the contractor depends according to Tab. 2.5 on the number of elementary operations $e \in \mathbb{N}_0$ in the corresponding subfunction $f_{(i)}(\alpha)$, $i = 1, \ldots, m$. In our case, evaluation of each subfunction $f_{(i)}(\alpha)$ requires pq exponentiations, pq multiplications, and q additions. Overall, this consequently results in

$$e = 2pq + q = q(2p + 1)$$

elementary operations, which yields the computational complexities shown in Tab. 3.7. □

The computational complexity of the `rescale` operation for different contractors is summarized in Tab. 3.7.

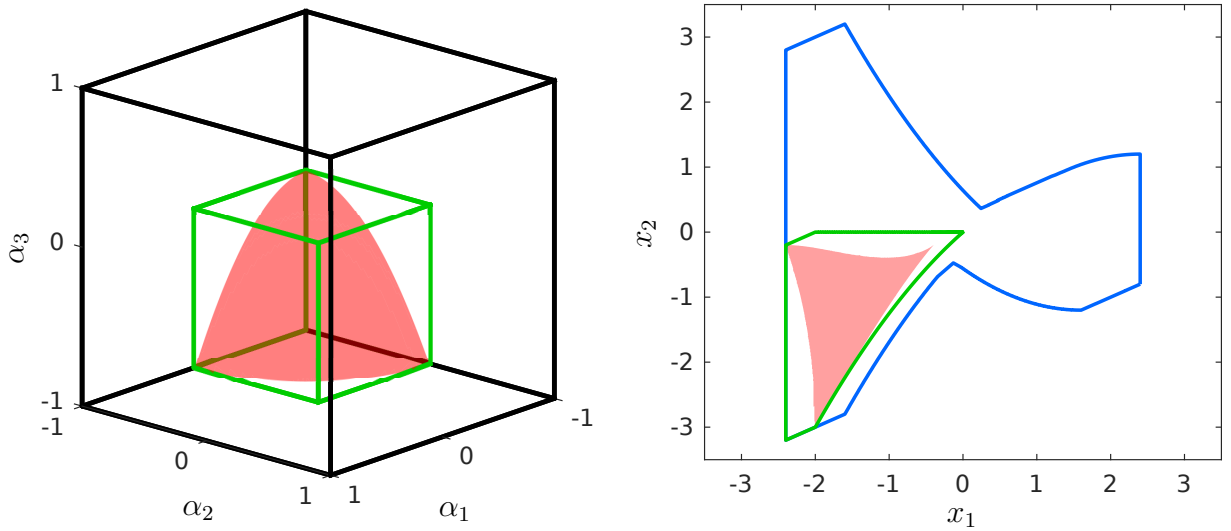

Figure 3.10: Visualization of resclaing using Prop. 3.2.7 for the CPZ $\mathcal{CPZ}$ from Example 3.2.8 (red, right), where the corresponding constraint is visualized on the left. The unconstrained polynomial zonotope before rescaling is shown in blue, and the unconstrained polynomial zonotope after rescaling is shown in green.

Let us demonstrate rescaling by an example:

Example 3.2.8. *We consider the CPZ*

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 & 0 & 0.4 \\ 0 & -2 & 1 & 0.2 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 2 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \end{bmatrix}, -2, \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right\rangle_{CPZ},$$

which is visualized in Fig. 3.10. As depicted on the left side of Fig. 3.10, the constraint only intersects a small part of the factor domain $\alpha_1, \alpha_2, \alpha_3 \in [-1, 1]$, so that the domain can be contracted to $\alpha_1, \alpha_2, \alpha_3 \in [-1, 0]$. Rescaling using Prop. 3.2.7 therefore significantly reduces the size of the unconstrained polynomial zonotope, as visualized on the right side of Fig. 3.10.

3.2.3 Conversion from other Set Representations

In this section we demonstrate how other set representations can be converted to CPZs. We first show that each polynomial zonotope can be represented as a CPZ:

Proposition 3.2.9. *(Conversion Polynomial Zonotope) A sparse polynomial zonotope $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ can be equivalently represented by a CPZ*

$$\mathcal{PZ} = \left\langle c, \begin{bmatrix} G & G_I \end{bmatrix}, \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix}, [\,], [\,], [\,] \right\rangle_{CPZ}.$$

The computational complexity of the conversion is $\mathcal{O}(1)$.

Proof. According to Prop. 3.1.4, each SPZ can be equivalently represented by a SPZ without independent generators:

$$\langle c, G, G_I, E, id \rangle_{PZ} \overset{\text{Prop. 3.1.4}}{=} \left\langle c, \begin{bmatrix} G & G_I \end{bmatrix}, [\,], \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix}, \begin{bmatrix} id & \texttt{uniqueID}(q) \end{bmatrix} \right\rangle_{PZ}.$$

The remainder of the conversion is trivial since a SPZ is simply a CPZ without constraints (see Def. 3.1.1 and Def. 3.2.1).

Complexity: The complexity of the conversion is $\mathcal{O}(1)$ since the construction of the CPZ only requires concatenations and initializations. □

According to Prop. 3.1.12, the set defined by a Taylor model can be equivalently represented as a SPZ. Moreover, according to Prop. 3.1.10 and Prop. 3.1.9 any interval and any zonotope can be represented as a SPZ. It therefore holds according to Prop. 3.2.9 that any Taylor model, any zonotope, and any interval can equivalently be represented by a CPZ. Next, we prove that any constrained zonotope can be represented as a CPZ:

Proposition 3.2.10. *(Conversion Constrained Zonotope) A constrained zonotope* $\mathcal{CZ} = \langle c, G, A, b \rangle_{CZ} \subset \mathbb{R}^n$ *can be equivalently represented by a CPZ*

$$\mathcal{CZ} = \langle c, G, I_l, A, b, I_l \rangle_{CPZ}.$$

The computational complexity of the conversion is $\mathcal{O}(1)$.

Proof. With the definition of constrained zonotopes in Def. 2.2.5 and the identity in (3.47) we obtain

$$\mathcal{CZ} \overset{\text{Def. 2.2.5}}{=} \left\{ c + \sum_{k=1}^{l} \alpha_k G_{(\cdot,k)} \;\middle|\; \sum_{k=1}^{l} \alpha_k A_{(\cdot,k)} = b,\ \alpha_k \in [-1,1] \right\} \overset{(3.47)}{=}$$

$$\left\{ c + \sum_{i=1}^{l} \left(\prod_{k=1}^{l} \alpha_k^{I_{l(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{l} \left(\prod_{k=1}^{l} \alpha_k^{I_{l(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [-1,1] \right\}$$

$$= \langle c, G, I_l, A, b, I_l \rangle_{CPZ},$$

which concludes the proof.

Complexity: The complexity of the conversion is $\mathcal{O}(1)$ since the construction of the CPZ only involves initializations. □

There are two possibilities to represent a bounded polytope as a CPZ: According to Prop. 3.1.11, every bounded polytope can equivalently be represented as a SPZ. Therefore, any bounded polytope can be converted to a CPZ by first representing it as a SPZ followed its conversion to a CPZ according to Prop. 3.2.9. In addition, it holds according to [32, Thm. 1] that any bounded polytope can be represented as a constrained zonotope. Therefore, the second possibility for the conversion of a bounded polytope to a CPZ is to first represent the polytope as a constrained zonotope, and then convert it to a CPZ using Prop. 3.2.10. Which of the two methods results in the more compact representation depends on the polytope. Now, we prove that any ellipsoid can be converted to a CPZ:

Proposition 3.2.11. *(Conversion Ellipsoid) An ellipsoid $\mathcal{E} = \langle c, Q \rangle_E \subset \mathbb{R}^n$ can be equivalently represented by a CPZ*

$$\mathcal{E} = \left\langle c, \underbrace{V \begin{bmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \end{bmatrix}}_{G}, \underbrace{\begin{bmatrix} I_n \\ \mathbf{0} \end{bmatrix}}_{E}, \underbrace{\begin{bmatrix} -0.5 & \mathbf{1} \end{bmatrix}}_{A}, \underbrace{0.5}_{b}, \underbrace{\begin{bmatrix} \mathbf{0} & 2I_n \\ 1 & \mathbf{0} \end{bmatrix}}_{R} \right\rangle_{CPZ}, \tag{3.50}$$

where the eigenvalues $\lambda_1, \ldots, \lambda_n$, the matrix of eigenvalues D, and the matrix of eigenvectors V are obtained by the eigenvalue decomposition of the matrix Q according to Def. 2.4.2:

$$Q = V \underbrace{\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}}_{D} V^T. \tag{3.51}$$

The computational complexity of the conversion is $\mathcal{O}(n^3)$.

Proof. The matrices A, R and the vector b in (3.50) define the constraint

$$-0.5\alpha_{n+1} + \alpha_1^2 + \ldots + \alpha_n^2 = 0.5. \tag{3.52}$$

Since $\alpha_{n+1} \in [-1, 1]$, (3.52) is equivalent to the constraint

$$0 \leq \alpha_1^2 + \ldots + \alpha_n^2 \leq 1. \tag{3.53}$$

Using the eigenvalue decomposition of the matrix Q from (3.51) we obtain

$$Q^{-1} \overset{(3.51)}{=} (VDV^T)^{-1} = VD^{-1}V^T, \tag{3.54}$$

where $V^{-1} = V^T$ holds since V is an orthonormal matrix. Inserting (3.54) into the definition of an ellipsoid in Def. 2.2.6 yields

$$\begin{aligned} \mathcal{E} &\overset{\text{Def. 2.2.6}}{=} \left\{ x \mid (x-c)^T Q^{-1} (x-c) \leq 1 \right\} = \left\{ c + x \mid x^T Q^{-1} x \leq 1 \right\} \overset{(3.54)}{=} \\ &\left\{ c + x \mid (V^T x)^T D^{-1} (V^T x) \leq 1 \right\} \overset{z := V^T x}{=} \\ &\left\{ c + Vz \mid z^T D^{-1} z \leq 1 \right\} \overset{(3.51)}{=} \left\{ c + Vz \;\middle|\; \frac{z_{(1)}^2}{\lambda_1} + \ldots + \frac{z_{(n)}^2}{\lambda_n} \leq 1 \right\}. \end{aligned} \tag{3.55}$$

We define the factors α_k of the CPZ as $\alpha_k = \frac{z_{(k)}}{\sqrt{\lambda_k}}$, $k = 1, \ldots, n$, so that

$$z_{(k)} = \sqrt{\lambda_k}\, \alpha_k. \tag{3.56}$$

Inserting (3.56) into (3.55) finally yields

$$\mathcal{E} \overset{(3.55)}{=} \left\{ c + Vz \;\middle|\; \frac{z_{(1)}^2}{\lambda_1} + \ldots + \frac{z_{(n)}^2}{\lambda_n} \leq 1 \right\} \overset{(3.56)}{=} \left\{ c + \sum_{k=1}^{n} \sqrt{\lambda_k}\ \alpha_k V_{(\cdot,k)} \;\middle|\; \alpha_1^2 + \ldots + \alpha_n^2 \leq 1 \right\}$$

$$\overset{\substack{(3.52)\\(3.53)}}{=} \left\{ c + \sum_{k=1}^{n} \sqrt{\lambda_k}\ \alpha_k V_{(\cdot,k)} \;\middle|\; -0.5\alpha_{n+1} + \alpha_1^2 + \ldots + \alpha_n^2 = 0.5,\ \alpha_1, \ldots, \alpha_{n+1} \in [-1, 1] \right\}$$

$$\overset{(3.50)}{=} \langle c, G, E, A, b, R \rangle_{CPZ},$$

which concludes the proof.

Complexity: Computation of the eigenvalue decomposition $Q = VDV^T$ in (3.51) has complexity $\mathcal{O}(n^3)$ according to Tab. 2.1. Moreover, computation of the generator matrix G in (3.50) requires n^2 multiplications and the calculation of n square roots and therefore has complexity $\mathcal{O}(n^2) + \mathcal{O}(n) = \mathcal{O}(n^2)$. Since all other required operations are concatenations or initializations, the overall complexity of the conversion is $\mathcal{O}(n^2) + \mathcal{O}(n^3) = \mathcal{O}(n^3)$. □

In summary, CPZs can equivalently represent all common set representations except support functions, level sets, and star sets.

3.2.4 Enclosure by other Set Representations

To speed up computations, one often encloses sets by simpler set representations in set-based computing. In this section, we therefore show how to enclose CPZs by polynomial zonotopes, zonotopes, constraint zonotopes, and support functions. To demonstrate the tightness of the enclosures, we use the CPZ

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.5 & 1 & 0.5 \\ 0 & 1 & 1 & 0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -0.5 & 0.5 \end{bmatrix}, 0.5, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 1 & 0 \end{bmatrix} \right\rangle_{CPZ} \tag{3.57}$$

as a running example throughout this section. Let us first present how to compute an enclosing polynomial zonotope of a CPZ:

Proposition 3.2.12. *(Polynomial Zonotope Enclosure) Given a CPZ $\mathcal{CPZ} \subset \mathbb{R}^n$, the operation* `polyZonotope` *returns a SPZ that encloses $\mathcal{CPZ}$:*

$$\mathcal{CPZ} \subseteq \texttt{polyZonotope}(\mathcal{CPZ}) = \langle c, G, [\,], E, \texttt{uniqueID}(p) \rangle_{PZ},$$

where the `rescale` *operation as defined in Prop. 3.2.7 is applied to reduce the over-approximation*

$$\langle c, G, E, A, b, R \rangle_{CPZ} = \texttt{rescale}(\mathcal{CPZ}),$$

and $p \in \mathbb{N}_0$ denotes the number of rows of matrix E. The computational complexity with respect to the dimension n is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$.

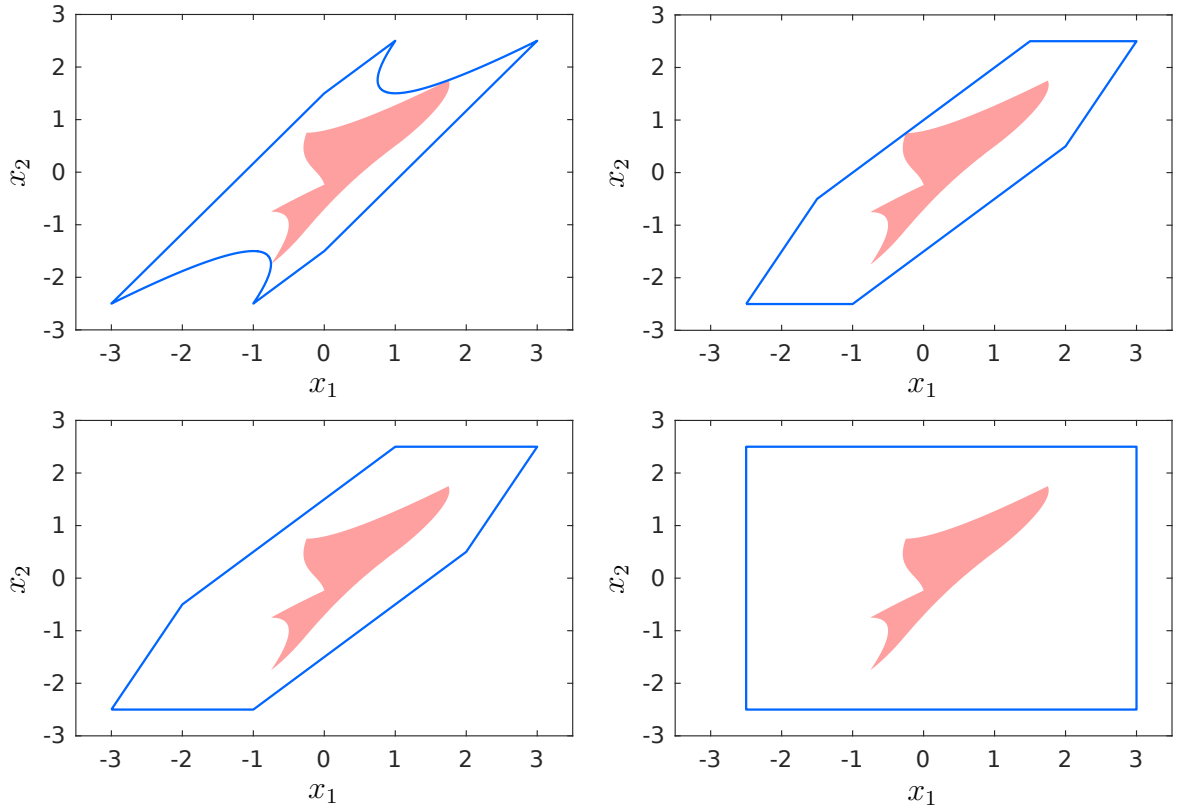

Figure 3.11: Enclosure of the CPZ in (3.57) with a polynomial zonotope (top, left), a constrained zonotope (top, right), a zonotope (bottom, left), and an interval (bottom, right).

Proof. According to Prop. 3.2.7, the `rescale` operation only changes the representation of the set, but not the set itself. The enclosing SPZ is then obtained by simply removing the constraints from the rescaled CPZ, which results in an over-approximation.

Complexity: The computational complexity is identical to the complexity of the `rescale` operation, which is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$ according to Prop. 3.2.7. □

The enclosing polynomial zonotope for the CPZ in (3.57) is shown in Fig. 3.11 and the computational complexity of Prop. 3.2.12 for different contractors is summarized in Tab. 3.8. An alternative method for computing a polynomial zonotope enclosure of an CPZ is to reduce all constraints of the CPZ using Prop. 3.2.31. While this is computationally more demanding, if often results in a tighter enclosure. It is also possible to combine both methods. For this, we first reduce all constraints for which we can show that the resulting over-approximation is small using Prop. 3.2.31, and then simply drop the remaining constraints as done in Prop. 3.2.12. Next, we show how to enclose a CPZ by a constrained zonotope:

Proposition 3.2.13. *(Constrained Zonotope Enclosure) Given a CPZ* $\mathcal{CPZ} \subset \mathbb{R}^n$*, the operation* `conZonotope` *returns a constrained zonotope that encloses* $\mathcal{CPZ}$*:*

$$\mathcal{CPZ} \subseteq \texttt{conZonotope}(\mathcal{CPZ}) = \underbrace{\langle c_z, G_z, A_z, -b_z \rangle_{CZ}}_{\mathcal{CZ}},$$

where

$$\langle c, G, E, A, b, R\rangle_{CPZ} = \texttt{rescale}(\mathcal{CPZ}), \quad id = \texttt{uniqueID}(p),$$

$$\underbrace{\left\langle \begin{bmatrix} c_z \\ b_z \end{bmatrix}, \begin{bmatrix} G_z \\ A_z \end{bmatrix} \right\rangle_Z}_{\mathcal{Z}^+} = \texttt{zonotope}\Bigg(\texttt{compact}\Bigg(\underbrace{\left\langle \begin{bmatrix} c \\ -b \end{bmatrix}, \begin{bmatrix} G & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}, [\,], \begin{bmatrix} E & R \end{bmatrix}, id \right\rangle_{PZ}}_{\mathcal{PZ}^+}\Bigg)\Bigg),$$

and the zonotope enclosure of the SPZ $\mathcal{PZ}^+$ is calculated using Prop. 3.1.14. The `rescale` *operation as defined in Prop. 3.2.7 and the compact operation for SPZs as defined in Prop. 3.1.7 are applied to reduce the over-approximation. The computational complexity with respect to the dimension n is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$.*

Proof. To calculate an enclosing constrained zonotope we compute a zonotope enclosure of the corresponding lifted polynomial zonotope as defined in Lemma 3.2.4. Back-transformation of the lifted zonotope to the original space then yields an enclosing constrained zonotope:

$$\forall x \in \mathbb{R}^n : \quad (x \in \mathcal{CPZ}) \overset{\text{Lemma 3.2.4}}{\Rightarrow} \left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathcal{PZ}^+ \right)$$

$$\overset{\mathcal{PZ}^+ \subseteq \mathcal{Z}^+}{\Rightarrow} \left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathcal{Z}^+ \right) \overset{\text{Lemma 3.2.4}}{\Rightarrow} (x \in \mathcal{CZ}),$$

where we omitted the operations `rescale` and `compact` since they only change the representation of the set, but not the set itself.

Complexity: The computational complexity for the `rescale` operation is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$ according to Prop. 3.2.7. Let $n^+ = n + m$, $p^+ = p$, and $h^+ = h + q$ denote the dimension, the number of dependent factors, and the number of dependent generators of the lifted polynomial zonotope $\mathcal{PZ}^+$. According to Prop. 3.1.7, the `compact` operation for SPZs has complexity $\mathcal{O}(h^+(n^+ + p^+ \log(h^+)))$, and the complexity for computing a zonotope enclosure of a SPZ using Prop. 3.1.14 is $\mathcal{O}(h^+(p^+ + n^+))$ according to (3.5). The overall computational complexity is therefore

$$\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(h^+(n^+ + p^+ \log(h^+))) + \mathcal{O}(h^+(p^+ + n^+))$$

$$= \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(h^+(n^+ + p^+ \log(h^+)))$$

$$\overset{\substack{n^+=n+m,\ p^+=p \\ h^+=h+q}}{=} \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}((h+q)(n+m+p\log(h+q))),$$

which is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$ using Assumption 3.2.3. □

The enclosing constrained zonotope for the CPZ in (3.57) is shown in Fig. 3.11 and the computational complexity of Prop. 3.2.13 for different contractors is summarized in Tab. 3.8.

Table 3.8: Computational complexity with respect to the dimension $n \in \mathbb{N}$ for polynomial zonotope, zonotope, constrained zonotope, and support function enclosure of CPZs for the different contractors presented in Sec. 2.8.

Method	Polynomial Zonotope	Constrained Zonotope	Zonotope	Support Function
Forward-backward	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^5)$
Extremal functions	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^5)$
Parallel linearization	$\mathcal{O}(n^{4.5})$	$\mathcal{O}(n^{4.5})$	$\mathcal{O}(n^{4.5})$	$\mathcal{O}(n^5)$

Now, we consider the enclosure of a CPZ by a zonotope:

Proposition 3.2.14. *(Zonotope Enclosure) Given a CPZ $\mathcal{CPZ} \subset \mathbb{R}^n$, the operation* `zonotope` returns a zonotope that encloses $\mathcal{CPZ}$:

$$\mathcal{CPZ} \subseteq \texttt{zonotope}(\mathcal{CPZ}) = \langle c_z, G_z \rangle_Z = \texttt{zonotope}\big(\underbrace{\texttt{polyZonotope}(\mathcal{CPZ})}_{\mathcal{PZ}}\big),$$

where the polynomial zonotope enclosure of $\mathcal{CPZ}$ is computed using Prop. 3.2.12 and the zonotope enclosure of the SPZ $\mathcal{PZ}$ is computed using Prop. 3.1.14. The computational complexity with respect to the dimension n is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$.

Proof. According to Prop. 3.2.12, it holds that $\mathcal{PZ} = \texttt{polyZonotope}(\mathcal{CPZ}) \supseteq \mathcal{CPZ}$. Moreover, since $\langle c_z, G_z \rangle_Z = \texttt{zonotope}(\mathcal{PZ}) \supseteq \mathcal{PZ}$ according to Prop. 3.1.14, we have

$$\langle c_z, G_z \rangle_Z = \texttt{zonotope}(\mathcal{PZ}) \overset{\text{Prop. 3.1.14}}{\supseteq} \mathcal{PZ} = \texttt{polyZonotope}(\mathcal{CPZ}) \overset{\text{Prop. 3.2.12}}{\supseteq} \mathcal{CPZ},$$

which concludes the proof.

Complexity: Computation of an enclosing polynomial zonotope for a CPZ has complexity $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$ according to Prop. 3.2.12 and computation of a zonotope enclosure of a SPZ has complexity $\mathcal{O}(n^2)$ according to Prop. 3.1.14. Consequently, we obtain

$$\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(n^2) = \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$$

for the overall complexity. □

The enclosing zonotope for the CPZ in (3.57) is shown in Fig. 3.11 and the computational complexity of Prop. 3.2.14 for different contractors is summarized in Tab. 3.8. An alternative method for computing a zonotope enclosure is to first enclose the CPZ by a constrained zonotope using Prop. 3.2.13, and then successively remove all constraints as described in [32, Appendix], which yields a zonotope. Which method performs better depends on the CPZ.

Finally, we show how to enclose the support function of a CPZ. Computing the exact support function of a CPZ is identical to solving a polynomial optimization problem, which consists of a polynomial objective function and polynomial constraints. However, computing the global optimum of a polynomial optimization problem up to a desired accuracy requires to split the state space, and therefore has exponential complexity with respect to the dimension [81, 82]. Instead of computing the exact support function, we therefore compute a tight enclosure using quadratic programming:

Proposition 3.2.15. *(Support Function Enclosure) Given a CPZ $\mathcal{CPZ} \subset \mathbb{R}^n$, an over-approximation of its support function for a given direction $d \in \mathbb{R}^n$ can be computed as*

$$s_{\mathcal{CPZ}}(d) \leq c_z - \texttt{quadProg}\left(\begin{bmatrix} -Q^- & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \begin{bmatrix} -f \\ -g_z \end{bmatrix}, \begin{bmatrix} I_{p+p_z} \\ -I_{p+p_z} \end{bmatrix}, \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}, \begin{bmatrix} M & A_z \end{bmatrix}, -b_z \right).$$

To construct the matrices $Q^- \in \mathbb{R}^{p \times p}$, $M \in \mathbb{R}^{m \times p}$, $A_z \in \mathbb{R}^{m \times p_z}$ and the vectors $f \in \mathbb{R}^p$, $g_z \in \mathbb{R}^{p_z}$, $b_z \in \mathbb{R}^m$ for the quadratic program, we first calculate $d \otimes \mathcal{CPZ}$ using Prop. 3.2.16, and then construct the corresponding lifted polynomial zonotope $\mathcal{PZ}^+$ as defined in Lemma 3.2.4:

$$\langle c, g, E, A, b, R \rangle_{CPZ} = \texttt{rescale}(d \otimes \mathcal{CPZ}), \quad id = \texttt{uniqueID}(p),$$

$$\left\langle \begin{bmatrix} \overline{c} \\ \overline{b} \end{bmatrix}, \begin{bmatrix} \overline{g} \\ \overline{A} \end{bmatrix}, [\,], \overline{E}, id \right\rangle_{PZ} = \texttt{compact}\Bigg(\underbrace{\left\langle \begin{bmatrix} c \\ -b \end{bmatrix}, \begin{bmatrix} g & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}, [\,], \begin{bmatrix} E & R \end{bmatrix}, id \right\rangle_{PZ}}_{\mathcal{PZ}^+} \Bigg), \tag{3.58}$$

where the `rescale` *operation as defined in Prop. 3.2.7 and the* `compact` *operation as defined in Prop. 3.1.7 are applied to reduce the over-approximation. Next, we extract the linear and quadratic parts from the generators and the constraints:*

$$\begin{aligned} &\forall i, j \in \{1, \ldots, p\} : \quad Q_{(i,j)} = \begin{cases} 0 + \sum_{k \in \mathcal{K}_i} \overline{g}_{(k)}, & i = j \\ 0 + 0.5 \sum_{k \in \mathcal{H}_{i,j}} \overline{g}_{(k)}, & otherwise \end{cases}, \quad f_{(i)} = 0 + \sum_{k \in \mathcal{G}_i} \overline{g}_{(k)} \\ &\forall l \in \{1, \ldots, m\}, \ \forall i = \{1, \ldots, p\} : \quad M_{(l,i)} = 0 + \sum_{k \in \mathcal{G}_i} \overline{A}_{(l,k)}, \end{aligned} \tag{3.59}$$

where the sets $\mathcal{K}_i$, $\mathcal{H}_{i,j}$, and $\mathcal{G}_i$ are defined as follows:

$$\begin{aligned} &\forall i, j \in \{1, \ldots, p\} : \\ &\mathcal{K}_i = \left\{ k \;\middle|\; \sum_{l=1}^{p} \overline{E}_{(l,k)} = 2, \ \overline{E}_{(i,k)} = 2 \right\}, \quad \mathcal{G}_i = \left\{ k \;\middle|\; \sum_{l=1}^{p} \overline{E}_{(l,k)} = 1, \ \overline{E}_{(i,k)} = 1 \right\}, \\ &\mathcal{H}_{i,j} = \left\{ k \;\middle|\; \sum_{l=1}^{p} \overline{E}_{(l,k)} = 2, \ \overline{E}_{(i,k)} = 1, \ \overline{E}_{(j,k)} = 1 \right\}. \end{aligned}$$

To obtain a convex quadratic program we compute the eigenvalue decomposition of the matrix Q according to Def. 2.4.2

$$Q = V \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{bmatrix} V^T, \tag{3.60}$$

which we use to split Q into a positive semi-definite matrix Q^+ and a negative semi-definite matrix Q^-:

$$Q^+ = V \begin{bmatrix} max(\lambda_1, 0) & & 0 \\ & \ddots & \\ 0 & & max(\lambda_p, 0) \end{bmatrix} V^T,\ Q^- = V \begin{bmatrix} min(\lambda_1, 0) & & 0 \\ & \ddots & \\ 0 & & min(\lambda_p, 0) \end{bmatrix} V^T.$$

Finally, we enclose all remaining parts of $\mathcal{CPZ}$ with a zonotope using Prop. 3.1.14:

$$\left\langle \begin{bmatrix} c_z \\ b_z \end{bmatrix}, \begin{bmatrix} g_z \\ A_z \end{bmatrix} \right\rangle_Z = \texttt{zonotope}\big(\texttt{compact}\big(\widehat{\mathcal{PZ}}^+\big)\big), \tag{3.61}$$

where the `compact` *operation as defined in Prop. 3.1.7 is applied to reduce the over-approximation, the lifted polynomial zonotope $\widehat{\mathcal{PZ}}^+$ is defined as*

$$\widehat{\mathcal{PZ}}^+ = \left\langle \begin{bmatrix} \overline{c} \\ \overline{b} \end{bmatrix}, \begin{bmatrix} \overline{g}_{(\cdot,\mathcal{F})} & g_q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \overline{A}_{(\cdot,\mathcal{N})} \end{bmatrix}, \begin{bmatrix} \overline{E}_{(\cdot,\mathcal{F})} & E_q & \overline{E}_{(\cdot,\mathcal{N})} \end{bmatrix} \right\rangle_{PZ},$$

$$\mathcal{Q} = \bigcup_{i,j \in \{1,\dots,p\}} \mathcal{K}_i \cup \mathcal{H}_{i,j}, \quad \mathcal{G} = \bigcup_{i \in \{1,\dots,p\}} \mathcal{G}_i, \quad \mathcal{N} = \{1,\dots,\overline{h}\} \setminus \mathcal{G}, \quad \mathcal{F} = \mathcal{N} \setminus \mathcal{Q},$$

$$\big\langle \mathbf{0}, g_q, [\,], E_q, id \big\rangle_{PZ} = sq\big(\{Q^+\}, \mathcal{PZ}\big), \quad \mathcal{PZ} = \big\langle \mathbf{0}, I_p, [\,], I_p, id \big\rangle_{PZ}$$

with $\overline{h}$ denoting the number of columns of matrix $\overline{E}$, and the quadratic map $sq(\{Q^+\}, \mathcal{PZ})$ is calculated using Prop. 3.1.30. The computational complexity with respect to the dimension n is $\mathcal{O}(n^5) + \mathcal{O}(\texttt{contract})$.

Proof. To obtain an over-approximation of the support function, the CPZ $d \otimes \mathcal{CPZ}$ is first enclosed by the simpler set $\mathcal{S}$ consisting of quadratic monomials only:

$$d \otimes \mathcal{CPZ} \overset{(3.58)}{=} \left\{ \overline{c} + \sum_{i=1}^{\overline{h}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{g}_{(i)} \;\middle|\; \sum_{i=1}^{\overline{h}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{A}_{(\cdot,i)} = -\overline{b},\ \alpha_k \in [-1,1] \right\}$$

$$= \Bigg\{ \overline{c} + \underbrace{\sum_{i \in \mathcal{Q}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{g}_{(i)}}_{\overset{(3.59)}{=} \alpha^T Q \alpha = \alpha^T Q^+ \alpha + \alpha^T Q^- \alpha} + \underbrace{\sum_{i \in \mathcal{G}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{g}_{(i)}}_{\overset{(3.59)}{=} f^T \alpha} + \sum_{i \in \mathcal{F}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{g}_{(i)} \;\Bigg|$$
$$\underbrace{\sum_{i \in \mathcal{G}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{A}_{(\cdot,i)}}_{\overset{(3.59)}{=} M\alpha} + \sum_{i \in \mathcal{N}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{A}_{(\cdot,i)} = -\overline{b},\ \alpha_k \in [-1,1] \Bigg\}$$

$$= \Bigg\{ \alpha^T Q^- \alpha + f^T \alpha + \underbrace{\overline{c} + \alpha^T Q^+ \alpha + \sum_{i \in \mathcal{F}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{g}_{(i)}}_{\overset{(3.61)}{\subseteq} \{c_z + g_z^T \beta \;|\; \beta \in [-\mathbf{1}, \mathbf{1}]\}} \;\Bigg|$$

$$M\alpha + \underbrace{\sum_{i\in\mathcal{N}} \left(\prod_{k=1}^{p} \alpha_k^{\overline{E}_{(k,i)}} \right) \overline{A}_{(\cdot,i)}}_{\overset{(3.61)}{\subseteq} \{A_z\beta \mid \beta\in[-\mathbf{1},\mathbf{1}]\}} = -\overline{b},\ \alpha_k \in [-1,1] \Bigg\}$$

$$\subseteq \underbrace{\left\{ c_z + \alpha^T Q^- \alpha + f^T\alpha + g_z^T\beta \mid M\alpha + A_z\beta = -b_z,\ \alpha,\beta\in[-\mathbf{1},\mathbf{1}] \right\}}_{\mathcal{S}}, \tag{3.62}$$

where $\alpha = [\alpha_1\ \dots\ \alpha_p]^T$. We omitted the `rescale` and `compact` operations in (3.62) since they only change the representation of the set, but not the set itself. Inserting the relation $d \otimes \mathcal{CPZ} \subseteq \mathcal{S}$ from (3.62) into the definition of the support function in Def. 2.2.7 finally yields

$$s_{\mathcal{CPZ}}(d) \overset{\text{Def. 2.2.7}}{=} \max_{x\in\mathcal{CPZ}} d^T x = \max_{y\in d\otimes\mathcal{CPZ}} y \overset{d\otimes\mathcal{CPZ}\subseteq\mathcal{S}}{\leq} \max_{y\in\mathcal{S}}\ y$$

$$\overset{(3.62)}{=} c_z + \max_{\substack{\alpha,\beta\in[-\mathbf{1},\mathbf{1}]\\ M\alpha+A_z\beta=-b_z}} \alpha^T Q^-\alpha + f^T\alpha + g_z^T\beta = c_z - \min_{\substack{\alpha,\beta\in[-\mathbf{1},\mathbf{1}]\\ M\alpha+A_z\beta=-b_z}} -\alpha^T Q^-\alpha - f^T\alpha - g_z^T\beta$$

$$\overset{\text{Def. 2.4.7}}{=} c_z - \texttt{quadProg}\left(\begin{bmatrix} -Q^- & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \begin{bmatrix} -f \\ -g_z \end{bmatrix}, \begin{bmatrix} I_{p+p_z} \\ -I_{p+p_z} \end{bmatrix}, \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}, \begin{bmatrix} M & A_z \end{bmatrix}, -b_z \right),$$

which concludes the proof.

Complexity: Let us begin with the quadratic program. Since the program has $p + p_z \leq p + 3\overline{h} \leq p + 3h + 3q$ variables, solving the quadratic program has complexity $\mathcal{O}((p+3h+3q)^5)$ according to Tab. 2.1, which is $\mathcal{O}(n^5)$ using Assumption 3.2.3. Next, we consider the computations in (3.58). Computation of the linear map $d \otimes \mathcal{CPZ}$ using Prop. 3.2.16 has complexity $\mathcal{O}(wn^2)$, which is $\mathcal{O}(n^2)$ for our case since $w = 1$. In addition, the computational complexity for the `rescale` operation is $\mathcal{O}(n^4\log(n)) + \mathcal{O}(\texttt{contract})$ according to Prop. 3.2.7. According to Prop. 3.1.7, the `compact` operation for SPZs has complexity $\mathcal{O}(h^+(n^+ + p^+\log(h^+)))$, where $n^+ = m+1$, $p^+ = p$, and $h^+ = h+q$ denote the dimension, the number of dependent factors, and the number of dependent generators of the lifted polynomial zonotope $\mathcal{PZ}^+$. The overall complexity for the computations in (3.58) is therefore

$$\mathcal{O}(n^2) + \mathcal{O}(n^4\log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(h^+(n^+ + p^+\log(h^+))) \overset{\substack{n^+=m+1,\ p^+=p\\ h^+=h+q}}{=}$$

$$\mathcal{O}(n^4\log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}((h+q)(m+1+p\log(h+q))) \overset{\text{Assumption 3.2.3}}{=}$$

$$\mathcal{O}(n^4\log(n)) + \mathcal{O}(\texttt{contract}).$$

Moreover, construction of the matrices Q and M and the vector f in (3.59) has complexity $\mathcal{O}(p\overline{h})$. Since $\overline{h} \leq h+q$ we have $\mathcal{O}(p\overline{h}) \leq \mathcal{O}(p(h+q))$, which results in a worst-case

complexity of $\mathcal{O}(n^2)$ using Assumption 3.2.3. The eigenvalue decomposition of the matrix Q in (3.60) has complexity $\mathcal{O}(p^3)$ according to Tab. 2.1. Since $\mathcal{O}(p^3)$ is also the complexity for the matrix multiplications required to construct Q^+ and Q^-, we obtain an overall complexity of $3 \cdot \mathcal{O}(p^3) = \mathcal{O}(p^3)$, which is $\mathcal{O}(n^3)$ using Assumption 3.2.3. Finally, we consider the computation of the zonotope in (3.61). Computation of the quadratic map $sq(\{Q^+\}, \mathcal{PZ})$ using Prop. 3.1.30 has complexity $\mathcal{O}(n^3(w + \log(n)))$, which is $\mathcal{O}(n^3 \log(n))$ in our case since $w = 1$. Furthermore, according to Prop. 3.1.7, the `compact` operation for SPZs has complexity $\mathcal{O}(\widehat{h}^+(\widehat{n}^+ + \widehat{p}^+ \log(\widehat{h}^+)))$ and the complexity for computing a zonotope enclosure of a SPZ using Prop. 3.1.14 is $\mathcal{O}(\widehat{h}^+(\widehat{p}^+ + \widehat{n}^+))$ according to (3.5), where $\widehat{n}^+ = m + 1$, $\widehat{p}^+ = p$, and $\widehat{h}^+ \leq 3\overline{h} \leq 3h + 3q$ denote the dimension, the number of dependent factors, and the number of dependent generators of the lifted polynomial zonotope $\widehat{\mathcal{PZ}}^+$. The overall complexity for computing the zonotope in (3.61) is therefore

$$\begin{aligned}
&\mathcal{O}(n^3 \log(n)) + \mathcal{O}(\widehat{h}^+(\widehat{n}^+ + \widehat{p}^+ \log(\widehat{h}^+)) + \mathcal{O}(\widehat{h}^+(\widehat{p}^+ + \widehat{n}^+)) \\
&= \mathcal{O}(n^3 \log(n))) + \mathcal{O}(\widehat{h}^+(\widehat{n}^+ + \widehat{p}^+ \log(\widehat{h}^+))) \overset{\substack{\widehat{n}^+=m+1,\ \widehat{p}^+=p \\ \widehat{h}^+\leq 3h+3q}}{=} \\
&= \mathcal{O}(n^3 \log(n)) + \mathcal{O}((3h + 3q)(m + 1 + p \log(3h + 3q))) \overset{\text{Assumption } 3.2.3}{=} \mathcal{O}(n^3 \log(n)).
\end{aligned}$$

Combining all the complexities from the single parts of the support function computation finally yields

$$\begin{aligned}
\mathcal{O}(n^5) + \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(n^2) + \mathcal{O}(n^3) + \mathcal{O}(n^3 \log(n)) \\
= \mathcal{O}(n^5) + \mathcal{O}(\texttt{contract})
\end{aligned}$$

for the overall complexity. □

The computational complexity of Prop. 3.2.15 for different contractors is summarized in Tab. 3.8. As described for SPZs in Sec. 3.1.4, a template polyhedron or an interval enclosing a CPZ can be easily computed by evaluating the support function for a set of discrete directions. The enclosing interval for the CPZ in (3.57) is shown in Fig. 3.11. To improve the tightness of the support function enclosure, one can for example iteratively split the CPZ multiple times using the `split` operation as defined in Prop. 3.2.35.

3.2.5 Basic Set Operations

In this section, we derive closed-form expressions for all basic set operations introduced in Sec. 2.1. We begin with the linear map:

Proposition 3.2.16. *(Linear Map) Given a CPZ $\mathcal{CPZ} = \langle c, G, E, A, b, R \rangle_{CPZ} \subset \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{w \times n}$, the linear map is*

$$M \otimes \mathcal{CPZ} = \left\langle Mc, MG, E, A, b, R \right\rangle_{CPZ},$$

which has complexity $\mathcal{O}(wn^2)$ with respect to the dimension n, where w is the number of rows of matrix M. The resulting CPZ is regular if $\mathcal{CPZ}$ is regular.

Proof. The result follows directly from inserting the definition of CPZs in Def. 3.2.1 into the definition of the linear map in (2.1):

$$M \otimes \mathcal{CPZ} \overset{(2.1)}{=} \{Ms \mid s \in \mathcal{CPZ}\} \overset{\text{Def. 3.2.1}}{=}$$

$$\left\{ Mc + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) MG_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [-1,1] \right\}$$

$$= \langle Mc, MG, E, A, b, R \rangle_{CPZ}.$$

Complexity: The two matrix multiplications Mc and MG have complexity $\mathcal{O}(wn) + \mathcal{O}(wnh) = \mathcal{O}(wnh)$ according to Tab. 2.1, which is $\mathcal{O}(wn^2)$ using Assumption 3.2.3. □

Next, we consider the Minkowski sum of two CPZs:

Proposition 3.2.17. *(Minkowski Sum) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$, their Minkowski sum is*

$$\mathcal{CPZ}_1 \oplus \mathcal{CPZ}_2 = \left\langle c_1 + c_2, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix} \right\rangle_{CPZ},$$

which has complexity $\mathcal{O}(n)$ with respect to the dimension n. The resulting CPZ is regular if $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$ are regular.

Proof. The result is obtained by inserting the definition of CPZs in Def. 3.2.1 into the definition of the Minkowski sum in (2.2):

$$\mathcal{CPZ}_1 \oplus \mathcal{CPZ}_2 \overset{(2.2)}{=} \left\{ s_1 + s_2 \;\middle|\; s_1 \in \mathcal{CPZ}_1,\ s_2 \in \mathcal{CPZ}_2 \right\} \overset{\text{Def. 3.2.1}}{=}$$

$$\left\{ c_1 + c_2 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} \;\middle|\; \right.$$
$$\left. \sum_{i=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_k^{R_{1(k,i)}} \right) A_{1(\cdot,i)} = b_1,\ \sum_{i=1}^{q_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{R_{2(k,i)}} \right) A_{2(\cdot,i)} = b_2,\ \alpha_k, \alpha_{p_1+k} \in [-1,1] \right\}$$

$$\overset{(3.45),(3.46)}{=} \left\langle c_1 + c_2, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix} \right\rangle_{CPZ},$$

where we used the identities in (3.45) and (3.46).

Complexity: The computation of the new constant offset $c_1 + c_2$ has complexity $\mathcal{O}(n)$. Since all other operations required for the construction of the resulting CPZ are concatenations, it holds that this is also the overall complexity. □

As for SPZs, two CPZ with identical factors can be added in a dependency-preserving way by using the exact addition rather than the Minkowski sum:

Proposition 3.2.18. *(Exact Addition) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$ with $p_1 = p_2$, their exact addition is defined as*

$$\mathcal{CPZ}_1 \boxplus \mathcal{CPZ}_2 = \left\langle c, \begin{bmatrix} G_1 & G_2 \end{bmatrix}, \begin{bmatrix} E_1 & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & R_2 \end{bmatrix} \right\rangle_{CPZ},$$

which has complexity $\mathcal{O}(n^2 \log(n))$ with respect to the dimension n. The `compactGen` *and* `compactCon` *operations as defined in Prop. 3.2.5 and Prop. 3.2.6 are applied to make the resulting CPZ regular.*

Proof. The result is similar to the result for the Minkowski sum in Prop. 3.2.17, with the difference that we exploit the equality of the factors for the two CPZs to preserve dependencies.

Complexity: The `compactGen` operation has complexity $\mathcal{O}(h(n + p\log(h)))$ according to Prop. 3.2.5 and the `compactCon` operation has complexity $\mathcal{O}(q(m + p\log(q)))$ according to Prop. 3.2.6, where $p = p_1 = p_2$, $h = h_1 + h_2$, $m = m_1 + m_2$, $q = q_1 + q_2$ denote the number of factors, the number of generators, the number of constraints, and the number of constraint generators of the resulting CPZ. The overall complexity is therefore

$$\mathcal{O}(h(n + p\log(h))) + \mathcal{O}(q(m + p\log(q))) \overset{\substack{p=p_1=p_2,\ h=h_1+h_2 \\ m=m_1+m_2,\ q=q_1+q_2}}{=}$$

$$\mathcal{O}((h_1 + h_2)(n + p_1 \log(h_1 + h_2))) + \mathcal{O}((q_1 + q_2)(m_1 + m_2 + p_1 \log(q_1 + q_2))),$$

which is $\mathcal{O}(n^2 \log(n))$ using Assumption 3.2.3. □

Let us now present a closed-form expression for the Cartesian product:

Proposition 3.2.19. *(Cartesian Product) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^w$, their Cartesian product is*

$$\mathcal{CPZ}_1 \times \mathcal{CPZ}_2 = \left\langle \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \begin{bmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{bmatrix}, \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix} \right\rangle_{CPZ},$$

which has complexity $\mathcal{O}(1)$. The resulting CPZ is regular if $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$ are regular.

Proof. The result is obtained by inserting the definition of CPZs in Def. 3.2.1 into the definition of the Cartesian product in (2.4):

$$\mathcal{CPZ}_1 \times \mathcal{CPZ}_2 \overset{(2.4)}{=} \left\{ [s_1^T \ s_2^T]^T \;\middle|\; s_1 \in \mathcal{CPZ}_1.\ s_2 \in \mathcal{CPZ}_2 \right\} \overset{\text{Def. } 3.2.1}{=}$$

$$\left\{ \begin{bmatrix} c_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ c_2 \end{bmatrix} + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) \begin{bmatrix} G_{1(\cdot,i)} \\ \mathbf{0} \end{bmatrix} + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) \begin{bmatrix} \mathbf{0} \\ G_{2(\cdot,i)} \end{bmatrix} \;\middle|\; \right.$$

$$\sum_{i=1}^{q_1}\left(\prod_{k=1}^{p_1}\alpha_k^{R_{1(k,i)}}\right)A_{1(\cdot,i)}=b_1,\ \sum_{i=1}^{q_2}\left(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{R_{2(k,i)}}\right)A_{2(\cdot,i)}=b_2,\ \alpha_k,\alpha_{p_1+k}\in[-1,1]\Bigg\}$$

$$\overset{(3.45),(3.46)}{=}\left\langle\begin{bmatrix}c_1\\c_2\end{bmatrix}\begin{bmatrix}G_1&\mathbf{0}\\\mathbf{0}&G_2\end{bmatrix},\begin{bmatrix}E_1&\mathbf{0}\\\mathbf{0}&E_2\end{bmatrix},\begin{bmatrix}A_1&\mathbf{0}\\\mathbf{0}&A_2\end{bmatrix},\begin{bmatrix}b_1\\b_2\end{bmatrix},\begin{bmatrix}R_1&\mathbf{0}\\\mathbf{0}&R_2\end{bmatrix}\right\rangle_{CPZ},$$

where we used the identities in (3.45) and(3.46).

Complexity: The construction of the resulting CPZ only involves concatenations and therefore has complexity $\mathcal{O}(1)$. □

Before we consider the convex hull, we first derive a closed-form expression for the linear combination of two CPZs, since we can reuse this result for the convex hull:

Proposition 3.2.20. *(Linear Combination) Given two CPZs, $\mathcal{CPZ}_1=\langle c_1,G_1,E_1,A_1,b_1,R_1\rangle_{CPZ}\subset\mathbb{R}^n$ and $\mathcal{CPZ}_2=\langle c_2,G_2,E_2,A_2,b_2,R_2\rangle_{CPZ}\subset\mathbb{R}^n$, their linear combination is*

$$comb(\mathcal{CPZ}_1,\mathcal{CPZ}_2)=\Bigg\langle\frac{1}{2}(c_1+c_2),\frac{1}{2}\begin{bmatrix}(c_1-c_2)&G_1&G_1&G_2&-G_2\end{bmatrix},\begin{bmatrix}\mathbf{0}&E_1&E_1&\mathbf{0}&\mathbf{0}\\\mathbf{0}&\mathbf{0}&\mathbf{0}&E_2&E_2\\1&\mathbf{0}&\mathbf{1}&\mathbf{0}&\mathbf{1}\end{bmatrix},\begin{bmatrix}A_1&\mathbf{0}\\\mathbf{0}&A_2\end{bmatrix},\begin{bmatrix}b_1\\b_2\end{bmatrix},\begin{bmatrix}R_1&\mathbf{0}\\\mathbf{0}&R_2\end{bmatrix}\Bigg\rangle_{CPZ},$$

which has complexity $\mathcal{O}(n^2)$ with respect to the dimension n. The resulting CPZ is regular if $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$ are regular.

Proof. The result is obtained by inserting the definition of CPZs in Def. (3.2.1) into the definition of the linear combination in (2.11):

$$comb(\mathcal{CPZ}_1,\mathcal{CPZ}_2)\overset{(2.11)}{=}$$

$$\left\{\frac{1}{2}(1+\lambda)s_1+\frac{1}{2}(1-\lambda)s_2\ \middle|\ s_1\in\mathcal{CPZ}_1,\ s_2\in\mathcal{CPZ}_2,\ \lambda\in[-1,1]\right\}\overset{\text{Def. 3.2.1}}{=}$$

$$\Bigg\{\frac{1}{2}(c_1+c_2)+\frac{1}{2}(c_1-c_2)\lambda+\frac{1}{2}\sum_{i=1}^{h_1}\left(\prod_{k=1}^{p_1}\alpha_k^{E_{1(k,i)}}\right)G_{1(\cdot,i)}+$$
$$\frac{1}{2}\sum_{i=1}^{h_1}\lambda\left(\prod_{k=1}^{p_1}\alpha_k^{E_{1(k,i)}}\right)G_{1(\cdot,i)}+\frac{1}{2}\sum_{i=1}^{h_2}\left(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{E_{2(k,i)}}\right)G_{2(\cdot,i)}-\frac{1}{2}\sum_{i=1}^{h_2}\lambda\left(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{E_{2(k,i)}}\right)G_{2(\cdot,i)}\ \Bigg|$$
$$\sum_{i=1}^{q_1}\left(\prod_{k=1}^{p_1}\alpha_k^{R_{1(k,i)}}\right)A_{1(\cdot,i)}=b_1,\ \sum_{i=1}^{q_2}\left(\prod_{k=1}^{p_2}\alpha_{p_1+k}^{R_{2(k,i)}}\right)A_{2(\cdot,i)}=b_2,\ \alpha_k,\alpha_{p_1+k},\lambda\in[-1,1]\Bigg\}$$

$$\overset{\substack{(3.45),(3.46)\\ \alpha_{p_1+p_2+1}:=\lambda}}{=}\left\langle\frac{1}{2}(c_1+c_2),\frac{1}{2}\begin{bmatrix}(c_1-c_2)&G_1&G_1&G_2&-G_2\end{bmatrix},\right.$$

$$\left\langle \begin{bmatrix} \mathbf{0} & E_1 & E_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & E_2 & E_2 \\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix} \right\rangle_{CPZ},$$

where we used the identities in (3.45) and(3.46). For the transformation in the last line, we substituted λ with an additional factor $\alpha_{p_1+p_2+1}$. Since $\lambda \in [-1, 1]$ and $\alpha_{p_1+p_2+1} \in [-1, 1]$, the substitution does not change the set.

Complexity: Construction of the constant offset $c = 0.5(c_1 + c_2)$ requires n additions and n multiplications. Moreover, construction of the generator matrix $0.5\begin{bmatrix}(c_1 - c_2) & G_1 & G_1 & G_2 & -G_2\end{bmatrix}$ requires n subtractions and $n(2h_1 + 2h_2 + 1)$ multiplications. This results in an overall complexity of

$$\mathcal{O}(2n) + \mathcal{O}(n(2h_1 + 2h_2 + 2)) = \mathcal{O}(n(h_1 + h_2)),$$

which is $\mathcal{O}(n^2)$ using Assumption 3.2.3. □

The convex hull of two CPZs can be computed as follows:

Proposition 3.2.21. *(Convex Hull) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1\rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2\rangle_{CPZ} \subset \mathbb{R}^n$, their convex hull is*

$$conv(\mathcal{CPZ}_1, \mathcal{CPZ}_2) = \left\langle a \cdot c, \begin{bmatrix}\overline{c} & \overline{G} & \overline{G}\end{bmatrix}, \begin{bmatrix} \mathbf{0} & \overline{E} & \overline{E} \\ I_a & \mathbf{0} & \widehat{E} \end{bmatrix}, \begin{bmatrix} \overline{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \begin{bmatrix} \overline{b} \\ -n \end{bmatrix}, \begin{bmatrix} \overline{R} & \mathbf{0} \\ \mathbf{0} & I_a \end{bmatrix} \right\rangle_{CPZ}$$

with

$$\langle c, G, E, A, b, R\rangle_{CPZ} = comb(\mathcal{CPZ}_1, \mathcal{CPZ}_2),\ a = n + 1,\ \overline{c} = \begin{bmatrix} c & \dots & c \end{bmatrix} \in \mathbb{R}^{n\times a},$$

$$\overline{G} = \begin{bmatrix} G & \dots & G \end{bmatrix} \in \mathbb{R}^{n\times ah},\ \overline{E} = \begin{bmatrix} E & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & E \end{bmatrix} \in \mathbb{R}^{ap\times ah},\ \widehat{E} = \begin{bmatrix} \mathbf{1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{1} \end{bmatrix} \in \mathbb{R}^{a\times ah},$$

$$\overline{A} = \begin{bmatrix} A & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & A \end{bmatrix} \in \mathbb{R}^{am\times aq},\ \overline{b} = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \in \mathbb{R}^{am},\ \overline{R} = \begin{bmatrix} R & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & R \end{bmatrix} \in \mathbb{R}^{ap\times aq}, \tag{3.63}$$

where the linear combination $comb(\mathcal{CPZ}_1, \mathcal{CPZ}_2)$ is calculated using Prop. 3.2.20, and the scalars p, h, q, and m denote the number of factors, the number of generators, the number of constraint generators, and the number of constraints of the CPZ $\langle c, G, E, A, b, R\rangle_{CPZ}$. The computational complexity with respect to the dimension n is $\mathcal{O}(n^2)$, and the resulting CPZ is regular if $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$ are regular.

Proof. According to the definition of the convex hull in (2.5), the definition of the union in (2.8), and the definition of the linear combination in (2.11), it holds that

$$\mathcal{CPZ}_1 \cup \mathcal{CPZ}_2 \subseteq comb(\mathcal{CPZ}_1, \mathcal{CPZ}_2) \subseteq conv(\mathcal{CPZ}_1, \mathcal{CPZ}_2). \tag{3.64}$$

The relation in (3.64) allows us to substitute the union in the definition of the convex hull in (2.5) with the linear combination. This yields a resulting CPZ with less factors than if we would have used the union according to Prop. 3.2.25, which is often advantageous:

$$conv(\mathcal{CPZ}_1, \mathcal{CPZ}_2) \overset{(2.5)}{=} \left\{ \sum_{j=1}^{n+1} \lambda_j \, s_j \;\middle|\; s_j \in \mathcal{CPZ}_1 \cup \mathcal{CPZ}_2,\ \lambda_j \geq 0,\ \sum_{j=1}^{n+1} \lambda_j = 1 \right\} \overset{(3.64)}{=}$$

$$\left\{ \sum_{j=1}^{n+1} (1 + \widehat{\lambda}_j) \, s_j \;\middle|\; s_j \in comb(\mathcal{CPZ}_1, \mathcal{CPZ}_2),\ \sum_{j=1}^{n+1} (1 + \widehat{\lambda}_j) = 1,\ \widehat{\lambda}_j \in [-1, 1] \right\} \overset{\text{Def. 3.2.1, (3.63)}}{=}$$

$$\left\{ \sum_{j=1}^{n+1} (1 + \widehat{\lambda}_j) \left(c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(j-1)p+k}^{E_{(k,i)}} \right) G_{(\cdot,i)} \right) \;\middle|\; \alpha_{(j-1)p+k}, \widehat{\lambda}_j \in [-1, 1], \right.$$
$$\left. \sum_{j=1}^{n+1} (1 + \widehat{\lambda}_j) = 1,\ \forall j \in \{1, \ldots, n+1\} :\ \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_{(j-1)p+k}^{R_{(k,i)}} \right) A_{(\cdot,i)} = b \right\} =$$

$$\left\{ (n+1)c + \sum_{j=1}^{n+1} \widehat{\lambda}_j \, c + \sum_{j=1}^{n+1} \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(j-1)p+k}^{E_{(k,i)}} \right) G_{(\cdot,i)} \right.$$
$$+ \sum_{j=1}^{n+1} \sum_{i=1}^{h} \widehat{\lambda}_j \left(\prod_{k=1}^{p} \alpha_{(j-1)p+k}^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_{(j-1)p+k}, \widehat{\lambda}_j \in [-1, 1],$$
$$\left. \forall j \in \{1, \ldots, n+1\} :\ \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_{(j-1)p+k}^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \sum_{j=1}^{n+1} \widehat{\lambda}_j = -n \right\}$$

$$\overset{\substack{(3.45),(3.46) \\ \alpha_{ap+j} := \widehat{\lambda}_j}}{=} \left\langle a \cdot c, \begin{bmatrix} \overline{c} & \overline{G} & \overline{G} \end{bmatrix}, \begin{bmatrix} \mathbf{0} & \overline{E} & \overline{E} \\ I_a & \mathbf{0} & \widehat{E} \end{bmatrix}, \begin{bmatrix} \overline{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \begin{bmatrix} \overline{b} \\ -n \end{bmatrix}, \begin{bmatrix} \overline{R} & \mathbf{0} \\ \mathbf{0} & I_a \end{bmatrix} \right\rangle_{CPZ},$$

where we used the identities in (3.45) and (3.46). For the transformation in the last line we substituted the scalars $\widehat{\lambda}_j$ by additional factors α_{ap+j}. Since $\widehat{\lambda}_j \in [-1, 1]$ and $\alpha_{ap+j} \in [-1, 1]$, the substitution does not change the set.

Complexity: The calculation of the linear combination $comb(\mathcal{CPZ}_1, \mathcal{CPZ}_2)$ has complexity $\mathcal{O}(n^2)$ according to Prop. 3.2.20. Moreover, the construction of the constant offset $a \cdot c$ requires n multiplications and therefore has complexity $\mathcal{O}(n)$. Since all other operations that are required are initializations and concatenations which have constant complexity $\mathcal{O}(1)$, the overall complexity for the computation of the convex hull is $\mathcal{O}(n^2) + \mathcal{O}(n) + \mathcal{O}(1) = \mathcal{O}(n^2)$. □

For the convex hull $conv(\mathcal{CPZ}) = conv(\mathcal{CPZ}, \mathcal{CPZ})$ of a single CPZ $\mathcal{CPZ}$, we can exploit that $\mathcal{CPZ} \cup \mathcal{CPZ} = \mathcal{CPZ}$ holds to obtain a more compact representation. Next, we demonstrate how to compute the quadratic map for CPZs. As shown in Appendix A, all higher-order polynomial maps can be computed using a sequence of quadratic maps, so

that all polynomial maps of CPZs can be calculated based on the quadratic map. Unlike for SPZs, we defined CPZs without unique identifiers to simplify the notation. Due to this, possible dependencies between different CPZs are not preserved, so that in general $sq(\mathcal{Q}, \mathcal{CPZ}) \neq sq(\mathcal{Q}, \mathcal{CPZ}, \mathcal{CPZ})$ for a CPZ $\mathcal{CPZ}$. Therefore, we provide separate formulas for computing the quadratic map $sq(\mathcal{Q}, \mathcal{CPZ})$ of a single CPZ and for computing the mixed quadratic map $sq(\mathcal{Q}, \mathcal{CPZ}, \mathcal{CPZ})$ for two CPZs $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$:

Proposition 3.2.22. *(Quadratic Map) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$, and a discrete set of matrices $\mathcal{Q} = \{Q_1, \ldots, Q_w\}$ with $Q_i \in \mathbb{R}^{n \times n}, i = 1, \ldots, w$, the quadratic map of a single CPZ is*

$$sq(\mathcal{Q}, \mathcal{CPZ}_1) = \left\langle \overline{c}, \begin{bmatrix} \widehat{G} & \overline{G}_1 & \ldots & \overline{G}_{h_1} \end{bmatrix}, \begin{bmatrix} E & \overline{E}_1 & \ldots & \overline{E}_{h_1} \end{bmatrix}, A_1, b_1, R_1 \right\rangle_{CPZ}, \tag{3.65}$$

where

$$\begin{aligned} \overline{c} &= \begin{bmatrix} c_1^T Q_1 c_1 \\ \vdots \\ c_1^T Q_w c_1 \end{bmatrix}, \quad \widehat{G} = \underbrace{\begin{bmatrix} c_1^T Q_1 G_1 \\ \vdots \\ c_1^T Q_w G_1 \end{bmatrix}}_{\widehat{G}_1} + \underbrace{\begin{bmatrix} c_1^T Q_1^T G_1 \\ \vdots \\ c_1^T Q_w^T G_1 \end{bmatrix}}_{\widehat{G}_2}, \\ \overline{E}_j &= E_1 + E_{1(\cdot,j)} \cdot \mathbf{1}, \quad \overline{G}_j = \begin{bmatrix} G_{1(\cdot,j)}^T Q_1 G_1 \\ \vdots \\ G_{1(\cdot,j)}^T Q_w G_1 \end{bmatrix}, \ j = 1, \ldots, h_1, \end{aligned} \tag{3.66}$$

and the quadratic map of two different CPZs is

$$\begin{aligned} sq(\mathcal{Q}, \mathcal{CPZ}_1, \mathcal{CPZ}_2) = \Bigg\langle \breve{c}, \begin{bmatrix} \widehat{G}_3 & \widehat{G}_4 & \breve{G}_1 & \ldots & \breve{G}_{h_1} \end{bmatrix}, \\ \begin{bmatrix} \mathbf{0} & E_1 & E_{(\cdot,1)} \cdot \mathbf{1} & \ldots & E_{(\cdot,h_1)} \cdot \mathbf{1} \\ E_2 & \mathbf{0} & E_2 & \ldots & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix} \Bigg\rangle_{CPZ}, \end{aligned} \tag{3.67}$$

where

$$\breve{c} = \begin{bmatrix} c_1^T Q_1 c_2 \\ \vdots \\ c^T Q_w c \end{bmatrix}, \ \widehat{G}_3 = \begin{bmatrix} c_1^T Q_1 G_2 \\ \vdots \\ c_1^T Q_w G_2 \end{bmatrix}, \ \widehat{G}_4 = \begin{bmatrix} c_2^T Q_1^T G_1 \\ \vdots \\ c_2^T Q_w^T G_1 \end{bmatrix},$$

$$\breve{G}_j = \begin{bmatrix} G_{1(\cdot,j)}^T Q_1 G_2 \\ \vdots \\ G_{1(\cdot,j)}^T Q_w G_2 \end{bmatrix}, \quad j = 1, \ldots, h_1.$$

The `compactGen` *operation as defined in Prop. 3.2.5 make the resulting CPZs regular. The computational complexity of* (3.65) *and* (3.67) *with respect to the dimension n is $\mathcal{O}(n^3(w + \log(n)))$, where w is the dimension of the resulting CPZs.*

Proof. The result in (3.65) is obtained by inserting the definition of CPZs in Def. 3.2.1 into the definition of the quadratic map in (2.6), which yields

$$sq(\mathcal{Q}, \mathcal{CPZ}_1) \overset{(2.6)}{=} \{x \mid x_{(i)} = s^T Q_i s,\ s \in \mathcal{CPZ}_1,\ i = 1, \dots, w\} \overset{\text{Def. } 3.2.1}{=}$$

$$\left\{ x \;\middle|\; x_{(i)} = \left(c_1 + \sum_{j=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,j)}} \right) G_{1(\cdot,j)} \right)^T Q_i \left(c_1 + \sum_{l=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,l)}} \right) G_{1(\cdot,l)} \right), \right.$$
$$\left. \sum_{j=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_k^{R_{1(k,j)}} \right) A_{1(\cdot,j)} = b_1,\ i = 1, \dots, w,\ \alpha_k \in [-1, 1] \right\} =$$

$$\left\{ x \;\middle|\; x_{(i)} = \underbrace{c_1^T Q_i c_1}_{\overset{(3.66)}{=} \overline{c}_{(i)}} + \sum_{l=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,l)}} \right) \underbrace{c_1^T Q_i G_{1(\cdot,l)}}_{\overset{(3.66)}{=} \widehat{G}_{1(i,l)}} + \sum_{j=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,j)}} \right) \underbrace{G_{1(\cdot,j)}^T Q_i c_1}_{\overset{(3.66)}{=} \widehat{G}_{2(i,j)}} + \right.$$
$$\sum_{j=1}^{h_1} \sum_{l=1}^{h_1} \left(\prod_{k=1}^{p_1} \underbrace{\alpha_k^{E_{1(k,j)} + E_{1(k,l)}}}_{\overset{(3.66)}{=} \alpha_k^{\overline{E}_{j(k,l)}}} \right) \underbrace{G_{1(\cdot,j)}^T Q_i G_{1(\cdot,l)}}_{\overset{(3.66)}{=} \overline{G}_{j(i,l)}},$$
$$\left. \sum_{j=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_k^{R_{1(k,j)}} \right) A_{1(\cdot,j)} = b_1,\ i = 1, \dots, w,\ \alpha_k \in [-1, 1] \right\} =$$

$$\left\langle \overline{c}, \big[\underbrace{\widehat{G}_1 + \widehat{G}_2}_{\widehat{G}} \ \overline{G}_1 \ \dots \ \overline{G}_{h_1} \big], \big[E \ \ \overline{E}_1 \ \ \dots \ \ \overline{E}_{h_1} \big], A_1, b_1, R_1 \right\rangle_{CPZ}.$$

Note that only the generator matrix, but not the exponent matrix, is different for each dimension $x_{(i)}$. The proof for the mixed quadratic map in (3.67) is very similar to the proof for the quadratic map with a single CPZ shown above and therefore omitted at this point.

Complexity: We begin with the derivation of the computational complexity of the quadratic map with a single CPZ in (3.65). The construction of the constant offset $\overline{c}$ in (3.66) has complexity $\mathcal{O}(wn^2)$ and the construction of the matrix $\widehat{G}$ in (3.66) has complexity $\mathcal{O}(n^2 h_1 w)$. Moreover, construction of the matrices $\overline{E}_j$ in (3.66) has complexity $\mathcal{O}(h_1^2 p_1)$, and construction of the matrices $\overline{G}_j$ in (3.66) has complexity $\mathcal{O}(n^2 h_1 w) + \mathcal{O}(n h_1^2 w)$ if the results for $Q_i G_1$ are stored and reused. Subsequent application of the `compactGen` operation has complexity $\mathcal{O}(\overline{h}(\overline{n} + \overline{p}\log(\overline{h}))$ according to Prop. 3.2.5, where $\overline{n} = w$, $\overline{p} = p_1$, and $\overline{h} = h_1^2 + h_1$ denote the dimension, the number of factors, and the number of generators of the resulting CPZ in (3.65). The resulting overall complexity is therefore

$$\mathcal{O}(wn^2) + \mathcal{O}(n^2 h_1 w) + \mathcal{O}(h_1^2 p_1) + \mathcal{O}(n^2 h_1 w) + \mathcal{O}(n h_1^2 w) + \mathcal{O}(\overline{h}(\overline{n} + \overline{p}\log(\overline{h})))$$
$$\overset{\substack{\overline{n}=w,\ \overline{p}=p_1 \\ \overline{h}=h_1^2+h_1}}{=} \mathcal{O}(n h_1 w(n + h_1)) + \mathcal{O}(h_1^2 p_1) + \mathcal{O}((h_1^2 + h_1)(w + p_1 \log(h_1^2 + h_1))),$$

which is $\mathcal{O}(n^3(w + \log(n)))$ using Assumption 3.2.3. Due to the similarities between the computation of the quadratic map for a single CPZ in (3.65) and the mixed quadratic map in (3.67), the complexity for the mixed quadratic map is $\mathcal{O}(n^3(w + \log(n)))$, too. □

Contrary to SPZs, CPZs are also closed under intersection:

Proposition 3.2.23. *(Intersection) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$, their intersection is*

$$\mathcal{CPZ}_1 \cap \mathcal{CPZ}_2 = \left\langle c_1, G_1, \begin{bmatrix} E_1 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & G_1 & -G_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ c_2 - c_1 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} & E_1 & \mathbf{0} \\ \mathbf{0} & R_2 & \mathbf{0} & E_2 \end{bmatrix} \right\rangle_{CPZ},$$

which has complexity $\mathcal{O}(n^2 \log(n))$ with respect to the dimension n. The `compactCon` *operation as defined in Prop. 3.2.6 is applied to make the resulting CPZ regular.*

Proof. The outline of the proof is inspired by [32, Prop. 1]. We compute the intersection by restricting the factors α_k of $\mathcal{CPZ}_1$ to values that belong to points that are located inside $\mathcal{CPZ}_2$, which is identical to adding the equality constraint

$$\underbrace{c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)}}_{x \in \mathcal{CPZ}_1} = \underbrace{c_2 + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)}}_{x \in \mathcal{CPZ}_2}$$

to $\mathcal{CPZ}_1$:

$$\begin{aligned} \mathcal{CPZ}_1 \cap \mathcal{CPZ}_2 = \Bigg\{ & c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} \;\Bigg|\; \sum_{i=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_k^{R_{1(k,i)}} \right) A_{1(\cdot,i)} = b_1, \;, \\ & \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} - \sum_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} = c_2 - c_1, \\ & \sum_{i=1}^{q_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{R_{2(k,i)}} \right) A_{2(\cdot,i)} = b_2, \; \alpha_k, \alpha_{p_1+k} \in [-1, 1] \Bigg\} \end{aligned}$$

$$\overset{(3.46)}{=} \left\langle c_1, G_1, \begin{bmatrix} E_1 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & G_1 & -G_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ c_2 - c_1 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} & E_1 & \mathbf{0} \\ \mathbf{0} & R_2 & \mathbf{0} & E_2 \end{bmatrix} \right\rangle_{CPZ},$$

where we used the identity in (3.46).

Complexity: The computation of $c_2 - c_1$ has complexity $\mathcal{O}(n)$. Moreover, subsequent application of the `compactCon` operation has complexity $\mathcal{O}(q(m + p \log(q)))$ according to Prop. 3.2.6, where $p = p_1 + p_2$, $q = q_1 + q_2 + h_1 + h_2$, and $m = m_1 + m_2 + n$ denote

the number of factors, number of constraint generators, and number of constraints of the resulting CPZ. The overall complexity is therefore

$$\mathcal{O}(n) + \mathcal{O}(q(m + p\log(q))) \overset{\substack{p=p_1+p_2 \\ q=q_1+q_2+h_1+h_2 \\ m=m_1+m_2+}}{=} \tag{3.68}$$
$$\mathcal{O}(n) + \mathcal{O}((q_1 + q_2 + h_1 + h_2)(m_1 + m_2 + n + (p_1 + p_2)\log(q_1 + q_2 + h_1 + h_2))),$$

which is $\mathcal{O}(n^2 \log(n))$ using Assumption 3.2.3. □

In set-based computing one often requires intersections between different set representations. For set representations that can be converted to CPZs, this can be done using Prop. 3.2.23. Level sets, however, can in general not be equivalently represented as a CPZs. We therefore provide a separate formula for tightly enclosing the intersection between a CPZ and a level set:

Proposition 3.2.24. *(Intersection Level Set) Given a CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$ *and two level sets* $\mathcal{LS}_1 = \langle w(x), \leq\rangle_{LS} \subset \mathbb{R}^n$ *and* $\mathcal{LS}_2 = \langle w(x), =\rangle_{LS} \subset \mathbb{R}^n$ *with* $w : \mathbb{R}^n \to \mathbb{R}^o$*, the intersections* $\mathcal{CPZ} \cap \mathcal{LS}_1$ *and* $\mathcal{CPZ} \cap \mathcal{LS}_2$ *can be tightly enclosed by*

$$\mathcal{CPZ} \cap \mathcal{LS}_1 \subseteq \left\langle c, G, \begin{bmatrix} E \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & KG & \widehat{G} & \overline{G}_1 \end{bmatrix}, \begin{bmatrix} b \\ \overline{b}_1 \end{bmatrix}, \begin{bmatrix} R & E & \widehat{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_o \end{bmatrix} \right\rangle_{CPZ}$$

$$\mathcal{CPZ} \cap \mathcal{LS}_2 \subseteq \left\langle c, G, \begin{bmatrix} E \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & KG & \widehat{G} & \overline{G}_2 \end{bmatrix}, \begin{bmatrix} b \\ \overline{b}_2 \end{bmatrix}, \begin{bmatrix} R & E & \widehat{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_o \end{bmatrix} \right\rangle_{CPZ},$$

where

$$K = \nabla w(c), \quad \mathcal{Q} = \nabla^2 w(c), \quad \langle \mathbf{0}, \widehat{G}, \widehat{E}, A, b, R\rangle_{CPZ} = 0.5\ sq\big(\mathcal{Q}, \mathcal{CPZ} \oplus (-c)\big),$$

$$\mathcal{I} = \texttt{interval}(\mathcal{CPZ}), \quad \boldsymbol{\mathcal{D}} = \texttt{bound}(\nabla^3 w(x), \mathcal{I}), \quad [l_r, u_r] = \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-c)\big),$$

$$[l, u] = \texttt{bound}(w(x), \mathcal{I}), \quad \overline{G}_1 = 0.5\ \texttt{diag}(u_r - l_r - l), \quad \overline{G}_2 = 0.5\ \texttt{diag}(u_r - l_r)$$

$$\overline{b}_1 = 0.5(l - l_r - u_r) - w(c), \quad \overline{b}_2 = -0.5(l_r + u_r) - w(c).$$

The computational complexity is $\mathcal{O}(n^6) + on^3 \cdot \mathcal{O}(\texttt{bound})$*, where* n *is the dimension and* o *is the number of constraints of the level set.*

Proof. Since $[l, u] = \texttt{bound}(w(x), \mathcal{I})$ and $\mathcal{CPZ} \subseteq \mathcal{I}$ it holds according to the definition of range bounding in Def. 2.7.1 that $l \leq \min_{x \in \mathcal{CPZ}} w(x)$, which yields

$$\mathcal{CPZ} \cap \mathcal{LS}_1 \overset{(2.7)}{=} \{x \in \mathcal{CPZ} \mid w(x) \leq \mathbf{0}\} = \{x \in \mathcal{CPZ} \mid l \leq w(x) \leq \mathbf{0}\}. \tag{3.69}$$

By using a Taylor series expansion with order 2 of the function $w(x)$ around the expansion point c we can equivalently express $l \leq w(x) \leq \mathbf{0}$ by the $i = 1, \ldots, o$ constraints

$$l_{(i)} \leq w_{(i)}(c) + \underbrace{\left.\frac{\partial w_{(i)}(x)}{\partial x}\right|_c}_{K_{(i,\cdot)}} (x - c) + \frac{1}{2}(x - c)^T \underbrace{\left.\frac{\partial^2 w_{(i)}(x)}{\partial x^2}\right|_c}_{Q_i} (x - c) \oplus \underbrace{\mathcal{L}_i(x)}_{\subseteq [l_{r(i)}, u_{r(i)}]} \leq 0, \tag{3.70}$$

where the Lagrange remainder $\mathcal{L}(x) = \mathcal{L}_1(x) \times \ldots \times \mathcal{L}_o(x)$ is enclosed by

$$\forall x \in \mathcal{CPZ}: \quad \mathcal{L}(x) = \left\{ s \;\middle|\; s_{(i)} = \frac{\left((x-c)^T\nabla\right)^3 w_{(i)}(\widehat{x})}{3!},\ \widehat{x} = c + \lambda(x-c),\ \lambda \in [0,1] \right\}$$

$$\overset{\mathcal{CPZ} \subseteq \mathcal{I}}{\subseteq} \frac{1}{6}\left\{ s \;\middle|\; s_{(i)} = \left((x-c)^T\nabla\right)^3 w_{(i)}(\widehat{x}),\ x, \widehat{x} \in \mathcal{I} \right\} \overset{(2.10)}{=} \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-c)\big) = [l_r, u_r].$$

Inserting (3.70) into (3.69) finally yields

$$\mathcal{CPZ} \cap \mathcal{LS}_1 \overset{(3.69)}{=} \{x \in \mathcal{CPZ} \mid l \leq w(x) \leq \mathbf{0}\} \overset{(3.70)}{\subseteq}$$

$$\Big\{ x \in \mathcal{CPZ} \;\Big|\; \forall i = 1, \ldots, o:\ l_{(i)} - u_{r(i)} \leq w_{(i)}(c) + K_{(i,\cdot)}(x-c) + 0.5(x-c)^T Q_i (x-c) \leq -l_{r(i)} \Big\} =$$

$$\Big\{ x \in \mathcal{CPZ} \;\Big|\; \forall i = 1, \ldots, o:\ K_{(i,\cdot)}(x-c) + 0.5 \underbrace{(x-c)^T Q_i (x-c)}_{\in\, sq(\mathcal{Q}, \mathcal{CPZ} \oplus (-c))} + \underbrace{0.5(u_{r(i)} - l_{r(i)} - l_{(i)})}_{\overline{G}_{1(i,\cdot)}} \alpha_{p+i} = \underbrace{0.5(l_{(i)} - l_{r(i)} - u_{r(i)}) - w_{(i)}(c)}_{\overline{b}_{1(i)}},\ \alpha_{p+i} \in [-1,1] \Big\} \overset{\text{Def. 3.2.1}}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) KG_{(\cdot,i)} + \sum_{i=1}^{h^2+h} \left(\prod_{k=1}^{p} \alpha_k^{\widehat{E}_{(k,i)}} \right) \widehat{G}_{(\cdot,i)} + \sum_{i=1}^{o} \overline{G}_{1(\cdot,i)} \alpha_{p+i} = \overline{b}_1,\ \alpha_k, \alpha_{p+i} \in [-1,1] \right\}$$

$$= \left\langle c, G, E, \begin{bmatrix} A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & KG & \widehat{G} & \overline{G}_1 \end{bmatrix}, \begin{bmatrix} b \\ \overline{b}_1 \end{bmatrix}, \begin{bmatrix} R & E & \widehat{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_o \end{bmatrix} \right\rangle_{CPZ},$$

where $\mathcal{Q} = \{Q_1, \ldots, Q_o\}$. The result for $\mathcal{PZ} \cap \mathcal{LS}_2$ is obtained by inserting $l = \mathbf{0}$, which corresponds to the constraint $\mathbf{0} \leq w(x) \leq \mathbf{0}$ that is identical to $w(x) = \mathbf{0}$, into the formula for $\mathcal{PZ} \cap \mathcal{LS}_1$.

Complexity: Computation of the quadratic map $sq(\mathcal{Q}, \mathcal{CPZ} \oplus (-c))$ using Prop. 3.2.22 has complexity $\mathcal{O}(n^3(o + \log(n)))$. Moreover, calculation of the interval enclosure $\texttt{interval}(\mathcal{CPZ})$ requires the evaluation of $2n$ support functions using Prop. 3.2.15, which results for all contractors considered in this thesis in a complexity of $2n \cdot \mathcal{O}(n^5) = \mathcal{O}(n^6)$ according to Tab. 3.8. Since $\nabla^3 w(x)$ consists of on^3 scalar functions, construction of $\boldsymbol{\mathcal{D}}$ using range bounding has complexity $on^3 \cdot \mathcal{O}(\texttt{bound})$. In addition, computation of the cubic map $1/6\ poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-c))$ as defined in (2.10) using interval arithmetic has complexity $\mathcal{O}(on^3)$. Computation of $\texttt{bound}(w(x), \mathcal{I})$ has complexity $o \cdot \mathcal{O}(\texttt{bound})$ since $w: \mathbb{R}^n \to \mathbb{R}^o$. Finally, construction of the matrices $\overline{G}_1$,$\overline{G}_2$ and the vectors $\overline{b}_1$,$\overline{b}_2$ has complexity $\mathcal{O}(o)$ and

the matrix multiplication KG has complexity $\mathcal{O}(ohn)$ according to Tab. 2.1. Summarizing the complexities for the single parts yields an overall complexity of

$$\begin{aligned}\mathcal{O}(n^3(o+\log(n))) + \mathcal{O}(n^6) + on^3 \cdot \mathcal{O}(\texttt{bound}) \\ + \mathcal{O}(on^3) + o \cdot \mathcal{O}(\texttt{bound}) + \mathcal{O}(o) + \mathcal{O}(ohn),\end{aligned}$$

which is $\mathcal{O}(n^6) + on^3 \cdot \mathcal{O}(\texttt{bound})$ using Assumption 3.2.3. □

For level sets defined by polynomial functions $w(x)$, the intersection with a CPZ can be calculated exactly by considering a sufficiently high order for the Taylor series expansion in Prop. 3.2.24. The union of two CPZs can be computed as follows:

Proposition 3.2.25. *(Union) Given two CPZs, $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1 R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$, their union is*

$$\mathcal{CPZ}_1 \cup \mathcal{CPZ}_2 = \left\langle 0.5(c_1+c_2), \begin{bmatrix}0.5(c_1-c_2) & G_1 & G_2\end{bmatrix}, \begin{bmatrix}1 & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & E_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & E_2\end{bmatrix}, \right.$$
$$\left. \begin{bmatrix}\widehat{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \overline{A} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & A_1 & \mathbf{0} & -0.5\ b_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & A_2 & 0.5\ b_2\end{bmatrix}, \begin{bmatrix}\widehat{b} \\ \overline{b} \\ 0.5\ b_1 \\ 0.5\ b_2\end{bmatrix}, \begin{bmatrix}\widehat{R} & \overline{R} & \begin{bmatrix}\mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & 0 \\ R_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & R_2 & \mathbf{0}\end{bmatrix}\end{bmatrix} \right\rangle_{CPZ},$$

where

$$\widehat{A} = 1, \quad \widehat{b} = 1, \quad \widehat{R} = \begin{bmatrix}1 \\ 1 \\ \mathbf{0} \\ \mathbf{0}\end{bmatrix}, \quad H = \begin{bmatrix}\begin{bmatrix}2 & \dots & 2\end{bmatrix} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \begin{bmatrix}2 & \dots & 2\end{bmatrix} \\ 2I_{p_2} & \dots & 2I_{p_2}\end{bmatrix},$$

$$\overline{A} = \begin{bmatrix}1 & -1 & \frac{1}{2p_1}\mathbf{1} & -\frac{1}{2p_1}\mathbf{1} & -\frac{1}{2p_2}\mathbf{1} & -\frac{1}{2p_2}\mathbf{1} & -\frac{1}{4p_1p_2}\mathbf{1} & \frac{1}{4p_1p_2}\mathbf{1}\end{bmatrix}, \quad \overline{b} = 0,$$

$$\overline{R} = \begin{bmatrix}\begin{bmatrix}1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 2I_{p_1} & 2I_{p_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 2I_{p_2} & 2I_{p_2}\end{bmatrix} & \begin{bmatrix}\mathbf{0} \\ \mathbf{0} \\ H\end{bmatrix} & \begin{bmatrix}\mathbf{1} \\ \mathbf{0} \\ H\end{bmatrix}\end{bmatrix},$$

which has complexity $\mathcal{O}(n^3 \log(n))$ with respect to the dimension n. The `compactCon` *operation as defined in Prop. 3.2.6 is applied to make the resulting CPZ regular.*

Proof. A detailed proof of the above formula for computing the union of two CPZs is provided in Appendix B.

Complexity: We first consider the assembly of the resulting CPZ. The computation of the vectors $0.5(c_1 + c_2)$ and $0.5(c_1 - c_2)$ requires n additions, n subtractions, and $2n$ multiplications. Moreover, computation of $-0.5\ b_1$, $0.5\ b_1$ and $0.5\ b_2$ requires $2m_1 + m_2$ multiplications, and computation of the matrix $\overline{A}$ requires 3 multiplications and 2 divisions.

Since the construction of the remaining matrices only involves concatenations, the resulting complexity for the construction of the CPZ is $\mathcal{O}(4n + 2m_1 + m_2 + 5)$, which is $\mathcal{O}(n)$ using Assumption 3.2.3. Next, we consider the subsequent application of the `compactCon` operation. The constraint generator matrix A for the resulting CPZ has $q = 1 + \widehat{q} + \overline{q} + q_1 + q_2 = 4 + 2p_1 + 2p_2 + 2p_1p_2 + q_1 + q_2$ columns since $\widehat{A}$ has $\widehat{q} = 1$ column, $\overline{A}$ has $\overline{q} = 2 + 2p_1 + 2p_2 + 2p_1p_2$ columns, A_1 has q_1 columns, and A_2 has q_2 columns. Moreover, the matrix A has $m = \widehat{m} + \overline{m} + m_1 + m_2 = 2 + m_1 + m_2$ rows since $\widehat{A}$ has $\widehat{m} = 1$ row, $\overline{A}$ has $\overline{m} = 1$ rows, A_1 has m_1 rows, and A_2 has m_2 rows. The number of factors of the resulting CPZ is $p = p_1 + p_2 + 2$. Since the complexity of the `compactCon` operation is $\mathcal{O}(q(m + p\log(q))$ according to Prop. 3.2.6, the subsequent application of `compactCon` has complexity

$$\mathcal{O}\big((4 + 2p_1 + 2p_2 + 2p_1p_2 + q_1 + q_2)(2 + m_1 + m_2 + (p_1 + p_2 + 2)\log(4 + 2p_1 + 2p_2 + 2p_1p_2 + q_1 + q_2))\big),$$

which is $\mathcal{O}(n^3\log(n))$ using Assumption 3.2.3. Combining the complexities for the assembly of the resulting CPZ and the subsequent application of the `compactCon` operation finally yields an overall complexity of $\mathcal{O}(n) + \mathcal{O}(n^3\log(n)) = \mathcal{O}(n^3\log(n))$. □

As a last set operation, we consider the Minkowski difference. While it is yet unclear how to compute the Minkowski difference of two CPZs, we specify a closed-form expression for the computation of the Minkowski difference of a CPZ with a polytope. Using this formula, a tight inner-approximation of the Minkowski difference of two CPZs can be obtained by first enclosing the minuend by a polytope as described in Sec. 3.2.4.

Proposition 3.2.26. *(Minkowski Difference Polytope) Given a CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$ *and the V-representation of a polytope* $\mathcal{P} = \langle [v_1 \ \dots \ v_s]\rangle_V \subset \mathbb{R}^n$*, their Minkowski difference* $\mathcal{CPZ} \ominus \mathcal{P}$ *is*

$$\mathcal{CPZ} \ominus \mathcal{P} = \left\langle c - v_1, G, \overline{E}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & R_2 \end{bmatrix} \right\rangle_{CPZ}, \tag{3.71}$$

where

$$\overline{E} = \begin{bmatrix} E \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \in \mathbb{N}_0^{ps \times h}, \quad R_1 = \begin{bmatrix} R & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & R \end{bmatrix} \in \mathbb{N}_0^{ps \times qs}, \quad R_2 = \begin{bmatrix} E & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & E \end{bmatrix} \in \mathbb{N}_0^{ps \times hs},$$

$$A_1 = \begin{bmatrix} A & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & A \end{bmatrix} \in \mathbb{R}^{ms \times qs}, \quad A_2 = \begin{bmatrix} G & -G & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ G & \mathbf{0} & \dots & -G \end{bmatrix} \in \mathbb{R}^{n(s-1) \times hs},$$

$$b_1 = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \in \mathbb{R}^{ms}, \quad b_2 = \begin{bmatrix} v_1 - v_2 \\ \vdots \\ v_1 - v_s \end{bmatrix} \in \mathbb{R}^{n(s-1)}.$$

The `compactCon` *operation as defined in Prop. 3.2.6 is applied to make the resulting CPZ regular. The computational complexity is* $\mathcal{O}(n^2s^2\log(ns))$*, where* n *is the dimension and* s *is the number of polytope vertices.*

Proof. According to [83, Lemma 1], which is derived from [84, Thm. 2.1], the Minkowski difference $\mathcal{CPZ} \ominus \mathcal{P}$ can be computed as

$$\begin{aligned} \mathcal{CPZ} \ominus \mathcal{P} &= \bigcap_{i \in \{1,\dots,s\}} \big(\mathcal{CPZ} \oplus (-v_i)\big) \\ &= \big(\mathcal{CPZ} \oplus (-v_1)\big) \cap \big(\mathcal{CPZ} \oplus (-v_2)\big) \cap \ldots \cap \big(\mathcal{CPZ} \oplus (-v_s)\big), \end{aligned} \tag{3.72}$$

where $v_i \in \mathbb{R}^n$ are the polytope vertices. Using Prop. 3.2.23, we obtain for the first intersection between $\mathcal{CPZ} \oplus (-v_1)$ and $\mathcal{CPZ} \oplus (-v_2)$ in (3.72)

$$\big(\mathcal{CPZ} \oplus (-v_1)\big) \cap \big(\mathcal{CPZ} \oplus (-v_2)\big) =$$

$$\langle c - v_1, G, E, A, b, R\rangle_{CPZ} \cap \langle c - v_2, G, E, A, b, R\rangle_{CPZ} \overset{\text{Prop. 3.2.23}}{=}$$

$$\left\langle c - v_1, G, \begin{bmatrix} E \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} A & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & G & -G \end{bmatrix}, \underbrace{\begin{bmatrix} b \\ b \\ (c - v_2) - (c - v_1) \end{bmatrix}}_{= v_1 - v_2}, \begin{bmatrix} R & \mathbf{0} & E & \mathbf{0} \\ \mathbf{0} & R & \mathbf{0} & E \end{bmatrix} \right\rangle_{CPZ}.$$

Repeated application of Prop. 3.2.23 for all intersections in (3.72) then yields the equation in (3.71).

Complexity: Computation of the constant offset $c - v_1$ requires n subtractions, and computation of the vector b_2 requires $n(s-1)$ subtractions. The construction of the resulting set in (3.71) has therefore complexity $\mathcal{O}(n) + \mathcal{O}(n(s-1)) = \mathcal{O}(ns)$. Subsequent application of the `compactCon` operation has complexity $\mathcal{O}(\overline{q}(\overline{m} + \overline{p}\log(\overline{q})))$ according to Prop. 3.2.6, where $\overline{p} = ps$, $\overline{m} = (m+n)s - n$, and $\overline{q} = (h+q)s$ denote the number of factors, the number of constraints, and the number of constraint generators of the resulting CPZ in (3.71). For the overall complexity we therefore obtain

$$\mathcal{O}(ns) + \mathcal{O}\big(\overline{q}(\overline{m} + \overline{p}\log(\overline{q}))\big) \overset{\substack{\overline{p}=ps,\ \overline{q}=(h+q)s \\ \overline{m}=(m+n)s-n}}{=}$$

$$\mathcal{O}(ns) + \mathcal{O}\big((h+q)s((m+n)s - n + ps\log((h+q)s))\big),$$

which is $\mathcal{O}(n^2 s^2 \log(ns))$ using Assumption 3.2.3. □

In Tab. 3.9, the growth of CPZ factors, generators, constraints, and constraint generators is summarized for all set operations presented in this section. Since many operations on CPZs significantly increase the representation size, one requires efficient reduction techniques when computing with CPZs. For this, we present in Sec. 3.2.7 the operations `reduce` in Prop. 3.2.29 and `reduceCon` in Prop. 3.2.31, which reduce the number of generators and the number of constraints of a CPZ, respectively.

Let us conclude this section with a discussion on how to extend CPZs with independent generators. Given CPZs $\mathcal{CPZ}_1, \mathcal{CPZ}_2 \subset \mathbb{R}^n$, the addition of independent generators $G_1 \in \mathbb{R}^{n \times l_1}$, $G_2 \in \mathbb{R}^{n \times l_2}$ yields the sets $\mathcal{S}_1 = \mathcal{CPZ}_1 \oplus \langle \mathbf{0}, G_{I,1}\rangle_Z$ and $\mathcal{S}_2 = \mathcal{CPZ}_2 \oplus \langle \mathbf{0}, G_{I,2}\rangle_Z$. For the set operations linear map, Minkowski sum, exact addition, Cartesian product, linear

Table 3.9: Growth of the number of factors p, the number of generators h, the number of constraints m, and the number of constraint generators q for basic set operations on CPZs, where n denotes the dimension, o denotes the number of level set constraints, s denotes the number of polytope vertices, and we use the shorthands $a = n + 1$ and $b = q_1 + q_2 + 2(p_1 + p_2 + p_1p_2) + 4$.

Set Operation	**Factors**	**Generators**	**Constraints**	**Constraint Generators**
Linear map	p	h	m	q
Minkowski sum	$p_1 + p_2$	$h_1 + h_2$	$m_1 + m_2$	$q_1 + q_2$
Exact addition	$p = p_1 = p_2$	$h_1 + h_2$	$m_1 + m_2$	$q_1 + q_2$
Cartesian product	$p_1 + p_2$	$h_1 + h_2$	$m_1 + m_2$	$q_1 + q_2$
Linear combination	$p_1 + p_2 + 1$	$2h_1 + 2h_2 + 1$	$m_1 + m_2$	$q_1 + q_2$
Convex hull	$a(p_1 + p_2 + 2)$	$a(4h_1{+}4h_2{+}3)$	$a(m_1{+}m_2){+}1$	$a(q_1 + q_2 + 1)$
Quadratic map	p	$h^2 + h$	m	q
Mixed quad. map	$p_1 + p_2$	$h_1h_2 + h_1 + h_2$	$m_1 + m_2$	$q_1 + q_2$
Intersection	$p_1 + p_2$	h_1	$m_1 + m_2 + n$	$q_1{+}q_2{+}h_1{+}h_2$
Intersection level set	$p + o$	h	$m + o$	$q{+}2h{+}h^2{+}o$
Union	$p_1 + p_2 + 2$	$h_1 + h_2 + 1$	$m_1 + m_2 + 2$	b
Mink. diff. polytope	ps	h	$(m + n)s - n$	$(h + q)s$

combination, convex hull, and quadratic map, independent generators can be included in the same way as for SPZs in Sec. 3.1.5. Moreover, with independent generators a tight enclosure of the intersection can be computed as

$$\mathcal{S}_1 \cap \mathcal{S}_2 \subseteq \big(\mathcal{CPZ}_1 \oplus \langle \mathbf{0}, \overline{G}_{I,1}, I_n, [\,], [\,], [\,]\rangle_{CPZ}\big) \cap \big(\mathcal{CPZ}_2 \oplus \langle \mathbf{0}, \overline{G}_{I,2}, I_n, [\,], [\,], [\,]\rangle_{CPZ}\big),$$

where we reduce the zonotopes corresponding to the independent generators to order 1 using one of the methods in Sec. 2.6 to minimize the number of factors of the resulting CPZ:

$$\langle \mathbf{0}, \overline{G}_{I,1}\rangle_Z = \texttt{reduce}\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, 1\big), \quad \langle \mathbf{0}, \overline{G}_{I,2}\rangle_Z = \texttt{reduce}\big(\langle \mathbf{0}, G_{I,2}\rangle_Z, 1\big).$$

In addition, the union can be tightly enclosed by

$$\mathcal{S}_1 \cup \mathcal{S}_2 \subseteq \big(\mathcal{CPZ}_1 \cup \mathcal{CPZ}_2\big) \oplus conv\big(\langle \mathbf{0}, G_{I,1}\rangle_Z, \langle \mathbf{0}, G_{I,2}\rangle_Z\big),$$

where the enclosure of the convex hull of two zonotopes is computed using Prop. 3.1.23. For the Minkowski difference, one can just omit the independent generators, which yields an inner-approximation.

3.2.6 Intersection and Containment Checks

In this section we present necessary as well as sufficient conditions that can be applied for intersection and set containment checks on CPZs. We begin with a sufficient condition for an interval to be contained in a CPZ:

Proposition 3.2.27. *(Containment Check Interval) Given a CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$ *with a degree-of-freedom order* $\rho_f = \frac{p-m}{n} \geq 1$ *and an interval* $\mathcal{I} \subset \mathbb{R}^n$*, it holds that*

$$\big((\mathcal{I} \times 0 \cdot b) \subseteq \texttt{compact}(\mathcal{PZ}^+)\big) \Rightarrow (\mathcal{I} \subseteq \mathcal{CPZ}),$$

where $\mathcal{PZ}^+$ *is the lifted polynomial zonotope defined as in Lemma 3.2.4:*

$$\mathcal{PZ}^+ = \left\langle \begin{bmatrix} c \\ -b \end{bmatrix}, \begin{bmatrix} G & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}, [\,], \begin{bmatrix} E & R \end{bmatrix}, id \right\rangle_{PZ}.$$

The containment check $(\mathcal{I} \times 0 \cdot b) \subseteq \texttt{compact}(\mathcal{PZ}^+)$ *is performed using the criterion for SPZs in Prop. 3.1.36 and the* `compact` *operation as defined in Prop. 3.1.7 is applied to reduce the conservatism of the criterion.*

Proof. Using Lemma 3.2.4 we have

$$\begin{aligned}\big((\mathcal{I} \times 0 \cdot b) \subseteq \mathcal{PZ}^+\big) &\Rightarrow \left(\forall x \in \mathcal{I} : \begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathcal{PZ}^+\right) \\ &\overset{\text{Lemma 3.2.4}}{\Rightarrow} \big(\forall x \in \mathcal{I} : \ x \in \mathcal{CPZ}\big) \Rightarrow (\mathcal{I} \subseteq \mathcal{CPZ}),\end{aligned}$$

which concludes the proof. □

To extend Prop. 3.2.27 to CPZ in CPZ containment, one can first recursively split the contained CPZ multiple times using the `split` operation as defined later in Prop. 3.2.35. After enclosing all split sets with intervals as described in Sec. 3.2.4, one can then apply the criterion in Prop. 3.2.27 to prove containment. To check if a CPZ is contained in other set representations, such as ellipsoids or polytopes, one can use the same procedures as for SPZs, which are described in Sec. 3.2.6. Next, we consider the problem of testing if two CPZs intersect:

Proposition 3.2.28. *(Intersection Check) Given two CPZs,* $\mathcal{CPZ}_1, \mathcal{CPZ}_2 \subset \mathbb{R}^n$*, it holds that*

$$\big(\texttt{contract}(f(y), [-\mathbf{1}, \mathbf{1}]) = \emptyset\big) \Rightarrow (\mathcal{CPZ}_1 \cap \mathcal{CPZ}_2 = \emptyset),$$

where the function $f(y)$ *is defined as*

$$f(y) = -b + \sum_{i=1}^{q} \left(\prod_{k=1}^{p} y_{(k)}^{R_{(k,i)}} \right) A_{(\cdot,i)}, \qquad \langle c, G, E, A, b, R\rangle_{CPZ} = \mathcal{CPZ}_1 \cap \mathcal{CPZ}_2,$$

and the intersection $\mathcal{CPZ}_1 \cap \mathcal{CPZ}_2$ *is calculated using Prop. 3.2.23.*

Proof. The function $f(y)$ represents the polynomial constraint $f(y) = \mathbf{0}$ of the CPZ $\langle c, G, E, A, b, R\rangle_{CPZ} = \mathcal{CPZ}_1 \cap \mathcal{CPZ}_2$, with $y = [\alpha_1 \ \dots \ \alpha_p]^T$ storing the factors of the CPZ. Consequently, if the factor domain $y \in [-\mathbf{1}, \mathbf{1}]$ can be contracted to the empty set, then $\mathcal{CPZ}_1 \cap \mathcal{CPZ}_2$ is empty. □

Prop. 3.2.28 presents a criterion to prove that two CPZs do not intersect. Contrary, to show that two CPZs intersect, one can solve the polynomial optimization problem

$$\min_{y \in [-\mathbf{1},\mathbf{1}]} ||y||_2 \ \text{ s.t. } f(y) = \mathbf{0},$$

where $f(y)$ is defined as in Prop. 3.2.28, and then apply Prop. 3.2.27 to show that the corresponding point is contained in $\mathcal{CPZ}_1$ and $\mathcal{CPZ}_2$. For intersection checks with other set representations one can apply the same procedures as for SPZs, which are described in Sec. 3.1.6.

3.2.7 Auxiliary Set Operations

Finally, we present several useful auxiliary operations on CPZs. As shown in Tab. 3.9, many operations on CPZs significantly increase the representation size. For computational reasons an efficient strategy for representation size reduction is therefore crucial when computing with CPZs. Thus, we now introduce the operations `reduce` and `reduceCon` for reducing the number of generators and the number of constraints of a CPZ:

Proposition 3.2.29. *(Order Reduction) Given a CPZ $\mathcal{CPZ} \subset \mathbb{R}^n$ and a desired order $\rho_d \geq 2 \cdot \frac{n+m}{n}$, the operation* `reduce` *returns a CPZ with an order smaller than or equal to ρ_d that encloses $\mathcal{CPZ}$:*

$$\mathcal{CPZ} \subseteq \texttt{reduce}(\mathcal{CPZ}, \rho_d) = \Big\langle \overline{c}, \begin{bmatrix} \overline{G}_{(\cdot,\mathcal{H})} & \overline{G}_{I(\cdot,\mathcal{K})} \end{bmatrix}, \begin{bmatrix} \overline{E}_{(\cdot,\mathcal{H})} & \mathbf{0} \\ \mathbf{0} & \widehat{E}_{(\cdot,\mathcal{K})} \end{bmatrix}, \begin{bmatrix} \overline{A}_{(\cdot,\mathcal{G})} & \overline{A}_{I(\cdot,\mathcal{F})} \end{bmatrix}, -\overline{b}, \begin{bmatrix} \overline{E}_{(\cdot,\mathcal{G})} & \mathbf{0} \\ \mathbf{0} & \widehat{E}_{(\cdot,\mathcal{F})} \end{bmatrix} \Big\rangle_{CPZ},$$

where

$$\langle c, G, E, A, b, R\rangle_{CPZ} = \texttt{rescale}(\mathcal{CPZ}), \quad id = \texttt{uniqueID}(p),$$

$$\mathcal{PZ}^+ = \left\langle \begin{bmatrix} c \\ -b \end{bmatrix}, \begin{bmatrix} G & \mathbf{0} \\ \mathbf{0} & A \end{bmatrix}, [\,], \begin{bmatrix} E & R \end{bmatrix}, id \right\rangle_{PZ}, \quad \rho_d^+ = \frac{\rho_d n}{2(n+m)}$$

$$\left\langle \begin{bmatrix} \overline{c} \\ \overline{b} \end{bmatrix}, \begin{bmatrix} \overline{G} \\ \overline{A} \end{bmatrix}, \begin{bmatrix} \overline{G}_I \\ \overline{A}_I \end{bmatrix}, \overline{E}, \overline{id} \right\rangle_{PZ} = \texttt{reduce}\big(\texttt{compact}(\mathcal{PZ}^+), \rho_d^+\big), \quad \widehat{E} = I_{n+m},$$

and the sets $\mathcal{H}, \mathcal{K}, \mathcal{G}, \mathcal{F}$ defined as

$$\begin{aligned} \mathcal{H} &= \big\{i \mid \exists j \in \{1,\dots,n\} : \ \overline{G}_{(j,i)} \neq 0\big\}, \quad \mathcal{K} = \big\{i \mid \exists j \in \{1,\dots,n\} : \ \overline{G}_{I(j,i)} \neq 0\big\}, \\ \mathcal{G} &= \big\{i \mid \exists j \in \{1,\dots,m\} : \ \overline{A}_{(j,i)} \neq 0\big\}, \quad \mathcal{F} = \big\{i \mid \exists j \in \{1,\dots,m\} : \ \overline{A}_{I(j,i)} \neq 0\big\} \end{aligned} \tag{3.73}$$

store the indices of non-zero generators. The reduced order SPZ for $\mathcal{PZ}^+$ is calculated using Prop. 3.1.39. We apply the `rescale` *operation as defined in Prop. 3.2.7 and the* `compact` *operation for SPZs as defined in Prop. 3.1.7 are applied to reduce the over-approximation. The resulting CPZ is regular and the computational complexity with respect to the dimension n is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(\texttt{reduce})$.*

Proof. To calculate a reduced order CPZ we reduce the order of the corresponding lifted polynomial zonotope as defined in Lemma 3.2.4 using the `reduce` operation on SPZs in Prop. 3.1.39. Back-transformation of the lifted polynomial zonotope to the original space then yields an over-approximative CPZ, which can be proven using Lemma 3.2.4:

$$\forall x \in \mathbb{R}^n : \ (x \in \mathcal{CPZ}) \overset{\text{Lemma 3.2.4}}{\Rightarrow} \left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \mathcal{PZ}^+ \right) \overset{\mathcal{PZ}^+ \subseteq \texttt{reduce}(\mathcal{PZ}^+, \rho_d^+)}{\Rightarrow}$$

$$\left(\begin{bmatrix} x \\ \mathbf{0} \end{bmatrix} \in \texttt{reduce}(\mathcal{PZ}^+, \rho_d^+) \right) \overset{\text{Lemma 3.2.4}}{\Rightarrow} \big(x \in \texttt{reduce}(\mathcal{CPZ}, \rho_d)\big),$$

where we omitted the operations `rescale` and `compact` since they only change the representation of the set, but not the set itself. It remains to show that the order of the resulting CPZ is smaller than or equal to the desired order ρ_d. According to Prop. 3.1.39, we have

$$\frac{\overline{h} + \overline{q}}{n + m} \leq \rho_d^+ = \frac{\rho_d n}{2(n+m)}, \tag{3.74}$$

where $\overline{h}$ and $\overline{q}$ denote the number of columns of the matrices $\overline{G}$ and $\overline{G}_I$, respectively. Solving (3.74) for ρ_d yields

$$2 \cdot \frac{\overline{h} + \overline{q}}{n} \leq \rho_d, \tag{3.75}$$

so that

$$\rho = \frac{|\mathcal{H}| + |\mathcal{K}| + |\mathcal{G}| + |\mathcal{F}|}{n} \overset{(3.73)}{\leq} 2 \cdot \frac{\overline{h} + \overline{q}}{n} \overset{(3.75)}{\leq} \rho_d.$$

Complexity: The computational complexity for the `rescale` operation is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract})$ according to Prop. 3.2.7 and the generation of p unique identifiers using `uniqueID` has complexity $\mathcal{O}(p)$ according to Tab. 2.2. Moreover, construction of the sets $\mathcal{H}$, $\mathcal{K}$, $\mathcal{G}$, and $\mathcal{F}$ has complexity $\mathcal{O}((h+q)(n+m))$ in the worst case. Let $n^+ = n + m$, $p^+ = p$, $h^+ = h + q$, and $q^+ = 0$ denote the dimension, the number of dependent factors, the number of dependent generators, and the number of independent generators of the lifted polynomial zonotope $\mathcal{PZ}^+$. According to Prop. 3.1.7, the `compact` operation for SPZs has complexity $\mathcal{O}(h^+(n^+ + p^+ \log(h^+)))$, and the complexity of order reduction of a SPZ using Prop. 3.1.39 is $\mathcal{O}((h^+ + q^+)(n^+ + \log(h^+ + q^+))) + \mathcal{O}(p^+ h^+) + \mathcal{O}(\texttt{reduce})$ according to (3.37). The overall computational complexity is therefore

$$\begin{aligned} &\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(p) + \mathcal{O}((h+q)(n+m)) + \mathcal{O}\big(h^+(n^+ + p^+ \log(h^+))\big) \\ &\qquad + \mathcal{O}\big((h^+ + q^+)(n^+ + \log(h^+ + q^+))\big) + \mathcal{O}(p^+ h^+) + \mathcal{O}(\texttt{reduce}) \\ &\overset{\substack{n^+ = n+m,\ p^+ = p \\ h^+ = h+q,\ q^+ = 0}}{=} \mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) \\ &\qquad + \mathcal{O}\big((h+q)(n+m+p \log(h+q))\big) + \mathcal{O}(\texttt{reduce}), \end{aligned}$$

which is $\mathcal{O}(n^4 \log(n)) + \mathcal{O}(\texttt{contract}) + \mathcal{O}(\texttt{reduce})$ using Assumption 3.2.3. □

Table 3.10: Computational complexity for order reduction of CPZs using Prop. 3.2.29 with respect to the dimension $n \in \mathbb{N}$ for different the zonotope order reduction methods presented in Sec. 2.6 and the different contractors presented in Sec. 2.8, where $k \in \mathbb{N}_0$ is the number of generators that are reduced.

Contractor	**Girard**	**PCA**	**Scott**	**Exh. Search**
Forward-backward	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}\left(n^4 \log(n) + \binom{k}{n} k\right)$
Extremal functions	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}(n^4 \log(n))$	$\mathcal{O}\left(n^4 \log(n) + \binom{k}{n} k\right)$
Parallel linearization	$\mathcal{O}(n^{4.5})$	$\mathcal{O}(n^{4.5})$	$\mathcal{O}(n^{4.5})$	$\mathcal{O}\left(n^{4.5} + \binom{k}{n} k\right)$

The computational complexity of operation `reduce` for different contractors and different zonotope order reduction methods is summarized in Tab 3.10. One advantage of the order reduction method in Prop. 3.2.29 is that it maintains dependencies between the generators and constraints. On the other hand, one disadvantage is that the presented method cannot reduce to orders smaller than $2\frac{n+m}{n}$. Let us demonstrate order reduction using Prop. 3.2.29 by an example:

Example 3.2.30. *We consider the CPZ*

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \begin{bmatrix} 2.5 & 0 & 2 & 0.05 & 0.02 & -0.03 & 0 \\ 0 & -4 & 3 & 0.02 & -0.01 & 0 & 0.02 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 2 & 3 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 1 & 5 & 1 & 0.1 & -0.2 & 0.2 & 0.05 \end{bmatrix}, 0, \begin{bmatrix} 1 & 0 & 2 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 2 & 3 \end{bmatrix} \right\rangle_{CPZ},$$

which has order $\rho = 7$. Order reduction to the desired order $\rho_d = 6$ using Prop. 3.2.29 yields

$$\texttt{reduce}(\mathcal{CPZ}, 6) = \left\langle \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \begin{bmatrix} 2.5 & 0 & 2 & -0.031 & -0.09 & 0.008 \\ 0 & -4 & 3 & 0.03 & -0.03 & -0.024 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 & \mathbf{0} \\ 0 & 1 & 0 & \mathbf{0} \\ 3 & 0 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_3 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 1 & 5 & 1 & 0.549 & -0.003 & 0.002 \end{bmatrix}, 0, \begin{bmatrix} 1 & 0 & 2 & \mathbf{0} \\ 0 & 1 & 0 & \mathbf{0} \\ 3 & 0 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_3 \end{bmatrix} \right\rangle_{CPZ}$$

if PCA is used for zonotope order reduction (see Sec. 2.6). The tightness of the resulting reduced order CPZ is visualized in Fig. 3.12.

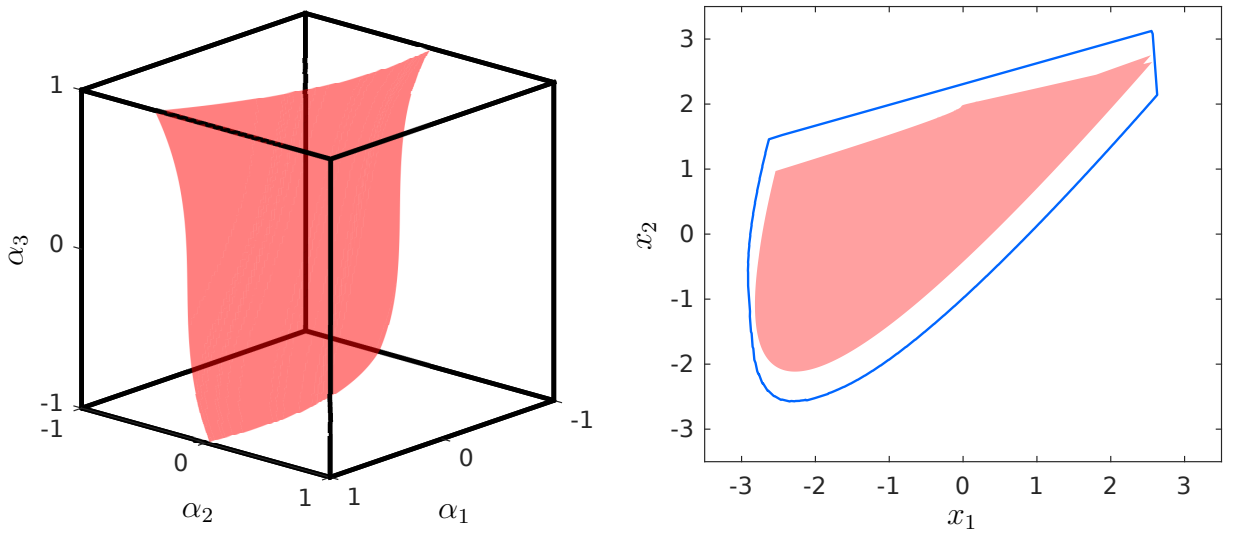

Figure 3.12: Visualization of order reduction using Prop. 3.2.29 for the CPZ from Example 3.2.30. The original CPZ $\mathcal{CPZ}$ (right) and the corresponding constraint (left) is depicted in red, and the reduced order CPZ $\texttt{reduce}(\mathcal{CPZ}, 6)$ (right) is depicted in blue.

Next, we show how one can reduce the number of constraints of a CPZ:

Proposition 3.2.31. *(Constraint Reduction) Given a CPZ* $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$*, the index of one constraint* $r \in \{1, \dots, m\}$*, and indices* $d, s \in \mathbb{N}$ *satisfying*

$$\forall i \in \{1, \dots, p\} : \; E_{(i,d)} = R_{(i,s)} \quad \text{and} \quad A_{(r,s)} \neq 0, \tag{3.76}$$

the operation `reduceCon` *removes the constraint with index* r *and returns a CPZ that encloses* $\mathcal{CPZ}$*:*

$$\mathcal{CPZ} \subseteq \texttt{reduceCon}(\mathcal{CPZ}, r, d, s) = \left\langle \overline{c}, \overline{G}, \overline{E}_{(\mathcal{N},\cdot)}, \overline{A}, \overline{b}, \overline{R}_{(\mathcal{N},\cdot)} \right\rangle_{CPZ},$$

where

$$\overline{G} = \begin{bmatrix} G_{(\cdot,\{1,\dots,d-1\})} & -\frac{1}{A_{(r,s)}} A_{(r,\mathcal{H})} G_{(\cdot,d)} & G_{(\cdot,\{d+1,\dots,h\})} \end{bmatrix}, \quad \overline{R} = R_{(\cdot,\mathcal{H})},$$

$$\overline{E} = \begin{bmatrix} E_{(\cdot,\{1,\dots,d-1\})} & \overline{R} & E_{(\cdot,\{d+1,\dots,h\})} \end{bmatrix}, \quad \overline{A} = A_{(\mathcal{K},\mathcal{H})} - \frac{1}{A_{(r,s)}} A_{(r,\mathcal{H})} A_{(\mathcal{K},s)},$$

$$\overline{c} = c + \frac{b_{(r)}}{A_{(r,s)}} G_{(\cdot,d)}, \quad \overline{b} = b_{(\mathcal{K})} - \frac{b_{(r)}}{A_{(r,s)}} A_{(\mathcal{K},s)},$$

and the sets $\mathcal{H}$*,* $\mathcal{K}$*, and* $\mathcal{N}$ *are defined as*

$$\begin{aligned} \mathcal{H} &= \{1, \dots, q\} \setminus s, \quad \mathcal{K} = \{1, \dots, m\} \setminus r, \\ \mathcal{N} &= \left\{ i \;\middle|\; \exists j, k : \; \overline{E}_{(i,j)} \neq 0 \vee \overline{R}_{(i,k)} \neq 0 \right\}. \end{aligned} \tag{3.77}$$

The `compactGen` *operation as defined in Prop. 3.2.5 is applied to make the resulting CPZ regular. The computational complexity with respect to the dimension* n *is* $\mathcal{O}(n^2 \log(n))$*.*

Proof. To remove the constraint with index r, we solve the corresponding equation for the variable part of the monomial with index s:

$$\prod_{k=1}^{p} \alpha_k^{R_{(k,s)}} = \frac{1}{A_{(r,s)}} \Big(- \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(r,i)} + b_{(r)} \Big). \tag{3.78}$$

Inserting the substitution in (3.78) into the definition of a CPZ in Def. 3.2.1 then yields

$$\begin{aligned}
\mathcal{CPZ} =& \Big\{ c + \sum_{i=1}^{h} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} \;\Big|\; \sum_{i=1}^{q} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(\cdot,i)} = b,\ \alpha_k \in [-1,1] \Big\} \\
\overset{(3.77)}{=} & \Big\{ c + \sum_{i=1}^{d-1} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} + \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,s)}} \Big) G_{(\cdot,d)} + \sum_{i=d+1}^{h} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} \;\Big|\; \\
& \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(\cdot,i)} + \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,s)}} \Big) A_{(\cdot,s)} = b,\ \alpha_k \in [-1,1] \Big\} \\
\overset{(3.78)}{\subseteq} & \Big\{ c + \sum_{i=1}^{d-1} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} + \frac{1}{A_{(r,s)}} \Big(- \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(r,i)} + b_{(r)} \Big) G_{(\cdot,d)} + \\
& \sum_{i=d+1}^{h} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} \;\Big|\; \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(\cdot,i)} + \\
& \frac{1}{A_{(r,s)}} \Big(- \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) A_{(r,i)} + b_{(r)} \Big) A_{(\cdot,s)} = b,\ \alpha_k \in [-1,1] \Big\} \\
\overset{\substack{\text{remove} \\ \text{constraint} \\ \text{"0=0"}}}{=} & \Big\{ \underbrace{c + \frac{b_{(r)}}{A_{(r,s)}} G_{(\cdot,d)}}_{\overline{c}} + \sum_{i=1}^{d-1} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} - \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) \frac{1}{A_{(r,s)}} A_{(r,i)} G_{(\cdot,d)} + \\
& \sum_{i=d+1}^{h} \Big(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \Big) G_{(\cdot,i)} \;\Big|\; \alpha_k \in [-1,1], \\
& \sum_{i \in \mathcal{H}} \Big(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \Big) \underbrace{\Big(A_{(\mathcal{K},i)} - \frac{1}{A_{(r,s)}} A_{(r,i)} A_{(\mathcal{K},s)} \Big)}_{\overline{A}_{(\cdot,i)}} = \underbrace{b_{(\mathcal{K})} - \frac{b_{(r)}}{A_{(r,s)}} A_{(\mathcal{K},s)}}_{\overline{b}} \Big\} \\
=& \big\langle \overline{c}, \overline{G}, \overline{E}_{(\mathcal{N},\cdot)}, \overline{A}, \overline{b}, \overline{R}_{(\mathcal{N},\cdot)} \big\rangle_{CPZ} = \texttt{reduceCon}(\mathcal{CPZ}, r, d, s).
\end{aligned} \tag{3.79}$$

The set $\mathcal{N}$ as defined in (3.77) only removes all-zero rows from the exponent matrix and the constraint exponent matrix, and therefore does not change the set. For the constraint

with index r, we obtain with the substitution from (3.78)

$$\sum_{i\in\mathcal{H}}\left(\prod_{k=1}^{p}\alpha_k^{R_{(k,i)}}\right)\underbrace{\left(A_{(r,i)}-\frac{1}{A_{(r,s)}}A_{(r,i)}A_{(r,s)}\right)}_{=0}=\underbrace{b_{(r)}-\frac{b_{(r)}}{A_{(r,s)}}A_{(r,s)}}_{=0}$$

the trivial constraint $0 = 0$, which is removed in (3.79) by restricting the indices of the constraints to the set $\mathcal{K}$ as defined in (3.77). The resulting CPZ therefore has one constraint less than the original CPZ.

Complexity: The construction of the matrix $\overline{G}$ requires nq multiplications, the construction of the matrix $\overline{A}$ requires $m(q-1)$ multiplications and $m(q-1)$ subtractions, and the construction of the vectors $\overline{c}$ and $\overline{b}$ requires $n+m-1$ multiplications and $n+m-1$ additions. Let $\overline{p} = p$, $\overline{h} = h+q-2$, and $\overline{q} = q-1$ denote the number of factors, the number of generators, and the number of constraint generators of the resulting CPZ $\langle\overline{c},\overline{G},\overline{E}_{(\mathcal{N},\cdot)},\overline{A},\overline{b},\overline{R}_{(\mathcal{N},\cdot)}\rangle_{CPZ}$. Then application of the `compactGen` operation has complexity $\mathcal{O}(\overline{h}(n+\overline{p}\log(\overline{h}))$ according to Prop. 3.2.5 and the construction of the set $\mathcal{N}$ in (3.77) has in the worst case complexity $\mathcal{O}(\overline{p}(\overline{h}+\overline{q}))$. The overall complexity is therefore

$$\begin{aligned}&\mathcal{O}(nq)+\mathcal{O}(2m(q-1))+\mathcal{O}(2n+2m-2)+\mathcal{O}\big(\overline{h}(n+\overline{p}\log(\overline{h}))\big)+\mathcal{O}\big(\overline{p}(\overline{h}+\overline{q})\big)\\ &\overset{\substack{\overline{p}=p\ \overline{q}=q-1\\ \overline{h}=h+q-2}}{=}\mathcal{O}(nq)+\mathcal{O}(mq)+\mathcal{O}\big((h+q-2)(n+p\log(h+q-2))\big)+\mathcal{O}\big(p(h+2q-3)\big)\\ &=\mathcal{O}(mq)+\mathcal{O}\big((h+q)(n+p\log(h+q))\big),\end{aligned}$$

which is $\mathcal{O}(n^2\log(n))$ using Assumption 3.2.3. □

The crucial point for Prop. 3.2.31 is the selection of the constraint r that is removed, as well as the selection of a suitable monomial with indices s, d that satisfies the conditions in (3.76) and can therefore be used for reduction. Clearly, we want to select the indices r, s, and d such that the over-approximation resulting from constraint reduction is minimized. Since it is computationally infeasible to determine the optimal indices for reduction, we instead present some heuristics on how to choose good values for r, s, and d. When removing a constraint from a CPZ using Prop. 3.2.31, there are two sources contributing to the resulting over-approximation:

1. Over-approximation due to lost bounds on factors
2. Over-approximation due to a loss of dependency

The over-approximation due to lost bounds results from the fact that due to the replacement of the selected monomial $\prod_{k=1}^{p}\alpha_k^{R_{(k,s)}}$ with the solved constraint in (3.78) we lose the ability to enforce the bounds $\prod_{k=1}^{p}\alpha_k^{R_{(k,s)}}\in\prod_{k=1}^{p}[-1,1]^{R_{(k,s)}}$. Consequently, if the solved constraint in (3.78) has feasible values that are outside the domain $\prod_{k=1}^{p}[-1,1]^{R_{(k,s)}}$

$$\left\{\frac{1}{A_{(r,s)}}\left(-\sum_{i\in\mathcal{H}}\left(\prod_{k=1}^{p}\alpha_k^{R_{(k,i)}}\right)A_{(r,i)}+b_{(r)}\right)\;\middle|\;\alpha_k\in[-1,1]\right\}\not\subseteq\prod_{k=1}^{p}[-1,1]^{R_{(k,s)}},\tag{3.80}$$

constraint reduction results in an over-approximation. Let us demonstrate this by an example:

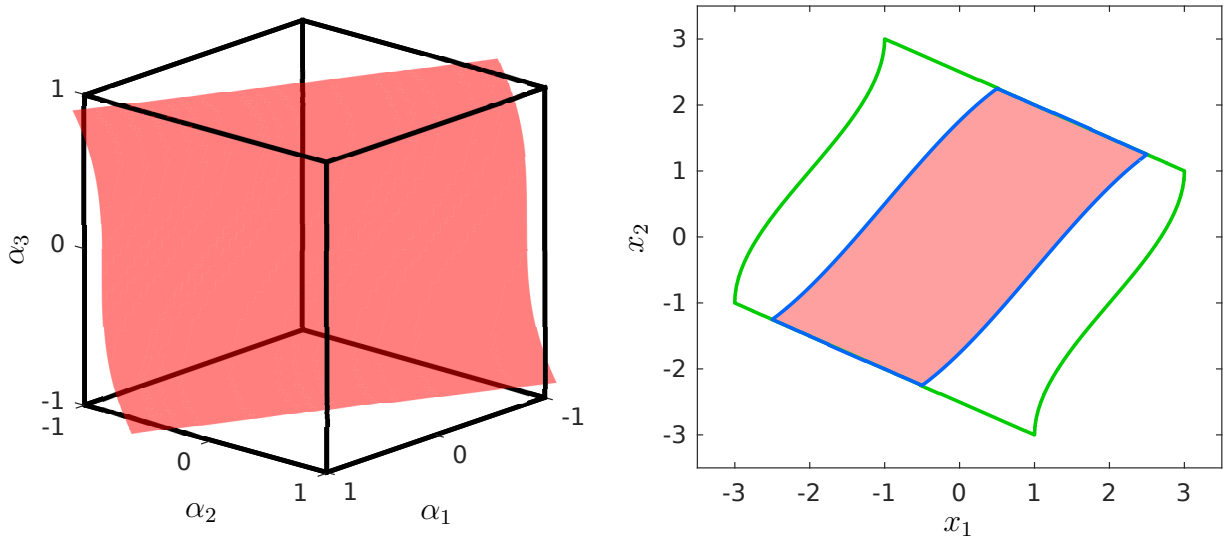

Figure 3.13: Visualization of constraint reduction using Prop. 3.2.31 for the CPZ $\mathcal{CPZ}$ from Example 3.2.32 (red, right), where the corresponding constraint is visualized on the left. While $\texttt{reduceCon}(\mathcal{CPZ}, 1, 1, 1)$ (green) results in an over-approximation, $\texttt{reduceCon}(\mathcal{CPZ}, 1, 2, 2)$ (blue) is exact.

Example 3.2.32. *We consider the CPZ*

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1.5 \\ 0 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 0.5 \end{bmatrix}, 0, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \right\rangle_{CPZ} =$$

$$\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \alpha_2 + \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \alpha_3 \;\middle|\; \alpha_1 + 2\alpha_2 + 0.5\alpha_3^3 = 0,\ \alpha_1, \alpha_2, \alpha_3 \in [-1, 1] \right\},$$

which is visualized in Fig. 3.13. We first choose the indices $s = 1$ and $d = 1$ that correspond to the monomial α_1 for constraint reduction. In this case, solving the constraint $\alpha_1 + 2\alpha_2 + 0.5\alpha_3^3 = 0$ for α_1 yields $\alpha_1 = -2\alpha_2 - 0.5\alpha_3^3$. As visible in Fig. 3.13 (left), the set of feasible values for the solved constraint has feasible values outside the domain $\alpha_1 \in [-1, 1]$:

$$\{-2\alpha_2 - 0.5\alpha_3^3 \mid \alpha_2, \alpha_3 \in [-1, 1]\} = [-2.5, 2.5] \not\subseteq [-1, 1].$$

Constraint reduction using Prop. 3.2.31 with the indices $s = 1$ and $d = 1$ therefore results in an over-approximation $\texttt{reduceCon}(\mathcal{CPZ}, 1, 1, 1) \supset \mathcal{CPZ}$ (see Fig. 3.13 (right)). Next, we consider the indices $s = 2$ and $d = 2$ that correspond to the monomial α_2. Solving the constraint for α_2 yields $\alpha_2 = -0.5\alpha_1 - 0.25\alpha_3^3$. Since the set of feasible values for the solved constraint is a subset of the domain $\alpha_2 \in [-1, 1]$

$$\{-0.5\alpha_1 - 0.25\alpha_3^3 \mid \alpha_1, \alpha_3 \in [-1, 1]\} = [-0.75, 0.75] \subset [-1, 1],$$

constraint reduction using Prop. 3.2.31 with the indices $s = 2$ and $d = 2$ does not result in an over-approximation, so that $\texttt{reduceCon}(\mathcal{CPZ}, 1, 2, 2) = \mathcal{CPZ}$.

Computing the exact bounds for the solved constraint in (3.80) is in general computationally infeasible. Instead, one can use range bounding to compute over-approximations

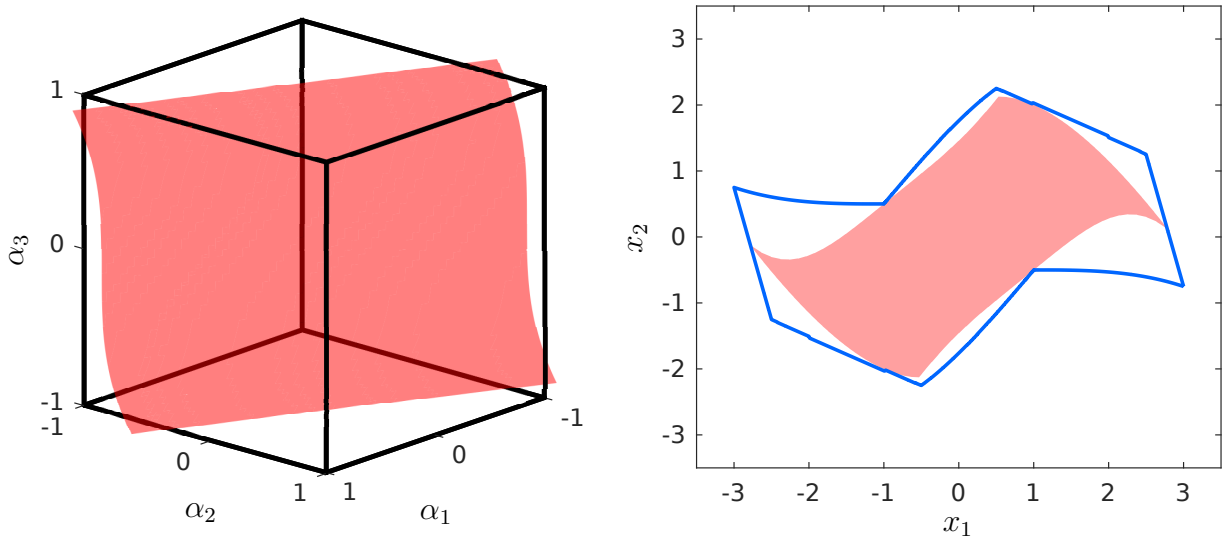

Figure 3.14: Visualization of constraint reduction using Prop. 3.2.31 for the CPZ $\mathcal{CPZ}$ from Example 3.2.33 (red, right), where the corresponding constraint is visualized on the left. Due to the loss of dependency, constraint reduction $\texttt{reduceCon}(\mathcal{CPZ}, 1, 2, 2)$ (blue) results in an over-approximation.

of the bounds. Another heuristic that we observed to perform well in practice is to first enclose the CPZ with a constrained zonotope using Prop. 3.2.13, and then select the constraint that is removed as well as the indices of the monomial that is used for reduction based on the constrained zonotope enclosure. For constrained zonotopes, sophisticated methods for selecting constraints and indices that result in the least over-approximation are available in [32, Appendix].

The second source contributing to the over-approximation during constraint reduction is the loss of dependency. This loss arises since if we substitute the monomial $\prod_{k=1}^{p} \alpha_k^{R_{(k,s)}}$ with the solved constraint in (3.78), the dependence between the factors α_k in $\prod_{k=1}^{p} \alpha_k^{R_{(k,s)}}$ and the factors α_k in other monomials gets lost. Let us demonstrate this by an example:

Example 3.2.33. *We consider the CPZ*

$$\mathcal{CPZ} = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1.5 & 0.5 \\ 0 & 1 & 2 & -2 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 0.5 \end{bmatrix}, 0, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \right\rangle_{CPZ},$$

which is identical to the CPZ from Example 3.2.32, except that we added an additional monomial $[0.5 \ -2]^T \alpha_2^2 \alpha_3$. *Since the polynomial constraint is identical to the constraint of the CPZ in Example 3.2.32, constraint reduction using the indices* $s = 2$ *and* $d = 2$ *does not result in an over-approximation due to lost bounds, as we demonstrated in Example 3.2.32. However, with the indices* $s = 2$ *and* $d = 2$, *we substitute* α_2 *by the solved constraint* $\alpha_2 = -0.5\alpha_1 - 0.25\alpha_3^3$, *so that the dependence between* α_2 *in the second monomial* $[0 \ 1]^T \alpha_2$ *and* α_2 *in the additional monomial* $[0.5 \ -2]^T \alpha_2^2 \alpha_3$ *gets lost. Due to this loss of dependency, constraint reduction using Prop. 3.2.31 with indices* $s = 2$ *and* $d = 2$ *results in an over-approximation, as visualized in Fig. 3.14.*

In the above example we could prevent the loss of dependency by substituting α_2^2 with $\alpha_2^2 = (-0.5\alpha_1 - 0.25\alpha_3^3)^2$, which corresponds to the square of the solved constraint $\alpha_2 = -0.5\alpha_1 - 0.25\alpha_3^3$. Similarly, it is possible to relax the constraint in (3.76) to

$$\forall i \in \{1, \dots, p\} : \; R^e_{(i,s)} = E_{(i,d)} \;\text{ and }\; A_{(r,s)} \neq 0,$$

which allows powers of the selected monomial with arbitrary exponents $e \in \mathbb{N}$. While this relaxation often allows us to reduce constraints with less over-approximation, taking powers of the solved constraint significantly increases the number of generators of the resulting CPZ, and therefore also the computational complexity for the subsequent application of the `compactGen` operation.

Now, we show how to obtain the CPZ corresponding to a subset of the factor domain, which is crucial for the `rescale` operation as defined in Prop. 3.2.7:

Proposition 3.2.34. *(Get Subset) Given a CPZ $\mathcal{CPZ} = \langle c, G, E, A, b, R\rangle_{CPZ} \subset \mathbb{R}^n$, the index of one factor $r \in \{1, \dots, p\}$, and an interval $[l, u] \subseteq [-1, 1]$, the operation* `getSubset` *substitutes the domain for the factor α_r with $\alpha_r \in [l, u]$, which yields a CPZ that is a subset of $\mathcal{CPZ}$:*

$$\texttt{getSubset}\big(\mathcal{CPZ}, r, [l, u]\big)$$

$$= \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\; \alpha_k \in [-1, 1],\; \alpha_r \in [l, u] \right\}$$

$$= \langle \overline{c}, \overline{G}, \overline{E}, \overline{A}, -\overline{b}, \overline{R} \rangle_{CPZ} \subseteq \mathcal{CPZ},$$

where $\overline{c}, \overline{G}, \overline{E}, \overline{A}, \overline{b}, \overline{R}$ are computed using `getSubset` *for SPZs as defined in Prop. 3.1.43*

$$\langle \overline{c}, \overline{G}, [\,], \overline{E}, id\rangle_{PZ} = \texttt{getSubset}\big(\langle c, G, [\,], E, id\rangle_{PZ}, r, [l, u]\big)$$
$$\langle \overline{b}, \overline{A}, [\,], \overline{R}, id\rangle_{PZ} = \texttt{getSubset}\big(\langle -b, A, [\,], R, id\rangle_{PZ}, r, [l, u]\big),$$

and $id =$ $\texttt{uniqueID}(p)$. The resulting CPZ is regular and the computational complexity with respect to the dimension is $\mathcal{O}(n^3 \log(n))$.

Proof. Since a CPZ without constraints $\langle c, G, E, [\,], [\,], [\,]\rangle_{CPZ}$ is identical to a SPZ, we can apply the `getSubset` operation for SPZs to obtain the subset. Moreover, due to the similarity of the constraints of the CPZ

$$\mathbf{0} \in \left\{ -b + \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1, 1] \right\}$$

and the definition of a SPZs in Def. 3.1.1

$$x \in \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\},$$

we can apply the `getSubset` operation for SPZs with the substitutions $c = -b$, $G = A$ and $E = R$ to obtain the subset for the constraints.

Complexity: According to (3.43), the complexity of the `getSubset` operation for SPZs is $\mathcal{O}(h\epsilon(n + p\log(h\epsilon)))$, where ϵ is the maximum entry of the exponent matrix. When applying `getSubset` to the constraints of the CPZ, we have to insert the substitutions $n = m$ and $h = q$. This yields an overall complexity of

$$\mathcal{O}\big(h\epsilon(n + p\log(h\epsilon))\big) + \mathcal{O}\big(q\epsilon(m + p\log(q\epsilon))\big),$$

which is $\mathcal{O}(n^3\log(n))$ using Assumption 3.2.3. □

As described in Sec. 3.2.4, the `split` operation can be applied to improve the support function enclosure. In addition, splitting is also crucial for CPZ in CPZ containment checks, as we explained in Sec. 3.2.6. We use the following implementation of the `split` operation for CPZs:

Proposition 3.2.35. *(Split) Given a CPZ $\mathcal{CPZ} \subset \mathbb{R}^n$ and the index of one factor $r \in \{1, \dots, p\}$, $\texttt{split}(\mathcal{CPZ}, r) = (\mathcal{CPZ}_1, \mathcal{CPZ}_2)$ returns two CPZs*

$$\begin{aligned}\mathcal{CPZ}_1 &= \texttt{getSubset}(\mathcal{CPZ}, r, [-1, 0]),\\ \mathcal{CPZ}_2 &= \texttt{getSubset}(\mathcal{CPZ}, r, [0, 1])\end{aligned}$$

that satisfy $\mathcal{CPZ}_1 \cup \mathcal{CPZ}_2 = \mathcal{CPZ}$, where `getSubset` is defined as in Prop. 3.2.34. The computational complexity with respect to the dimension is $\mathcal{O}(n^3\log(n))$.

Proof. The `split` operation for CPZs is based on the substitution of the selected factor α_r with two new dependent factors $\alpha_{r,1}$ and $\alpha_{r,2}$:

$$\begin{aligned}\big\{\alpha_r \mid \alpha_r \in [-1, 1]\big\} =& \big\{\alpha_{r,1} + \alpha_{r,2} \mid \alpha_{r,1} \in [-1, 0],\ \alpha_{r,2} \in [0, 1]\big\}\\ & \big\{\alpha_{r,1} \mid \alpha_{r,1} \in [-1, 0]\big\} \cup \big\{\alpha_{r,2} \mid \alpha_{r,2} \in [0, 1]\big\}.\end{aligned} \tag{3.81}$$

Inserting this substitution into the definition of CPZs in Def. 3.2.1 yields

$$\mathcal{CPZ} = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [-1, 1] \right\} \overset{(3.81)}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1\\k\neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) (\alpha_{r,1} + \alpha_{r,2})^{E_{(r,i)}} G_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1, 1], \right.$$

$$\left. \sum_{i=1}^{q} \left(\prod_{\substack{k=1\\k\neq r}}^{p} \alpha_k^{R_{(k,i)}} \right) (\alpha_{r,1} + \alpha_{r,2})^{R_{(r,i)}} A_{(\cdot,i)} = b,\ \alpha_{r,1} \in [-1, 0],\ \alpha_{r,2} \in [0, 1] \right\} \overset{(3.81)}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1\\k\neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \alpha_{r,1}^{E_{(r,i)}} G_{(\cdot,i)} \;\middle|\; \right.$$

$$\underbrace{\left\{\sum_{i=1}^{q}\left(\prod_{\substack{k=1\\k\neq r}}^{p}\alpha_k^{R_{(k,i)}}\right)\alpha_{r,1}^{R_{(r,i)}}A_{(\cdot,i)} = b,\ \alpha_k\in[-1,1],\ \alpha_{r,1}\in[-1,0]\right\}}_{=\mathcal{CPZ}_1=\texttt{getSubset}(\mathcal{CPZ},r,[-1,0])}$$

$$\cup\left\{c+\sum_{i=1}^{h}\left(\prod_{\substack{k=1\\k\neq r}}^{p}\alpha_k^{E_{(k,i)}}\right)\alpha_{r,2}^{E_{(r,i)}}G_{(\cdot,i)}\ \middle|\ \underbrace{\sum_{i=1}^{q}\left(\prod_{\substack{k=1\\k\neq r}}^{p}\alpha_k^{R_{(k,i)}}\right)\alpha_{r,2}^{R_{(r,i)}}A_{(\cdot,i)} = b,\ \alpha_k,\beta_j\in[-1,1],\ \alpha_{r,2}\in[0,1]}_{=\mathcal{CPZ}_2=\texttt{getSubset}(\mathcal{CPZ},r,[0,1])}\right\},$$

which concludes the proof.

Complexity: The operation `getSubset` has complexity $\mathcal{O}(n^3\log(n))$ according to Prop. 3.2.34. For the `split` operation `getSubset` has to be applied two times, resulting in a complexity of $2\cdot\mathcal{O}(n^3\log(n)) = \mathcal{O}(n^3\log(n))$. □

As for SPZs, the `split` operation for CPZs is not exact so that the resulting sets usually overlap. It is therefore desirable to choose the factor α_r that minimizes the size of the overlapping area for the split. While finding the optimal factor is computational infeasible for general CPZs, one heuristic that we observed to perform well is to select the factor α_r that corresponds to the longest generator of the lifted polynomial zonotope as defined in Lemma 3.2.4.

3.3 Z-Representation of Polytopes

As we demonstrated in Sec. 3.1.3, every bounded polytope can be equivalently represented as a polynomial zonotope. Motivated by this fundamental result, we present in this section[3] the *Z-representation* of polytopes, which stores the polynomial zonotope representation of a polytope in a very efficient way. The structure of the section is as follows: After introducing the Z-representation in Sec. 3.3.1, we present in Sec. 3.3.2 algorithms for the conversion between the Z-representation and other set representations, such as the V-representation of polytopes. Next, we derive basic set operations on the Z-representation in Sec. 3.3.3 and provide a criterion to check if a set in Z-representation is a polytope in Sec. 3.3.4. Finally, in Sec. 3.3.5, we compare the representation size of our novel Z-representation with other polytope representations. A summary of all operations on the Z-representation presented in this thesis is shown in Tab. 3.11.

3.3.1 Definition

Let us first define the Z-representation of bounded polytopes:

Definition 3.3.1. *(Z-Representation) Given a constant offset $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times h}$, the Z-representation defines the set*

$$\mathcal{P} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{(i,k)}} \in [-1, 1] \right\},$$

where the tuple $\mathbf{E} = (e_1, \ldots, e_h)$ storing the factor indices satisfies

$$\forall i \in \{1, \ldots, h\} : \; e_i \in \mathbb{N}_{\leq p}^{m_i},$$

$$\forall i \in \{1, \ldots, h\}, \; \forall j, k \in \{1, \ldots, m_i\} : \; (j \neq k) \Rightarrow \left(\mathbf{E}_{(i,j)} \neq \mathbf{E}_{(i,k)} \right).$$

The scalars $\alpha_{\mathbf{E}_{(i,k)}}$ are called factors, p is the number of factors, m_i is the length of the i-th element of $\mathbf{E}$, and h is the number of generators. The overall number of entries in $\mathbf{E}$ is

$$\mu = \sum_{i=1}^{h} m_i.$$

The Z-representation is regular if the tuple $\mathbf{E} = (e_1, \ldots, e_h)$ does not contain duplicate entries:

$$\forall i, j \in \{1, \ldots, h\} : \; (i \neq j \wedge m_i = m_j) \Rightarrow (e_i \neq e_j).$$

For a concise notation, we introduce the shorthand $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z$.

The Z-representation defines a special type of polynomial zonotope (see Def. 3.1.1) where the exponents of the factors α_k are restricted to the values 0 and 1. Through this restriction, the Z-representation requires less scalar numbers to represent a bounded polytope compared to other parameterizations of polynomial zonotopes. While every bounded polytope can be equivalently represented by the Z-representation, not every set defined by a Z-representation is a bounded polytope, as we illustrate by the following examples:

[3] This section is based on [85].

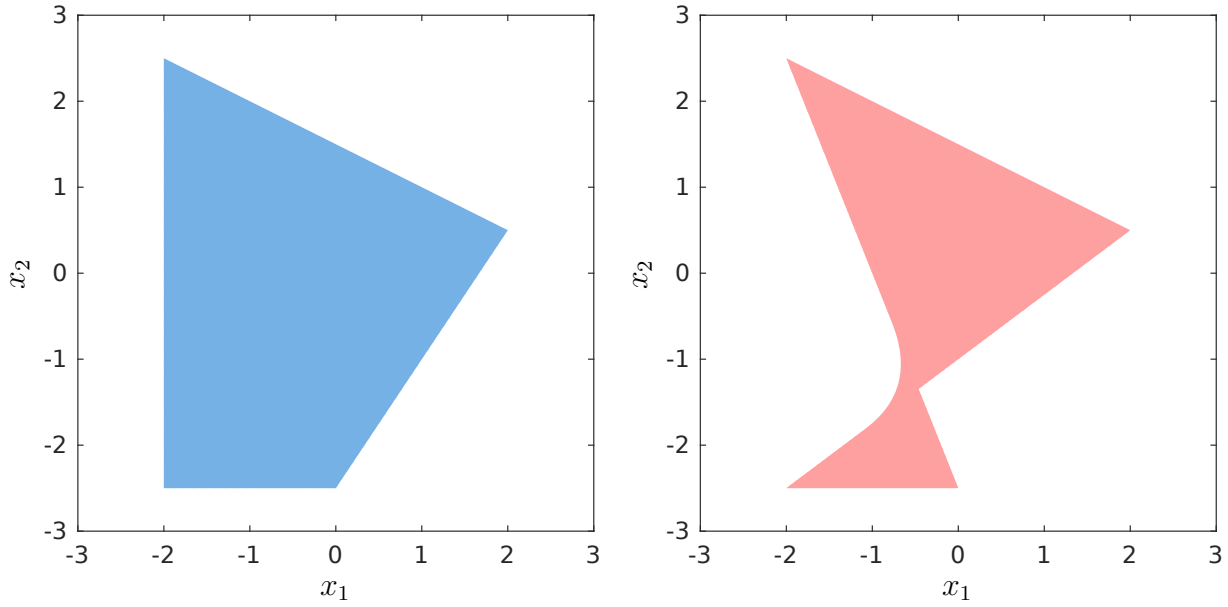

Figure 3.15: Visualization of the sets defined by the Z-representations from Example 3.3.2 (left) and from Example 3.3.3 (right).

Example 3.3.2. *The Z-representation*

$$\mathcal{P} = \left\langle \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 1.5 & -0.5 & -0.5 \\ -0.5 & -2 & 0.5 \end{bmatrix}, \left(1, 2, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \right\rangle_Z$$

defines the polytope

$$\mathcal{P} = \left\{ \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \alpha_1 + \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \alpha_2 + \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \alpha_1\alpha_2 \;\middle|\; \alpha_1, \alpha_2 \in [-1, 1] \right\},$$

which is visualized in Fig. 3.15 (left).

Example 3.3.3. *The Z-representation*

$$\mathcal{P} = \left\langle \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} -0.5 & -0.5 & 1.5 \\ 0.5 & -2 & -0.5 \end{bmatrix}, \left(1, 2, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \right\rangle_Z$$

defines the set

$$\mathcal{P} = \left\{ \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \alpha_1 + \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \alpha_2 + \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \alpha_1\alpha_2 \;\middle|\; \alpha_1, \alpha_2 \in [-1, 1] \right\},$$

which is not a polytope as shown in Fig. 3.15 (right).

A criterion to check if a set in Z-representation is a polytope is presented later in Sec. 3.3.4.

Table 3.11: Overview showing all set operations on the Z-representation presented in this thesis.

Set Operation	Reference	Page
Conversion V-representation to Z-representation	Alg. 4	123
Conversion Z-representation to V-representation	Alg. 5	128
Conversion Zonotope to Z-representation	Prop. 3.3.11	130
Conversion Z-representation to SPZ	Prop. 3.3.12	131
Linear map	Prop. 3.3.13	132
Minkowski sum	Prop. 3.3.14	132
Cartesian product	Prop. 3.3.15	133
Convex hull	Prop. 3.3.16	134
Polytope test	Prop. 3.3.18	137

3.3.2 Conversion from and to other Set Representations

Since none of the existing polytope representations permits the efficient computation of all basic set operations, algorithms for the conversion between different representations are important. In this section we present algorithms for converting polytopes from V-representation to Z-representation, and for the conversion from Z-representation to V-representation. Moreover, we show how to convert zonotopes to Z-representation, and how to convert a set in Z-representation to a SPZ. Let us begin with the formulation and proof of the main theorem for the Z-representation:

Theorem 3.3.4. *Every bounded polytope can be equivalently represented by the Z-representation.*

Proof. If the polytope is bounded, it can be described as the convex hull of its vertices (see Def. 2.2.3). Each vertex $v_i \in \mathbb{R}^n$ can be equivalently represented by the Z-representation $v_i = \langle v_i, [\,], \emptyset \rangle_Z$. Since the Z-representation is closed under the convex hull operation as shown later in Prop. 3.3.16, computation of the convex hull of all vertices results in a Z-representation of the polytope. □

Next, we provide an algorithm for the conversion of a polytope in V-representation to Z-representation in Alg. 4. The algorithm is structured as follows: First, all vertices of the polytope are converted to Z-representation during the for-loop in lines 2-4 and the result is stored in the tuple $\mathbf{K}$. The remainder of the algorithm can then be viewed as the exploration of a binary tree as shown in Fig. 3.17, where the nodes of the tree are polytopes in Z-representation. Each iteration of the outer while-loop in lines 5-19 of Alg. 4 represents the exploration of one level of the tree. For each of these levels, the inner while-loop in lines 7-14 visits all nodes at one level and computes the convex hull of two nodes using Prop. 3.3.16 to construct one node at the next higher level of the tree. If the root node of the binary tree is reached, all polytope vertices have been united by the convex hull, so that the root element is the desired Z-representation of the polytope $\mathcal{P}$.

Algorithm 4 Conversion from V-representation to Z-representation

Require: Bounded polytope in V-representation $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V$
Ensure: Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z$ of the polytope

1: $\mathbf{K} \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ to s **do**
3: $\quad \mathbf{K} \leftarrow \left(\mathbf{K},\ \langle v_i, [\,], \emptyset \rangle_Z\right)$
4: **end for**
5: **while** $|\mathbf{K}| > 1$ **do**
6: $\quad \widehat{\mathbf{K}} \leftarrow \emptyset$
7: $\quad$ **while** $|\mathbf{K}| \geq 2$ **do**
8: $\quad\quad \widehat{\mathbf{K}} \leftarrow \left(\widehat{\mathbf{K}}\ , conv(\mathbf{K}_{(1)}, \mathbf{K}_{(2)})\right)$ (convex hull computed using Prop. 3.3.16)
9: $\quad\quad$ **if** $|\mathbf{K}| > 2$ **then**
10: $\quad\quad\quad \mathbf{K} \leftarrow \left(\mathbf{K}_{(3)}, \dots, \mathbf{K}_{(|\mathbf{K}|)}\right)$
11: $\quad\quad$ **else**
12: $\quad\quad\quad \mathbf{K} \leftarrow \emptyset$
13: $\quad\quad$ **end if**
14: $\quad$ **end while**
15: $\quad$ **if** $|\mathbf{K}| = 1$ **then**
16: $\quad\quad \widehat{\mathbf{K}} \leftarrow \left(\widehat{\mathbf{K}},\ \mathbf{K}_{(1)}\right)$
17: $\quad$ **end if**
18: $\quad \mathbf{K} \leftarrow \widehat{\mathbf{K}}$
19: **end while**
20: $\langle c, G, \mathbf{E} \rangle_Z \leftarrow \mathbf{K}_{(1)}$

We demonstrate Alg. 4 by an example:

Example 3.3.5. *The conversion of the polytope*

$$\mathcal{P} = \left\langle \left[\begin{bmatrix} 1 \\ 6 \end{bmatrix} \ \begin{bmatrix} 4 \\ 7 \end{bmatrix} \ \begin{bmatrix} 5 \\ 6 \end{bmatrix} \ \begin{bmatrix} 6 \\ 2 \end{bmatrix} \ \begin{bmatrix} 3 \\ 1 \end{bmatrix} \ \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right] \right\rangle_V$$

from V-representation to Z-representation using Alg. 4 is visualized in Fig. 3.16. The algorithm terminates after 3 *iterations, where after the first iteration we obtain the tuple* $\mathbf{K} = (\mathcal{P}_1^{(1)}, \mathcal{P}_2^{(1)}, \mathcal{P}_3^{(1)})$ *in Line 18 of Alg. 4 with*

$$\mathcal{P}_1^{(1)} = \left\langle \begin{bmatrix} 2.5 \\ 6.5 \end{bmatrix}, \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix}, (1) \right\rangle_Z,\ \mathcal{P}_2^{(1)} = \left\langle \begin{bmatrix} 5.5 \\ 4 \end{bmatrix}, \begin{bmatrix} -0.5 \\ 2 \end{bmatrix}, (1) \right\rangle_Z,\ \mathcal{P}_3^{(1)} = \left\langle \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, (1) \right\rangle_Z,$$

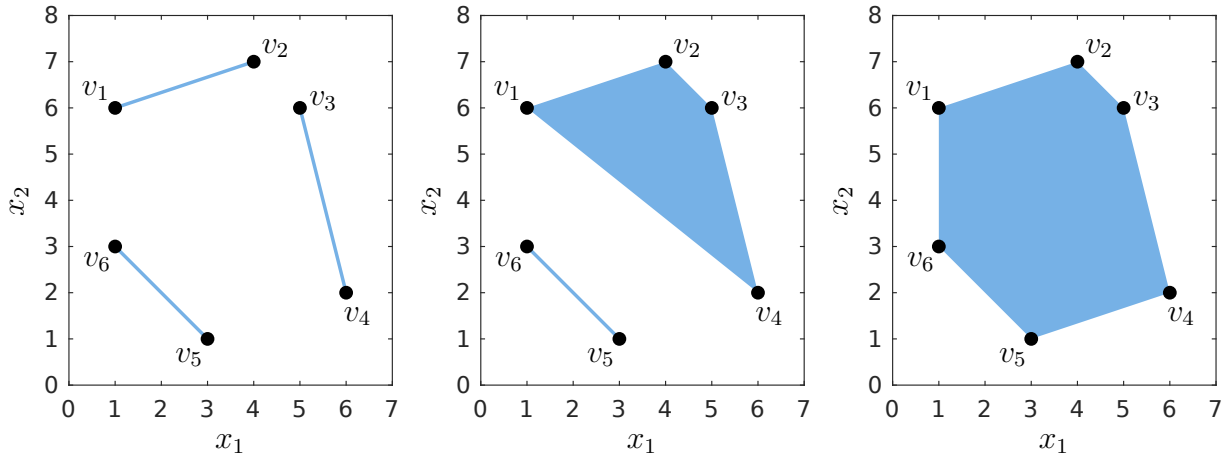

Figure 3.16: First (left), second (middle), and third (right) iteration of Alg. 4 applied to the polytope $\mathcal{P}$ from Example 3.3.5.

after the second iteration we obtain $\mathbf{K} = (\mathcal{P}_1^{(2)}, \mathcal{P}_2^{(2)})$ *with*

$$\mathcal{P}_1^{(2)} = \left\langle \begin{bmatrix} 4 \\ 5.25 \end{bmatrix}, \begin{bmatrix} -1.5 & -0.75 & -0.75 & -0.25 & 0.25 \\ 1.25 & -0.25 & -0.25 & 1 & -1 \end{bmatrix}, \left(3, 1, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, 2, \begin{bmatrix} 2 \\ 3 \end{bmatrix}\right) \right\rangle_Z, \quad \mathcal{P}_2^{(2)} = \mathcal{P}_3^{(1)},$$

and after the third iteration we obtain the final result

$$\mathcal{P} = \left\langle \begin{bmatrix} 3 \\ 3.6125 \end{bmatrix}, 0.5\ G, \left(5, 3, 1, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, 2, \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}, 4, \begin{bmatrix} 4 \\ 5 \end{bmatrix}\right) \right\rangle_Z,$$

where

$$G = \begin{bmatrix} 2 & -1.5 & -0.75 & -0.75 & -0.25 & 0.25 & -1.5 & -0.75 & -0.75 & -0.25 & 0.25 & 1 & -1 \\ 3.25 & 1.25 & -0.25 & -0.25 & 1 & -1 & 1.25 & -0.25 & -0.25 & 1 & -1 & -1 & 1 \end{bmatrix}$$

is the generator matrix.

The Z-representation of a polytope is not unique. If Alg. 4 is used for the conversion of a polytope given in V-representation, then the resulting Z-representation depends on the order of the vertices in the matrix $[v_1 \ \dots \ v_s]$, since this order defines which vertices are combined by the convex hull operation. Therefore, it might be meaningful to sort the matrix $[v_1 \ \dots \ v_s]$ before applying Alg. 4 in order to obtain a Z-representation with desirable properties. For example, to minimize the length of the vectors in the generator matrix of the Z-representation, the vertices have to be sorted such that vertices located close to each other are combined first. The computational complexity for the conversion with Alg. 4 is as follows:

Proposition 3.3.6. *The computational complexity of the conversion from V-representation to Z-representation using Alg. 4 is* $\mathcal{O}(s^2(n + \log(s)))$ *with respect to the number of polytope vertices* s*, where* n *is the dimension of the polytope.*

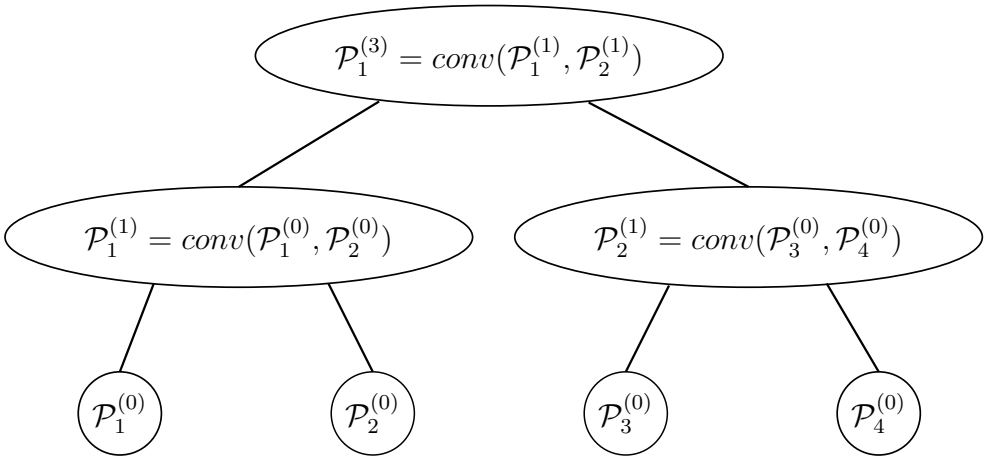

Figure 3.17: Example of a binary tree as explored by Alg. 4.

Proof. The for-loop in lines 2-4 of Alg. 4 can be ignored since it only involves initializations. Let us first consider the case where $s = 2^k$, $k \in \mathbb{N}$. Then each iteration of the outer while-loop in lines 5-19 of Alg. 4 corresponds to one level of a perfect binary tree with depth $k = \log(s)$ (see Fig. 3.17). Each node at level $i = 0, \ldots, k$ is a polytope in Z-representation $\mathcal{P}^{(i)} = \langle c^{(i)}, G^{(i)}, \mathbf{E}^{(i)} \rangle_Z$. From Tab. 3.12 we can derive the number of factors $p^{(i)}$, the number of generators $h^{(i)}$ and the number of entries $\mu^{(i)}$ in the tuple $\mathbf{E}^{(i)}$ of a node at level i of the binary tree; the values for a perfect binary tree on level i are:

$$\begin{aligned} p^{(i)} &= 2p^{(i-1)} + 1 = 2^i p^{(0)} + \sum_{j=0}^{i-1} 2^j, \quad h^{(i)} = 4h^{(i-1)} + 1 = 4^i h^{(0)} + \sum_{j=0}^{i-1} 4^j, \\ \mu^{(i)} &= 4\mu^{(i-1)} + 2h^{(i-1)} + 1 = 4^i \mu^{(0)} + \sum_{j=0}^{i-1} 4^j \left(1 + 2h^{(i-1-j)}\right). \end{aligned} \tag{3.82}$$

The nodes at the bottom level of the binary tree are the polytope vertices v_l represented in Z-representation $v_l = \langle v_l, [\,], \emptyset \rangle_Z$, so that $p^{(0)} = 0$, $h^{(0)} = 0$, and $\mu^{(0)} = 0$. Inserting these values into (3.82) and using the finite sum of the geometric series [86, Ch. 1.2.2.2]

$$\sum_{j=0}^{z} r^j = \frac{1 - r^{z+1}}{1 - r}, \quad r \in \mathbb{R},\ r \neq 1, \tag{3.83}$$

we obtain

$$\begin{aligned} p^{(i)} &= \sum_{j=0}^{i-1} 2^j \overset{(3.83)}{=} 2^i - 1, \quad h^{(i)} = \sum_{j=0}^{i-1} 4^j \overset{(3.83)}{=} \frac{4^i - 1}{3}, \\ \mu^{(i)} &= \sum_{j=0}^{i-1} 4^j \left(1 + \frac{2}{3}\left(4^{i-1-j} - 1\right)\right) = \frac{1}{3} \sum_{j=0}^{i-1} 4^j + \frac{2}{3} \sum_{j=0}^{i-1} 4^{i-1} \overset{(3.83)}{=} 4^i \left(\frac{1}{6} i + \frac{1}{9}\right) - \frac{1}{9}. \end{aligned} \tag{3.84}$$

Each level i of the tree contains 2^{k-i} nodes, where k is the depth of the tree. Consequently, at each level 2^{k-i} convex hulls have to be computed using Prop. 3.3.16, where each convex

hull involves according to (3.103) $4n+4nh^{(i-1)}+\mu^{(i-1)}$ elementary operations. The number of elementary operations O required for the conversion with Alg. 4 is therefore

$$O = \sum_{i=1}^{k} 2^{k-i} \left(4n + 4nh^{(i-1)} + \mu^{(i-1)}\right). \tag{3.85}$$

In the general case, the binary tree explored by Alg. 4 is not a perfect binary tree. However, the number of operations required for the general case is obviously smaller than the number of operations required for the exploration of a perfect binary tree with depth $\lceil \log(s) \rceil$. Since it holds that $\lceil \log(s) \rceil = \log(s) + a$ for some $a \in [0, 1]$, inserting $k = \lceil \log(s) \rceil = \log(s) + a$ for the tree depth in (3.85) yields

$$\begin{aligned}
O &= \sum_{i=1}^{\log(s)+a} 2^{\log(s)+a-i} \left(4n + 4nh^{(i-1)} + \mu^{(i-1)}\right) = 2^a \sum_{i=1}^{\log(s)+a} \frac{s}{2^i} \left(4n + 4nh^{(i-1)} + \mu^{(i-1)}\right) \\
&\overset{(3.84)}{=} 2^a \sum_{i=1}^{\log(s)+a} \frac{s}{2^i} \left(4n + \frac{4}{3}n\left(4^{i-1} - 1\right) + 4^{i-1}\left(\frac{1}{6}(i-1) + \frac{1}{9}\right) - \frac{1}{9}\right) \\
&= 2^a s \Bigg(\left(\frac{8}{3}n - \frac{1}{9}\right) \underbrace{\sum_{i=1}^{\log(s)+a} \frac{1}{2^i}}_{\overset{(3.83)}{=} 1 - \frac{1}{2^a s}} + \left(\frac{1}{3}n - \frac{1}{72}\right) \underbrace{\sum_{i=1}^{\log(s)+a} 2^i}_{\overset{(3.83)}{=} 2(2^a s - 1)} + \frac{1}{24} \underbrace{\sum_{i=1}^{\log(s)+a} i \cdot 2^i}_{\substack{\overset{(3.87)}{=} 2 \cdot (1 + 2^a s(a-1) \\ + 2^a \log(s) s)}} \Bigg) \\
&= \frac{4^a}{12} s^2 \log(s) + 4^a \left(\frac{1}{12}a + \frac{2}{3}n - \frac{1}{9}\right) s^2 + 2^{(a+1)} n s - \frac{8}{3}n + \frac{1}{9},
\end{aligned} \tag{3.86}$$

where we used the finite sum of the arithmetic-geometric series [86, Ch. 1.2.2.3]

$$\sum_{i=1}^{z} i \cdot r^i = r \cdot \frac{1 - (z+1)r^z + z r^{z+1}}{(1-r)^2}, \quad r \in \mathbb{R},\ r \neq 1. \tag{3.87}$$

Since $a \in [0, 1]$, the complexity of Alg. 4 with respect to the number of polytope vertices s is therefore $\mathcal{O}(s^2(n + \log(s)))$ according to (3.86). □

Next, we present an algorithm that converts a polytope in Z-representation to V-representation. Our algorithm is based on the following proposition:

Proposition 3.3.7. *Given a polytope $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z \subset \mathbb{R}^n$ in Z-representation, it holds that the polytope vertices are a subset of the finite set $\mathcal{K}$*

$$\texttt{vertices}(\mathcal{P}) \subseteq \mathcal{K},$$

where $\mathcal{K}$ is defined as

$$\mathcal{K} = \Bigg\{ \underbrace{c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \widehat{\alpha}_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)}}_{f_{\langle c,G,\mathbf{E}\rangle}(\widehat{\alpha})} \;\Bigg|\; \widehat{\alpha} = [\widehat{\alpha}_1 \;\; \dots \;\; \widehat{\alpha}_p]^T \in \texttt{vertices}(\mathcal{I}) \Bigg\}, \tag{3.88}$$

and $\mathcal{I} = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$.

Proof. According to the definition of the Z-representation in Def. 3.3.1, the polynomial function $f_{\langle c,G,\mathbf{E}\rangle} : \mathbb{R}^p \to \mathbb{R}^n$ in (3.88) maps a vector of factors $\alpha \in \mathbb{R}^p$ to a corresponding point $f_{\langle c,G,\mathbf{E}\rangle}(\alpha) \in \mathcal{P}$. We therefore have to show that each vertex v_j of the polytope $\mathcal{P}$ corresponds to one vertex $\widehat{\alpha}^{(j)}$ of the interval $\mathcal{I}$:

$$v_j = f_{\langle c,G,\mathbf{E}\rangle}(\widehat{\alpha}^{(j)}). \tag{3.89}$$

As shown in [87, Prop. 7.2(d)], for each vertex v_j of a polytope $\mathcal{P}$ there exists a vector $d_j \in \mathbb{R}^n$ such that

$$v_j = \arg\max_{x \in \mathcal{P}} \;\; d_j^T x. \tag{3.90}$$

With the relation in (3.89), (3.90) can be equivalently formulated as

$$v_j = f_{\langle c,G,\mathbf{E}\rangle}(\alpha^*), \qquad \alpha^* = \arg\max_{\alpha \in \mathcal{I}} \;\; \underbrace{d_j^T \; f_{\langle c,G,\mathbf{E}\rangle}(\alpha)}_{g(\alpha_1,\dots,\alpha_p)}, \tag{3.91}$$

where $\alpha = [\alpha_1 \;\; \dots \;\; \alpha_p]^T$. Consequently, we have to show that the point α^* where the polynomial function $g : \mathbb{R}^p \to \mathbb{R}$ in (3.91) reaches its extremum within the domain $\alpha \in \mathcal{I}$ is identical to a vertex $\widehat{\alpha}^{(j)}$ of the interval $\mathcal{I}$. Since $g(\alpha_1, \dots, \alpha_p)$ results from the Z-representation it doesn't contain polynomial exponents greater than 1, so that the partial derivative with respect to a variable α_i does not depend on α_i

$$\forall i \in \{1, \dots, p\} : \;\; \frac{\partial g(\alpha_1, \dots, \alpha_p)}{\partial \alpha_i} = \widehat{g}_i(\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_p), \tag{3.92}$$

which corresponds to a constant gradient. According to (3.92), the function $g(\alpha_1, \dots, \alpha_p)$ therefore reaches its extremum on the domain $\alpha_i \in [-1, 1]$ at either $\alpha_i^* = 1$ or $\alpha_i^* = -1$. Since this holds for all α_i, $i = 1, \dots, p$, function $g(\alpha_1, \dots, \alpha_p)$ reaches its extremum within the domain $\alpha \in \mathcal{I}$ at the point $\alpha^* = [\alpha_1^* \;\; \dots \;\; \alpha_p^*]^T$ with $\alpha_j^* \in \{-1, 1\}$, $j = 1, \dots, p$, which is a vertex of the interval $\mathcal{I}$. □

Based on Prop. 3.3.7, we now present an algorithm for the conversion form Z-representation to V-representation in Alg. 5. The algorithm is structured as follows: We first compute the vertices of the interval $\mathcal{I}$ in Line 2 of Alg. 5. In the for-loop in lines 4-9 we then calculate the corresponding potential polytope vertex for each of the 2^p interval vertices according to Prop. 3.3.7. We check if the potential vertex v is already part of the set $\mathcal{K}$ in Line 6 since this decreases the average runtime of the algorithm. The points stored in $\mathcal{K}$ define a potentially redundant V-representation of the polytope $\mathcal{P}$. Redundant points are removed by computation of the convex hull in Line 10 of Alg. 5, where the

Algorithm 5 Conversion from Z-representation to V-representation

Require: Bounded polytope in Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z$
Ensure: V-representation $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V$ of the polytope

1: $\mathcal{I} \leftarrow [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$
2: $\left\{ \widehat{\alpha}^{(1)}, \dots, \widehat{\alpha}^{(2^p)} \right\} \leftarrow \texttt{vertices}(\mathcal{I})$
3: $\mathcal{K} \leftarrow \emptyset$
4: **for** $j \leftarrow 1$ to 2^p **do**
5: $\quad v \leftarrow f_{\langle c,G,\mathbf{E} \rangle}(\widehat{\alpha}^{(j)})$ ($f_{\langle c,G,\mathbf{E} \rangle}(\alpha)$ defined as in (3.88))
6: $\quad$ **if** $v \notin \mathcal{K}$ **then**
7: $\quad\quad \mathcal{K} \leftarrow \mathcal{K} \cup v$
8: $\quad$ **end if**
9: **end for**
10: $[v_1 \ \dots \ v_s] \leftarrow \texttt{convHull}([\widehat{v}_1 \ \dots \ \widehat{v}_{|\mathcal{K}|}])$, where $\mathcal{K} = \{\widehat{v}_1, \dots, \widehat{v}_{|\mathcal{K}|}\}$
11: $\mathcal{P} \leftarrow \langle [v_1 \ \dots \ v_s] \rangle_V$

operation `convHull` as defined in Def. 2.4.5 returns the convex hull of a point cloud. For Z-representations that define non-convex sets (see Example 3.3.3), Alg. 5 returns the convex hull of the set. Since it is required for the derivation of the computation complexity of Alg. 5, we first derive a formula for the maximum number of generators and the maximum number of tuple entries for a regular Z-representation with a fixed number of factors:

Lemma 3.3.8. *Given a regular Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z$ with p factors, it holds that*

$$h \leq 2^p - 1 \quad \textit{and} \quad \mu \leq p \ 2^{p-1}$$

are upper bounds for the number of generators h of $\mathcal{P}$ and the number of entries μ in the tuple $\mathbf{E}$.

Proof. For the proof, we require the identities [88, Ch. 8.6, Eq. (7)]

$$\sum_{i=0}^{z} \binom{z}{i} = 2^z, \quad z \in \mathbb{N} \tag{3.93}$$

and [88, Ch. 8.6, Eq. (8)]

$$\sum_{i=0}^{z} i \binom{z}{i} = z \ 2^{z-1}, \quad z \in \mathbb{N}. \tag{3.94}$$

For a regular Z-representation (see Def. 3.3.1), the maximum number of generators is equal to the number of different monomials $\alpha_{\mathbf{E}_{(i,1)}} \cdot \ldots \cdot \alpha_{\mathbf{E}_{(i,m_i)}}$ that can be constructed with p factors $\alpha_{\mathbf{E}_{(i,k)}}$. Given a fixed m_i, there exist $\binom{p}{m_i}$ different monomials since there are $\binom{p}{m_i}$ possible combinations to choose m_i of the p factors $\alpha_{\mathbf{E}_{(i,k)}}$ without order. Summation over

all $m_i \in \{1, \dots, p\}$ therefore yields

$$h \leq \sum_{m_i=1}^{p} \binom{p}{m_i} = \left(\sum_{m_i=0}^{p} \binom{p}{i} \right) - 1 \overset{(3.93)}{=} 2^p - 1.$$

For each monomial, m_i entries have to be stored in the tuple $\mathbf{E}$. Consequently,

$$\mu \leq \sum_{m_i=1}^{p} m_i \binom{p}{m_i} = \sum_{m_i=0}^{p} m_i \binom{p}{m_i} \overset{(3.94)}{=} p\ 2^{p-1}$$

is an upper bound for the number of entries μ in the tuple $\mathbf{E}$. □

Using Lemma 3.3.8, we can now derive the computational complexity of Alg. 5:

Proposition 3.3.9. *The computational complexity of the conversion from regular Z-representation to V-representation using Alg. 5 is $\mathcal{O}(2^{p\lfloor n/2 \rfloor} + 4^p(p+n))$ with respect to the number of factors p of the Z-representation, where n is the dimension of the polytope.*

Proof. The interval $\mathcal{I} = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$ has 2^p vertices. Consequently, the for-loop in lines 4-9 of Alg. 5 consists of 2^p iterations. In each iteration, the potential vertex v is calculated in Line 5 using the function $f_{\langle c,G,\mathbf{E} \rangle}(\alpha)$ as defined in (3.88). This requires nh additions and at most $\mu - 1 + nh$ multiplications, resulting for all iterations in

$$\begin{aligned} O_1 = 2^p \left(2nh + \mu - 1\right) &\overset{\text{Lemma 3.3.8}}{\leq} 2^p \big(2n(2^p - 1) + p2^{p-1} - 1\big) \\ &= 4^p \left(0.5p + 2n\right) - 2^p(2n + 1) \end{aligned}$$

elementary operations. Moreover, the containment check $v \notin \mathcal{K}$ in Line 6 of Alg. 5 requires in the worst case

$$O_2 = \sum_{i=1}^{2^p} n(i-1) = -2^p n + n \sum_{i=1}^{2^p} i \overset{(3.95)}{=} -2^p n + \frac{n}{2}(4^p + 2^p) = \frac{n}{2}\left(4^p - 2^p\right)$$

comparisons of scalar numbers, where we used the finite sum of the arithmetic series [86, Ch. 1.2.2.1]

$$\sum_{i=1}^{z} i = \frac{z^2 + z}{2}. \tag{3.95}$$

The complexity of the computations in the for-loop in lines 4-9 of Alg. 5 is therefore $\mathcal{O}(O_1 + O_2) = \mathcal{O}(4^p(p+n))$ with respect to p. Computation of the convex hull in Line 10 has complexity $\mathcal{O}(m^{\lfloor n/2 \rfloor + 1})$ according to Tab. 2.1, where m is the number of points in the point cloud. In our case, the point cloud stored in the set $\mathcal{K}$ consists of $m = 2^p$ points in the worst case, resulting in complexity $\mathcal{O}((2^p)^{\lfloor n/2 \rfloor + 1})$ with respect to p. Combining the complexities from the for-loop and from the convex hull computation results in an overall complexity of

$$\mathcal{O}(O_1 + O_2) + \mathcal{O}\big((2^p)^{\lfloor n/2 \rfloor + 1}\big) = \mathcal{O}\big(2^{p\lfloor n/2 \rfloor} + 4^p(p+n)\big)$$

with respect to p. □

For general Z-representations, it is not possible to specify a relation between the number of polytope vertices and the number of factors. However, under the assumption that the Z-representation is obtained by conversion from V-representation using Alg. 4, we can compute an upper bound for the number of factors from the number of vertices, which enables us to derive the computation complexity with respect to the number of polytope vertices:

Proposition 3.3.10. *Given a polytope in Z-representation which is obtained by conversion from V-representation using Alg. 4, the computational complexity of the conversion from Z-representation to V-representation using Alg. 5 is $\mathcal{O}(4^{s\lfloor n/2 \rfloor} + 16^s(s+n))$ with respect to the number of polytope vertices s, where n is the dimension of the polytope.*

Proof. We first determine a relation between the number of factors p of the Z-representation and the number of polytope vertices s, where we use the assumption that the Z-representation of the polytope is obtained from Alg. 4. As visualized in Fig. 3.17, conversion from V-representation to Z-representation using Alg. 4 can be viewed as the exploration of a binary tree with depth $k = \lceil \log(s) \rceil = \log(s) + a$ for some $a \in [0, 1]$. Using (3.84), we obtain for the number of factors at the k-th level of the binary tree

$$p = p^{(k)} \overset{(3.84)}{=} 2^k - 1 \overset{k=\log(s)+a}{=} 2^a s - 1, \tag{3.96}$$

which is identical to the number of factors of the resulting Z-representation. Inserting this result into the computational complexity of Alg. 5 as specified in Prop. 3.3.9 then yields

$$\mathcal{O}(2^{p\lfloor n/2 \rfloor} + 4^p(p+n)) \overset{(3.96)}{=} \mathcal{O}\big(4^{as\lfloor n/2 \rfloor} + 16^{as}(2^a s + n)\big),$$

which is $\mathcal{O}(4^{s\lfloor n/2 \rfloor} + 16^s(s+n))$ since $a \in [0, 1]$. □

After showing how to convert between Z-representation and V-representation, we now consider the conversion from and to other set representations. We first specify how to convert a zonotope to Z-representation:

Proposition 3.3.11. *(Conversion Zonotope) A zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ can be equivalently represented by the Z-representation*

$$\mathcal{Z} = \big\langle c, G, \underbrace{(1, \ldots, l)}_{\mathbf{E}} \big\rangle_Z, \tag{3.97}$$

where l is the number of zonotope generators. The computational complexity of the conversion is $\mathcal{O}(1)$.

Proof. With the tuple $\mathbf{E} = (1, \ldots, l)$ in (3.97) it holds that

$$\sum_{i=1}^{l} \alpha_i G_{(\cdot,i)} = \sum_{i=1}^{l} \Big(\prod_{k=1}^{1} \alpha_{\mathbf{E}(i,k)} \Big) G_{(\cdot,i)}. \tag{3.98}$$

Inserting (3.98) into the definition of a zonotope in Def. 2.2.4 then yields

$$\mathcal{Z} \overset{\text{Def. 2.2.4}}{=} \left\{ c + \sum_{i=1}^{l} \alpha_i G_{(\cdot,i)} \;\middle|\; \alpha_i \in [-1,1] \right\}$$
$$\overset{(3.98)}{=} \left\{ c + \sum_{i=1}^{l} \left(\prod_{k=1}^{1} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{(i,k)}} \in [-1,1] \right\} \overset{\text{Def. 3.3.1}}{=} \langle c, G, \underbrace{(1,\ldots,l)}_{\mathbf{E}} \rangle_Z,$$

which concludes the proof.

Complexity: The conversion only involves initialization and therefore has constant complexity $\mathcal{O}(1)$. □

Finally, we show how to convert a set in Z-representation to a SPZ:

Proposition 3.3.12. *(Conversion to SPZ) A set in Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z \subset \mathbb{R}^n$ can be equivalently represented by a SPZ*

$$\mathcal{P} = \big\langle c, G, [\,], E, \texttt{uniqueID}(p) \big\rangle_{PZ},$$

where

$$E_{(i,j)} = \begin{cases} 1, & \exists k \in \{1,\ldots,m_j\} : \; \mathbf{E}_{(j,k)} = i \\ 0, & \textit{otherwise} \end{cases}, \quad i = 1,\ldots,p, \;\; j = 1,\ldots,h. \tag{3.99}$$

The computational complexity of the conversion is $\mathcal{O}(\mu + p)$, where p is the number of factors of the Z-representation and μ is the number of entries in the tuple $\mathbf{E}$.

Proof. With the exponent matrix E defined as in (3.99) it holds that

$$\sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \overset{(3.99)}{=} \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)}. \tag{3.100}$$

Inserting this into the definition of the Z-representation in Def. 3.3.1 yields

$$\mathcal{P} \overset{\text{Def. 3.3.1}}{=} \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{(i,k)}} \in [-1,1] \right\} \overset{(3.100)}{=}$$
$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_k \in [-1,1] \right\} \overset{\text{Def. 3.1.1}}{=} \big\langle c, G, [\,], E, \texttt{uniqueID}(p) \big\rangle_{PZ},$$

which concludes the proof.

Complexity: Construction of the exponent matrix E in (3.99) has complexity $\mathcal{O}(\mu)$ and generation of p unique identifiers using `uniqueID` has complexity $\mathcal{O}(p)$ according to Tab. 2.2, which results in an overall complexity of $\mathcal{O}(\mu) + \mathcal{O}(p) = \mathcal{O}(\mu + p)$. □

Since any polytope in H-representation can be equivalently represented by the V-representation, Alg. 4 and Alg. 5 presented in this section can also be used to convert between the Z-representation and the H-representation. Moreover, since any interval can be equivalently represented by a zonotope, it holds according to Prop. 3.3.11 that any interval can be equivalently represented by the Z-representation.

3.3.3 Basic Set Operations

In this section we derive closed-form expressions for the linear map, the Minkowski sum, the Cartesian product, and the convex hull on the Z-representation. For this, we require the following identity:

$$\left\{ c + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{m_{1,i}} \alpha_{\mathbf{E}_{1(i,k)}} \right) G_{1(\cdot,i)} + \sum_{i=1}^{h_2} \left(\prod_{k=1}^{m_{2,i}} \alpha_{\mathbf{E}_{2(i,k)}} \right) G_{2(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{1(i,k)}}, \alpha_{\mathbf{E}_{2(i,k)}} \in [-1,1] \right\}$$

$$= \left\langle c, [G_1, G_2], \left(\mathbf{E}_1, \mathbf{E}_2\right) \right\rangle_Z. \tag{3.101}$$

We begin with the linear map:

Proposition 3.3.13. *(Linear Map) Given a set in Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z \subset \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{w \times n}$, the linear map is*

$$M\mathcal{P} = \langle Mc, MG, \mathbf{E} \rangle_Z.$$

The resulting Z-representation is regular if $\mathcal{P}$ is regular, and the computational complexity is $\mathcal{O}(wnh)$, where n is the dimension, w is the number of rows of matrix M, and h is the number of generators.

Proof. The result follows directly from inserting the definition of the Z-representation in Def. 3.3.1 into the definition of the linear map in (2.1):

$$M \otimes \mathcal{P} \overset{(2.1)}{=} \{Ms \mid s \in \mathcal{P}\} \overset{\text{Def. } 3.3.1}{=}$$

$$\left\{ Mc + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) MG_{(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{(i,k)}} \in [-1,1] \right\} = \langle Mc, MG, \mathbf{E} \rangle_Z,$$

which concludes the proof.

Complexity: The two matrix multiplications Mc and MG have complexity $\mathcal{O}(wn) + \mathcal{O}(wnh) = \mathcal{O}(wnh)$ according to Tab. 2.1. □

Next, we consider the Minkowski sum:

Proposition 3.3.14. *(Minkowski Sum) Given two sets in Z-representation, $\mathcal{P}_1 = \langle c_1, G_1, \mathbf{E}_1 \rangle_Z \subset \mathbb{R}^n$ and $\mathcal{P}_2 = \langle c_2, G_2, \mathbf{E}_2 \rangle_Z \subset \mathbb{R}^n$, their Minkowski sum is*

$$\mathcal{P}_1 \oplus \mathcal{P}_2 = \left\langle c_1 + c_2, [G_1 \; G_2], \left(\mathbf{E}_1, \widehat{\mathbf{E}}_2\right) \right\rangle_Z,$$

where

$$\widehat{\mathbf{E}}_2 = \left(\mathbf{E}_{2(1)} + \mathbf{1} \cdot p_1, \; \ldots \; , \mathbf{E}_{2(h_2)} + \mathbf{1} \cdot p_1\right).$$

The resulting Z-representation is regular if $\mathcal{P}_1$ and $\mathcal{P}_2$ are regular, and the computational complexity is $\mathcal{O}(n + \mu_2)$, where n is the dimension and μ_2 is the number of entries in the tuple $\mathbf{E}_2$.

Proof. The proposition follows from inserting the definition of the Z-representation in Def. 3.3.1 into the definition of the Minkowski sum (2.2):

$$\mathcal{P}_1 \oplus \mathcal{P}_2 \overset{(2.2)}{=} \{s_1 + s_2 \mid s_1 \in \mathcal{P}_1,\ s_2 \in \mathcal{P}_2\} \overset{\text{Def. 3.3.1}}{=}$$

$$\Bigg\{ c_1 + c_2 + \sum_{i=1}^{h_1} \Bigg(\prod_{k=1}^{m_{1,i}} \alpha_{\mathbf{E}_{1(i,k)}} \Bigg) G_{1(\cdot,i)} + \sum_{i=1}^{h_2} \Bigg(\prod_{k=1}^{m_{2,i}} \underbrace{\alpha_{\mathbf{E}_{2(i,k)}+p_1}}_{\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}} \Bigg) G_{2(\cdot,i)} \;\Bigg|\; \alpha_{\mathbf{E}_{1(i,k)}}, \alpha_{\widehat{\mathbf{E}}_{2(i,k)}} \in [-1,1] \Bigg\}$$

$$\overset{(3.101)}{=} \left\langle c_1 + c_2, [G_1\ G_2], (\mathbf{E}_1, \widehat{\mathbf{E}}_2) \right\rangle_Z,$$

where we used the identity in (3.101).

Complexity: The complexity for the addition of the constant offsets is $\mathcal{O}(n)$ and the complexity for the construction of the tuple $\widehat{\mathbf{E}}_2$ is $\mathcal{O}(\mu_2)$, where μ_2 denotes the number of entries in $\widehat{\mathbf{E}}_2$ (see Def. 3.3.1). Thus, the overall complexity is $\mathcal{O}(n + \mu_2)$. □

We continue with the Cartesian product:

Proposition 3.3.15. *(Cartesian Product) Given two sets in Z-representation, $\mathcal{P}_1 = \langle c_1, G_1, \mathbf{E}_1 \rangle_Z \subset \mathbb{R}^n$ and $\mathcal{P}_2 = \langle c_2, G_2, \mathbf{E}_2 \rangle_Z \subset \mathbb{R}^n$, their Cartesian product is*

$$\mathcal{P}_1 \times \mathcal{P}_2 = \left\langle \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \begin{bmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{bmatrix}, (\mathbf{E}_1, \widehat{\mathbf{E}}_2) \right\rangle_Z,$$

where

$$\widehat{\mathbf{E}}_2 = \left(\mathbf{E}_{2(1)} + \mathbf{1} \cdot p_1,\ \dots\ , \mathbf{E}_{2(h_2)} + \mathbf{1} \cdot p_1 \right).$$

The resulting Z-representation is regular if $\mathcal{P}_1$ and $\mathcal{P}_2$ are regular, and the computational complexity is $\mathcal{O}(\mu_2)$, where μ_2 is the number of entries in the tuple $\mathbf{E}_2$.

Proof. The proposition follows from inserting the definition of the Z-representation in Def. 3.3.1 into the definition of the Cartesian product in (2.4):

$$\mathcal{P}_1 \times \mathcal{P}_2 \overset{(2.4)}{=} \{[s_1^T\ s_2^T]^T \mid s_1 \in \mathcal{P}_1,\ s_2 \in \mathcal{P}_2\} \overset{\text{Def. 3.3.1}}{=}$$

$$\Bigg\{ \begin{bmatrix} c_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ c_2 \end{bmatrix} + \sum_{i=1}^{h_1} \Bigg(\prod_{k=1}^{m_{1,i}} \alpha_{\mathbf{E}_{1(i,k)}} \Bigg) \begin{bmatrix} G_{1(\cdot,i)} \\ \mathbf{0} \end{bmatrix} + \sum_{i=1}^{h_2} \Bigg(\prod_{k=1}^{m_{2,i}} \underbrace{\alpha_{\mathbf{E}_{2(i,k)}+p_1}}_{\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}} \Bigg) \begin{bmatrix} \mathbf{0} \\ G_{2(\cdot,i)} \end{bmatrix} \;\Bigg|\; \alpha_{\mathbf{E}_{1(i,k)}}, \alpha_{\widehat{\mathbf{E}}_{2(i,k)}} \in [-1,1] \Bigg\}$$

$$\overset{(3.101)}{=} \left\langle \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \begin{bmatrix} G_1 & \mathbf{0} \\ \mathbf{0} & G_2 \end{bmatrix}, (\mathbf{E}_1, \widehat{\mathbf{E}}_2) \right\rangle_Z,$$

where we used the identity in (3.101).

Complexity: The complexity for the construction of the tuple $\widehat{\mathbf{E}}_2$ is $\mathcal{O}(\mu_2)$, where μ_2 denotes the number of entries in $\widehat{\mathbf{E}}_2$ (see Def. 3.3.1). Since all other required operations are concatenations and initialization, this is identical to the overall complexity. □

Finally, we consider the convex hull of two sets in Z-representation. If both sets represent polytopes, it holds according to Sec. 2.1 that the linear combination and the convex hull are identical since polytopes are convex. However, if one of the sets in Z-representation is not a polytope (see Example 3.3.3), linear combination and convex hull differ. For simplicity, we assume here that the sets in Z-representation are polytopes. If this is not the case, one can compute the convex hull using the equation for SPZs in Prop. 3.1.27.

Proposition 3.3.16. *(Convex Hull) Given two sets in Z-representation* $\mathcal{P}_1 = \langle c_1, G_1, \mathbf{E}_1 \rangle_Z \subset \mathbb{R}^n$ *and* $\mathcal{P}_2 = \langle c_2, G_2, \mathbf{E}_2 \rangle_Z \subset \mathbb{R}^n$ *which both represent polytopes, their convex hull is*

$$conv(\mathcal{P}_1, \mathcal{P}_2) = \left\langle \frac{1}{2}(c_1 + c_2), \frac{1}{2}\begin{bmatrix} (c_1 - c_2) & G_1 & G_1 & G_2 & -G_2 \end{bmatrix}, \left((p), \mathbf{E}_1, \overline{\mathbf{E}}_1, \widehat{\mathbf{E}}_2, \overline{\mathbf{E}}_2 \right) \right\rangle_Z,$$

where

$$\begin{aligned} \overline{\mathbf{E}}_1 &= \left([\mathbf{E}_{1(1)}^T \; p]^T, \; \dots \; , [\mathbf{E}_{1(h_1)}^T \; p]^T \right), \\ \widehat{\mathbf{E}}_2 &= \left(\mathbf{E}_{2(1)} + \mathbf{1} \cdot p_1, \; \dots \; , \mathbf{E}_{2(h_2)} + \mathbf{1} \cdot p_1 \right), \\ \overline{\mathbf{E}}_2 &= \left([\widehat{\mathbf{E}}_{2(1)}^T \; p]^T, \; \dots \; , [\widehat{\mathbf{E}}_{2(h_2)}^T \; p]^T \right), \end{aligned} \tag{3.102}$$

with $p = p_1 + p_2 + 1$. *The resulting Z-representation is regular if* $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *are regular, and the computational complexity is* $\mathcal{O}(n(h_1 + h_2) + \mu_2)$, *where* n *is the dimension,* h_1 *and* h_2 *are the number of generators of* $\mathcal{P}_1$ *and* $\mathcal{P}_2$, *and* μ_2 *is the number of entries in the tuple* $\mathbf{E}_2$.

Proof. Since we assume that $\mathcal{P}_1$ and $\mathcal{P}_2$ are polytopes and therefore convex sets, it holds according to Sec. 2.1 that the linear combination and the convex hull are identical. The result then follows from inserting the definition of the Z-representation in Def. 3.3.1 into the definition of the linear combination in (2.11):

$$conv(\mathcal{P}_1, \mathcal{P}_1) \overset{\text{Sec. 2.1}}{=} comb(\mathcal{P}_1, \mathcal{P}_2) \overset{(2.11)}{=}$$

$$\left\{ \frac{1}{2}(1+\lambda)s_1 + \frac{1}{2}(1-\lambda)s_2 \;\middle|\; s_1 \in \mathcal{P}_1, \; s_2 \in \mathcal{P}_2, \; \lambda \in [-1, 1] \right\} \overset{\text{Def. 3.3.1}}{=}$$

$$\left\{ \frac{1}{2}(c_1 + c_2) + \frac{1}{2}(c_1 - c_2)\lambda + \frac{1}{2}\sum_{i=1}^{h_1} \left(\prod_{k=1}^{m_{1,i}} \alpha_{\mathbf{E}_{1(i,k)}} \right) G_{1(\cdot,i)} \right.$$

$$+\frac{1}{2}\sum_{i=1}^{h_1}\lambda\Big(\prod_{k=1}^{m_{1,i}}\alpha_{\mathbf{E}_{1(i,k)}}\Big)G_{1(\cdot,i)}+\frac{1}{2}\sum_{i=1}^{h_2}\Big(\prod_{k=1}^{m_{2,i}}\underbrace{\alpha_{\mathbf{E}_{2(i,k)+p_1}}}_{\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}}\Big)G_{2(\cdot,i)}$$

$$-\frac{1}{2}\sum_{i=1}^{h_2}\lambda\Big(\prod_{k=1}^{m_{2,i}}\underbrace{\alpha_{\mathbf{E}_{2(i,k)+p_1}}}_{\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}}\Big)G_{2(\cdot,i)}\;\Big|\;\alpha_{\mathbf{E}_{1(i,k)}},\alpha_{\widehat{\mathbf{E}}_{2(i,k)}},\lambda\in[-1,1]\Big\}\overset{\substack{(3.101),\\ \alpha_p:=\lambda}}{=}$$

$$\left\langle\frac{1}{2}\left(c_1+c_2\right),\frac{1}{2}\begin{bmatrix}(c_1-c_2) & G_1 & G_1 & G_2 & -G_2\end{bmatrix},\Big((p),\mathbf{E}_1,\overline{\mathbf{E}}_1,\widehat{\mathbf{E}}_2,\overline{\mathbf{E}}_2\Big)\right\rangle_Z,$$

where we used the identity in (3.101). For the transformation in the last line, we substitute λ with an additional factor α_p. With this substitution and $\overline{\mathbf{E}}_1$ and $\overline{\mathbf{E}}_2$ defined as in (3.102) it holds that

$$\lambda\prod_{k=1}^{m_{1,i}}\alpha_{\mathbf{E}_{1(i,k)}}=\alpha_p\prod_{k=1}^{m_{1,i}}\alpha_{\mathbf{E}_{1(i,k)}}=\prod_{k=1}^{m_{1,i}+1}\alpha_{\overline{\mathbf{E}}_{1(i,k)}},$$
$$\lambda\prod_{k=1}^{m_{2,i}}\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}=\alpha_p\prod_{k=1}^{m_{2,i}}\alpha_{\widehat{\mathbf{E}}_{2(i,k)}}=\prod_{k=1}^{m_{2,i}+1}\alpha_{\overline{\mathbf{E}}_{2(i,k)}}.$$

Since $\lambda\in[-1,1]$ and $\alpha_p\in[-1,1]$, substituting λ with α_p does not change the set.

Complexity: The construction of the constant offset $0.5(c_1+c_2)$ has complexity $\mathcal{O}(2n)$, and the construction of the generator matrix $0.5[(c_1-c_2)\ G_1\ G_1\ G_2\ -G_2]$ has complexity $\mathcal{O}(2n(h_1+h_2+1))$. Moreover, the complexity for the construction of the tuple $\widehat{\mathbf{E}}_2$ is $\mathcal{O}(\mu_2)$, where μ_2 denotes the number of entries in $\mathbf{E}_2$ (see Def. 3.3.1). Summation of all complexities yields

$$\mathcal{O}(2n)+\mathcal{O}(2n(h_1+h_2+1))+\mathcal{O}(\mu_2)=\mathcal{O}(n(h_1+h_2)+\mu_2) \tag{3.103}$$

for the overall complexity of the convex hull operation. □

Contrary to previous set operations where the computation in Z-representation is straightforward, the calculation of the intersection is non-trivial. Currently, there exists no algorithm to compute the intersection directly in Z-representation without conversion to another polytope representation. As shown in Tab. 3.12, most set operations on the Z-representation increase the representation size. One major advantage of the Z-representation over the vertex and halfspace representation of polytopes is that it is possible to efficiently reduce the representation size by using the order reduction method for SPZs in Prop. 3.1.39.

3.3.4 Polytope Test

As demonstrated in Example. 3.3.3, not all sets in Z-representation are polytopes. The natural question that arises is therefore how we can test if a set in Z-representation is a polytope or not. In this section we derive a sufficient condition to show that a set in Z-representation is not a polytope.

Table 3.12: Growth of the number of factors p, the number of generators h, and the number of tuple entries μ for basic set operations on the Z-representation.

Set Operation	Factors	Generators	Tuple Entries
Linear map	p	h	μ
Minkowski sum	$p_1 + p_2$	$h_1 + h_2$	$\mu_1 + \mu_2$
Cartesian product	$p_1 + p_2$	$h_1 + h_2$	$\mu_1 + \mu_2$
Convex hull	$p_1 + p_2 + 1$	$2h_1 + 2h_2 + 1$	$2\mu_1 + 2\mu_2 + h_1 + h_2 + 1$

We first introduce the following Lemma, which we require for the proof of our criterion:

Lemma 3.3.17. *A set in Z-representation is a polytope if and only if it is convex.*

Proof. Trivially, every non-convex set in Z-representation is not a polytope since polytopes are always convex. It remains to show that all convex set in Z-representation are polytopes. Every convex set $\mathcal{P} \subset \mathbb{R}^n$ can be represented as the convex hull of its extreme points

$$\mathcal{P} = \left\{ \sum_{i=1}^{s} \beta_i x_i \;\middle|\; x_i \in \mathcal{K},\ \beta_i \geq 0,\ \sum_{i=1}^{s} \beta_i = 1 \right\}, \tag{3.104}$$

where the set of extreme points $\mathcal{K} \subset \mathbb{R}^n$ is defined as

$$\mathcal{K} = \left\{ x \;\middle|\; \exists d \in \mathbb{R}^n :\ x = \arg\max_{y \in \mathcal{P}} d^T y \right\}.$$

If $\mathcal{P}$ is a set in Z-representation $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z \subset \mathbb{R}^n$, then it follows from the calculation of the vertices as described in Alg. 5 that

$$\forall d \in \mathbb{R}^n :\quad \left(\arg\max_{y \in \mathcal{P}} \quad d^T y \right) \in \texttt{vertices}(\mathcal{P}) = \{v_1, \ldots, v_s\},$$

so that the set of extreme points is $\mathcal{K} = \{v_1, \ldots, v_s\}$. Inserting this result into the definition of a convex set in (3.104) yields,

$$\begin{aligned} \mathcal{P} &\overset{(3.104)}{=} \left\{ \sum_{i=1}^{s} \beta_i x_i \;\middle|\; x_i \in \mathcal{K},\ \beta_i \geq 0,\ \sum_{i=1}^{s} \beta_i = 1 \right\} \overset{\mathcal{K}=\{v_1,\ldots,v_s\}}{=} \\ &\left\{ \sum_{i=1}^{s} \beta_i v_i \;\middle|\; \beta_i \geq 0,\ \sum_{i=1}^{s} \beta_i = 1 \right\} \overset{\text{Def. 2.2.3}}{=} \langle [v_1 \ \ldots \ v_s] \rangle_V, \end{aligned}$$

which is identical to the definition of a polytope in Def. 2.2.3. □

Based on this lemma, we formulate the following criterion:

Proposition 3.3.18. *(Polytope Test) Given a set in Z-representation $\mathcal{P} = \langle c, G, \mathbf{E}\rangle_Z \subset \mathbb{R}^n$, it holds that*

$$\big(\exists \mathcal{I} \subseteq \mathcal{I}_2 :\ \texttt{contract}(f(x), \mathcal{I}_1 \times \mathcal{I}) = \emptyset\big) \Rightarrow \big(\mathcal{P} \neq conv(\mathcal{P})\big),$$

where $\mathcal{P} \neq conv(\mathcal{P})$ implies according to Lemma 3.3.17 that $\mathcal{P}$ is not a polytope. The sets $\mathcal{I}_1 = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$ and $\mathcal{I}_2 = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^{\widehat{p}}$ are intervals, and the function $f : \mathbb{R}^{p+\widehat{p}} \to \mathbb{R}^n$ is defined as

$$f(x) = \underbrace{c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} x_{(\mathbf{E}_{(i,k)})} \right) G_{(\cdot,i)}}_{b_{\langle c,G,\mathbf{E}\rangle}(\alpha_1)} - \Bigg(\underbrace{\widehat{c} + \sum_{i=1}^{\widehat{h}} \left(\prod_{k=1}^{\widehat{m}_i} x_{(p+\widehat{\mathbf{E}}_{(i,k)})} \right) \widehat{G}_{(\cdot,i)}}_{b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle}(\alpha_2)} \Bigg), \tag{3.105}$$

where $x = [\alpha_1^T\ \alpha_2^T]^T$ and

$$\left\langle \widehat{c}, \widehat{G}, \widehat{\mathbf{E}} \right\rangle_Z = conv(\mathcal{P}) = conv(\mathcal{P}, \mathcal{P})$$

is computed using Prop. 3.3.16.

Proof. With the functions $b_{\langle c,G,\mathbf{E}\rangle} :\ \mathbb{R}^p \to \mathbb{R}^n$ and $b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle} :\ \mathbb{R}^{\widehat{p}} \to \mathbb{R}^n$ in (3.105), the Z-representations $\mathcal{P}$ and $conv(\mathcal{P})$ can be equivalently represented as

$$\mathcal{P} = \left\{ b_{\langle c,G,\mathbf{E}\rangle}(\alpha_1) \;\middle|\; \alpha_1 \in \mathcal{I}_1 \right\}, \quad conv(\mathcal{P}) = \left\{ b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle}(\alpha_2) \;\middle|\; \alpha_2 \in \mathcal{I}_2 \right\}.$$

By restricting the factor domain $\mathcal{I}_2 = [-\mathbf{1}, \mathbf{1}]$ of $conv(\mathcal{P})$ to a subset $\mathcal{I} \subset \mathcal{I}_2$ we obtain a subset $\overline{\mathcal{P}} \subseteq conv(\mathcal{P})$:

$$\overline{\mathcal{P}} = \left\{ b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle}(\alpha) \;\middle|\; \alpha \in \mathcal{I} \right\} \overset{\mathcal{I} \subseteq \mathcal{I}_2}{\subseteq} \left\{ b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle}(\alpha) \;\middle|\; \alpha \in \mathcal{I}_2 \right\} = conv(\mathcal{P}).$$

Since a set in Z-representation is a special type of SPZ, we can compute the intersection between $\mathcal{P}$ and $\overline{\mathcal{P}}$ using Lemma 3.1.33, which yields

$$\begin{aligned} \mathcal{P} \cap \overline{\mathcal{P}} &= \Big\{ b_{\langle c,G,\mathbf{E}\rangle}(\alpha_1) \;\Big|\; \underbrace{b_{\langle c,G,\mathbf{E}\rangle}(\alpha_1) - b_{\langle \widehat{c},\widehat{G},\widehat{\mathbf{E}}\rangle}(\alpha_2)}_{\overset{(3.105)}{=} f(x)} = \mathbf{0},\ \alpha_1 \in \mathcal{I}_1,\ \alpha_2 \in \mathcal{I} \Big\} \\ &= \Big\{ b_{\langle c,G,\mathbf{E}\rangle}(\alpha_1) \;\Big|\; f(x) = \mathbf{0},\ x = [\alpha_1^T\ \alpha_2^T]^T \in \mathcal{I}_1 \times \mathcal{I} \Big\}. \end{aligned}$$

Consequently, it holds that

$$\big(\texttt{contract}(f(x), \mathcal{I}_1 \times \mathcal{I}) = \emptyset\big) \Rightarrow \big(\overline{\mathcal{P}} \cap \mathcal{P} = \emptyset\big), \tag{3.106}$$

which then yields

$$\begin{aligned} \big(\exists \mathcal{I} \subseteq \mathcal{I}_2 :\ \texttt{contract}(f(x), \mathcal{I}_1 \times \mathcal{I}) = \emptyset\big) &\overset{(3.106)}{\Rightarrow} \big(\exists \overline{\mathcal{P}} \subseteq conv(\mathcal{P}) :\ \overline{\mathcal{P}} \cap \mathcal{P} = \emptyset\big) \\ &\Rightarrow \big(\exists \overline{\mathcal{P}} \subseteq conv(\mathcal{P}) :\ \overline{\mathcal{P}} \not\subseteq \mathcal{P}\big) \Rightarrow \big(\mathcal{P} \neq conv(\mathcal{P})\big) \end{aligned}$$

as a final result. □

To determine a suitable interval $\mathcal{I} \subset \mathcal{I}_2$, the interval $\mathcal{I}_2$ can for example be recursively split until either the criterion in Prop. 3.3.18 is satisfied or a user-defined minimal interval width is reached.

3.3.5 Representation Size Comparison

In this section, we compare the representation size in V-representation, H-representation, and Z-representation for general polytopes, for polytopes defined by the convex hull of a zonotope and a single point, and for polytopes defined by the convex hull of two zonotopes. Given a polytope $\mathcal{P}$, we denote by $N_V(\mathcal{P})$, $N_H(\mathcal{P})$ and $N_Z(\mathcal{P})$ the number of scalar values required for V-, H-, and Z-representation, respectively. Moreover, we denote the number of i-dimensional polytope faces by $F_i(\mathcal{P})$. Let us first specify the representation size for V-, H-, and Z-representation:

Proposition 3.3.19. *The representation size of a polytope $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V \subset \mathbb{R}^n$ in V-representation is*

$$N_V(\mathcal{P}) = ns,$$

where s is the number of polytope vertices.

Proof. For each of the s vertices, a vector with n entries has to be stored. □

Proposition 3.3.20. *The representation size of a polytope $\mathcal{P} = \langle A, b \rangle_H \subset \mathbb{R}^n$ in H-representation is*

$$N_H(\mathcal{P}) = (n+1)F_{n-1}(\mathcal{P}),$$

where $F_{n-1}(\mathcal{P})$ is the number of polytope facets.

Proof. Since each of the $F_{n-1}(\mathcal{P})$ polytopes facets corresponds to one inequality constraint, it holds that $A \in \mathbb{R}^{F_{n-1}(\mathcal{P}) \times n}$ and $b \in \mathbb{R}^{F_{n-1}(\mathcal{P})}$. □

Proposition 3.3.21. *The representation size of a polytope $\mathcal{P} = \langle c, G, \mathbf{E} \rangle_Z \subset \mathbb{R}^n$ in Z-representation is*

$$N_Z(\mathcal{P}) = n(h+1) + \mu,$$

where h is the number of generators and μ is the number of entries in the tuple $\mathbf{E}$.

Proof. The constant offset $c \in \mathbb{R}^n$ consists of n scalar values, the generator matrix $G \in \mathbb{R}^{n \times h}$ of nh scalar values, and the tuple $\mathbf{E}$ of μ scalar values (see Def. 3.3.1). □

General Case

We first derive the representation size in V-, H- and Z-representation for the general case of a bounded n-dimensional polytope $\mathcal{P}$ with s vertices.

V-Representation: Trivially, the representation complexity of a polytope with s vertices in V-representation is

$$N_V(\mathcal{P}) = ns$$

according to Prop. 3.3.19.

H-Representation: As stated by McMullens upper bound theorem [89], the polytope that maximizes the number of facets for a fixed number of vertices is the cyclic polytope.

Consequently, an upper bound for the number of $n-1$ dimensional facets $F_{n-1}(\mathcal{P})$ for a polytope $\mathcal{P}$ with s vertices is according to [13, Ch. 4.7, Eq. (3)]

$$F_{n-1}(\mathcal{P}) \leq \binom{s - \lfloor \frac{n+1}{2} \rfloor}{s-n} + \binom{s - \lfloor \frac{n+2}{2} \rfloor}{s-n}. \tag{3.107}$$

Inserting (3.107) into Prop. 3.3.20 then yields the upper bound

$$N_H(\mathcal{P}) \overset{\text{Prop. 3.3.20}}{=} (n+1)F_{n-1}(\mathcal{P}) \overset{(3.107)}{\leq} (n+1)\left(\binom{s - \lfloor \frac{n+1}{2} \rfloor}{s-n} + \binom{s - \lfloor \frac{n+2}{2} \rfloor}{s-n}\right)$$

for the representation size in H-representation.

Z-Representation: The representation size of a polytope $\mathcal{P}$ in Z-representation is given by Prop. 3.3.21. It remains to relate the parameters h and μ to the number of polytope vertices s. For this, we assume that the Z-representation of $\mathcal{P}$ is calculated with Alg. 4. As visualized in Fig. 3.17, conversion from V-representation to Z-representation using Alg. 4 can be viewed as the exploration of a binary tree with depth $k = \lceil \log(s) \rceil$. Using (3.84), we obtain the following upper bounds for the number of generators h and the number of tuple entries μ of the resulting Z-representation:

$$\begin{aligned} h &\leq h^{(k)} \overset{(3.84)}{=} \frac{4^k - 1}{3} \overset{k=\lceil \log(s) \rceil}{=} \frac{4^{\lceil \log(s) \rceil} - 1}{3}, \\ \mu &\leq \mu^{(k)} \overset{(3.84)}{=} 4^k \left(\frac{1}{6}k + \frac{1}{9}\right) - \frac{1}{9} \overset{k=\lceil \log(s) \rceil}{=} 4^{\lceil \log(s) \rceil} \left(\frac{1}{6}\lceil \log(s) \rceil + \frac{1}{9}\right) - \frac{1}{9}. \end{aligned} \tag{3.108}$$

Inserting (3.108) into Prop. 3.3.21 then yields the upper bound

$$N_P(\mathcal{P}) \overset{\text{Prop. 3.3.21}}{=} n(h+1) + \mu \overset{(3.108)}{\leq} \frac{4^{\lceil \log(s) \rceil} + 2}{3} n + 4^{\lceil \log(s) \rceil} \left(\frac{1}{6}\lceil \log(s) \rceil + \frac{1}{9}\right) - \frac{1}{9}.$$

for the representation size in Z-representation.

A comparison of the representation sizes in V-, H-, and Z-representation for different number of vertices is shown in Fig. 3.18 (left) for a 3-dimensional polytope. Obviously, for the case of general polytopes the V-representation and the H-representation are more compact than the Z-representation.

Convex Hull of a Zonotope and a Point

Next, we consider the special case of a polytope $\mathcal{C} = conv(\mathcal{Z}, x)$ that can be described by the convex hull of a zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ with l generators and a single point $x \in \mathbb{R}^n$.

V-Representation: The number of vertices s of an n-dimensional zonotope with l generators is according to [90, Prop. 2.1.2]

$$s = 2 \sum_{i=0}^{\min(n,l)-1} \binom{l-1}{i}. \tag{3.109}$$

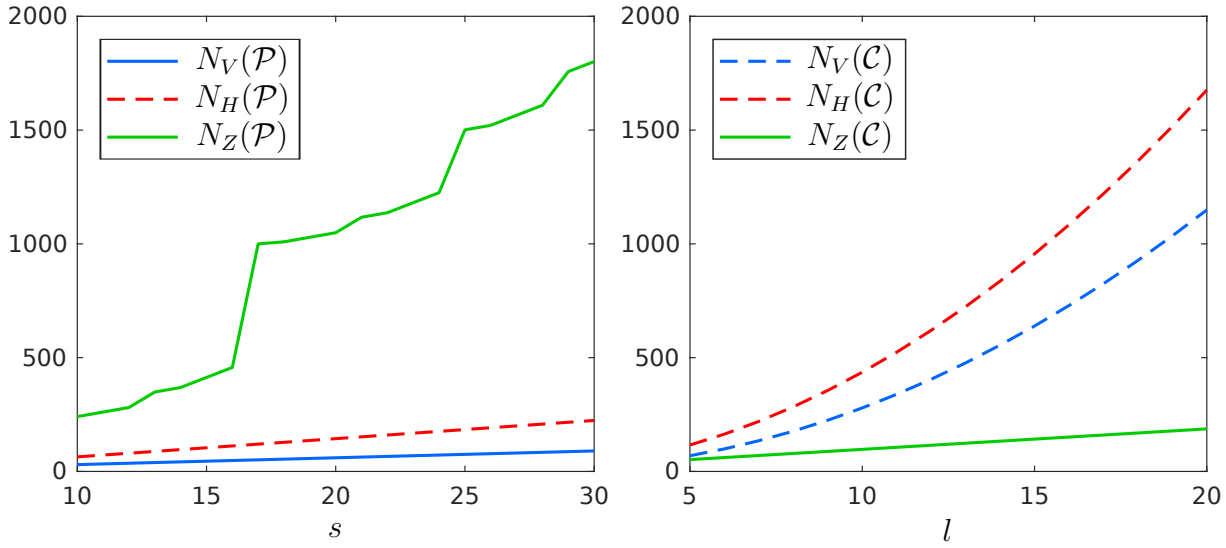

Figure 3.18: Representation sizes for a general 3-dimensional polytope $\mathcal{P}$ with s vertices (left) and for a 3-dimensional polytope $\mathcal{C} = conv(\mathcal{Z}, x)$ defined by the convex hull of a zonotope and a point (right). Exact values are depicted by solid lines, while upper bounds are depicted by dashed lines.

The polytope $\mathcal{C}$ defined by the convex hull of a zonotope and a point has at most $s+1$ vertices, so that according to Prop. 3.3.19

$$N_V(\mathcal{C}) \overset{\text{Prop. 3.3.19}}{\leq} n(s+1) \overset{(3.109)}{=} n\left(1 + 2\sum_{i=0}^{\min(n,l)-1}\binom{l-1}{i}\right)$$

is an upper bound for the representation size in V-representation.

H-Representation: The number of facets of an n-dimensional zonotope $\mathcal{Z}$ with l generators is [24, Ch. 2.2.1]

$$F_{n-1}(\mathcal{Z}) = 2\binom{l}{n-1}. \tag{3.110}$$

The maximum number of facets for the convex hull $\mathcal{C} = conv(\mathcal{Z}, x)$ is obtained in the case where $x \notin \mathcal{Z}$ and all facets of $\mathcal{Z}$ except for one facet $\mathcal{F}$ are facets of the convex hull $\mathcal{C}$. It therefore holds that

$$F_{n-1}(\mathcal{C}) \leq \underbrace{F_{n-1}(\mathcal{Z}) - 1}_{F_A} + \underbrace{F_{n-1}(conv(\mathcal{F}, x)) - 1}_{F_B}, \tag{3.111}$$

where F_A is the number of facets of $\mathcal{Z}$ that are facets of $\mathcal{C}$, and F_B is the number of additional facets resulting from the convex hull of the facet $\mathcal{F}$ with the point x. The number of facets for the convex hull of facet $\mathcal{F}$ and point x is identical to

$$F_{n-1}(conv(\mathcal{F}, x)) = F_{n-2}(\mathcal{F}) + 1, \tag{3.112}$$

where $F_{n-2}(\mathcal{F})$ is the number of $(n-2)$-dimensional faces of facet $\mathcal{F}$. The maximum for $F_{n-2}(\mathcal{F})$ is obtained in the case where $\mathcal{F}$ is a $(n-1)$-dimensional zonotope with l generators,

which yields the upper bound

$$F_{n-2}(\mathcal{F}) \overset{(3.110)}{\leq} 2\binom{l}{n-2}. \tag{3.113}$$

Overall, we therefore get the upper bound

$$N_H(\mathcal{C}) \overset{\text{Prop. 3.3.20}}{=} (n+1)F_{n-1}(\mathcal{C}) \overset{(3.111)}{\leq} (n+1)\big(F_{n-1}(\mathcal{Z}) + F_{n-1}(conv(\mathcal{F}, x)) - 2\big)$$

$$\overset{(3.112)}{=} (n+1)\big(F_{n-1}(\mathcal{Z}) + F_{n-2}(\mathcal{F}) - 1\big) \overset{\substack{(3.110)\\(3.113)}}{\leq} (n+1)\left(2\binom{l}{n-1} - 1 + 2\binom{l}{n-2}\right)$$

for the representation size in H-representation.

Z-Representation: According to Prop. 3.3.11, the zonotope $\mathcal{Z} = \langle c, G\rangle_Z$ can be equivalently represented by the Z-representation $\mathcal{Z} = \langle c, G, (1, \dots, l)\rangle_Z$ with $h_z = l$ generators and $\mu_z = l$ tuple entries. Moreover, the point $x \in \mathbb{R}^n$ can be represented by the Z-representation $x = \langle x, [\,], \emptyset\rangle_Z$ with $h_x = 0$ generators and $\mu_x = 0$ tuple entries. Computation of the convex hull $\mathcal{C} = conv(\mathcal{Z}, x)$ using Prop. 3.3.16 therefore results according to Tab. 3.12 in a Z-representation with $h = 2h_z + 2h_d + 1 = 2l + 1$ generators and $\mu = 2\mu_z + 2\mu_d + h_z + h_d + 1 = 3l + 1$ tuple entries. The representation size of $\mathcal{C}$ in Z-representation is therefore

$$N_Z(\mathcal{C}) \overset{\text{Prop. 3.3.21}}{=} n(h+1) + \mu \overset{\substack{h=2l+1\\ \mu=3l+1}}{=} 2n + 2ln + 3l + 1.$$

The comparison of the representation size in Fig. 3.18 (right) illustrates that for polytopes defined by the convex hull of a zonotope and a point the Z-representation is much more compact than other representations. We further demonstrate this by an example:

Example 3.3.22. *The representation sizes for the polytope $\mathcal{C} = conv(\mathcal{Z}, x)$ corresponding to the convex hull of the point $x = [2\ \mathbf{0}]^T \in \mathbb{R}^{20}$ and the zonotope $\mathcal{Z} = \langle \mathbf{0}, I_{20}\rangle_Z \subset \mathbb{R}^{20}$ are $N_V(\mathcal{C}) = 20971540$, $N_H(\mathcal{C}) = 1617$, and $N_Z(\mathcal{C}) = 901$.*

Convex Hull of Two Zonotopes

Finally, we consider the case of a polytope $\mathcal{C} = conv(\mathcal{Z}_1, \mathcal{Z}_2)$ defined by the convex hull of two full-dimensional zonotopes $\mathcal{Z}_1 = \langle c_1, G_1\rangle_Z \subset \mathbb{R}^n$ and $\mathcal{Z}_2 = \langle c_2, G_2\rangle_Z \subset \mathbb{R}^n$, which have l_1 and l_2 generators, respectively. While we can compute the exact representation size for the Z-representation, this is not possible for the V- and H-representation since the number of facets and vertices of $\mathcal{C}$ depends on the shape of the two zonotopes. We therefore consider the case where the zonotope with fewer generators encloses the second zonotope, which results in the minimum number of facets and vertices for $\mathcal{C}$ and therefore represents the best case for the V- and H-representation.

V-Representation: The minimum number of vertices s for the convex hull $\mathcal{C} = conv(\mathcal{Z}_1, \mathcal{Z}_2)$ is obtained in the case where the zonotope with less generators encloses

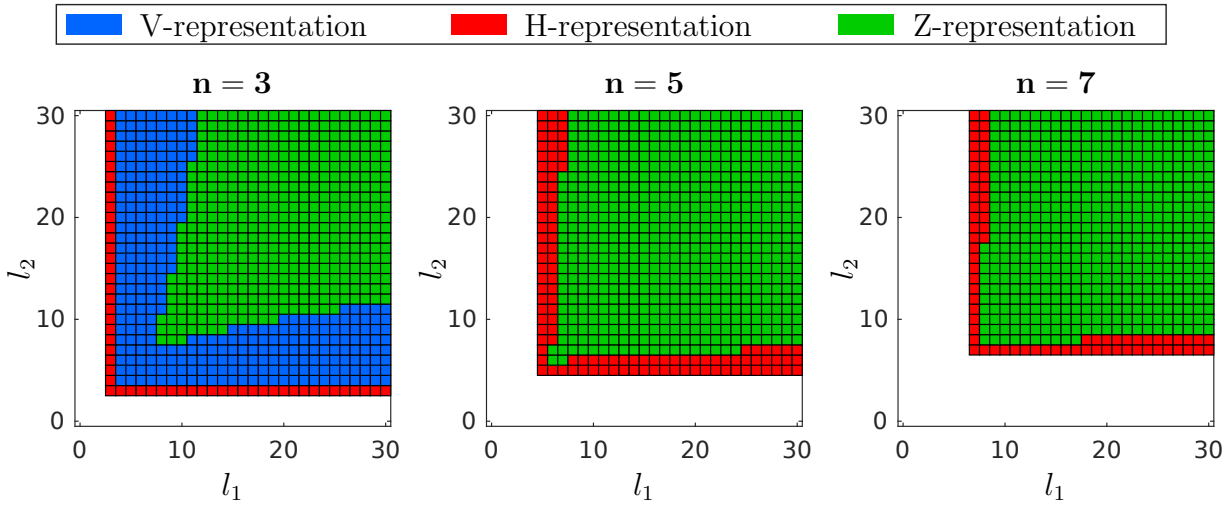

Figure 3.19: Polytope representation with the smallest representation size for a polytope $\mathcal{C} = conv(\mathcal{Z}_1, \mathcal{Z}_1)$ defined by the convex hull of two zonotopes for different dimensions n and numbers of generators l_1 and l_2.

the other zonotope. Using the equation in (3.109) for the number of zonotope vertices therefore yields the lower bound

$$N_V(\mathcal{C}) \overset{\text{Prop. 3.3.19}}{=} ns \overset{(3.109)}{\geq} 2n \sum_{i=0}^{\min(n,l_1,l_2)-1} \binom{\min(l_1,l_2)-1}{i}$$

for the representation size in V-representation.

H-Representation: The minimum number of facets for the convex hull $\mathcal{C} = conv(\mathcal{Z}_1, \mathcal{Z}_2)$ is obtained in the case where the zonotope with less generators encloses the other zonotope. Using the equation in (3.110) for the number of zonotope facets therefore yields the lower bound

$$N_H(\mathcal{C}) \overset{\text{Prop. 3.3.20}}{=} (n+1)F_{n-1}(\mathcal{C}) \overset{(3.110)}{\geq} 2(n+1)\binom{\min(l_1,l_2)}{n-1}$$

for the representation size in H-representation.

Z-Representation: According to Prop. 3.3.11, the two zonotopes $\mathcal{Z}_1 = \langle c_1, G_1 \rangle_Z$ and $\mathcal{Z}_2 = \langle c_2, G_2 \rangle_Z$ can be equivalently represented by the Z-representations $\mathcal{Z}_1 = \langle c_1, G_1, (1, \ldots, l_1) \rangle_Z$ and $\mathcal{Z}_2 = \langle c_2, G_2, (1, \ldots, l_2) \rangle_Z$, so that $h_1 = l_1$, $\mu_1 = l_1$ and $h_2 = l_2$, and $\mu_2 = l_2$. Computation of the convex hull $\mathcal{C} = conv(\mathcal{Z}_1, \mathcal{Z}_2)$ using Prop. 3.3.16 therefore results in a Z-representation with $h = 2h_1 + 2h_2 + 1 = 2l_1 + 2l_2 + 1$ generators and $\mu = 2\mu_1 + 2\mu_2 + h_1 + h_2 + 1 = 3l_1 + 3l_2 + 1$ tuple entries according to Tab. 3.12. The representation size of $\mathcal{C}$ in Z-representation is therefore

$$N_Z(\mathcal{C}) \overset{\text{Prop. 3.3.21}}{=} n(h+1) + \mu \overset{\substack{h=2l_1+2l_2+1 \\ \mu=3l_1+3l_2+1}}{=} 2n(l_1+l_2+1) + 3l_1 + 3l_2 + 1.$$

As shown in Fig. 3.19, for polytopes defined by the convex hull of two zonotopes the Z-representation is often the most compact representation, even thought we considered the best case resulting in the smallest representation size for the V-representation and the H-representation.

3.4 Summary

In this chapter we introduced the three novel set representations *sparse polynomial zonotopes*, *constrained polynomial zonotopes*, and the *Z-representation* of polytopes. All of these set representations are extensions to polynomial zonotopes as introduced in [47]. Due to their advantageous properties, our novel set representation are an excellent fit for the verification of complex cyber-physical systems.

Sparse polynomial zonotopes store polynomial zonotopes much more compactly than the non-sparse representation in [47]. A direct consequence of this is that operations on sparse polynomial zonotopes are more efficient compared to the non-sparse representation. In particular, the computational complexity for all relevant set operations on sparse polynomial zonotopes is at most polynomial with respect to the dimension. While sparse polynomial zonotope are in theory closed under linear map, Minkowski sum, Cartesian product, linear combination, convex hull, and quadratic map, exact computation of these set operations often significantly increased the computation time. We therefore use so-called independent generators to achieve a good trade-off between computation time and accuracy. With independent generators, we compute tight enclosures instead of the exact result for linear combination, convex hull, and quadratic map, which significantly accelerates the computations. Moreover, to keep track of dependencies between different sets we equipped sparse polynomial zonotopes with unique identifiers. Since sparse polynomial zonotopes consequently do not suffer from the dependency problem, one usually obtains more accurate results compared to other set representations.

Constrained polynomial zonotopes extend sparse polynomial zonotopes by adding polynomial equality constraints on the dependent factors. This extension enables us to additionally derive closed-form expressions for the intersection and union. Currently, constrained polynomial zonotopes are consequently the only set representation that is closed-under all relevant set operations with existing closed-form expressions for all operations. Moreover, as for sparse polynomial zonotopes, all relevant set operations on constraint polynomial zonotopes only have polynomial complexity with respect to the dimension, so that this novel set representation is well suited for computing with high-dimensional sets. In addition, we demonstrated that constrained polynomial zonotopes are generalizations of most other common set representations including intervals, zonotopes, polytopes, ellipsoids, polynomial zonotopes, and Taylor models, which further substantiates the relevance of this novel set representation.

Finally, we introduced the Z-representation of polytopes, which stores polynomial zonotopes representing polytopes very efficiently. While for general polytopes the V-representation and the H-representation are more compact than the Z-representation, we showed that for polytopes that are close to zonotopes, such as the convex hull of a zonotope and a point or the convex hull of two zonotopes, the Z-representation is much more compact than the other polytope representations. Moreover, we demonstrated that linear map, Minkowski sum, Cartesian product, and convex hull can be computed very efficiently in Z-representation, and we provided algorithms for converting between the Z-representation and the V-representation. Since not all sets in Z-representations are polytopes, we also presented a criterion to check if a set in Z-representation defines a polytope.

Chapter 4

Reachability Analysis

Using the novel set representations introduced in Chapter 3, we now present new approaches for reachability analysis of nonlinear continuous and hybrid systems. First, in Sec. 4.1, we demonstrate how sparse polynomial zonotopes improve reachability analysis for nonlinear continuous systems. Next, in Sec. 4.2, we present a novel method for the efficient extraction of reachable subsets. Based on this reachable subset approach, we afterward introduce a new technique for computing inner-approximations of reachable sets in Sec. 4.3. Finally, in Sec. 4.4, we propose a novel approach for reachability analysis of hybrid systems with nonlinear guard sets.

4.1 Outer-Approximations of Reachable Sets for Nonlinear Continuous Systems

In this section[1], we demonstrate the improvements for computing outer-approximations of reachable sets for nonlinear continuous systems resulting from the usage of SPZs as introduced in Sec. 3.1. In particular, we consider the conservative polynomialization algorithm [47] and show how SPZs increase the accuracy and reduce the computation time of this algorithm. The section is structured as follows: We first provide an overview of the state of the art for reachability analysis of nonlinear systems in Sec. 4.1.1, before we describe the conservative polynomialization algorithm in detail in Sec. 4.1.2. Next, we elaborate on the advantages resulting from SPZs in Sec. 4.1.3 and derive the computational complexity of the conservative polynomialization algorithm for SPZs in Sec. 4.1.4. Finally, we demonstrate the performance of SPZs for reachability analysis on several benchmark system in Sec. 4.1.5.

4.1.1 State of the Art

Let us first summarize the state of the art for computing outer-approximations of reachable sets for nonlinear continuous systems. Reachability algorithms for nonlinear systems can be categorized into four groups:

- Invariant generation

[1] This section is based on [76].

- Optimization-based approaches
- Abstraction in solution space
- Abstraction in state space

Since any invariant set which includes the initial set is also a reachable set, approaches for invariant generation can be used for reachability analysis [91–93]. However, the computation of invariant sets for nonlinear systems is challenging and the resulting invariants often only yield a very rough enclosure of the reachable set.

Optimization-based approaches reformulate reachability analysis as an optimization problem [52, 94, 95]. Thus, the approach in [94] optimizes the outward translation of polytope halfspaces to obtain a flowpipe. Halfspaces are also used in [95], where Bernstein polynomials are applied to abstract the optimization problem to linear programming. The approach in [52] uses a differential game formulation of reachability analysis to obtain an approximation of the reachable set as the solution to a partial-differential Hamilton-Jacobi equation, which requires optimization over the uncertain inputs.

Other approaches abstract the solution space directly: The work in [96] uses validated simulations for the construction of bounded flowpipes. Moreover, the approaches in [97,98] implement Runge-Kutta integration methods with affine arithmetic to obtain an enclosure of the reachable set. Taylor models computed from iterations, such as the Picard iteration, were initially proposed in [45, 99] and later extended to include uncertain parameter [100] and inputs [43].

Approaches based on an abstraction of the state space compute simplified differential equations to which a compensating uncertainty is added. Often, nonlinear ODEs are abstracted by a hybrid automaton with constant dynamics [101] or linear dynamics [102,103]. Other methods, such as the conservative linearization algorithm [104] and the simplex-based approach in [105], linearize the nonlinear dynamics on-the-fly. The conservative polynomialization algorithm [47] that we consider in this section extends conservative linearization by applying a polynomial abstraction of the nonlinear dynamics, which usually results in a tighter enclosure of the reachable set.

Common tools for reachability analysis of nonlinear systems are Adriadne [106], C2E2 [107], CORA [1], DynIbex [108], Flow* [44], Isabelle/HOL [109], and JuliaReach [110]. CORA is based on state space abstraction, and implements the conservative linearization and the conservative polynomialization algorithm. All other tools apply an abstraction in solution space, where Ariadne, Flow* and JuliaReach use Picard iteration, C2E2 constructs flowpipes based on validated simulation, and DynIbex and Isbelle/HOL implement Runge-Kutta integration methods with affine arithmetic.

4.1.2 Conservative Polynomialization Algorithm

In this section, we provide a detailed explanation of the conservative polynomialization algorithm introduced in [47]. The algorithm considers nonlinear systems $\dot{x}(t) = f(x(t), u(t))$ with uncertain inputs as defined in Def. 2.3.2. Since for general nonlinear systems the exact reachable set as defined in Def. 2.3.4 cannot be computed, the conservative polynomialization algorithm computes a tight over-approximation $\mathcal{R}^o(t) \supseteq \mathcal{R}(t)$ instead. Moreover,

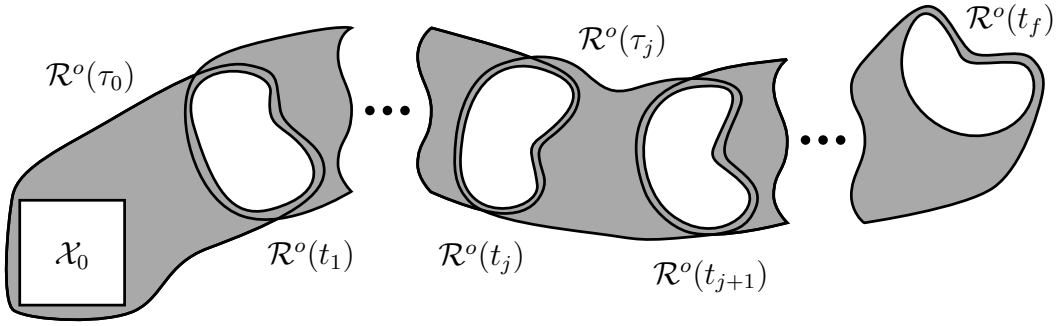

Figure 4.1: Computation of the reachable set for consecutive time intervals, where the time point reachable sets $\mathcal{R}(t_j)$ are depicted in white and the time interval reachable sets $\mathcal{R}(\tau_j)$ are depicted in gray.

as visualized in Fig. 4.1, the algorithm calculates the reachable set enclosure for consecutive time intervals $\tau_j = [t_j, t_{j+1}]$ with $t_{j+1} = t_j + \Delta t$, so that the reachable set enclosure for a time horizon t_f is given as $\mathcal{R}^o([0, t_f]) = \bigcup_{j=0}^{t_f/\Delta t - 1} \mathcal{R}^o(\tau_j)$, where $t_0 = 0$ and t_f is a multiple of the time step size Δt. Conservative polynomialization is an extension to the conservative linearization approach in [104]. The principle of conservative linearization is quite simple: In each time interval τ_j, the nonlinear function $f(x(t), u(t))$ that defines the system dynamics is linearized and the set of linearization errors on $\mathcal{R}^o(\tau_j)$ is treated as an additional uncertain input to the system, which then enables the calculation of the reachable set with a reachability algorithm for linear systems [24, Alg. 3]. The conservative polynomialization approach improves this method by abstracting the nonlinear function $f(x(t), u(t))$ by a Taylor expansion of order κ:

$$\dot{x}_{(i)}(t) = f_{(i)}(z(t)) \in \sum_{k=0}^{\kappa} \frac{\left((z(t) - z^*)^T \nabla\right)^k f_{(i)}(z^*)}{k!} \oplus \mathcal{L}_i(t), \quad i = 1, \ldots, n, \tag{4.1}$$

where we introduced $z(t) = [x(t)^T \; u(t)^T]^T$ to concisely denote the system dynamics as $\dot{x}(t) = f(z(t))$. Moreover, the vector $z^* = [x^{*T} \; u^{*T}]^T \in \mathbb{R}^{n+m}$ in (4.1) is the expansion point for the Taylor expansion and the set $\mathcal{L}(t) = \mathcal{L}_1(t) \times \ldots \times \mathcal{L}_n(t)$ defined as

$$\mathcal{L}(t) = \left\{ x \;\middle|\; x_{(i)} = \frac{\left((z(t) - z^*)^T \nabla\right)^{\kappa+1} f_{(i)}(\widehat{z})}{(\kappa+1)!},\ \widehat{z} = z^* + \lambda(z(t) - z^*),\ \lambda \in [0, 1] \right\} \tag{4.2}$$

is the Lagrange remainder. For simplicity, we focus at this point on the case with Taylor order $\kappa = 2$ since the extensions to higher orders is straightforward. For Taylor order $\kappa = 2$, the Taylor expansion in (4.1) becomes

$$\begin{aligned} \dot{x}_{(i)}(t) \in \; & f_{(i)}(z^*) + \underbrace{\left.\frac{\partial f_{(i)}(x,u)}{\partial x}\right|_{z^*}}_{A_{(i,\cdot)}} (x(t) - x^*) + \underbrace{\left.\frac{\partial f_{(i)}(x,u)}{\partial u}\right|_{z^*}}_{B_{(i,\cdot)}} (u(t) - u^*) \\ & + \frac{1}{2}(z(t) - z^*)^T \underbrace{\left.\frac{\partial^2 f_{(i)}(z)}{\partial z^2}\right|_{z^*}}_{Q_i} (z(t) - z^*) \oplus \mathcal{L}_i(t), \quad i = 1, \ldots, n. \end{aligned} \tag{4.3}$$

The crucial point of the conservative polynomialization algorithm is the division of the state vector $x(t) = x(t_j) + x^\Delta(t)$ and the input vector $u(t) = u^* + u^\Delta(t)$ into a static part $x(t_j)$, u^* that is constant over time for one time interval $\tau_j = [t_j, t_{j+1}]$ and a time-varying part $x^\Delta(t)$, $u^\Delta(t)$. This division allows one to combine the reachable set of the linearized system and the set of static linearization errors using the exact addition $\boxplus$ for SPZs as defined in Prop. 3.1.20 instead of the Minkowski sum, which results in a much tighter enclosure as we demonstrate later in Sec. 4.1.3. Inserting the substitutions $x(t) = x(t_j) + x^\Delta(t)$ and $u(t) = u^* + u^\Delta(t)$ into the Taylor series expansion in (4.3) yields

$$\begin{aligned}\dot{x}_{(i)}(t) \in\ & f_{(i)}(z^*) + A_{(i,\cdot)}\big(x(t) - x^*\big) + B_{(i,\cdot)}\big(u^* + u^\Delta(t) - u^*\big) \\ & + \frac{1}{2}\begin{bmatrix} x(t_j) + x^\Delta(t) - x^* \\ u^* + u^\Delta(t) - u^* \end{bmatrix}^T Q_i \begin{bmatrix} x(t_j) + x^\Delta(t) - x^* \\ u^* + u^\Delta(t) - u^* \end{bmatrix} \oplus \mathcal{L}_i(t) \\ \\ =\ & f_{(i)}(z^*) - A_{(i,\cdot)}x^* + A_{(i,\cdot)}x(t) + B_{(i,\cdot)}u^\Delta(t) \\ & + \frac{1}{2}\Bigg(\underbrace{\begin{bmatrix} x(t_j) - x^* \\ \mathbf{0} \end{bmatrix}}_{z_d(t_j)} + \underbrace{\begin{bmatrix} x^\Delta(t) \\ u^\Delta(t) \end{bmatrix}}_{z^\Delta(t)}\Bigg)^T Q_i \Bigg(\underbrace{\begin{bmatrix} x(t_j) - x^* \\ \mathbf{0} \end{bmatrix}}_{z_d(t_j)} + \underbrace{\begin{bmatrix} x^\Delta(t) \\ u^\Delta(t) \end{bmatrix}}_{z^\Delta(t)}\Bigg) \oplus \mathcal{L}_i(t) \qquad (4.4) \\ \\ =\ & A_{(i,\cdot)}x(t) + f_{(i)}(z^*) - A_{(i,\cdot)}x^* + \frac{1}{2} z_d(t_j)^T Q_i\, z_d(t_j) + B_{(i,\cdot)}u^\Delta(t) \\ & + \frac{1}{2}\big(z_d(t_j)^T Q_i\, z^\Delta(t) + z^\Delta(t)^T Q_i\, z_d(t_j) + z^\Delta(t)^T Q_i\, z^\Delta(t)\big) \oplus \mathcal{L}_i(t).\end{aligned}$$

For each time interval $\tau_j = [t_j, t_{j+1}]$, we have to compute an enclosure $\mathcal{R}^o(t_{j+1})$ of the reachable set at the end of the time interval based on the reachable set enclosure $\mathcal{R}^o(t_j)$ at the beginning of the time interval. To achieve this, the Taylor expansion in (4.4) has to be evaluated in a set-based manner. Obviously, the state $x(t_j)$ at the beginning of the time interval is part of the final reachable set from the previous time interval $x(t_j) \in \mathcal{R}^o(t_j)$. Moreover, the input vector $u(t)$ is part of the set of uncertain inputs $u(t) \in \mathcal{U}$. In addition, the time-varying part of the system state $x^\Delta(t) \in \mathcal{R}^\Delta(\tau_j)$ is contained in the reachable set $\mathcal{R}^\Delta(\tau_j)$ defined as

$$\mathcal{R}^\Delta(\tau) := \Big\{ \xi\big(t, x(t_j), u(\cdot)\big) - x(t_j) \;\Big|\; t \in \tau_j,\ x(t_j) \in \mathcal{R}^o(t_j),\ \forall t \in \tau_j :\ u(t) \in \mathcal{U} \Big\}, \qquad (4.5)$$

where $\xi(t, x(t_j), u(\cdot))$ as introduced in Def. 2.3.4 denotes the solution to the differential equation $\dot{x}(t) = f(x(t), u(t))$. Moreover, we have

$$u^\Delta(t) \in \underbrace{\mathcal{U} \oplus (-u^*)}_{\mathcal{U}^\Delta},\quad z_d(t_j) \in \underbrace{\big(\mathcal{R}^o(t_j) \oplus (-x^*)\big) \times \mathbf{0}}_{\mathcal{Z}(t_j)},\quad z^\Delta(t) \in \underbrace{\mathcal{R}^\Delta(\tau_j) \times \mathcal{U}^\Delta}_{\mathcal{Z}^\Delta(\tau_j)}. \qquad (4.6)$$

Inserting (4.6) into (4.4) then yields the differential inclusion

$$\dot{x}(t) \in\ Ax(t) + \underbrace{f(z^*) - Ax^* + \frac{1}{2}\begin{bmatrix} z_d(t_j)^T Q_1\, z_d(t_j) \\ \vdots \\ z_d(t_j)^T Q_n\, z_d(t_j) \end{bmatrix}}_{\in \mathcal{V}(t_j)}$$

$$\underbrace{+Bu^{\Delta}(t)+\frac{1}{2}\begin{bmatrix} z_d(t_j)^T Q_1\, z^{\Delta}(t)+z^{\Delta}(t)^T Q_1\, z_d(t_j)+z^{\Delta}(t)^T Q_1\, z^{\Delta}(t) \\ \vdots \\ z_d(t_j)^T Q_n\, z^{\Delta}(t)+z^{\Delta}(t)^T Q_n\, z_d(t_j)+z^{\Delta}(t)^T Q_n\, z^{\Delta}(t) \end{bmatrix}\oplus\mathcal{L}(t)}_{\in\mathcal{V}^{\Delta}(\tau_j)}, \tag{4.7}$$

where the set of static linearization errors $\mathcal{V}(t_j)$ is

$$\mathcal{V}(t_j)=(f(z^*)-Ax^*)\oplus\frac{1}{2}sq\big(\mathcal{Q},\mathcal{Z}(t_j)\big), \tag{4.8}$$

with $\mathcal{Q}=\{Q_1,\ldots,Q_n\}$ and the set of dynamic linearization errors $\mathcal{V}^{\Delta}(\tau_j)$ is

$$\begin{aligned}\mathcal{V}^{\Delta}(\tau_j)=&\Big(B\otimes\mathcal{U}^{\Delta}\Big)\oplus\frac{1}{2}sq\big(\mathcal{Q},\mathcal{Z}(t_j),\mathcal{Z}^{\Delta}(\tau_j)\big)\oplus\frac{1}{2}sq\big(\mathcal{Q},\mathcal{Z}^{\Delta}(\tau_j),\mathcal{Z}(t_j)\big)\oplus\\&\frac{1}{2}sq\big(\mathcal{Q},\mathcal{Z}^{\Delta}(\tau_j)\big)\oplus\mathcal{L}(t).\end{aligned} \tag{4.9}$$

Since the differential inclusion $\dot{x}(t)\in Ax(t)\oplus\mathcal{V}(t_j)\oplus\mathcal{V}^{\Delta}(\tau_j)$ in (4.7) is linear in $x(t)$ during the time interval τ_j, the solution can be calculated as

$$\begin{aligned}x(t_{j+1})\in e^{A\Delta t}x(t_j)\oplus\int_0^{\Delta t}e^{A(\Delta t-t)}\otimes\big(\mathcal{V}(t_j)\oplus\mathcal{V}^{\Delta}(\tau_j)\big)dt=\\ e^{A\Delta t}x(t_j)\oplus\Bigg(\underbrace{\bigg(\int_0^{\Delta t}e^{A(\Delta t-t)}dt\bigg)}_{\Gamma(\Delta t)}\otimes\mathcal{V}(t_j)\Bigg)\oplus\underbrace{\int_0^{\Delta t}e^{A(\Delta t-t)}\otimes\mathcal{V}^{\Delta}(\tau_j)dt}_{\subseteq\mathcal{R}^{p,\Delta}(\tau_j)},\end{aligned} \tag{4.10}$$

where we exploited that $\mathcal{V}(t_j)$ is constant over time and can therefore be moved outside the integral. If the matrix A is invertible, the matrix $\Gamma(\Delta t)$ can be computed as $\Gamma(\Delta t)=A^{-1}(e^{A\Delta t}-I_n)$. Otherwise, one can calculate $\Gamma(\Delta t)$ using the power series of the exponential matrix [24, Eq. (A.2)]. Inserting $x(t_j)\in\mathcal{R}^o(t_j)$ into (4.10) finally yields an enclosure of the reachable set $\mathcal{R}^o(t_{j+1})$ at the end of the time interval $\tau_j=[t_j,t_{j+1}]$:

$$\mathcal{R}(t_{j+1})\supseteq\big(e^{A\Delta t}\otimes\mathcal{R}^o(t_j)\big)\boxplus\big(\Gamma(\Delta t)\otimes\mathcal{V}(t_j)\big)\oplus\mathcal{R}^{p,\Delta}(\tau_j). \tag{4.11}$$

What remains to show is how the expansion point of the Taylor series z^* is chosen and how enclosures of the Lagrange remainder $\mathcal{L}(\tau_j)$, the reachable set due to time-varying linearization errors $\mathcal{R}^{p,\Delta}(\tau_j)$, and the reachable set of time varying states $\mathcal{R}^{\Delta}(\tau_j)$ can be calculated. The expansion point is heuristically selected as $z^*=[x^{*T}\ u^{*T}]^T$ with $x^*=x_c+0.5\cdot\Delta t\cdot f(x_c,u^*)\approx\texttt{center}(\mathcal{R}^o(\tau_j))$, $u^*=\texttt{center}(\mathcal{U})$, and $x_c=\texttt{center}(\mathcal{R}(t_j))$, which approximately corresponds to the center of the time interval reachable set and the input set. Moreover, using

$$\boldsymbol{\mathcal{D}}=\texttt{bound}\big(\nabla^3 f(z),\mathcal{I}\big),\quad\mathcal{I}=\texttt{interval}\big(\mathcal{R}^o(\tau_j)\times\mathcal{U}\big), \tag{4.12}$$

we obtain with Taylor order $\kappa = 2$ the following enclosure of the Lagrange remainder $\mathcal{L}(\tau_j)$ as defined in (4.2):

$$\begin{aligned}\mathcal{L}(\tau_j) &\overset{(4.2)}{=} \Big\{ x \;\Big|\; x_{(i)} = \frac{\big((z(t)-z^*)^T\nabla\big)^3 f_{(i)}(\widehat{z})}{3!}, \\ &\qquad\qquad \widehat{z} = z^* + \lambda(z(t)-z^*),\ \lambda \in [0,1],\ t \in \tau_j \Big\} \\ &\overset{\mathcal{R}^o(\tau_j)\times\mathcal{U} \subseteq \mathcal{I}}{\subseteq} \frac{1}{6}\Big\{ x \;\Big|\; x_{(i)} = \big((z(t)-z^*)^T\nabla\big)^3 f_{(i)}(\widehat{z}),\ z(t), \widehat{z} \in \mathcal{I} \Big\} \\ &\overset{(2.10)}{=} \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*)\big),\end{aligned} \tag{4.13}$$

where $\boldsymbol{\mathcal{D}}$ is calculated with range bounding as introduced in Sec. 2.7, and the polynomial map $poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*))$ as defined in (2.10) is evaluated using interval arithmetic [37]. Next, we consider the time-varying inputs. Given a convex set $\mathcal{S} \subset \mathbb{R}^n$ of time varying-inputs, an enclosure of the reachable set due to time-varying inputs can be computed as

$$\int_0^{\Delta t} \Big(e^{A(\Delta t - t)} \otimes \mathcal{S}\Big) dt \subseteq \underbrace{\bigoplus_{k=0}^{\nu} \frac{\Delta t^{k+1}}{(k+1)!} \left(A^k \otimes \mathcal{S}\right) \oplus \left(\Delta t \cdot \boldsymbol{\mathcal{E}} \otimes \mathcal{S}\right)}_{\texttt{reachVarInput}(\mathcal{S}, A, \Delta t, \nu)} \tag{4.14}$$

according to [24, Eq. (3.7)], with

$$\boldsymbol{\mathcal{E}} = [-\mathbf{1}, \mathbf{1}] \frac{(||A||_\infty \Delta t)^{\nu+1}}{(\nu+1)!} \frac{1}{1-\epsilon}, \quad \epsilon = \frac{||A||_\infty \Delta t}{\nu + 2},$$

where the number of Taylor terms for the exponential matrix ν has to be chosen large enough such that $\epsilon < 1$ holds. We denote the calculation of the reachable set due to time-varying inputs with the operation $\texttt{reachVarInput}(\mathcal{S}, A, \Delta t, \mu)$. Consequently, the reachable set due to time-varying linearization errors in (4.10) can be computed as $\mathcal{R}^{p,\Delta}(\tau_j) = \texttt{reachVarInput}(\mathcal{V}^\Delta(\tau_j), A, \Delta t, \nu)$. Finally, an enclosure of the reachable set for the time-varying system state $\mathcal{R}^\Delta(\tau_j)$ as defined in (4.5) can be calculated using a slight modification of the reachability algorithm for linear systems in [24, Alg. 3]:

$$\mathcal{R}^\Delta(\tau_j) = \underbrace{comb\big(\mathbf{0}, \mathcal{R}^\Delta(t_{j+1})\big) \oplus \left(\boldsymbol{\mathcal{F}} \otimes \mathcal{R}^d(t_j)\right) \oplus \widehat{\boldsymbol{\mathcal{F}}} f(z^*)}_{\mathcal{R}^{s,\Delta}(\tau_j)} \oplus \mathcal{R}^p(\tau_j), \tag{4.15}$$

where

$$\Psi(\tau_j) = \texttt{interval}\big(\mathcal{V}(t_j) \oplus \mathcal{V}^\Delta(\tau_j)\big) \oplus (-f(z^*) + Ax^*),$$

$$\mathcal{R}^\Delta(t_{j+1}) = \big((e^{A\Delta t} - I_n) \otimes \mathcal{R}^d(t_j)\big) \oplus \Gamma(\Delta t) f(z^*),$$

$$\mathcal{R}^d(t_j) = \mathcal{R}^o(t_j) + (-x^*), \quad \mathcal{R}^p(\tau_j) = \texttt{reachVarInput}(\Psi(\tau_j), A, \Delta t, \nu), \tag{4.16}$$

$$\boldsymbol{\mathcal{F}} = \bigoplus_{k=2}^{\nu} \left[(k^{\frac{-k}{k-1}} - k^{\frac{-1}{k-1}})\Delta t^k, 0 \right] \frac{A^k}{k!} \oplus \boldsymbol{\mathcal{E}},$$
$$\widehat{\boldsymbol{\mathcal{F}}} = \bigoplus_{k=2}^{\nu} \left[\left((k+1)^{\frac{-k-1}{k}} - (k+1)^{\frac{-1}{k}}\right)\Delta t^{k+1}, 0 \right] \frac{A^k}{(k+1)!} \oplus \Delta t \cdot \boldsymbol{\mathcal{E}}.$$

The resulting overall conservative polynomialization algorithm is shown in Alg. 6. To fully exploit the advantages of SPZs, Alg. 6 is slightly modified from the original conservative polynomialization algorithm in [47, Alg. 1]. We shortly explain the structure of the algorithm: The while-loop in lines 2-28 of Alg. 6 iterates over all time intervals τ_j until the time horizon t_f is reached. For each time interval we first abstract the nonlinear equation $f(x(t), u(t))$ by a Taylor expansion in Line 4. In lines 5-19 we then compute the set of linearization errors $\Psi(\tau_j)$ on the time interval reachable set $\mathcal{R}^o(\tau_j)$. The problem we are facing here is that we need $\mathcal{R}^o(\tau_j)$ to calculate $\Psi(\tau_j)$, but on the other hand, we also need $\Psi(\tau_j)$ to calculate $\mathcal{R}^o(\tau_j)$. To resolve this mutual dependence, we first compute $\mathcal{R}^o(\tau_j)$ using the set of linearization errors from the previous time step ($\Psi(\tau_j) = \Psi(\tau_{j-1})$, see Line 27) as an initial guess in Line 14. Next, we use $\mathcal{R}^o(\tau_j)$ to calculate $\Psi(\tau_j)$ in Line 18. In the next iteration of the repeat-until loop we then enlarge the set of linearization errors in Line 11 and calculate $\mathcal{R}^o(\tau_j)$ and $\Psi(\tau_j)$ using the enlarged set $\overline{\Psi}(\tau_j)$. We repeat this process until the enlarged set of linearization errors $\overline{\Psi}(\tau_j)$ contains the set of actual linearization errors $\Psi(\tau_j)$ (see Line 19), which guarantees that $\mathcal{R}^o(\tau_j)$ and $\Psi(\tau_j)$ are both over-approximative. After we computed the set of linearization errors, we finally calculate an enclosure of the reachable set $\mathcal{R}^o(t_{j+1})$ for the next point in time in Line 21, which we then use as the new initial set for the next time interval τ_{j+1}.

For SPZs, many of the operations in Alg. 6 increase the order and therefore also the representation size (see Tab. 3.4). For computational reasons, we therefore apply the operation `reduce` in Line 22 to reduce the order to the desired order ρ_d. Moreover, many of the operations in Alg. 6, like for example the Minkowski sum with a zonotope or order reduction, increase the size of the independent part of the SPZ. Since computations on the independent part are often over-approximative while computations on the dependent part are exact, we apply the `restructure` operation as introduced in Prop. 3.1.41 in Line 24 of Alg. 6 to redefine independent generators as dependent generators, which usually increases the accuracy of the computed reachable set enclosure. The restructuring is heuristically triggered using the operation `volRatio` defined as

$$\texttt{volRatio}\big(\langle c, G, G_I, E, id\rangle_{PZ}\big) := \frac{\texttt{volume}\big(\texttt{interval}(\langle \mathbf{0}, G_I\rangle_Z)\big)}{\texttt{volume}\big(\texttt{interval}(\langle G, [\,], E, id\rangle_{PZ})\big)}, \tag{4.17}$$

which returns an approximation of the volume of the dependent part of a SPZ divided by the volume of the independent part.

Algorithm 6 Conservative Polynomialization Algorithm

Require: Initial set $\mathcal{X}_0$, input set $\mathcal{U} = \langle c_u, G_u \rangle_Z$, time horizon t_f, time step size Δt, enlargement factor λ, number of Taylor terms for the exponential matrix ν, maximum zonotope order ρ_d, maximum number of dependent factors p_d, maximum volume ratio μ_d.

Ensure: Tight enclosure of the reachable set $\mathcal{R}^o([0,t_f]) \supseteq \mathcal{R}([0,t_f])$.

$t_0 \leftarrow 0,\ j \leftarrow 0,\ \mathcal{R}^{union} \leftarrow \emptyset,\ \mathcal{R}^o(0) \leftarrow \mathcal{X}_0,\ \mathcal{U}_s \leftarrow \mathbf{0},\ \Psi(\tau_0) \leftarrow \mathbf{0},\ \mathcal{U}^\Delta \leftarrow \mathcal{U} \oplus (-c_u)$

while $t_j < t_f$ **do**

  $x_c \leftarrow \texttt{center}(\mathcal{R}(t_j)),\ x^* \leftarrow x_c + 0.5 \cdot \Delta t \cdot f(x_c, c_u),\ z^* \leftarrow [x^{*T}\ c_u^T]^T$

  $[A\ B] \leftarrow \nabla f(z^*),\ \mathcal{Q} \leftarrow \nabla^2 f(z^*)$

  $\mathcal{R}^d(t_j) \leftarrow \mathcal{R}^o(t_j) \oplus (-x^*),\ \mathcal{R}_z^d(t_j) = \texttt{zonotope}(\mathcal{R}^d(t_j))$

  $\mathcal{Z}(t_j) \leftarrow \mathcal{R}^d(t_j) \times \mathcal{U}_s,\ \mathcal{Z}_z(t_j) \leftarrow \mathcal{R}_z^d(t_j) \times \mathcal{U}_s$

  $\mathcal{R}^\Delta(t_{j+1}) \leftarrow \big((e^{A\Delta t} - I_n) \otimes \mathcal{R}^d(t_j)\big) \oplus \Gamma(\Delta t) f(z^*)$ (see (4.16))

  $\mathcal{R}^{s,\Delta}(\tau_j) \leftarrow comb\big(\mathbf{0}, \mathcal{R}^\Delta(t_{j+1})\big) \oplus \big(\boldsymbol{\mathcal{F}} \otimes \mathcal{R}_z^d(t_j)\big) \oplus \widehat{\boldsymbol{\mathcal{F}}} f(z^*)$ (see (4.15))

  $\mathcal{V}(t_j) \leftarrow \big(f(z^*) - Ax^*\big) \oplus \frac{1}{2} sq(\mathcal{Q}, \mathcal{Z}(t_j))$ (see (4.8))

  repeat

    $\overline{\Psi}(\tau_j) \leftarrow (\lambda \cdot I_n) \otimes \Psi(\tau_j)$

    $\mathcal{R}^p(\tau_j) \leftarrow \texttt{reachVarInput}(\overline{\Psi}(\tau_j), A, \Delta t, \nu)$ (see (4.14))

    $\mathcal{R}^\Delta(\tau_j) \leftarrow \texttt{zonotope}\big(\mathcal{R}^{s,\Delta}(\tau_j) \oplus \mathcal{R}^p(\tau_j)\big),\ \mathcal{Z}^\Delta(\tau_j) \leftarrow \mathcal{R}^\Delta(\tau_j) \times \mathcal{U}^\Delta$ (see (4.15))

    $\mathcal{R}^o(\tau_j) \leftarrow \mathcal{R}^o(t_j) \oplus \mathcal{R}^\Delta(\tau_j)$

    $\mathcal{I} \leftarrow \texttt{interval}\big(\mathcal{R}^o(\tau_j) \times \mathcal{U}\big),\ \boldsymbol{\mathcal{D}} \leftarrow \texttt{bound}\big(\nabla^3 f(z), \mathcal{I}\big)$ (see (4.12))

    $\mathcal{L}(\tau_j) \leftarrow \frac{1}{6} poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*))$ (see (4.13))

    $\mathcal{V}^\Delta(\tau_i) \leftarrow \big(B \otimes \mathcal{U}^\Delta\big) \oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}_z(t_j), \mathcal{Z}^\Delta(\tau_j)\big) \oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}^\Delta(\tau_j), \mathcal{Z}_z(t_j)\big)$
    $\oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}^\Delta(\tau_j)\big) \oplus \mathcal{L}(\tau_j)$ (see (4.9))

    $\Psi(\tau_j) \leftarrow \texttt{interval}\big(\mathcal{V}(t_j) \oplus \mathcal{V}^\Delta(\tau_j)\big) \oplus (-f(z^*) + Ax^*)$ (see (4.16))

  until $\Psi(\tau_j) \subseteq \overline{\Psi}(\tau_j)$

  $\mathcal{R}^{p,\Delta}(\tau_j) \leftarrow \texttt{reachVarInput}(\mathcal{V}^\Delta(\tau_j), A, \Delta t, \nu)$ (see (4.14))

  $\mathcal{R}^o(t_{j+1}) \leftarrow \big(e^{A\Delta t} \otimes \mathcal{R}^o(t_j)\big) \boxplus \big(\Gamma(\Delta t) \otimes \mathcal{V}(t_j)\big) \oplus \mathcal{R}^{p,\Delta}(\tau_j)$ (see (4.11))

  $\mathcal{R}^o(t_{j+1}) \leftarrow \texttt{reduce}(\mathcal{R}^o(t_{j+1}), \rho_d)$

  if $\texttt{volRatio}(\mathcal{R}^o(t_{j+1})) > \mu_d$ **then** (see (4.17))

    $\mathcal{R}^o(t_{j+1}) \leftarrow \texttt{restructure}(\mathcal{R}^o(t_{j+1}), p_d)$

  end if

  $\mathcal{R}^{union} \leftarrow \mathcal{R}^{union} \cup \mathcal{R}^o(\tau_j)$

  $t_{j+1} \leftarrow t_j + \Delta t,\ \Psi(\tau_{j+1}) \leftarrow \Psi(\tau_j),\ j \leftarrow j+1$

end while

$\mathcal{R}^o([0,t_f]) \leftarrow \mathcal{R}^{union}$

In addition to the standard operations on SPZs introduced in Sec. 3.1, Alg. 6 requires some additional operations, which we shortly explain now. For the `center` operation used in Line 3 of Alg. 6, we return for SPZs the constant offset since there exists no closed-form expression for the computation of the arithmetic center of a SPZ. Moreover, the Minkowski addition of a SPZ and a single point can be easily realized by just adding the point to the constant offset. Construction of the set $\boldsymbol{\mathcal{F}} \otimes \mathcal{R}^d(t_j)$ in Line 8 of Alg. 6 requires the computation of the linear map of an interval matrix and a zonotope, which can be implemented according to [24, Thm. 3.3]. Moreover, the linear map of the interval matrix $\widehat{\boldsymbol{\mathcal{F}}}$ and the vector $f(z^*)$ in Line 8 of Alg. 6 can be evaluated using interval arithmetic [37]. The resulting set is an interval, which can be converted to a zonotope using [24, Prop. 2.1]. Finally, the quadratic maps on zonotopes in Line 17 of Alg. 6 can be calculated with the quadratic map for SPZs in Prop. 3.1.31 followed by a zonotope enclosure using Prop. 3.1.14.

4.1.3 Advantages of using Sparse Polynomial Zonotopes

After recapitulating the conservative polynomialization algorithm, we now demonstrate the advantages for this algorithm resulting from the usage of SPZs. In particular, using SPZs instead of zonotopes for Alg. 6 has two major advantages:

- With SPZs the quadratic map $sq(\mathcal{Q}, \mathcal{Z}(t_j))$ in Line 9 of Alg. 6 can be evaluated without over-approximation.
- Since SPZs preserve dependencies, the reachable set of the linearized system $e^{A\Delta t} \otimes \mathcal{R}^o(t_j)$ and the reachable set due to the static linearization error $\Gamma(\Delta t) \otimes \mathcal{V}(t_j)$ can be combined using the exact addition $\boxplus$ instead of the Minkowski sum $\oplus$ in Line 21 of Alg. 6.

Let us explain these two advantages in detail. As shown in Tab. 1.1, zonotopes are not closed under quadratic maps. Consequently, the quadratic map has to be over-approximated by a zonotope when computing with zonotopes. While zonotopes are convex, the result of the quadratic map is usually non-convex, so that the enclosure by a zonotope often yields quite large over-approximation errors. SPZs without independent generators, on the other hand, are closed under quadratic maps according to Prop. 3.1.30, which allows us to compute $sq(\mathcal{Q}, \mathcal{Z}(t_j))$ exactly. For the general case with independent generators we compute an enclosure using Prop. 3.1.31 for computational reasons. However, this enclosure is in general still much tighter than an enclosing zonotope.

Moreover, SPZs preserve dependencies since they store a unique identifier for each dependent factor. Therefore, relations between the reachable states of the linearized system $e^{A\Delta t} \otimes \mathcal{R}^o(t_j)$ and the corresponding points in the set of static linearization errors $\mathcal{V}(t_j)$ are preserved, so that for each state in the reachable set $e^{A\Delta t} \otimes \mathcal{R}^o(t_j)$ the corresponding static linearization error is known. Consequently, we can combine the reachable set of the linearized system $e^{A\Delta t} \otimes \mathcal{R}^o(t_j)$ and the reachable set due to static linearization errors $\Gamma(\Delta t) \otimes \mathcal{V}(t_j)$ using the exact addition operation $\boxplus$ as introduced in Prop. 3.1.20 instead of the Minkowski sum, which results in a much tighter enclosure of the reachable set. While zonotopes are also dependency-preserving as shown in Tab. 4.3, the zonotope enclosure of

the quadratic map introduces new factors that do not appear in the original set, which destroys most of the dependencies.

Let us now demonstrate the advantages of SPZs with two examples. We begin with a simple one-dimensional system:

Example 4.1.1. *We consider the one-dimensional system* $\dot{x} = f(x) = -x + x^2$*, the initial set* $\mathcal{X}_0 = \{\alpha_1 \mid \alpha_1 \in [-1, 1]\}$*, and the time step size* $\Delta t = 1$*. Computation of the Taylor series expansion at* $z^* = 0$ *in Line 4 of Alg. 6 results in the values* $f(z^*) = 0$*,* $A = \nabla f(z^*) = -1$*, and* $\mathcal{Q} = \nabla^2 f(z^*) = 2$*. The quadratic map in Line 9 evaluates to* $\mathcal{V}(0) = 0.5 \cdot sq(\mathcal{Q}, \mathcal{X}_0) = \{\alpha_1^2 \mid \alpha_1 \in [-1, 1]\}$ *for SPZs. On the other hand, if we use zonotopes, the quadratic map has to be enclosed by the zonotope* $\mathcal{V}(0) = 0.5 \cdot sq(\mathcal{Q}, \mathcal{X}_0) \subseteq \{0.5 + 0.5\alpha_2 \mid \alpha_2 \in [-1, 1]\}$*. While in this simple case the enclosure by the zonotope does not result in an over-approximation, the introduction of the new factor* α_2 *destroys the dependencies between the sets* $\mathcal{X}_0$ *and* $\mathcal{V}(0)$*. Consequently, for zonotopes the sets* $e^{A\Delta t} \otimes \mathcal{X}_0$ *and* $\Gamma(\Delta t) \otimes \mathcal{V}(0)$ *in Line 21 of Alg. 6 have to be added using the Minkowski sum instead of the exact addition, which results in an over-approximation due to the loss of dependency. With zonotopes, we therefore obtain the rather rough enclosure*

$$\begin{aligned}
&\left(e^{A\Delta t} \otimes \mathcal{X}_0\right) \oplus \left(\Gamma(\Delta t) \otimes \mathcal{V}(0)\right) \\
&\quad \subseteq \left(e^{-1} \otimes \{\alpha_1 \mid \alpha_1 \in [-1, 1]\}\right) \oplus \left((1 - e^{-1}) \otimes \{0.5 + 0.5\alpha_2 \mid \alpha_2 \in [-1, 1]\}\right) \\
&\quad = \left\{0.368\alpha_1 + 0.632(0.5 + 0.5\alpha_2) \mid \alpha_1, \alpha_2 \in [-1, 1]\right\} = [-0.368, 1].
\end{aligned}$$

With SPZs, on the other hand, we obtain

$$\begin{aligned}
&\left(e^{A\Delta t} \otimes \mathcal{X}_0\right) \boxplus \left(\Gamma(\Delta t) \otimes \mathcal{V}(0)\right) \\
&\quad = \left(e^{-1} \otimes \{\alpha_1 \mid \alpha_1 \in [-1, 1]\}\right) \boxplus \left((1 - e^{-1}) \otimes \{\alpha_1^2 \mid \alpha_1 \in [-1, 1]\}\right) \\
&\quad = \left\{0.368\alpha_1 + 0.632\alpha_1^2 \mid \alpha_1 \in [-1, 1]\right\} = [-0.054, 1],
\end{aligned}$$

which is the exact result.

Next, we consider an exemplary two-dimensional system:

Example 4.1.2. *We consider the nonlinear system*

$$\begin{aligned}
\dot{x}_1 &= x_1 x_2 \\
\dot{x}_2 &= (1 - x_1^2)x_2 - x_1,
\end{aligned}$$

the initial set $\mathcal{X}_0 = [-1.4, 0.6] \times [0.6, 1.4]$*, and the time step size* $\Delta t = 1$*. The visualization of the set of static linearization errors* $\mathcal{V}(0)$ *and the reachable set* $\left(e^{A\Delta t} \otimes \mathcal{X}_0\right) \boxplus \left(\Gamma(\Delta t) \otimes \mathcal{V}(0)\right)$ *for the first time interval of the conservative polynomialization algorithm in Fig. 4.2 demonstrates the conservatism introduced by the enclosure of the quadratic map and the loss of dependency when computing with zonotopes.*

We will further quantify the advantages resulting from the usage of SPZs for conservative polynomialization in Sec. 4.1.5 on several numerical examples.

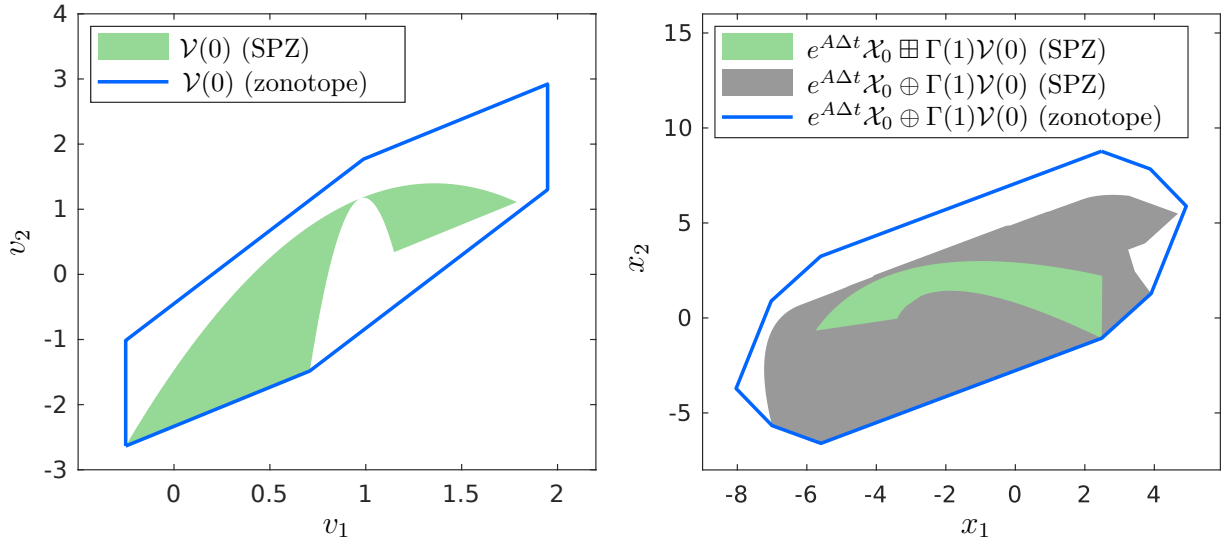

Figure 4.2: Visualization of the set of static linearization errors (left) and the reachable set (right) in the first time interval of the conservative polynomialization algorithm for the nonlinear system from Example 4.1.2 computed with zonotopes and SPZs.

4.1.4 Computational Complexity

We now derive the computational complexity of the conservative polynomialization algorithm as specified in Alg. 6 for the case that SPZs are used to represent the reachable set. To specify the computational complexity with respect to the system dimension, we require the following assumption:

Assumption 4.1.3. *Given a nonlinear system defined by the differential equation* $\dot{x}(t) = f(x(t), u(t))$ *with* $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, *and* $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, *where* $e \in \mathbb{N}_0$ *denotes the maximum number of elementary operations required for the evaluation of one subfunction* $f_{(i)} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $i = 1, \ldots, n$, *we assume for the derivation of the computational complexity that*

$$e = c_e n, \quad m = c_m n,$$

with constants $c_e, c_m \in \mathbb{R}_{\geq 0}$. *In addition, we assume that that taking the derivative of a subfunction* $f_{(i)}(x(t), u(t))$ *only changes the number of required elementary operations by a constant factor.*

Clearly, this assumption does not always hold. If $f_{(i)}(x(t), u(t))$ is for example a non-sparse quadratic function, we have $e = 0.5(n^2 + n)$, which violates Assumption 4.1.3. However, in practice, Assumption 4.1.3 is satisfied for most nonlinear systems. Since the number of time intervals $t_f/\Delta t$ and the number of required iterations for the repeat-until loop in lines 10-19 of Alg. 6 do in general not depend on the system dimension, each line of the algorithm is executed a constant number of times. The overall computational complexity of the algorithm is therefore identical to the computational complexity of the most expensive line.

Let us begin with the complexity of set operations on SPZs that are required for Alg. 6. The linear maps with quadratic matrices in Line 7 and Line 21 of Alg. 6 have complexity $\mathcal{O}(n^3)$ according to Prop. 3.1.18. Moreover, the Minkowski sums with zonotopes or

single points as required in lines5, 7-9, 13, 14, 18, and 21 have complexity $\mathcal{O}(n)$ according to Prop. 3.1.19. The exact addition $\boxplus$ as applied in Line 21 of Alg. 6 has complexity $\mathcal{O}(n^2 \log(n))$ according to Prop. 3.1.20. In addition, the Cartesian products with zonotopes or single points in lines 6, 13, and 15 have complexity $\mathcal{O}(1)$ according to Prop. 3.1.22. The linear combination in Line 8 has complexity $\mathcal{O}(n^2)$ according to Prop. 3.1.26 and the quadratic map $sq(\mathcal{Q}, \mathcal{Z}(t_j))$ in Line 9 has complexity $\mathcal{O}((n+m)^3(n+\log(n))) = \mathcal{O}(n(n+m)^3)$ according to Prop. 3.1.31 since $\mathcal{Z}(t_j) \subset \mathbb{R}^{n+m}$. Moreover, operation `reduce` in Line 22 and operation `restructure` in Line 24 both have complexity $\mathcal{O}(n^2)$ according to Tab. 3.5 if Girard's method is used for zonotope order reduction. Finally, the zonotope enclosures in Line 5 and Line 13 have complexity $\mathcal{O}(n^2)$ according to Prop. 3.1.14 and the interval enclosures in Line 15 and Line 18 have complexity $\mathcal{O}((n+m)^3) + \mathcal{O}(n^3) = \mathcal{O}((n+m)^3)$ according to Tab. 3.3 if interval arithmetic is used for range bounding. The resulting overall complexity of operations on SPZs is therefore

$$\begin{aligned} &\underbrace{\mathcal{O}(n^3)}_{\otimes} + \underbrace{\mathcal{O}(n)}_{\oplus} + \underbrace{\mathcal{O}(n^2\log(n))}_{\boxplus} + \underbrace{\mathcal{O}(1)}_{\times} + \underbrace{\mathcal{O}(n^2)}_{comb} + \underbrace{\mathcal{O}(n(n+m)^3)}_{sq} \\ &\quad + \underbrace{\mathcal{O}(n^2)}_{\texttt{reduce}} + \underbrace{\mathcal{O}(n^2)}_{\texttt{restructure}} + \underbrace{\mathcal{O}(n^2)}_{\texttt{zonotope}} + \underbrace{\mathcal{O}((n+m)^3)}_{\texttt{interval}} = \mathcal{O}(n(n+m)^3). \end{aligned} \tag{4.18}$$

The most expensive operation that does not involve operations on SPZs is the construction of the Lagrange remainder $\mathcal{L}(\tau_j)$ in Line 16 of Alg. 6, which involves the computation of $\boldsymbol{\mathcal{D}} = \texttt{bound}(\nabla^3 f(z), \mathcal{I})$ using range bounding and the calculation of the cubic map $poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*))$. Since $\nabla^3 f(z)$ consists of $n(n+m)^3$ scalar functions, the construction of $\boldsymbol{\mathcal{D}}$ has complexity $\mathcal{O}(ne(n+m)^3)$ according to Tab. 2.4 if interval arithmetic is used for range bounding. Moreover, computation of the cubic map $poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*))$ as defined in (2.10) using interval arithmetic has complexity $\mathcal{O}(n(n+m)^3)$, so that the overall complexity for the construction of the Lagrange remainder $\mathcal{L}(\tau_j)$ is $\mathcal{O}(ne(n+m)^3) + \mathcal{O}(n(n+m)^3) = \mathcal{O}(ne(n+m)^3)$. By combining this with the complexity for SPZ operations in (4.18) we finally obtain the resulting overall complexity of Alg. 6 as

$$\underbrace{\mathcal{O}(n(n+m)^3)}_{\text{SPZ operations}} + \underbrace{\mathcal{O}(ne(n+m)^3)}_{\text{Lagrange remainder}} = \mathcal{O}(ne(n+m)^3), \tag{4.19}$$

which is $\mathcal{O}(n^5)$ using Assumption 4.1.3. Clearly, (4.19) shows that due to the computational efficiency of SPZs the overall complexity of reachability analysis using the conservative polynomialization algorithm is actually dominated by the computation of the Lagrange remainder, and not by set operations on SPZs.

4.1.5 Numerical Examples

Finally, we demonstrate the performance of SPZs for reachability analysis based on the conservative polynomialization algorithm on four different benchmarks. We mainly focus on benchmarks from the ARCH competition [111, 112], which is the major platform for performance comparisons of reachability analysis tools. The competition contains challenging benchmarks that define the limit of difficulty that can currently be handled by the different tools.

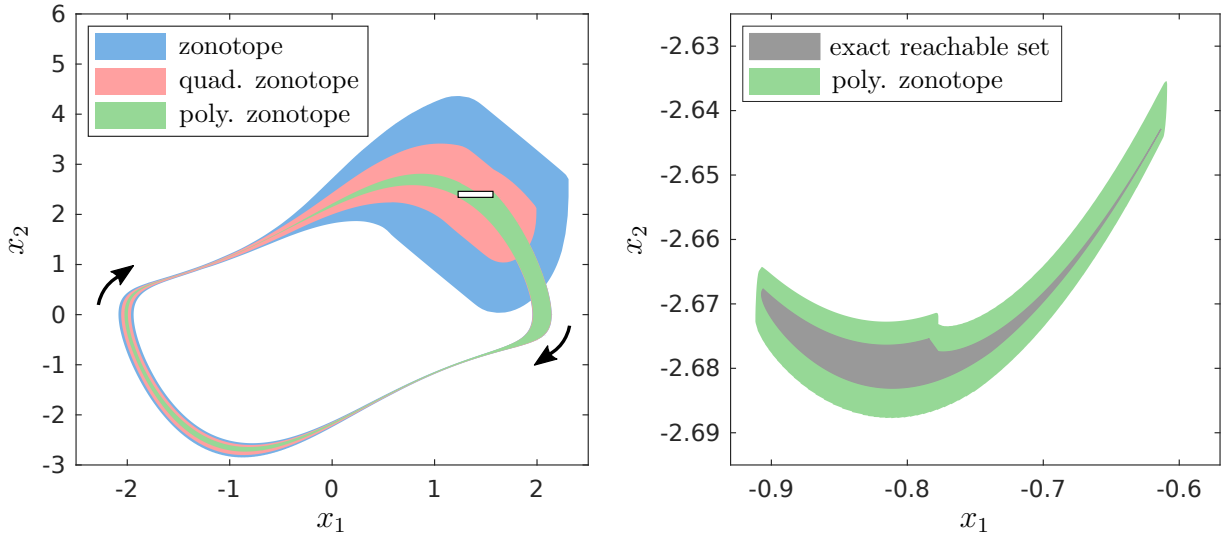

Figure 4.3: Reachable set for the Van-der-Pol oscillator calculated with different set representations, where the initial set is depicted in white with a black border. A comparison of the exact reachable set at time $t = 3.15$s with the reachable set enclosure calculated using SPZs is shown on the right.

Van-der-Pol Oscillator

The system considered first is the Van-der-Pol oscillator taken from the 2019 ARCH competition [111, Sec. 3.1]:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= (1 - x_1^2)x_2 - x_1. \end{aligned} \tag{4.20}$$

For this system, we compare the results for reachability analysis with Alg. 6 using zonotopes, the quadratic zonotopes from [47], and SPZs. We consider the initial set $\mathcal{X}_0 = [1.23, 1.57] \times [2.34, 2.46]$ and execute Alg. 6 with the parameter values $\Delta t = 0.005$s, $\lambda = 1.1$, $\nu = 4$, $\rho_d = 50$, $p_d = 100$, and $\mu_d = 0.01$. For a fair comparison, we use the same parameter values for every set representation. The resulting reachable sets are shown in Fig. 4.3 (left). It is clearly visible that the stability of the limit cycle can only be verified with SPZs when reachable sets are not split. The computation time is 9.33 seconds for zonotopes, 13.38 seconds for quadratic zonotopes, and 16.52 seconds for SPZs. An impression on how tight the reachable set can be enclosed with SPZs is provided in Fig. 4.3 (right), where the reachable set at time $t = 3.15$s computed with a time step size of $\Delta t = 0.0001$s and a maximum volume ratio of $\mu_d = 0.001$ is compared to the exact reachable set of the system. The figure also demonstrates how well the SPZ approximates the shape of the exact reachable set.

Drivetrain

For the second numerical example we examine a drivetrain, which is again a benchmark from the 2019 ARCH competition [112, Sec. 3.3]. We consider the case with two rotating

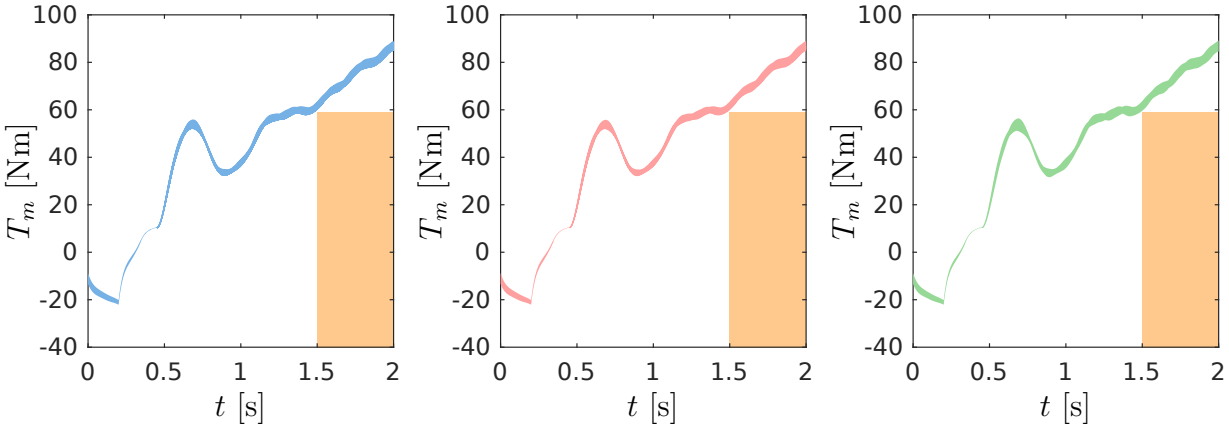

Figure 4.4: Reachable sets for the drivetrain benchmark calculated with zonotopes (left), quadratic zonotopes (middle), and SPZs (right). The unsafe set defined by the specification is depicted in orange.

masses, resulting in a system dimension of $n = 11$. The model is a hybrid system with linear dynamics. However, we apply the approach in [113] for calculating the intersections with guard sets, which is based on a time-triggered conversion of guards and results in highly nonlinear dynamics due to the time-scaling process. The initial set is given by the zonotope $\mathcal{X}_0 = \langle c, 0.5\, g \rangle_Z$, where $c \in \mathbb{R}^n$ and $g \in \mathbb{R}^n$ are defined as in [112, Sec. 3.3]. Moreover, we consider the same extreme acceleration maneuver as in [112, Sec. 3.3]. As a specification, we require that the engine torque after 1.5 seconds is at least 59Nm, which can be formally specified as $\forall t \geq 1.5\text{s} : \; T_m \geq 59\text{Nm}$. The results for the drivetrain model are shown in Fig. 4.4. We explicitly consider the possibility of splitting the reachable sets along the largest generator vector so that the specification can be verified with all set representations. However, splitting sets prolongs the computation time: With quadratic zonotopes, verification takes 93 seconds and with zonotopes verification takes 221 seconds. It was only possible with SPZs to verify the specification without splitting, which results in a computation time of 15 seconds. This is six times faster compared to quadratic zonotopes and more than 14 times faster compared to zonotopes.

Spacecraft Rendezvous

As a third numerical example, we consider the docking maneuver of a spacecraft taken from the 2019 ARCH competition [111, Sec. 3.4]. The model is a hybrid system with nonlinear dynamics, where the four system states are the planar positions s_x, s_y and corresponding velocities v_x, v_y of the spacecraft. The three discrete modes are *approaching*, *rendezvous attempt*, and *aborting*. We consider the same initial set and the same specifications as in [111, Sec. 3.4]. In particular, the specifications require that in mode *rendezvous attempt* the spacecraft is located inside the line-of-sight cone and the absolute velocity stays below 3.3m/min. Moreover, in mode *aborting* the spacecraft should not collide with the space station that is located at the origin. We apply Alg. 6 using SPZs and the parameter values $\Delta t = 0.2$min (mode *approaching* and *abortion*), $\Delta t = 0.05$min (mode *rendezvous attempt*), $\lambda = 1.1$, $\nu = 5$, $\rho_d = 10$, $p_d = 10$, and $\mu_d = 1$. To calculate the intersections

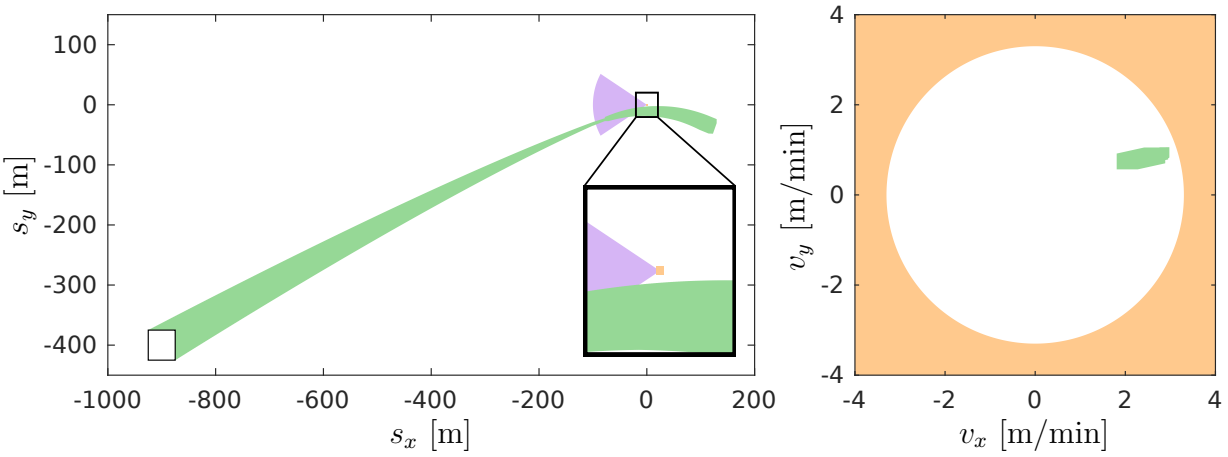

Figure 4.5: Reachable set for the spacecraft rendezvous benchmark (green), where the initial set is depicted in white with a black border, the line-of-sight cone is depicted in purple, and the unsafe sets defined by the velocity constraint and the spacestation are depicted in orange.

Table 4.1: Computation times in seconds for the spacecraft rendezvous benchmark. The results for the different tools are taken from [111, Tab. 4]. The computation times are measured on the machines of the participants, which are listed in [111, Appendix A].

Tool	Computation Time	Set Representation	Language
Ariadne [106]	172	Taylor models	C++
CORA [1]	11.8	Zonotopes	MATLAB
DynIbex [108]	294	Zonotopes	C++
Flow* [44]	18.7	Taylor models	C++
Isabelle/HOL [109]	295	Zonotopes	SML
Our approach	10.1	SPZs	MATLAB

between the reachable set and the guard sets, we use the approach that we present later in Sec. 4.4. The visualization of the reachable set in Fig. 4.5 demonstrates that all specifications are satisfied. To compare the performance of SPZs with other reachability tools, we consider the results from the 2019 ARCH competition [111]. The comparison in Tab. 4.1 shows that using SPZs actually results in the smallest computation time.

Transcriptional Regulator Network

To demonstrate the scalability of reachability analysis using SPZs, we consider the benchmark in [114, Sec. VIII.D] describing a transcriptional regulator network with N genes. For a network with N genes the system has $n = 2N$ dimensions. We consider the case without an artificial guard set so that the benchmark represents a continuous nonlinear system with uncertain inputs. Moreover, we use the same initial set, time horizon, and set of uncertain inputs as in [114, Sec. VIII.D]. We compute the reachable set with SPZs using Alg. 6 together with the parameter values $\Delta t = 0.1\text{min}$, $\lambda = 1.1$, $\nu = 3$, $\rho_d = 10$,

Table 4.2: Computation times in seconds for the transcriptional regulator network benchmark for different system dimensions.

System Dimension	$n = 12$	$n = 24$	$n = 36$	$n = 48$
Computation Time	6	20	54	122

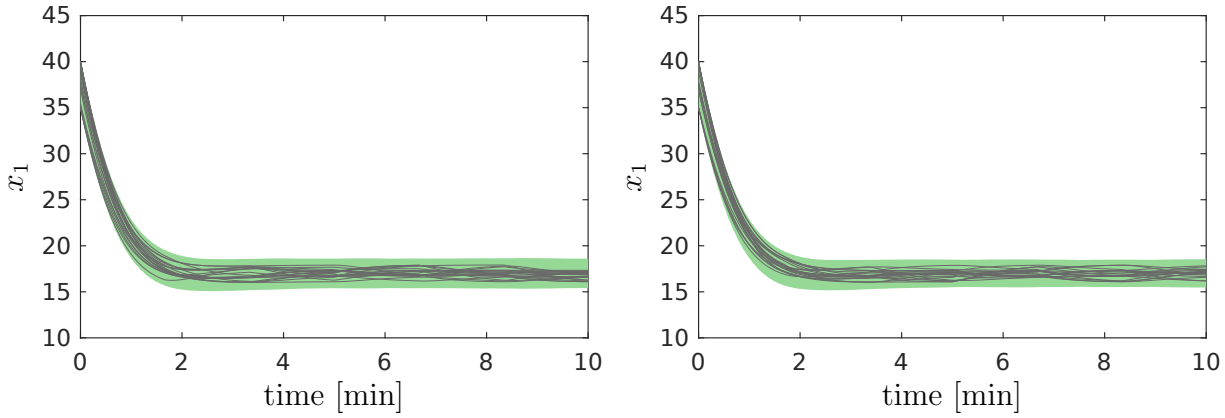

Figure 4.6: Reachable set for the transcriptional regulator network for the system dimensions $n = 12$ (left) and $n = 48$ (right). Simulations for random initial points and random inputs are depicted in gray.

$p_d = 50$, and $\mu_d = 1$. The reachable set is visualized in Fig. 4.6 and the computation times for different system dimensions are listed in Tab. 4.2. Even for a system dimension of 48, the computation of the reachable set with SPZs takes only 122 seconds. Moreover, as shown in Fig. 4.6, the accuracy of the calculated enclosure of the reachable set does not get worse for higher system dimensions.

4.2 Reachable Subsets

As we demonstrated in Sec. 4.1.3, the main advantage of the conservative polynomialization algorithm for reachability analysis of nonlinear systems is that it preserves relations between the reachable states of the linearized system and the corresponding points in the set of linearization errors. As a consequence, the reachable set due to linearization errors can be added using the exact addition instead of the Minkowski sum, which yields a much tighter enclosure. In this section[2], we generalize this concept and prove that the conservative polynomialization algorithm indeed also preserves relations between initial states and reachable states if a dependency-preserving set representation, such as SPZs, is used. These relations can be exploited for the computationally efficient extraction of reachable subsets. That is, given the reachable set $\mathcal{R}_{\mathcal{X}_0}(t)$ for an initial set $\mathcal{X}_0$, we can extract the reachable set $\mathcal{R}_{\widehat{\mathcal{X}}_0}(t)$ for any initial set $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$ by evaluating an analytical equation (see Fig. 4.7), which is much faster than computing $\mathcal{R}_{\widehat{\mathcal{X}}_0}(t)$ with a reachability algorithm. This novel method offers great advantages for applications where reachable sets have to be computed for many different subsets $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$, like safety falsification, optimization over reachable sets, and motion-primitive-based control.

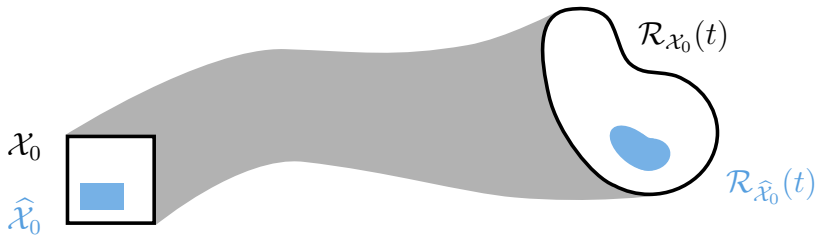

Figure 4.7: Given a reachable set $\mathcal{R}_{\mathcal{X}_0}(t)$ for a set of initial states $\mathcal{X}_0$ and a subset of initial states $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$, we can obtain $\mathcal{R}_{\widehat{\mathcal{X}}_0}(t)$ without any reachability analysis.

The structure of the section is as follows: First, we provide an overview about related approaches in Sec. 4.2.1. Next, in Sec. 4.2.2, we formally define the properties a set representation has to fulfill to be dependency-preserving. Afterward, we prove in Sec. 4.2.3 that the conservative polynomialization algorithm is dependency-preserving and show how this property can be exploited to efficiently extract reachable subsets. The computational complexity of reachable subset extraction is derived in Sec. 4.2.4, and in Sec. 4.2.5 we finally demonstrate the advantages resulting from our novel reachable subset approach on multiple different applications.

4.2.1 State of the Art

While our approach is to the best of our knowledge the first one to explicitly consider the extraction of reachable subsets, there exist many related approaches in the literature, for which we provide an overview in this section.

Let us first consider approaches from different fields of research that utilize dependency preservation. For range bounding, the loss of relation known as the *dependency problem* is a big issue, as we already mentioned in Sec. 2.7. While intervals as used for interval

[2]This section is based on [115].

arithmetic [37, Ch. 2.3] are not dependency-preserving, affine arithmetic [30] and Taylor models [41] explicitly preserve relations between different variables, and consequently often enable the computation of much tighter bounds. Moreover, for *abstract interpretation*, the work in [29] demonstrates how the parameterization of zonotopes can be utilized to preserve relations between inputs and outputs of computer programs.

Dependency preservation is also used in combination with reachability analysis. In [116], the relation between the initial and reachable states is used to compute inner-approximations of reachable sets. Similarly, the approaches in [39, 43] utilize this relation to tightly enclose the intersections with guard sets for hybrid system reachability analysis. For linear systems with piecewise constant inputs, [51] extracts initial states and input signals resulting in a violation of the specification from the computed reachable set by solving a linear program. A similar concept is used in [117], where reachability analysis is utilized to efficiently determine falsifying trajectories for hybrid systems with linear continuous dynamics.

The problem of finding falsifying initial states and input signals is known as *safety falsification*. In addition to the previously mentioned reachability-based falsification approaches [51, 117], there exist many other safety falsification techniques that are based on other concepts. Since we evaluate later in Sec. 4.2.5 the performance of safety falsification using our novel reachable subset method in comparison with other state of the art falsification approaches, we provide a short overview. Monte-Carlo methods [118, 119] randomly sample from the parameter space to determine falsifying trajectories. The cross-entropy technique in [120] constructs a function that estimates the robustness for each parameter. Moreover, ant-colony optimization in [121] divides the parameter space into multiple discrete regions and then propagates samples between these regions to determine regions corresponding to parameter values that most likely result in a falsification. Two common toolboxes for safety falsification are S-TaLiRo [122] and Breach [123], which implement multiple of the above-mentioned approaches.

4.2.2 Dependency-Preserving Set Representations

To prove that the conservative polynomialization algorithm preserves dependencies when using SPZs, we first have to show that all set operations used by the algorithm are dependency-preserving for SPZs. A prerequisite for preserving relations between points in different sets is that the points inside the sets are parameterized. As shown in Tab. 4.3, not all set representations fulfill this requirement. We demonstrate this exemplary for the halfspace representation and the vertex representation of a polytope:

Example 4.2.1. *Given a polytope $\mathcal{P} \subset \mathbb{R}^n$, its halfspace representation is according to Def. 2.2.2 defined as*

$$\mathcal{P} = \langle A, b \rangle_H = \{x \in \mathbb{R}^n \mid Ax \leq b\},$$

and its vertex representation is according to Def. 2.2.3 defined as

$$\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V = \left\{ \sum_{i=1}^{s} \beta_i v_i \;\middle|\; \beta_i \geq 0, \ \sum_{i=1}^{s} \beta_i = 1 \right\}.$$

For the vertex representation, each point $x \in \mathcal{P}$ can be parameterized by specific values $\overline{\beta}_i$ so that

$$x = \sum_{i=1}^{s} \overline{\beta}_i v_i \quad \textit{with} \quad \overline{\beta}_i \geq 0, \ \sum_{i=1}^{s} \overline{\beta}_i = 1. \tag{4.21}$$

For the halfspace representation, on the other hand, such a parameterization is not possible.

In general, the above parameterization of the vertex representation is not unique [124]. For parameterized sets, we introduce evaluation functions:

Definition 4.2.2. *(Evaluation Function) Given a set $\mathcal{S} \subset \mathbb{R}^n$ that is parameterized by the parameter vector $d \in \mathcal{D}$, the evaluation function $\underline{\lfloor \mathcal{S} \rfloor} : \mathcal{D} \to 2^{\mathcal{S}}$ returns the set $\overline{\mathcal{S}}$ that corresponds to a specific value $\overline{d} \in \mathcal{D}$ of the parameter vector d:*

$$\underline{\lfloor \mathcal{S} \rfloor}(\overline{d}) = \overline{\mathcal{S}},$$

where the parameter domain $\mathcal{D}$ satisfies

$$\bigcup_{d \in \mathcal{D}} \underline{\lfloor \mathcal{S} \rfloor}(d) = \mathcal{S}.$$

We use the shorthand notation

$$\underline{\lfloor \mathcal{S} \rfloor}(\widehat{\mathcal{D}}) = \left\{ \underline{\lfloor \mathcal{S} \rfloor}(d) \;\middle|\; d \in \widehat{\mathcal{D}} \right\}$$

for the evaluation function applied to a set of parameters $d \in \widehat{\mathcal{D}} \subseteq \mathcal{D}$.

If the bracket-notation $\underline{\lfloor \mathcal{S} \rfloor}(\overline{d})$ for the evaluation function is used in combination with set operations, all set operations inside the brackets are evaluated first. For example, given a set $\mathcal{S} \subset \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{w \times n}$, we use the notation $\underline{\lfloor M \otimes \mathcal{S} \rfloor}(\overline{d})$ as a shorthand for $\underline{\lfloor \overline{\mathcal{S}} \rfloor}(\overline{d})$ with $\overline{\mathcal{S}} = M \otimes \mathcal{S}$. Let us exemplary demonstrate the evaluation function for the vertex representation of polytopes:

Example 4.2.3. *For a polytope $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V$ in vertex representation, the parameter domain is*

$$\mathcal{D} = \left\{ \beta \in \mathbb{R}^s \;\middle|\; \beta_{(i)} \geq 0, \ \sum_{i=1}^{s} \beta_{(i)} = 1 \right\}$$

and

$$\underline{\lfloor \mathcal{P} \rfloor}(\beta) = \left\{ \sum_{i=1}^{s} \beta_{(i)} v_i \right\} \tag{4.22}$$

is the evaluation function.

After introducing parameterization and evaluation functions as preliminaries, we are now ready to formally define dependency preservation:

Definition 4.2.4. *(Dependency Preservation) Given an implementation of a set operation* `op` *and a set $\mathcal{S} \subset \mathbb{R}^n$ parameterized by $d \in \mathcal{D}$, we call the implementation of* `op` *dependency-preserving if*

$$\forall d \in \mathcal{D}: \quad \texttt{op}\left(\underline{\lfloor \mathcal{S} \rfloor}(d)\right) \subseteq \underline{\lfloor \texttt{op}(\mathcal{S}) \rfloor}(d)$$

holds.

Table 4.3: Characterization of set representations with respect to parameterization and dependency preservation of set operations.

Set Representation	Parameterization	Dependency Preservation			
		Linear Map	Minkowski Sum	Cartesian Product	Quadratic Map
Intervals	×				
Zonotopes	√	√	√	√	√
Polytopes (H-Rep.)	×				
Polytopes (V-Rep.)	√	√	×	×	×
Polytopes (Z-Rep.)	√	√	√	√	√
Con. Zonotopes	√	√	√	√	√
Zonotope Bundles	√	√	√	√	√
Ellipsoids	×				
Support Functions	×				
Taylor Models	√	√	√	√	√
Level Sets	×				
Star Sets	√	√	√	√	×
Sparse Poly. Zono.	√	√	√	√	√
Con. Poly. Zono.	√	√	√	√	√

A summary of the set operations that are dependency-preserving for different set representations is shown in Tab. 4.3. Note that even if a set representation is not closed under a set operation, a closed-form expression for enclosing the result of the set operation can still be dependency preserving. For example, while zonotopes are not closed under quadratic maps, the enclosure of the result from the quadratic map with a zonotope is dependency preserving. Again, we demonstrate dependency-preservation with the example of the vertex representation:

Example 4.2.5. *Given a scalar $M \in \mathbb{R}$ and a one-dimensional polytope $\mathcal{P} = \langle [v_1 \; v_2] \rangle_V \subset \mathbb{R}$, its linear map is computed as*

$$M \otimes \mathcal{P} = \left\langle [Mv_1 \; Mv_2] \right\rangle_V. \tag{4.23}$$

Let us introduce the polytope $\mathcal{P} = \langle [-1 \; 3] \rangle_V$, the point $x = 2 \in \mathcal{P}$, and the scalar $M = 2$. According to (4.21)*, the point $x \in \mathcal{P}$ can be parameterized by the values $\overline{\beta} = [0.25 \; 0.75]^T$, so that $x = \underline{\mathcal{P}\rfloor}(\overline{\beta})$. Computation of the linear map according to* (4.23) *yields*

$$M \otimes \mathcal{P} = 2 \otimes \left\langle [-1 \; 3] \right\rangle_V \overset{(4.23)}{=} \left\langle [\underbrace{-2}_{\widehat{v}_1} \; \underbrace{6}_{\widehat{v}_2}] \right\rangle_V. \tag{4.24}$$

If we evaluate the result for $\overline{\beta}$ corresponding to the point x, we obtain

$$\lfloor \underline{M \otimes \mathcal{P}} \rfloor(\overline{\beta}) \overset{(4.22)}{=} \sum_{i=1}^{2} \overline{\beta}_{(i)} \widehat{v}_i \overset{(4.24)}{=} 0.25 \cdot (-2) + 0.75 \cdot 6 = 4 = Mx = M \otimes \lfloor \underline{\mathcal{P}} \rfloor(\overline{\beta}),$$

which demonstrates that the implementation of the linear map in (4.23) *is dependency-preserving according to Def. 4.2.4.*

Next, we consider the quadratic map, which is not dependency-preserving for the vertex representation:

Example 4.2.6. *Given a scalar $Q \in \mathbb{R}$ and a one-dimensional polytope $\mathcal{P} = \langle [v_1 \ v_2] \rangle_V$, its quadratic map is computed as*

$$sq(\{Q\}, \mathcal{P}) \overset{(2.6)}{=} \left\{ x^T Q x \mid x \in \mathcal{P} \right\} = \begin{cases} \left\langle \, [0 \ \max(|v_1|, |v_2|)^2 Q] \, \right\rangle_V, & 0 \in \mathcal{P} \\ \left\langle \, [\min(v_1^2, v_2^2) Q \ \max(v_1^2, v_2^2) Q] \, \right\rangle_V, & \textit{otherwise} \end{cases}. \tag{4.25}$$

As in Example 4.2.5, we consider the polytope $\mathcal{P} = \langle [-1 \ 3] \rangle_V$ and the point $x = 2 \in \mathcal{P}$, which can according to (4.21) *be parameterized by the values $\overline{\beta} = [0.25 \ 0.75]^T$, so that $x = \lfloor \underline{\mathcal{P}} \rfloor(\overline{\beta})$. Computation of the quadratic map according to* (4.25) *for the value $Q = 2$ yields*

$$sq(\{Q\}, \mathcal{P}) = sq\big(\{2\}, \langle [-1 \ 3] \rangle_V\big) \overset{(4.25)}{=} \big\langle [\underbrace{0}_{\widehat{v}_1} \ \underbrace{18}_{\widehat{v}_2}] \big\rangle_V. \tag{4.26}$$

If we evaluate the computed quadratic map for $\overline{\beta}$ corresponding to the point x, we obtain

$$\lfloor \underline{sq(\{Q\}, \mathcal{P})} \rfloor(\overline{\beta}) \overset{(4.22)}{=} \sum_{i=1}^{2} \overline{\beta}_{(i)} \widehat{v}_i \overset{(4.26)}{=} 0.25 \cdot 0 + 0.75 \cdot 18 = 13.5$$
$$\neq 8 = sq(\{Q\}, x) = sq\big(\{Q\}, \lfloor \underline{\mathcal{P}} \rfloor(\overline{\beta})\big),$$

which is not identical to $sq(\{Q\}, \lfloor \underline{\mathcal{P}} \rfloor(\overline{\beta}))$. The implementation of the quadratic map in (4.25) *is therefore not dependency-preserving according to Def. 4.2.4.*

The overall conservative polynomialization algorithm in Alg. 6 consists of a sequence of basic set operations. We therefore have to derive some results on the composition of dependency-preserving set operations, for which we require the following assumption:

Assumption 4.2.7. *Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$, we assume that all unary set operations* `un` *satisfy*

$$\texttt{un}(\mathcal{S}_1) \subseteq \texttt{un}(\mathcal{S}_2). \tag{4.27}$$

Moreover, given sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ and $\mathcal{S}_3, \mathcal{S}_4 \subset \mathbb{R}^w$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and $\mathcal{S}_3 \subseteq \mathcal{S}_4$, we assume that

$$\texttt{bin}(\mathcal{S}_1, \mathcal{S}_3) \subseteq \texttt{bin}(\mathcal{S}_2, \mathcal{S}_4) \tag{4.28}$$

is satisfied for all binary set operations `bin`.

Clearly, some set operations, like for example the Minkowski difference as defined in (2.3), violate Assumption 4.2.7. For the set operations used by the conservative polynomialization algorithm, however, Assumption 4.2.7 is satisfied. Moreover, Assumption 4.2.7 equivalently holds for the composition of set operations:

Lemma 4.2.8. *Given two set operations* $\mathtt{A}$ *and* $\mathtt{B}$ *that satisfy Assumption 4.2.7, their composition* $\mathtt{A} \circ \mathtt{B}$ *satisfies Assumption 4.2.7, too.*

Proof. If $\mathtt{A}$ and $\mathtt{B}$ are unary set representations, it holds for sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$ that

$$(\mathtt{A} \circ \mathtt{B})(\mathcal{S}_1) = \mathtt{A}\Big(\underbrace{\mathtt{B}(\mathcal{S}_1)}_{\overset{\text{Ass. 4.2.7}}{\subseteq}\, \mathtt{B}(\mathcal{S}_2)}\Big) \overset{\text{Assumption 4.2.7}}{\subseteq} \mathtt{A}(\mathtt{B}(\mathcal{S}_2)) = (\mathtt{A} \circ \mathtt{B})(\mathcal{S}_2),$$

which is identical to the conditions in Assumption 4.2.7. Trivially, this result also holds for compositions involving binary set operations. □

Finally, we show that the composition of two dependency-preserving set operations is dependency-preserving:

Lemma 4.2.9. *Given two dependency-preserving set operations* $\mathtt{A}$ *and* $\mathtt{B}$*, their composition* $\mathtt{A} \circ \mathtt{B}$ *is dependency-preserving as well.*

Proof. For simplicity, we consider the case where $\mathtt{A}$ and $\mathtt{B}$ are both unary set operations, since the extension to compositions involving binary set operations is straightforward. Consequently, we have to prove that

$$\forall d \in \mathcal{D}: \quad (\mathtt{A} \circ \mathtt{B})\left(\llcorner \mathcal{S} \lrcorner(d)\right) \subseteq \llcorner \underline{(\mathtt{A} \circ \mathtt{B})(\mathcal{S})} \lrcorner(d) \tag{4.29}$$

holds for all sets $\mathcal{S} \subset \mathbb{R}^n$ parameterized by $d \in \mathcal{D}$. Since $\mathtt{B}$ is dependency-preserving, we have according to Def. 4.2.4 that

$$\forall d \in \mathcal{D}: \quad \mathtt{B}\left(\llcorner \mathcal{S} \lrcorner(d)\right) \subseteq \llcorner \underline{\mathtt{B}(\mathcal{S})} \lrcorner(d). \tag{4.30}$$

Then, using (4.30) and Assumption 4.2.7, we obtain

$$\forall d \in \mathcal{D}: \quad \mathtt{A}\Big(\underbrace{\mathtt{B}\left(\llcorner \mathcal{S} \lrcorner(d)\right)}_{\overset{(4.30)}{\subseteq}\, \llcorner \underline{\mathtt{B}(\mathcal{S})} \lrcorner(d)}\Big) \overset{\text{Assumption 4.2.7}}{\subseteq} \mathtt{A}\Big(\llcorner \underline{\mathtt{B}(\mathcal{S})} \lrcorner(d)\Big). \tag{4.31}$$

Finally, since $\mathtt{A}$ is dependency-preserving, it holds according to Def. 4.2.4 that

$$\forall d \in \mathcal{D}: \quad (\mathtt{A} \circ \mathtt{B})(\llcorner \underline{\mathcal{S}} \lrcorner(d)) = \mathtt{A}\left(\mathtt{B}\left(\llcorner \underline{\mathcal{S}} \lrcorner(d)\right)\right) \overset{(4.31)}{\subseteq} \mathtt{A}\Big(\llcorner \underline{\mathtt{B}(\mathcal{S})} \lrcorner(d)\Big)$$

$$\overset{\text{Def. 4.2.4}}{\subseteq} \llcorner \underline{\mathtt{A}(\mathtt{B}(\mathcal{S}))} \lrcorner(d) = \llcorner \underline{(\mathtt{A} \circ \mathtt{B})(\mathcal{S})} \lrcorner(d),$$

which is identical to (4.29) and therefore concludes the proof. □

While the results derived up to now hold for general set representations, we now specialize on SPZs. We first consider the parameterization and the evaluation function of SPZs. In general, SPZs as defined in Def. 3.1.1 are parameterized by the dependent factors α and the independent factors β. However, as shown in Prop. 3.1.4, every SPZ can be equivalently represented as a SPZ without independent generators. For simplicity, we therefore only use the dependent factors α for the parameterization, since without loss of generality we can assume that the initial set for reachability analysis is a SPZ without independent generators. Consequently, we obtain the following evaluation function for SPZs:

Definition 4.2.10. *(Evaluation Function SPZ) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, its evaluation function is*

$$\underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) := c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\},$$

where the parameter domain is $\mathcal{D} = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$.

It is trivial to see that the result of the evaluation function as defined in Def. 4.2.10 can be represented as a zonotope

$$\underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) = \left\langle c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)},\ G_I \right\rangle_Z.$$

The simple definition of the evaluation function in Def. 4.2.10 is only applicable if the dependent factors of the SPZ do not change. However, some operations on SPZs, such as for example the `restructure` operation, potentially increase or decrease the number of dependent factors, which would impede dependency preservation if the evaluation function in Def. 4.2.10 is used. Luckily, SPZs store a unique identifier for each dependent factor, which allows us to define an extended version of the evaluation function as follows:

Definition 4.2.11. *(Extended Evaluation Function SPZ) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, a parameter vector* $\widehat{\alpha} \in \mathbb{R}^w$*, and an identifier vector* $\widehat{id} \in \mathbb{N}^{1 \times w}$ *satisfying* $\forall i, j \in \{1, \ldots, w\} : \ \widehat{id}_{(i)} \neq \widehat{id}_{(j)}$*, the extended evaluation function of* $\mathcal{PZ}$ *is*

$$\underline{\lfloor \mathcal{PZ} \rfloor}\left(\widehat{\alpha}, \widehat{id}\right) := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \widetilde{\alpha}_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \breve{\alpha}_1, \ldots, \breve{\alpha}_p \in [-1,1] \right\} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\},$$

where

$$\forall k \in \{1, \ldots, p\} : \quad \widetilde{\alpha}_{(k)} = \begin{cases} \widehat{\alpha}_{(i)}, & \widehat{id}_{(i)} = id_{(k)} \\ \breve{\alpha}_k, & \textit{otherwise} \end{cases}$$

and the parameter domain is $\mathcal{D} = [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^w$.

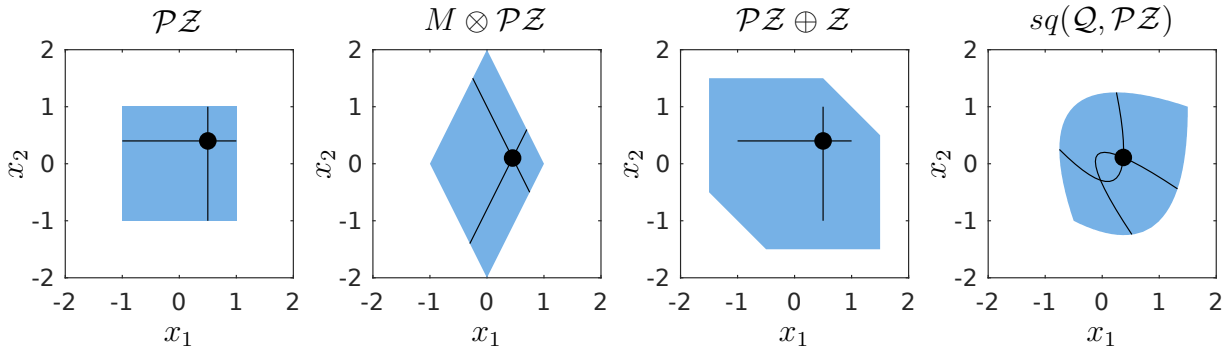

Figure 4.8: Sets resulting from the application of different set operations to the SPZ $\mathcal{PZ}$ from Example 4.2.12. The point x and its transformations are depicted by the black dots. Black lines correspond to the case where $\alpha_1 = 0.5 = const.$, $\alpha_2 \in [-1, 1]$ and the case where $\alpha_2 = 0.4 = const.$, $\alpha_1 \in [-1, 1]$.

We again use the shorthand $\llcorner\underline{\mathcal{PZ}}\lrcorner(\widehat{\mathcal{D}}, \widehat{id})$ to denote the application of the extended evaluation function to a set of parameters $\widehat{\alpha} \in \widehat{\mathcal{D}} \subset \mathcal{D}$. Note that the simple definition of the evaluation function in Def. 4.2.10 is just a special case of the general definition in Def. 4.2.11 with $\llcorner\underline{\mathcal{PZ}}\lrcorner(\alpha) = \llcorner\underline{\mathcal{PZ}}\lrcorner(\alpha, id)$. The set resulting from the extended evaluation function in Def. 4.2.11 can be represented as a SPZ, which we demonstrate later in Sec. 4.2.4. Since the evaluation function of a SPZ as defined in Def. 4.2.10 and Def. 4.2.11 is nonlinear, finding a parameterization for a point inside a SPZ can be computationally demanding. However, for many applications, such as reachability analysis, the initial set is often a multi-dimensional interval. In this case the parameterization is unique and trivial to compute, as we demonstrate with the following example:

Example 4.2.12. *We consider a SPZ $\mathcal{PZ}$ and a point x defined as*

$$\mathcal{PZ} = \left\{ \underbrace{\begin{bmatrix}1\\0\end{bmatrix}\alpha_{(1)} + \begin{bmatrix}0\\1\end{bmatrix}\alpha_{(2)}}_{\llcorner\underline{\mathcal{PZ}}\lrcorner(\alpha)} \;\middle|\; \alpha_{(1)}, \alpha_{(2)} \in [-1, 1] \right\} \quad \textit{and} \quad x = \begin{bmatrix}0.5\\0.4\end{bmatrix}.$$

Trivially, the point x can be parameterized with $\overline{\alpha} = [0.5\ 0.4]^T$, so that $x = \llcorner\underline{\mathcal{PZ}}\lrcorner(\overline{\alpha})$.

For the conservative polynomialization algorithm in Alg. 6 we require the set operations linear map, Minkowski sum with a zonotope, Cartesian product with a zonotope, quadratic map, order reduction and restructuring. To show that the algorithm is dependency-preserving, we consequently have to prove that all these operations are dependency-preserving for SPZs. We exemplary demonstrate this here for the Minkowski sum with a zonotope, and provide the proofs for the remaining set operations in Appendix C.

Proposition 4.2.13. *(Dependency Preservation Minkowski Sum) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$ and a zonotopes $\mathcal{Z} = \langle c_z, G_z\rangle_Z \subset \mathbb{R}^n$, it holds that the implementation of the Minkowski sum $\mathcal{PZ} \oplus \mathcal{Z}$ in Prop. 3.1.19 is dependency-preserving.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \llcorner\underline{\mathcal{PZ}}\lrcorner(\alpha) \oplus \mathcal{Z} = \llcorner\underline{\mathcal{PZ} \oplus \mathcal{Z}}\lrcorner(\alpha). \tag{4.32}$$

Inserting the evaluation function for SPZs as defined in Def. 4.2.10, the definition of zonotopes in Def. 2.2.4, and the implementation of the Minkowski sum in Prop. 3.1.19 into (4.32) yields

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) \oplus \mathcal{Z}$$

$$\overset{\text{Def. 4.2.10}}{=} c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\} \oplus \mathcal{Z}$$

$$\overset{\text{Def. 2.2.4}}{=} c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\}$$
$$\oplus \left\{ c_z + \sum_{j=1}^{l} \beta_{q+j} G_{z(\cdot,j)} \;\middle|\; \beta_{q+j} \in [-1,1] \right\} \overset{(2.2)}{=}$$

$$c + c_z + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} + \sum_{j=1}^{l} \beta_{q+j} G_{z(\cdot,j)} \;\middle|\; \beta_j, \beta_{q+j} \in [-1,1] \right\}$$

$$\overset{\text{Def. 4.2.10}}{=} \underline{\lfloor \langle c + c_z, G, [G_I \; G_z], E, id \rangle_{PZ} \rfloor}(\alpha) \overset{\text{Prop. 3.1.19}}{=} \underline{\lfloor \mathcal{PZ} \oplus \mathcal{Z} \rfloor}(\alpha),$$

which is identical to (4.32) and therefore concludes the proof. □

A visualization of dependency preservation of SPZs for different set operations is shown in Fig. 4.8.

4.2.3 Extraction of Reachable Subsets

After introducing dependency preservation for set operations in the previous section, we now prove that the conservative polynomialization algorithm is dependency-preserving for SPZs. Let us first introduce some notations. The computation of a tight enclosure of the reachable set using the conservative polynomialization algorithm as specified in Alg. 6 is denoted by the operation $\mathcal{R}^o(t_f) = \texttt{reach}(\mathcal{X}_0)$. Moreover, we denote the expansion point z^* in the j-th time interval of Alg. 6 by z_j^*. For the derivations in this section we require a small variation of the algorithm in which the sets of linearization errors $\overline{\Psi}(\tau_j)$ and the expansion points z_j^* for all time intervals are provided as additional input arguments and consequently do not have to be computed. We denote this with the operation $\mathcal{R}^o(t_f) = \texttt{reach}(\mathcal{X}_0, \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*)$, where $N = t_f/\Delta t$ is the number of time intervals. Note that if $\overline{\Psi}(\tau_j)$ is provided as an additional input argument, the repeat-until loop in lines 10-19 of Alg. 6 is not necessary any more, so that we only have to consider one iteration of the loop. Moreover, if the sets $\overline{\Psi}(\tau_j)$ enclose the exact linearization errors, which we can guarantee in our case, providing $\overline{\Psi}(\tau_j)$ and z_j^* as additional input arguments does not destroy the correctness of the algorithm. Finally, the computations for

a single time interval of Alg. 6, which corresponds to a single iteration of the while-loop in lines 2-28, are summarized by the operation $\mathcal{R}^o(t_{j+1}) = \texttt{post}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j), z_j^*)$, so that the overall reachability algorithm can be equivalently described as the N-times consecutive execution of the `post` operation.

The outline of the proof is as follows: We first show that the `post` operation is dependency-preserving for SPZs since it represents a composition of dependency-preserving set operations. Since Alg. 6 can be viewed as the N-times consecutive execution of the `post` operation, it then holds that Alg. 6 is dependency-preserving, too, which enables the efficient extraction of reachable subsets as we finally prove in our main theorem. Before we consider the `post` operation, we first study the computation of the set of dynamic linearization errors $\mathcal{V}^\Delta(\tau_j)$:

Lemma 4.2.14. *The operation $\mathcal{V}^\Delta(\tau_j) = \texttt{abstrErr}(\mathcal{R}^o(t_j), \Psi(\tau_j))$ denoting the computation of the set of dynamic linearization errors $\mathcal{V}^\Delta(\tau_j)$ in Alg. 6 satisfies Assumption 4.2.7.*

Proof. A flowchart visualizing the computations required for operation `abstrErr` is shown in Fig. 4.9, where the operation `reachDelta` is defined as

$$\begin{aligned}\texttt{reachDelta}\big(\mathcal{R}^d(t_j)\big) = \mathit{comb}\big(\mathbf{0}, \big((e^{A\Delta t} - I_n) \otimes \mathcal{R}^d(t_j)\big) \oplus \Gamma(\Delta t) f(z^*)\big) \\ \oplus \big(\boldsymbol{\mathcal{F}} \otimes \texttt{zonotope}(\mathcal{R}^d(t_j)\big) \oplus \widehat{\boldsymbol{\mathcal{F}}} f(z^*),\end{aligned} \tag{4.33}$$

the operation `lagrRem` is defined as

$$\texttt{lagrRem}(\mathcal{I}) = \frac{1}{6}\mathit{poly}\big(\texttt{bound}(\nabla^3 f(z), \mathcal{I}), \mathcal{I} \oplus (-z^*)\big), \tag{4.34}$$

and the operation `dynErr` is defined as

$$\begin{aligned}\texttt{dynErr}\big(\mathcal{Z}_z(t_j), \mathcal{Z}^\Delta(\tau_j)\big) =& \big(B \otimes \mathcal{U}^\Delta\big) \oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}_z(t_j), \mathcal{Z}^\Delta(\tau_j)\big) \\ &\oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}^\Delta(\tau_j), \mathcal{Z}_z(t_j)\big) \oplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}^\Delta(\tau_j)\big).\end{aligned} \tag{4.35}$$

As Fig. 4.9 demonstrates, `abstrErr` is a composition of operations that satisfy Assumption 4.2.7, so that according to Lemma 4.2.8 operation `abstrErr` satisfies Assumption 4.2.7. □

Using Lemma 4.2.14, we can now show that the `post` operation is dependency-preserving:

Lemma 4.2.15. *Given a set $\mathcal{R}^o(t_j) = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$ represented as a SPZ, a set of linearization errors $\overline{\Psi}(\tau_j) \subset \mathbb{R}^n$, and an expansion point $z_j^* \in \mathbb{R}^{n+m}$, we have*

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{post}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j), z_j^*\big) \subseteq \underline{\lfloor \texttt{post}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j), z_j^*) \rfloor}(\alpha, id),$$

so that operation `post` is dependency-preserving according to Def. 4.2.4.

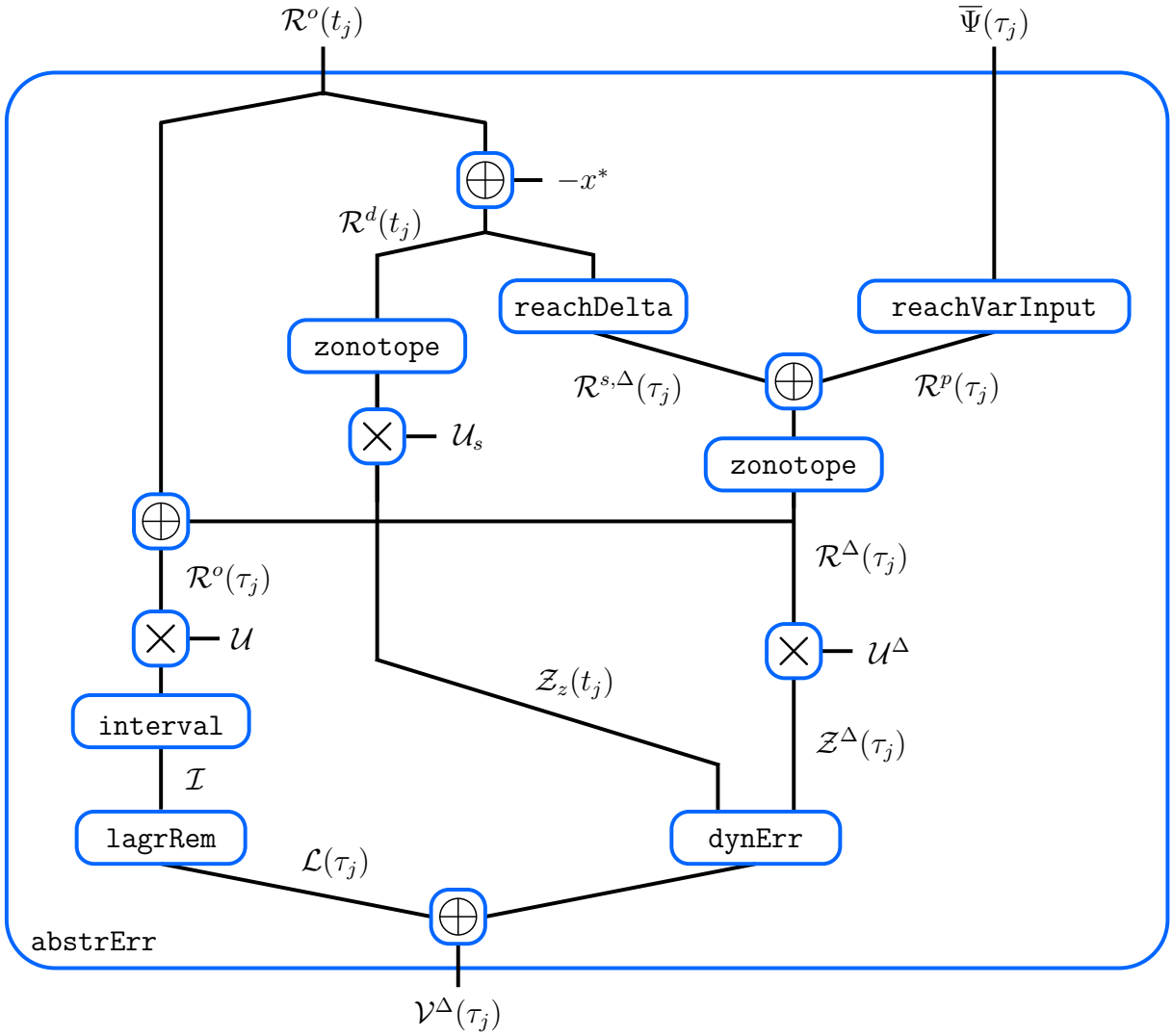

Figure 4.9: Flowchart for the operation `abstrErr` that computes the set of dynamic linearization errors $\mathcal{V}^\Delta(\tau_j)$ for the conservative polynomialization algorithm in Alg. 6, where operations that satisfy Assumption 4.2.7 are depicted in blue. The operations `reachDelta`, `reachVarInput`, `lagrRem`, and `dynErr` are defined as in (4.33), (4.14), (4.34), and (4.35), respectively.

Proof. As shown in Fig. 4.10, the operation `post` representing a single iteration of the conservative polynomialization algorithm in Alg. 6 is defined as

$$\begin{aligned}&\texttt{post}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j), z_j^*\big) = \\ &\qquad \texttt{restructure}\big(\texttt{reduce}\big(\texttt{stat}(\mathcal{R}^o(t_j)) \oplus \texttt{dyn}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)), \rho_d\big), p_d\big),\end{aligned}$$

where the operations `stat` and `dyn` are given as

$$\begin{aligned}\texttt{stat}\big(\mathcal{R}^o(t_j)\big) = &\big(e^{A\Delta t} \otimes \mathcal{R}^o(t_j)\big) \boxplus \\ &\big(\Gamma(\Delta t) \otimes ((f(z^*) - Ax^*) \oplus 0.5sq(\mathcal{Q}, (\mathcal{R}^o(t_j) \oplus (-x^*)) \times \mathcal{U}_s)\big)\end{aligned}$$

and

$$\texttt{dyn}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)\big) = \texttt{reachVarInput}\big(\texttt{abstrErr}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j))\big).$$

The visualization in Fig. 4.10 demonstrates that the operation `stat` is defined by the composition of set operations that are dependency-preserving for SPZs, which proves according to Lemma 4.2.9 that `stat` is dependency-preserving, too:

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{stat}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id)\big) \subseteq \underline{\lfloor \texttt{stat}\big(\mathcal{R}^o(t_j)\big) \rfloor}(\alpha, id). \tag{4.36}$$

Moreover, since operation `abstrErr` satisfies Assumption 4.2.7 according to Lemma 4.2.14, operation `dyn` is defined by a composition of set operations that satisfy Assumption 4.2.7, as it is shown in Fig. 4.10. According to Lemma 4.2.8, operation `dyn` therefore satisfies Assumption 4.2.7, so that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{dyn}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j)\big) \subseteq \texttt{dyn}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)\big)$$

since $\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id) \subseteq \mathcal{R}^o(t_j)$, which implies that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]\ \exists \mathcal{S} \subset \mathbb{R}^n: \quad \texttt{dyn}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j)\big) \oplus \mathcal{S} = \texttt{dyn}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)\big). \tag{4.37}$$

Using (4.36) and (4.37), we obtain for the Minkowski sum $\texttt{stat}(\mathcal{R}^o(t_j)) \oplus \texttt{dyn}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j))$ the result

$$\begin{aligned}
\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \ & \texttt{stat}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id)\big) \oplus \texttt{dyn}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j)\big) \overset{(4.36)}{\subseteq} \\
& \underline{\lfloor \texttt{stat}\big(\mathcal{R}^o(t_j)\big) \rfloor}(\alpha, id) \oplus \texttt{dyn}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j)\big) \subseteq \\
& \underline{\lfloor \texttt{stat}\big(\mathcal{R}^o(t_j)\big) \rfloor}(\alpha, id) \oplus \texttt{dyn}\big(\underline{\lfloor \mathcal{R}^o(t_j) \rfloor}(\alpha, id), \overline{\Psi}(\tau_j)\big) \oplus \mathcal{S} \overset{(4.37)}{=} \\
& \underline{\lfloor \texttt{stat}\big(\mathcal{R}^o(t_j)\big) \rfloor}(\alpha, id) \oplus \texttt{dyn}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)\big) \overset{\text{Prop. 4.2.13}}{=} \\
& \underline{\lfloor \texttt{stat}\big(\mathcal{R}^o(t_j)\big) \oplus \texttt{dyn}\big(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j)\big) \rfloor}(\alpha, id),
\end{aligned}$$

which proves that $\texttt{stat}(\mathcal{R}^o(t_j)) \oplus \texttt{dyn}(\mathcal{R}^o(t_j), \overline{\Psi}(\tau_j))$ is dependency-preserving for SPZs according to Def. 4.2.4. Consequently, as visualized in Fig. 4.10, the operation `post` is defined by a composition of dependency-preserving set operations and therefore is dependency-preserving due to Lemma 4.2.9. □

Next, we extend the results for a single time interval in Lemma 4.2.15 to the overall algorithm:

Lemma 4.2.16. *Given an initial set $\mathcal{X}_0 = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ represented as a SPZ, it holds that*

$$\begin{aligned}
\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \ & \texttt{reach}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id), \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*\big) \\
& \qquad \subseteq \underline{\lfloor \texttt{reach}(\mathcal{X}_0) \rfloor}(\alpha, id),
\end{aligned}$$

where $\overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1})$ and $z_0^*, \ldots, z_{N-1}^*$ are the sets of linearization errors and expansions points resulting from the computation of $\texttt{reach}(\mathcal{X}_0)$.

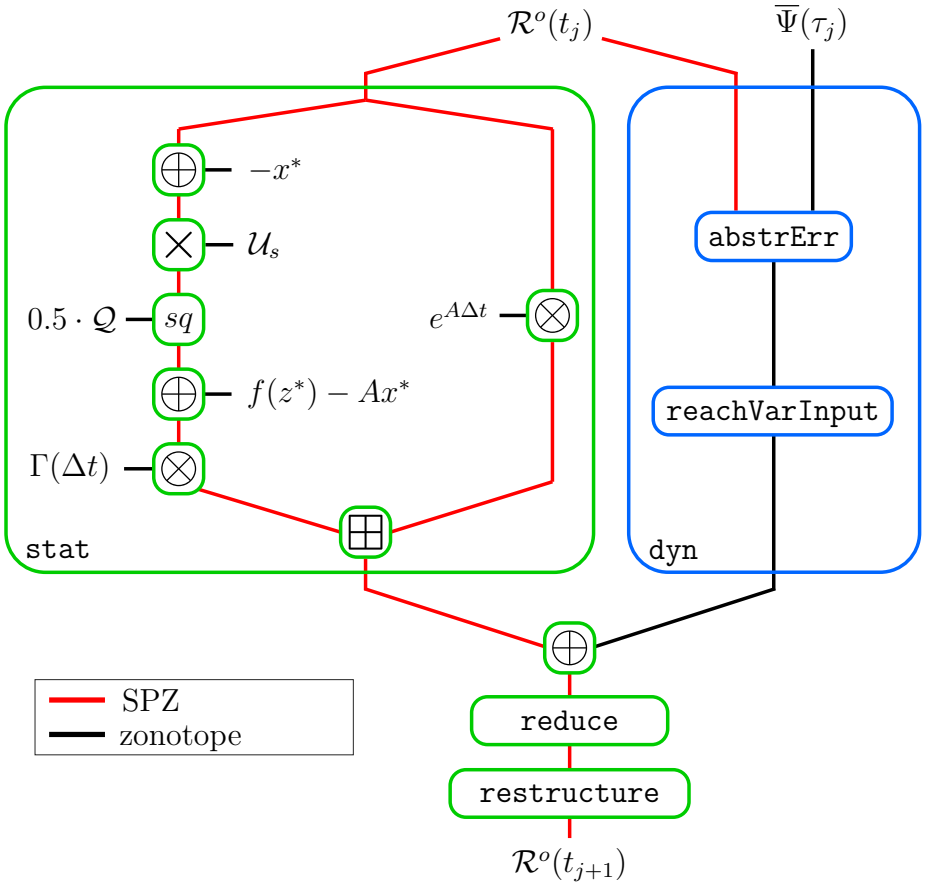

Figure 4.10: Flowchart for a single time interval of the conservative polynomialization algorithm in Alg. 6, where operations that are dependency-preserving for SPZs are depicted in green and operations that satisfy Assumption 4.2.7 are depicted in blue.

Proof. Clearly, the conservative polynomialization algorithm in Alg. 6 can be equivalently expressed as the N-times consecutive execution of the `post` operation:

$$\mathcal{R}^o(t_f) = \underbrace{\texttt{post}\big(\ \ldots\ \texttt{post}(\mathcal{X}_0, \overline{\Psi}(\tau_0), z_0^*)\ \ldots\ , \overline{\Psi}(\tau_{N-1}), z_{N-1}^*\big)}_{\texttt{reach}(\mathcal{X}_0, \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*)}.$$

Consequently, since the operation `post` is dependency-preserving according to Lemma 4.2.15 and the composition of dependency-preserving operations yields a dependency-preserving operation due to Lemma 4.2.9, we have that `reach` is dependency-preserving:

$$\begin{aligned}\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \ \texttt{reach}\left(\lfloor \mathcal{X}_0 \rfloor(\alpha, id), \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*\right) \\ \subseteq \underbrace{\lfloor \texttt{reach}(\mathcal{X}_0, \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*) \rfloor(\alpha, id)}_{= \lfloor \texttt{reach}(\mathcal{X}_0) \rfloor(\alpha, id)},\end{aligned}$$

where $\texttt{reach}(\mathcal{X}_0, \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*) = \texttt{reach}(\mathcal{X}_0)$ holds since the sets of linearization errors $\overline{\Psi}(\tau_j)$ and the expansion points z_j^* are obtained from the computation

of $\texttt{reach}(\mathcal{X}_0)$, which corresponds to the original version of conservative polynomialization algorithm in Alg. 6 without additional input arguments. □

Finally, we formulate the main result:

Theorem 4.2.17. *Given an initial set $\mathcal{X}_0 = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ represented as a SPZ, it holds that $\underline{\lfloor \texttt{reach}(\mathcal{X}_0) \rfloor}(\alpha, id)$ is an outer-approximation of the exact reachable set $\mathcal{R}_{x_0}(t_f)$ starting from $x_0 = \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id)$:*

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \mathcal{R}_{x_0}(t_f) \subseteq \underline{\lfloor \texttt{reach}(\mathcal{X}_0) \rfloor}(\alpha, id) \quad \textit{with} \quad x_0 = \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id),$$

where operation reach *denotes the computation of an outer-approximation of the reachable set using the conservative polynomialization algorithm as specified in Alg. 6.*

Proof. As shown in Sec. 4.1.2, Alg. 6 always computes an outer-approximation of the reachable set, independent of the choice for the expansion points $z_0^*, \ldots, z_{N-1}^*$. Moreover, since $x_0 = \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id) \in \mathcal{X}_0$, it holds that the sets of linearization errors $\overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1})$ calculated during $\texttt{reach}(\mathcal{X}_0)$ enclose the sets of linearization errors for the reachable set starting from x_0. Consequently, we have that $\texttt{reach}(\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id), \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*)$ is an outer-approximation of the exact reachable set starting from x_0:

$$\mathcal{R}_{x_0}(t_f) \subseteq \texttt{reach}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id), \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*\big). \tag{4.38}$$

Using (4.38) and Lemma 4.2.16, we obtain

$$\begin{aligned} \forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \ \mathcal{R}_{x_0}(t_f) &\overset{(4.38)}{\subseteq} \texttt{reach}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id), \overline{\Psi}(\tau_0), \ldots, \overline{\Psi}(\tau_{N-1}), z_0^*, \ldots, z_{N-1}^*\big) \\ &\overset{\text{Lemma 4.2.16}}{\subseteq} \underline{\lfloor \texttt{reach}(\mathcal{X}_0) \rfloor}(\alpha, id), \end{aligned}$$

which concludes the proof. □

Since Thm. 4.2.17 holds for all points $x_0 \in \mathcal{X}_0$ inside the initial set $\mathcal{X}_0$, it is obvious that Thm. 4.2.17 equally holds for all sets $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$. In addition, Thm. 4.2.17 also holds for all reachable sets at intermediate time points $\mathcal{R}^o(t_j)$, $j = 0, \ldots, N$. The time interval reachable set $\mathcal{R}^o(\tau_j)$ is computed in Line 14 of Alg. 6 by a Minkowski sum $\mathcal{R}^o(t_j) \oplus \mathcal{R}^\Delta(\tau_j)$ of the time point reachable set $\mathcal{R}^o(t_j)$ and the zonotope $\mathcal{R}^\Delta(\tau_j)$. Since the Minkowski sum with a zonotope is dependency-preserving for SPZs according to Prop. 4.2.13, we consequently have that Thm. 4.2.17 equally holds for the all time interval reachable sets $\mathcal{R}^o(\tau_j)$, $j = 0, \ldots, N-1$. Let us finally demonstrate the extraction of reachable subsets using Thm. 4.2.17 with an example:

Example 4.2.18. *We consider the Van-der-Pol oscillator in* (4.20) *with the initial set*

$$\mathcal{X}_0 = \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \alpha_2 \;\middle|\; \alpha_1, \alpha_2 \in [-1, 1] \right\}$$

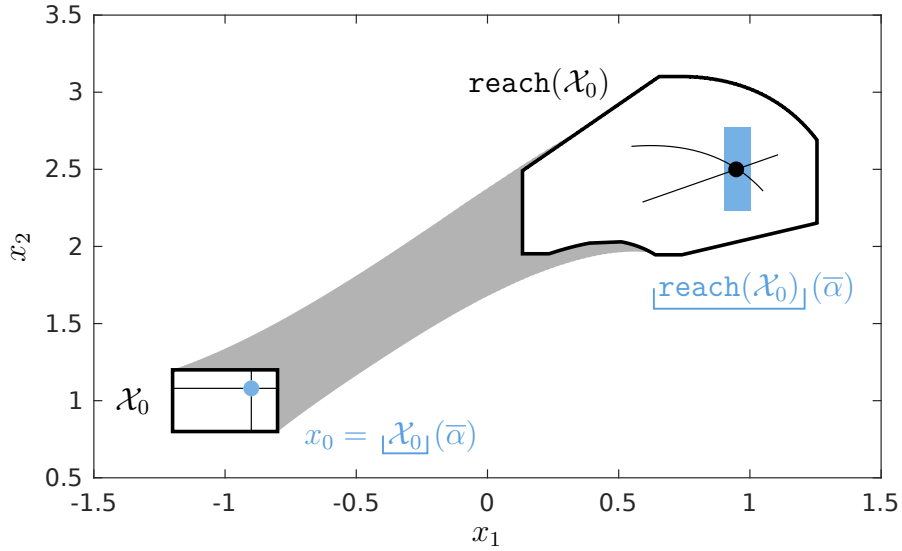

Figure 4.11: Visualization of the reachable set from Example 4.2.18, where the exact reachable set $\mathcal{R}_{x_0}(t_f)$ for the initial point x_0 is depicted by a black dot and the extracted reachable subset is shown in blue. The black lines correspond to the case where $\alpha_1 = 0.5 = const.$, $\alpha_2 = [-1, 1]$ and the case where $\alpha_2 = 0.4 = const.$, $\alpha_1 \in [-1, 1]$.

and the initial point $x_0 = [-0.9\ 1.08]^T \in \mathcal{X}_0$. *The initial point* x_0 *can be parameterized with* $\overline{\alpha} = [0.5\ 0.4]^T$, *so that* $x_0 = \lfloor \mathcal{X}_0 \rfloor(\overline{\alpha})$. *Computation of the reachable set for a time horizon of* $t_f = 1\text{s}$ *with the conservative polynomialization algorithm in Alg. 6 yields the SPZ*

$$\mathcal{R}^o(t_f) = \texttt{reach}(\mathcal{X}_0) = \left\{ \begin{bmatrix} 0.73 \\ 2.52 \end{bmatrix} + \begin{bmatrix} 0.25 \\ -0.1 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0.26 \\ 0.2 \end{bmatrix} \alpha_2 - \begin{bmatrix} 0.04 \\ 0.09 \end{bmatrix} \alpha_1^2 - \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \alpha_1 \alpha_2 + \begin{bmatrix} 0.05 \\ 0 \end{bmatrix} \beta_1 + \begin{bmatrix} 0 \\ 0.27 \end{bmatrix} \beta_2 \;\middle|\; \alpha_1, \alpha_2, \beta_1, \beta_2 \in [-1, 1] \right\}.$$

To extract the reachable subset for the initial point x_0, *we evaluate* $\mathcal{R}^o(t_f)$ *with the parameter values* $\overline{\alpha}$ *corresponding to* x_0. *Since the number of dependent factors did not change, it holds that* $\lfloor \texttt{reach}(\mathcal{X}_0) \rfloor(\overline{\alpha}, id) = \lfloor \texttt{reach}(\mathcal{X}_0) \rfloor(\overline{\alpha})$, *so that we can use the simple definition of the evaluation function in Def. 4.2.10. This yields the set*

$$\lfloor \mathcal{R}^o(t_f) \rfloor(\overline{\alpha}) = \lfloor \texttt{reach}(\mathcal{X}_0) \rfloor(\overline{\alpha}) = \left\{ \begin{bmatrix} 0.949 \\ 2.5075 \end{bmatrix} + \begin{bmatrix} 0.05 \\ 0 \end{bmatrix} \beta_1 + \begin{bmatrix} 0 \\ 0.27 \end{bmatrix} \beta_2 \;\middle|\; \beta_1, \beta_2 \in [-1, 1] \right\},$$

which encloses the exact reachable set $\mathcal{R}_{x_0}(t_f)$ *starting from the initial point* x_0 *according to Thm. 4.2.17. A visualization of the extracted outer-approximation of the reachable subset is shown in Fig. 4.11.*

4.2.4 Computational Complexity

We now derive the computational complexity for the extraction of reachable subsets. For this, we first show how the extended evaluation function for SPZs as defined in Def. 4.2.11 can be implemented:

Proposition 4.2.19. *(Extended Evaluation Function SPZ) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$*, a parameter vector* $\widehat{\alpha} \in \mathbb{R}^w$*, and an identifier vector* $\widehat{id} \in \mathbb{N}^{1\times w}$ *satisfying* $\forall i, j \in \{1, \ldots, w\} : \ \widehat{id}_{(i)} \neq \widehat{id}_{(j)}$*, the extended evaluation function* $\underline{\lfloor \mathcal{PZ} \rfloor}(\widehat{\alpha}, \widehat{id})$ *as defined in Def. 4.2.11 can be computed as*

$$\underline{\lfloor \mathcal{PZ} \rfloor}(\widehat{\alpha}, \widehat{id}) = \langle c, G\,D, G_I, E_{(\mathcal{K},\cdot)}, id_{(\mathcal{K})} \rangle_{PZ},$$

where

$$\mathcal{H} = \left\{ i \;\middle|\; \exists j \in \{1, \ldots, w\} : \ \widehat{id}_{(j)} = id_{(i)} \right\}, \quad \mathcal{K} = \{1, \ldots, p\} \setminus \mathcal{H}$$

$$D = \mathtt{diag}\left(\left[\prod_{k\in\mathcal{H}} \alpha_{(k)}^{E_{(k,1)}} \ \ldots \ \prod_{k\in\mathcal{H}} \alpha_{(k)}^{E_{(k,h)}} \right] \right), \ \forall k \in \{1, \ldots, p\} : \ \alpha_{(k)} = \begin{cases} \widehat{\alpha}_{(i)}, & \widehat{id}_{(i)} = id_{(k)} \\ 0, & \textit{otherwise} \end{cases}.$$

The `compact` *operation as defined in Prop. 3.1.7 is applied to make the resulting SPZ regular. The computational complexity with respect to the dimension* n *is* $\mathcal{O}(nw + n^2 \log(n))$*, where* w *is the length of the parameter vector* $\widehat{\alpha}$*.*

Proof. The result is obtained by splitting the dependent factors into factors that are replaced by numerical values ($k \in \mathcal{H}$) and remaining factors ($k \in \mathcal{K}$):

$$\underline{\lfloor \mathcal{PZ} \rfloor}(\widehat{\alpha}, \widehat{id}) \overset{\text{Def. 4.2.11}}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \widetilde{\alpha}_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \breve{\alpha}_1, \ldots, \breve{\alpha}_p \in [-1, 1] \right\} \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1, 1] \right\} =$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k\in\mathcal{K}} \breve{\alpha}_k^{E_{(k,i)}} \right) \underbrace{\left(\prod_{k\in\mathcal{H}} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)}}_{D_{(i,i)}\, G_{(\cdot,i)}} \;\middle|\; \breve{\alpha}_k \in [-1, 1] \right\} \oplus \langle \mathbf{0}, G_I \rangle_Z$$

$$\overset{\text{Def. 3.1.1}}{=} \langle c, G\,D, G_I, E_{(\mathcal{K},\cdot)}, id_{(\mathcal{K})} \rangle_{PZ},$$

where

$$\forall k \in \{1, \ldots, p\} : \ \ \widetilde{\alpha}_{(k)} = \begin{cases} \widehat{\alpha}_{(i)}, & \widehat{id}_{(i)} = id_{(k)} \\ \breve{\alpha}_k, & \text{otherwise} \end{cases}$$

is defined as in Def. 4.2.11.

Complexity: Construction of the sets $\mathcal{H}$, $\mathcal{K}$ and the vector $\alpha \in \mathbb{R}^p$ has complexity $\mathcal{O}(wp)$. Computation of one diagonal entry $\prod_{k\in\mathcal{H}} \alpha_{(k)}^{E_{(k,i)}}$ of the matrix D requires at most w exponentiations and $w - 1$ multiplications since $|\mathcal{H}| \leq w$, so that the complexity for the construction of $D \in \mathbb{R}^{h\times h}$ is $h \cdot \mathcal{O}(2w - 1) = \mathcal{O}(wh)$. Moreover, since D is a diagonal matrix, the matrix multiplication $G\,D$ has complexity $\mathcal{O}(nh)$. The resulting SPZ has h generators and at most p dependent factors, so that the subsequent application of the

`compact` operation has complexity $\mathcal{O}(h(n+p\log(h)))$ according to Prop. 3.1.7. The overall complexity for the evaluation function of a SPZ is therefore

$$\mathcal{O}(wp) + \mathcal{O}(wh) + \mathcal{O}(nh) + \mathcal{O}(h(n + p\log(h))),$$

which is $\mathcal{O}(nw + n^2\log(n))$ using Assumption 3.1.3. □

The result of the evaluation function for an interval $[l, u] \subseteq [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$ of possible parameter values $\alpha \in [l, u]$ can be implemented by successive application of the `getSubset` operation as specified in Prop. 3.1.43:

$$\underline{\lfloor \mathcal{PZ} \rfloor}([l, u]) = \texttt{getSubset}\big(\ \ldots\ \texttt{getSubset}(\mathcal{PZ}, 1, [l_{(1)}, u_{(1)}])\ \ldots\ , p, [l_{(p)}, u_{(p)}]\big), \quad (4.39)$$

where we considered the definition of the evaluation function in Def. 4.2.10 for simplicity since handling the extended evaluation function is straightforward. According to Thm. 4.2.17, we can extract reachable subsets by applying the implementation of the extended evaluation function in Prop. 4.2.19 to the computed outer-approximation $\mathcal{R}^o(t_f)$ of the final reachable set. The computational complexity of reachable subset extraction is therefore $\mathcal{O}(nw+n^2\log(n))$ according to Prop. 4.2.19, which is $\mathcal{O}(n^2\log(n))$ using Assumption 3.1.3 and the fact the parameter vector has dimension $w = p$, where p is the number of dependent factors of the SPZ representing the initial set $\mathcal{X}_0$. Since the computational complexity for reachability analysis using the conservative polynomialization algorithm is $\mathcal{O}(n^5)$ according to Sec. 4.1.4, our novel method for the extraction of reachable subsets is computationally much more efficient than computing the reachable subset with a reachability algorithm. We will further substantiate this by several numerical examples in the next section.

4.2.5 Numerical Examples

Our novel reachable subset approach results in significant improvements for many different applications. We exemplary demonstrate this for safety falsification, optimization over reachable set, and motion-primitive-based control. As shown in Tab. 4.4, for all of these applications using reachable subsets results in significant speed-ups compared to the previous solution without reachable subsets.

Safety Falsification

If the reachable set does not fulfill a given specification, it would be helpful to know one concrete trajectory that demonstrates the violation. The task of finding such a trajectory defined by an initial point and an input signal is known as safety falsification. For simplicity, we consider a specification defined by a linear inequality constraint $a^T x \leq b$, $a \in \mathbb{R}^n, b \in \mathbb{R}$. However, the presented concept is easily expendable to arbitrary complex specifications including temporal logic expressions by considering suitable robustness measures that quantify the amount of violation [125]. With our novel reachable subset approach in Thm. 4.2.17, the initial point x_0^* resulting in the largest violation of the specification can be determined by solving the optimization problem

$$x_0^* = \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha^*, id) \ \text{ with } \ \alpha^* = \underset{\alpha \in [-\mathbf{1}, \mathbf{1}]}{\arg\max} \ \ a^T \otimes \underline{\lfloor \mathcal{R}^o(\tau_j) \rfloor}(\alpha, id), \quad (4.40)$$

Table 4.4: Comparison of the computation times in seconds between the new approach using reachable subsets and the previous solution without reachable subsets for different applications.

Application	**Computation Time**	
	New Approach	**Previous Solution**
Safety falsification (Van-der-Pol)	0.12	0.38
Safety falsification (quadrocopter)	6.39	37.8
Optimization over reachable sets	0.13	172
Motion-primitive-based control	0.06	21.4

where $\mathcal{R}^o(\tau_j)$ is the reachable set of the time interval τ_j for which the violation of the specification is the largest and *id* is the identifier vector of the SPZ that represents the initial set $\mathcal{X}_0$. With the initial point from (4.40), safety falsification then reduces to the task of finding a suitable input signal. Let us demonstrate the advantages of our novel safety falsification approach by an example for which we compare our results with the safety falsification toolbox S-TaLiRo [122]:

Example 4.2.20. *We consider the reachability problem from Example 4.2.18 featuring the Van-der-Pol oscillator in combination with the specification* $[1\ 2]\,x \leq 6.4$. *As shown in Fig. 4.12, the final reachable set* $\mathcal{R}^o(t_f)$ *violates the specification the most. Inserting the numerical values for* $\mathcal{R}^o(t_f)$ *from Example 4.2.18 into* (4.40) *yields the optimization problem*

$$\alpha^* = \underset{\alpha\in[-\mathbf{1},\mathbf{1}]}{\arg\max}\ \ a^T \otimes \underline{\lfloor \mathcal{R}^o(t_f) \rfloor}(\alpha, id) \overset{\text{Example 4.2.18}}{=}$$

$$\underset{\alpha\in[-\mathbf{1},\mathbf{1}]}{\arg\max}\ \ [1\ 2] \otimes \underline{\left\lfloor \left\langle \begin{bmatrix} 0.73 \\ 2.52 \end{bmatrix}, \begin{bmatrix} 0.25 & 0.26 & -0.04 & 0 \\ -0.1 & 0.2 & -0.09 & -0.1 \end{bmatrix}, \begin{bmatrix} 0.05 & 0 \\ 0 & 0.27 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, [1\ 2] \right\rangle_{PZ} \right\rfloor}(\alpha, id) \tag{4.41}$$

$$\overset{\text{Prop. 3.1.18}}{=} \underset{\alpha\in[-\mathbf{1},\mathbf{1}]}{\arg\max}\ 0.05\alpha_{(1)} + 0.66\alpha_{(2)} - 0.22\alpha_{(1)}^2 - 0.2\alpha_{(1)}\alpha_{(2)},$$

where we omit the constant part resulting from the constant offset and the independent generators of the SPZ since it does not change the result. The optimization problem in (4.41) *can be solved efficiently using a nonlinear programming solver. We use MATLAB's fmincon with the interior-point algorithm*[3]*, which yields the solution* $\alpha^* = [-0.34\ 1]^T$ *and requires a computation time of* 0.12 *seconds. Since the reachability problem from Example 4.2.18 does not include uncertain inputs, it is sufficient to determine a suitable initial point to obtain a falsifying trajectory. With the value for the parameter vector* α^* *we get*

[3] `https://www.mathworks.com/help/optim/ug/fmincon.html`

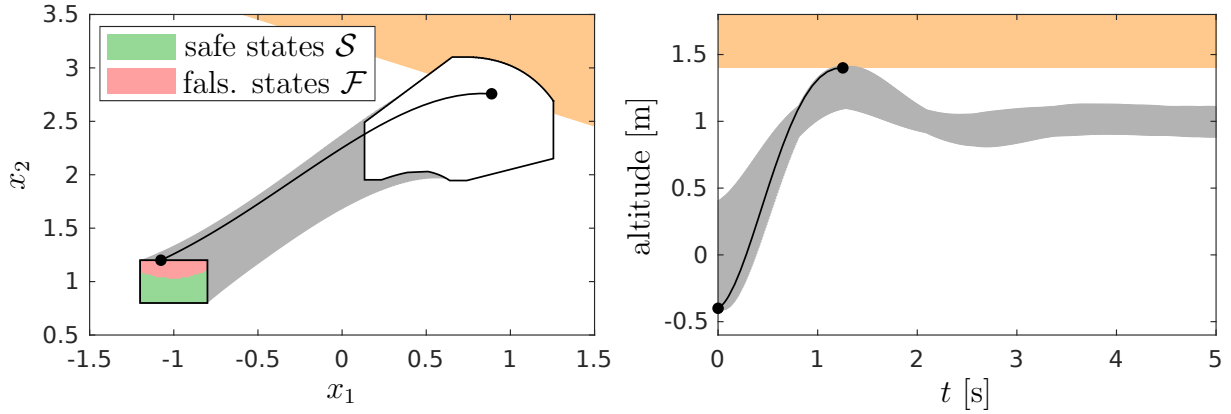

Figure 4.12: Falsifying trajectories (black) for the Van-der-Pol oscillator (left) in Example 4.2.20 and the quadrocopter benchmark (right) in Example 4.2.21, where the unsafe sets defined by the specification are depicted in orange.

the initial point $x_0^* = \underline{\mathcal{X}_0}(\alpha^*, id) = [-1.068\ 1.2]^T$ *with* $id = [1\ 2]$ *according to* (4.40) *and Example 4.2.18. The corresponding falsifying trajectory is visualized in Fig. 4.12. As a comparison, we determined a falsifying trajectory using the simulated annealing algorithm from S-TaLiRo, which is based on Monte-Carlo sampling. Since the simulated annealing algorithm is non-deterministic, we compute the average computation time from* 10 *executions, which requires* 0.38 *seconds on average. Our novel falsification approach using reachable subsets is therefore significantly faster than S-TaLiRo.*

To demonstrate the scalability of our novel approach, we consider a second high-dimensional example, which also has uncertain inputs:

Example 4.2.21. *We consider the 12-dimensional quadrocopter benchmark from the 2019 ARCH competition [111, Sec. 3.3]. To increase the difficulty, we append the system model by adding uncertain additive inputs* $u = [u_1\ u_2\ u_3]^T \in \mathcal{U} = [-0.3, 0.3]\text{m/s}^2 \times [-0.3, 0.3]\text{m/s}^2 \times [-0.3, 0.3]\text{m/s}^2$ *to the differential equations for the states* x_4*,* x_5 *and* x_6*. The specification for the benchmark is* $x_3 \leq 1.4\text{m}$*, which describes that the altitude of the quadrocopter corresponding to the state* x_3 *should stay below* 1.4m *during the considered time horizon of* $t_f = 5\text{s}$*. To determine a falsifying trajectory, we first solve* (4.41) *using MATLAB's fmincon with the interior-point algorithm to determine a suitable initial state. Next, we determine a suitable input signal using the simulated annealing algorithm from S-TaLiRo. The resulting falsifying trajectory is visualized in Fig. 4.12. The overall computation time for falsification is* 6.39 *seconds, where the computation of the initial point using our novel reachable subset approach takes* 0.13 *seconds. As a comparison, we determine both, the initial state and the input signal, using the simulated annealing algorithm from S-TaLiRo. This takes* 37.8 *seconds and is therefore significantly slower than the reachable subset approach. As in Example 4.2.20, we averaged the computation time for S-TaLiRo over* 10 *executions since the simulated annealing algorithm is non-deterministic.*

In addition to obtaining a falsifying trajectory, it would also be helpful to know which states from the initial set result in a violation of the specification, and which states are safe. For simplicity, let us consider the case in which only the reachable set $\mathcal{R}^o(\tau_j)$ for one

single time interval τ_j violates the specification $a^T x \leq b$. According to Thm. 4.2.17, the states inside the set $\mathcal{S} \subseteq \mathcal{X}_0$ defined as

$$\mathcal{S} := \bigcup_{\alpha \in \mathcal{B}} \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id) \;\text{ with }\; \mathcal{B} = \left\{ \alpha \in [-\mathbf{1}, \mathbf{1}] \;\middle|\; a^T \otimes \underline{\lfloor \mathcal{R}^o(\tau_j) \rfloor}(\alpha, id) \leq b \right\} \tag{4.42}$$

are guaranteed to fulfill the specification. Consequently, the set $\mathcal{F} := \mathcal{X}_0 \setminus \mathcal{S} \subseteq \mathcal{X}_0$ contains the initial states that potentially result in a violation. Since both, the initial set $\mathcal{X}_0 = \langle c_0, G_0, [\,], E_0, id \rangle_{PZ}$ and the reachable set $\mathcal{R}^o(\tau_j) = \langle c, G, G_I, E, id \rangle_{PZ}$ are represented by SPZs, it holds that the set of safe states as defined in (4.42) can be represented as a CPZ:

$$\mathcal{S} \overset{(4.42)}{=} \bigcup_{\alpha \in \mathcal{B}} \underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha, id) = \bigcup_{\alpha \in \mathcal{B}} \underline{\lfloor \langle c_0, G_0, [\,], E_0, id \rangle_{PZ} \rfloor}(\alpha, id) \overset{\text{Def. } 4.2.11}{=}$$

$$\left\{ c_0 + \sum_{i=1}^{h_0} \left(\prod_{k=1}^{p_0} \alpha_{(k)}^{E_{0(k,i)}} \right) G_{0(\cdot,i)} \;\middle|\; a^T c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) a^T G_{(\cdot,i)} \right.$$
$$\left. \underbrace{+ \sum_{j=1}^{q} \beta_j \; a^T G_{I(\cdot,j)} \leq b, \;\; \alpha_{(k)}, \beta_j \in [-1, 1]}_{a^T \otimes \underline{\lfloor \mathcal{R}^o(\tau_j) \rfloor}(\alpha, id)} \right\} =$$

$$\left\{ c_0 + \sum_{i=1}^{h_0} \left(\prod_{k=1}^{p_0} \alpha_{(k)}^{E_{0(k,i)}} \right) G_{0(\cdot,i)} \;\middle|\; a^T c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) a^T G_{(\cdot,i)} \right.$$
$$\left. + \sum_{j=1}^{q} \beta_j \; a^T G_{I(\cdot,j)} = \frac{b+d}{2} + \frac{b-d}{2} \lambda, \;\; \alpha_{(k)}, \beta_j, \lambda \in [-1, 1] \right\} = \tag{4.43}$$

$$\left\{ c_0 + \sum_{i=1}^{h_0} \left(\prod_{k=1}^{p_0} \alpha_k^{E_{0(k,i)}} \right) G_{0(\cdot,i)} \;\middle|\; \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) a^T G_{(\cdot,i)} \right.$$
$$\left. + \left(\sum_{j=1}^{q} \left| a^T G_{I(\cdot,j)} \right| - \frac{b-d}{2} \right) \alpha_{p+1} = \frac{b+d}{2} - a^T c, \;\; \alpha_1, \ldots, \alpha_{p+1} \in [-1, 1] \right\} =$$

$$\left\langle c_0, G_0, \begin{bmatrix} E_0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} a^T G & a^T G_I & \left(\sum_{i=1}^{q} \left| a^T G_{I(\cdot,j)} \right| - \frac{b-d}{2} \right) \end{bmatrix}, \frac{b+d}{2} - a^T c, \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \right\rangle_{CPZ},$$

where the support function enclosure

$$d \leq -s_{\mathcal{H}}(-1) \;\text{ with }\; \mathcal{H} = a^T \otimes \mathcal{R}^o(\tau_j)$$

is computed using Prop. 3.1.16 and we assumed for simplicity that the SPZs $\mathcal{X}_0$ and $\mathcal{R}^o(\tau_j)$ have identical identifier vectors. For the general case where the reachable set violates the specification for more than one time interval, the set of safe states can be computed by

intersection using Prop. 3.2.23. Equivalently, the set of falsifying states $\mathcal{F}$ can be obtained by applying (4.43) to the inverted constraint $a^T x \geq b$ and combining the CPZs for different time intervals with the union according to Prop. 3.2.25. Again, the concept presented here is easily expendable to more complex specifications. For the reachability problem and specification in Example 4.2.20 only the final reachable set $\mathcal{R}^o(t_f)$ violates the specification, so that with $d = 4.1$ and using (4.43) we obtain the set

$$\mathcal{S} = \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \alpha_2 \;\middle|\; 0.05\alpha_1 + 0.66\alpha_2 - 0.22\alpha_1^2 - 0.2\alpha_1\alpha_2 - 0.56\alpha_3 = -0.52,\ \alpha_1, \alpha_2, \alpha_3 \in [-1, 1] \right\},$$

which is visualized in Fig. 4.12.

Optimization over Reachable Sets

Since reachability analysis is computational expensive, solving nonlinear optimization problems for which a reachable set has to be computed in every evaluation of the cost and/or constraint function is often infeasible. However, with our new reachable subset approach it is possible to achieve major speed-ups for optimizing over reachable sets. As an example, we consider the problem of finding the largest subset $\mathcal{X}_0^* \subseteq \mathcal{X}_0$ of the initial set $\mathcal{X}_0$ such that the final reachable set $\mathcal{R}^o_{\mathcal{X}_0^*}(t_f) = \texttt{reach}(\mathcal{X}_0^*)$ satisfies a linear inequality constraint $a^T x \leq b$. To solve this problem, we optimize the lower bound $\underline{\alpha}$ and the upper bound $\overline{\alpha}$ of the interval domain for the dependent factors:

$$\max_{-\mathbf{1} \leq \underline{\alpha} \leq \overline{\alpha} \leq \mathbf{1}} \texttt{volume}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}([\underline{\alpha}, \overline{\alpha}], id)\big) \;\text{ s.t. }\; \forall x \in a^T \otimes \texttt{reach}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}([\underline{\alpha}, \overline{\alpha}], id)\big) : x \leq b, \tag{4.44}$$

where id is the identifier of the SPZ representing the initial set $\mathcal{X}_0$. If we solve (4.44) directly using a nonlinear programming solver, we have to execute reachability analysis for every evaluation of the constraint function, which is computationally expensive. However, with our novel reachable subset approach specified in Thm. 4.2.17, we can reformulate (4.44) as

$$\max_{-\mathbf{1} \leq \underline{\alpha} \leq \overline{\alpha} \leq \mathbf{1}} \texttt{volume}\big(\underline{\lfloor \mathcal{X}_0 \rfloor}([\underline{\alpha}, \overline{\alpha}], id)\big) \;\text{ s.t. }\; \forall \alpha \in [\underline{\alpha}, \overline{\alpha}] : \underline{\lfloor a^T \otimes \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\alpha, id) \leq b, \tag{4.45}$$

which avoids the execution of reachability analysis in every evaluation of the constraint function and is therefore computationally much more efficient to solve. Let us demonstrate this with a concrete example:

Example 4.2.22. *As in Example 4.2.20, we again consider the reachability problem from Example 4.2.18 featuring the Van-der-Pol oscillator together with the specification* $[1\ 2]\, x \leq 6.4$. *With the numerical values from Example 4.2.18, we obtain for* (4.45)

$$\begin{aligned} \max_{-\mathbf{1} \leq \underline{\alpha} \leq \overline{\alpha} \leq \mathbf{1}} \;& 0.4^2 \big(\overline{\alpha}_{(1)} - \underline{\alpha}_{(1)}\big)\big(\overline{\alpha}_{(2)} - \underline{\alpha}_{(2)}\big) \\ \textit{s.t.} \;& \forall \alpha \in [\underline{\alpha}, \overline{\alpha}] : \; 6.36 + 0.05\alpha_{(1)} + 0.66\alpha_{(2)} - 0.22\alpha_{(1)}^2 - 0.2\alpha_{(1)}\alpha_{(2)} \leq 6.4. \end{aligned} \tag{4.46}$$

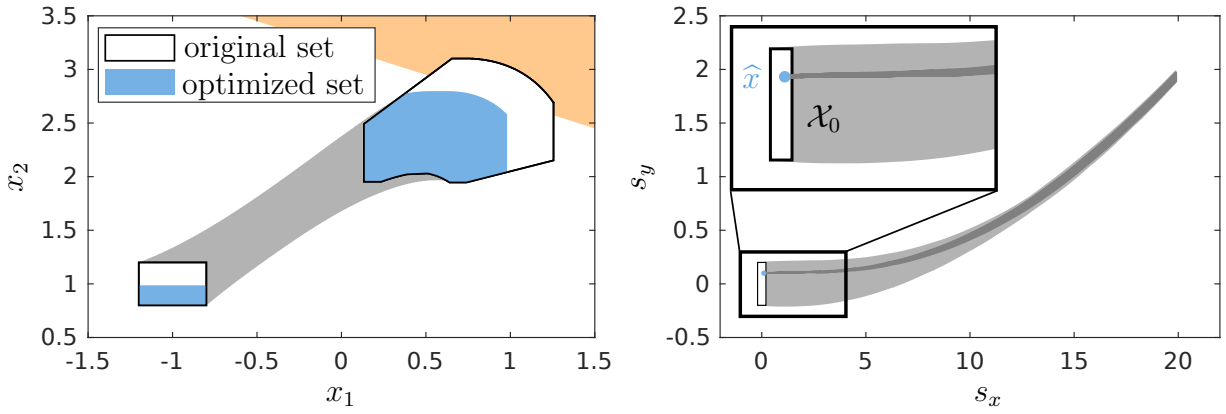

Figure 4.13: On the left the results for optimization over reachable sets from Example 4.2.22 are visualized, where the unsafe set defined by the specification is depicted in orange. On the right the extracted reachable subset (dark gray) for the motion primitive from Example 4.2.23 is shown, where the offline-computed reachable set is depicted in light gray.

Solving the optimization problem (4.46) *takes* 0.13 *seconds, while solving the optimization problem* (4.44) *takes* 172 *seconds, where in both cases we used MATLAB's fmincon with the interior-point algorithm and applied range bounding (see Sec. 2.7) to evaluate the universal quantifications. This demonstrates that by using the reachable subset approach we can achieve significant speed-ups for optimization over reachable sets. The optimized sets resulting from* (4.46) *are visualized in Fig. 4.13.*

Motion-Primitive-Based Control

Maneuver automata represent an efficient method for generating motion plans that are guaranteed to be robustly save despite disturbances [126]. For this, control laws $u_{ctr}(x)$ for many motion primitives are synthesized offline using for example one of the approaches in [127–129]. Next, the reachable set of the controlled system starting from an initial set $\mathcal{X}_0$ is computed for each motion primitive to guarantee robust safety despite disturbances. The resulting motion primitives consisting of a controller and the corresponding reachable set are then stored in a maneuver automaton that saves which motion primitives can be connected to one another. With this offline-constructed maneuver automaton, online motion planning tasks can be solved efficiently by simply concatenating the motion primitives, where the reachable sets are used for collision checking. To ensure that motion primitives can be connected safely and to cover the whole state space, an initial set $\mathcal{X}_0$ instead of a single initial point is used when computing the reachable sets offline. As a consequence, the resulting reachable sets are often quite large, which makes it difficult to find a collision-free motion plan. However, during online application we obtain a single measurement of the system state $\widehat{x} \in \mathcal{X}_0$. Clearly, using the smaller reachable set starting from $\widehat{x}$ would significantly increase the chances for finding a collision-free motion plan. While computation of the reachable set for $\widehat{x}$ using a reachability algorithm is too slow to be real-time capable for most systems, with our novel approach we can extract the reachable subset for $\widehat{x}$ directly from the offline-computed reachable set, which is computationally much more efficient.

Let us demonstrate this by an example:

Example 4.2.23. *We consider the kinematic single-track model of an autonomous car in [127, Sec. 6]:*

$$\begin{aligned}\dot{v} &= u_{ctr,1}(x) + u_1\\ \dot{\phi} &= u_{ctr,2}(x) + u_2\\ \dot{s}_x &= v\cos(\phi)\\ \dot{s}_y &= v\sin(\phi),\end{aligned}$$

where the system state $x = [v\ \phi\ s_x\ s_y]^T$ consists of the velocity v, the orientation ϕ, and the x and y positions of the cars reference point s_x and s_y. The uncertain inputs $u = [u_1\ u_2]^T$ are bounded by the set $\mathcal{U} = [-0.5, 0.5]\text{m/s}^2 \times [-0.02, 0.02]\text{rad/s}$. We use the approach in [127, Sec. 5] to synthesize a control law $u_{ctr}(x) = [u_{ctr,1}(x)\ u_{ctr,2}(x)]^T$ for the turn-left maneuver described in [127, Sec. 6], and compute the reachable set for the controlled system starting from the initial set $\mathcal{X}_0 = [19.8, 20.2]\text{m/s} \times [-0.02, 0.02]\text{rad} \times [-0.2, 0.2]\text{m} \times [-0.2, 0.2]\text{m}$ for a time horizon of $t_f = 1\text{s}$ using the conservative polynomialization algorithm in Alg. 6. For this motion primitive, extraction of the reachable subset for the measured state $\widehat{x} = [20.1\text{m/s}\ 0.01\text{rad}\ 0.1\text{m}\ 0.1\text{m}]^T$ using Thm. 4.2.17 takes only 0.06 seconds, which is fast enough to be done online. On the other hand, calculation of the reachable set starting from $\widehat{x}$ using Alg. 6 takes 21.4 seconds and is therefore by far too slow to be computed online. The extracted reachable subset is visualized in Fig. 4.13.

Clearly, for all three applications that we considered in this section, our novel reachable subset approach resulted in significant improvements and speed-ups. Moreover, the fact that the presented applications belong to different research domains further demonstrates the broad applicability of the approach.

4.3 Inner-Approximations of Reachable Sets for Nonlinear Continuous Systems

Outer-approximations of reachable sets can be used to verify that a system satisfies a given specification: If the outer-approximation does not intersect any of the unsafe regions defined by the specification, safety is guaranteed. However, if the outer-approximation intersects an unsafe region, it is not clear if the system is really unsafe, or if the intersection only occurs due to the over-approximation. To solve this problem, one can use inner-approximations of reachable sets: If the inner-approximation intersects an unsafe region, it is guaranteed that the specification is violated. Another application of inner-approximations is backward reachability analysis since the backward reachable set can simply be calculated by computing an inner-approximation of the time-inverted system. In this section[4], we introduce a novel method for computing non-convex inner-approximations of reachable sets for nonlinear continuous systems. For our approach we first compute an outer-approximation of the reachable set using the conservative polynomialization algorithm described in Sec. 4.1 and then apply the reachable subset method introduced in Sec. 4.2 to compute an inner-approximation.

The structure of this section is as follows: First, we summarize the state of the art for computing inner-approximations in Sec. 4.3.1. In the main part in Sec. 4.3.2, we introduce our novel approach for computing inner-approximations for nonlinear systems without inputs and then show in Sec. 4.3.3 how to extend this approach to handle systems with uncertain inputs. Afterward, we derive the computational complexity of our novel method in Sec. 4.3.4. Finally, in Sec. 4.3.5, we consider several benchmark systems for which we compare the performance of our new approach to other state of the art methods.

4.3.1 State of the Art

As shown in Sec. 4.1.1, the computation of outer-approximations of reachable sets for nonlinear continuous systems is a well-studied problem for which many approaches are available. The problem of calculating inner-approximations of reachable sets has been studied far less. Efficient techniques for computing inner-approximations for linear continuous systems exist for quite some time: The approach in [131] uses zonotopes to compute inner-approximations for linear time-invariant systems with piecewise constant inputs. For time-varying linear systems, the method in [132] computes inner-approximations represented by ellipsoids. The authors in [133] compute inner-approximations for piecewise-affine systems using linear matrix inequalities.

Only recently, methods for computing inner-approximations for nonlinear continuous systems have been developed: In [134] a criterion on when an interval is part of the inner-approximation is provided. Based on this criterion, an inner-approximation of the reachable set represented by a union of intervals can be computed. However, this method is computationally expensive for high-dimensional systems due to the curse of dimensionality. The same authors present another approach in [135], where they represent inner-approximation with a single parallelotope. Since reachable sets of nonlinear systems, how-

[4] This section is based on [130].

ever, usually have complex non-convex shapes, the accuracy achievable with parallelotope inner-approximations is often quite low.

Several approaches compute inner-approximations represented by sub-zero level sets based on the Hamilton-Jacobi framework [52]: The method in [136] calculates inner-approximations by under-approximating the evolution function. In [53] and [137], inner-approximations for polynomial systems are obtained by solving semi-definite programs. Since the size of the semi-definite program grows rapidly with the system dimension, the approaches in [53, 137] are computationally expensive for high-dimensional systems.

Other approaches use the time-inverted dynamics to compute inner-approximations: The method in [18] computes inner-approximations represented by polytopes using linear programming. To achieve this, a set which encloses the boundary is computed first. Afterward, a polytope outer-approximation of the reachable set is contracted until it is inclosed by the set enclosing the boundary. In [138], this approach is extended to handle delay differential equations. Since reachable sets of nonlinear systems are in general non-convex, inner-approximations represented by convex polytopes as computed in [18, 138] are often not very accurate. The approach in [116] first computes a backward-flowpipe based on Picard iteration. This flowpipe is then used to propagate the inequality constraints that define the initial set forward in time, which yields a non-convex inner-approximation represented by the intersection of polynomial inequality constraints. However, the intersection of polynomial inequality constrains can result in sets that are not connected. To determine a valid inner-approximation, the method in [116] therefore requires to prove that the resulting set is connected, which is computationally demanding.

The approach in [139] demonstrates that inner-approximations for the projection of the reachable set onto the coordinate axes can be computed very efficiently for autonomous nonlinear systems. In [140], the authors extend this approach to systems with uncertain inputs. While inner-approximations of the projection are often useful for simple verification tasks, they are in general not sufficient to answer more complex verification queries.

4.3.2 Computing Non-Convex Inner-Approximations

We now present our novel approach for computing tight non-convex inner-approximations $\mathcal{R}^i(t) \subseteq \mathcal{R}(t)$ of the exact reachable set as defined in Def. 2.3.4. For simplicity, we first consider autonomous nonlinear systems defined as $\dot{x}(t) = f(x(t))$, and show later in Sec. 4.3.3 how our method can be extended to handle nonlinear systems with uncertain inputs as defined in Def. 2.3.2. Different from Sec. 4.1 where we computed outer-approximations of the time point reachable set $\mathcal{R}^o(t_j)$ and the time interval reachable set $\mathcal{R}^o(\tau_j)$ for consecutive time intervals, here we focus on calculating an inner-approximation of the final reachable set $\mathcal{R}^i(t_f)$, where t_f is the final time. Our approach is based on the following theorem from [116, Sec. III]:

Theorem 4.3.1. *Given a set $\mathcal{B} \supseteq \partial\mathcal{R}(t)$ that encloses the boundary $\partial\mathcal{R}(t)$ of the exact reachable set, every connected set $\mathcal{C}$ that does not intersect $\mathcal{B}$ and contains some state of the exact reachable set $\mathcal{R}(t) \subset \mathbb{R}^n$ is an inner-approximation of the reachable set:*

$$\forall \mathcal{C} \subset \mathbb{R}^n : \ \big(\mathcal{C} \cap \mathcal{B} = \emptyset\big) \wedge \big(\mathcal{C} \cap \mathcal{R}(t) \neq \emptyset\big) \Rightarrow \big(\mathcal{C} \subseteq \mathcal{R}(t)\big).$$

The theorem holds for arbitrary times $t \in \mathbb{R}_{\geq 0}$.

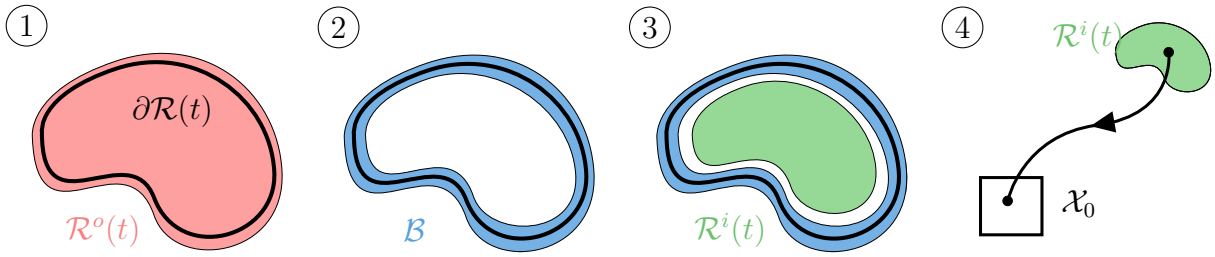

Figure 4.14: Visualization of the procedure applied to calculate inner-approximations of reachable sets.

To compute an inner-approximation $\mathcal{R}^i(t_f)$ of the reachable set at time t_f, we follow the steps visualized in Fig. 4.14:

(1) We first compute an outer-approximation $\mathcal{R}^o(t_f)$ of the reachable set with the conservative polynomialization algorithm in Alg. 6 using SPZs as a set representation.

(2) Next, we compute a set $\mathcal{B}$ that encloses the boundary $\partial\mathcal{R}(t_f)$ of the exact reachable set using the reachable subset approach from Sec. 4.2.

(3) Afterwards, we scale the size of the set $\mathcal{R}^o(t_f)$ to obtain a set $\mathcal{C}$ that does not intersect the set $\mathcal{B}$.

(4) Finally, we simulate the backward flow of the system using the center of $\mathcal{C}$ as a starting point to verify that $\mathcal{C}$ satisfies the conditions from Thm. 4.3.1, which then proves that $\mathcal{C}$ is a valid inner-approximation $\mathcal{R}^i(t_f)$ of the reachable set at time t_f.

This procedure for computing inner-approximations is inspired by [18]. Next, we describe the steps 2, 3, and 4 in detail.

Enclosure of the Boundary

To compute a set which encloses the boundary of the exact reachable set, we require the following well-known theorem from [141, Corollary 1]:

Theorem 4.3.2. *Given a non-empty compact initial set $\mathcal{X}_0 \subset \mathbb{R}^n$, it holds that the reachable set $\mathcal{R}_{\partial\mathcal{X}_0}(t)$ for the boundary of $\mathcal{X}_0$ is identical to the boundary of the reachable set $\mathcal{R}_{\mathcal{X}_0}(t)$ for $\mathcal{X}_0$:*

$$\mathcal{R}_{\partial\mathcal{X}_0}(t) = \partial\mathcal{R}_{\mathcal{X}_0}(t).$$

The theorem holds for arbitrary times $t \in \mathbb{R}_{\geq 0}$.

Based on Thm. 4.3.2, we can efficiently extract an enclosure of the boundary from the outer-approximation of the reachable set using the reachable subset approach in Sec. 4.2:

Proposition 4.3.3. *Given a SPZ $\mathcal{X}_0 = \langle c, G, G_I, E, id\rangle_{PZ} \subset \mathbb{R}^n$, a parameter domain $\mathcal{D} \subseteq [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$ satisfying $\partial\mathcal{X}_0 = \underline{\lfloor \mathcal{X}_0 \rfloor}(\mathcal{D})$, and an outer-approximation of the reachable set $\mathcal{R}^o_{\mathcal{X}_0}(t)$ calculated with the conservative polynomialization algorithm in Alg. 6, it holds that*

$$\partial\mathcal{R}_{\mathcal{X}_0}(t) \subseteq \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t) \rfloor}(\mathcal{D}, id),$$

where $\underline{\lfloor \mathcal{X}_0 \rfloor}(\mathcal{D})$ *denotes the evaluation function as defined in Def. 4.2.10 and* $\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t) \rfloor}(\mathcal{D}, id)$ *denotes the extended evaluation function for SPZs as defined in Def. 4.2.11. The proposition holds for arbitrary times* $t \in \mathbb{R}_{\geq 0}$.

Proof. Since the outer-approximation of the reachable set $\mathcal{R}^o_{\mathcal{X}_0}(t)$ is calculated with the conservative polynomialization algorithm using SPZs, we can apply the reachable subset approach introduced in Sec. 4.2. According to Thm. 4.2.17, it therefore holds that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \mathcal{R}_{\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha)}(t) \subseteq \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t) \rfloor}(\alpha, id). \tag{4.47}$$

With $\partial \mathcal{X}_0 = \underline{\lfloor \mathcal{X}_0 \rfloor}(\mathcal{D})$ we then obtain

$$\partial \mathcal{R}_{\mathcal{X}_0}(t) \overset{\text{Thm. 4.3.2}}{=} \mathcal{R}_{\partial \mathcal{X}_0}(t) = \bigcup_{\alpha \in \mathcal{D}} \mathcal{R}_{\underline{\lfloor \mathcal{X}_0 \rfloor}(\alpha)}(t) \overset{(4.47)}{\subseteq} \bigcup_{\alpha \in \mathcal{D}} \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t) \rfloor}(\alpha, id) = \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t) \rfloor}(\mathcal{D}, id),$$

which concludes the proof. □

Using Prop. 4.3.3, a set $\mathcal{B}$ that encloses the boundary can simply be obtained as $\mathcal{B} = \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\mathcal{D}, id)$. It remains to show how to compute $\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\mathcal{D}, id)$ in detail. For simplicity, we assume that the initial set $\mathcal{X}_0$ is an interval and discuss later how to handle initial sets represented by other set representations. An interval $\mathcal{X}_0 = [l, u] \subset \mathbb{R}^n$ can be equivalently represented by the SPZ

$$\mathcal{X}_0 = \big\langle 0.5(l+u), \mathtt{diag}(0.5(u-l)), [\,], I_n, \underbrace{\mathtt{uniqueID}(n)}_{id} \big\rangle_{PZ}$$

according to Prop. 3.1.10. Consequently, the factor domain $\mathcal{D}$ that corresponds to the boundary $\partial \mathcal{X}_0$ of the initial set is

$$\mathcal{D} = \partial[-\mathbf{1}, \mathbf{1}] = \bigcup_{i=1}^{n} \Big(\underbrace{\{\alpha \in [-\mathbf{1}, \mathbf{1}] \mid \alpha_{(i)} = 1\}}_{\mathcal{F}_{i,1}} \cup \underbrace{\{\alpha \in [-\mathbf{1}, \mathbf{1}] \mid \alpha_{(i)} = -1\}}_{\mathcal{F}_{i,2}} \Big), \tag{4.48}$$

which corresponds to the facets of the factor hypercube $[-\mathbf{1}, \mathbf{1}]$. Inserting (4.48) into the result from Prop. 4.3.3 finally yields

$$\begin{aligned} \mathcal{B} &= \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\mathcal{D}, id) \overset{(4.48)}{=} \bigcup_{i=1}^{n} \Big(\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\mathcal{F}_{i,1}, id) \cup \underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(\mathcal{F}_{i,2}, id) \Big) \\ &= \bigcup_{i=1}^{n} \Big(\underbrace{\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(1, id_{(i)})}_{\mathcal{PZ}_i} \cup \underbrace{\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}(-1, id_{(i)})}_{\mathcal{PZ}_{n+i}} \Big) = \bigcup_{k=1}^{2n} \mathcal{PZ}_k, \end{aligned} \tag{4.49}$$

where the SPZs $\mathcal{PZ}_k$ are computed using the implementation of the extended evaluation function in Prop. 4.2.19. According to (4.49), the boundary enclosure $\mathcal{B}$ can therefore be represented as the union of $2n$ SPZs. For initial sets represented as zonotopes, the factor domain $\mathcal{D}$ belonging to the boundary $\partial \mathcal{X}_0$ can be determined by converting the zonotope to a polytope as described in [24, Thm. 2.1] followed by a selection of faces from the factor hypercube $[-\mathbf{1}, \mathbf{1}]$ that correspond to facets of the polytope. Initial sets represented as polytopes can be converted to SPZs using Alg. 4, where again $\mathcal{D}$ can be determined by selecting the faces of the factor hypercube $[-\mathbf{1}, \mathbf{1}]$ that correspond to facets of the polytope.

Computation of the Inner-Approximation

To find a suitable set $\mathcal{C}$ that satisfies Thm. 4.3.1, we scale the previously computed outer-approximation $\mathcal{R}^o_{\mathcal{X}_0}(t_f) = \langle c, G, G_I, E, id \rangle_{PZ}$ by optimizing the lower bound $\underline{\alpha} \in \mathbb{R}^p$ and upper bound $\overline{\alpha} \in \mathbb{R}^p$ for the domain of dependent factors $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ of the SPZ:

$$\max_{-\mathbf{1} \leq \underline{\alpha} \leq \overline{\alpha} \leq \mathbf{1}} \ \texttt{volume}\big(\llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])\big) \ \text{ s.t. } \ \llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}]) \cap \mathcal{B} = \emptyset, \tag{4.50}$$

where as a heuristic we scale the SPZ $\mathcal{W} = \langle c, G, [\,], E, id \rangle_{PZ}$ obtained by removing the independent generators from $\mathcal{R}^o_{\mathcal{X}_0}(t_f)$ for computational reasons. With the optimized bounds $\underline{\alpha}$ and $\overline{\alpha}$ from (4.50), a suitable set $\mathcal{C}$ can then be calculated as $\mathcal{C} = \llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])$. However, solving the optimization problem in (4.50) exactly is computationally expensive. Instead, we therefore compute a feasible and close to optimal solution using Alg. 7. The tightness of the inner-approximation obtained from Alg. 7 for several numerical examples is demonstrated later in Sec. 4.3.5. Alg. 7 iterates over the $2n$ SPZs $\mathcal{PZ}_k$ from the boundary enclosure $\mathcal{B}$ in (4.49) and adapts in each iteration $\underline{\alpha}, \overline{\alpha}$ such that the intersection between $\llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])$ and $\mathcal{PZ}_k$ is empty as required by (4.50). Next, we explain the single steps of Alg. 7 in detail. According to Lemma 3.1.33, computation of the intersection between the two SPZs $\mathcal{W} = \langle c, G, [\,], E, id \rangle_{PZ}$ and $\mathcal{PZ}_k = \langle \widehat{c}, \widehat{G}, \widehat{G}_I, \widehat{E}, \widehat{id} \rangle_{PZ}$ results in a nonlinear constraint

$$f(y) = c - \widehat{c} + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} y_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} - \sum_{i=1}^{\widehat{h}} \left(\prod_{k=1}^{\widehat{p}} y_{(p+k)}^{\widehat{E}_{(k,i)}} \right) \widehat{G}_{(\cdot,i)} - \sum_{j=1}^{\widehat{q}} y_{(p+\widehat{p}+j)} \widehat{G}_{I(\cdot,j)} = \mathbf{0} \tag{4.51}$$

on the dependent factors $\alpha \in \mathbb{R}^p$ of the SPZ $\mathcal{W}$, where the vector $y = [\alpha \ \widehat{\alpha} \ \widehat{\beta}]^T$ concatenates α with the dependent factors $\widehat{\alpha}$ and the independent factors $\widehat{\beta}$ of the SPZ $\mathcal{PZ}_k$. The values of α that satisfy the constraint $f(y) = \mathbf{0}$ correspond to points that intersect the set $\mathcal{PZ}_k$ which encloses a part of the boundary. For computational reasons, we first compute in Line 5 of Alg. 7 an interval enclosure $\mathcal{I}$ of all values α that satisfy $f(y) = \mathbf{0}$ by applying a contractor as introduced in Sec. 2.8 to the domain $y = [\alpha \ \widehat{\alpha} \ \widehat{\beta}]^T \in [\underline{\alpha}, \overline{\alpha}] \times [-\mathbf{1}, \mathbf{1}] \times [-\mathbf{1}, \mathbf{1}]$. Afterward, we remove the interval $\mathcal{I}$ from the domain $[\underline{\alpha}, \overline{\alpha}]$ in Line 6 of Alg. 7, so that the set $\llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])$ corresponding to the updated factor domain $[\underline{\alpha}, \overline{\alpha}]$ does not intersect the set $\mathcal{PZ}_k$ anymore. Since the set $\mathcal{B}$ enclosing the boundary of the exact reachable set is identical to the union of the sets $\mathcal{PZ}_k$, $k = 1, \ldots, 2n$, it holds that $\llcorner \underline{\mathcal{R}^o_{\mathcal{X}_0}(t_f)} \lrcorner([\underline{\alpha}, \overline{\alpha}])$ does not intersect $\mathcal{B}$ if it does not intersect any of the sets $\mathcal{PZ}_k$:

$$\Big(\llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}]) \cap \underbrace{\mathcal{B}}_{= \bigcup_{k=1}^{2n} \mathcal{PZ}_k} = \emptyset \Big) \Leftrightarrow \Big(\forall k \in \{1, \ldots, 2n\} : \ \llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}]) \cap \mathcal{PZ}_k = \emptyset \Big).$$

Finally, in lines 8-11, we compute the set $\mathcal{C} = \llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])$ by calculating the evaluation function $\llcorner \underline{\mathcal{W}} \lrcorner([\underline{\alpha}, \overline{\alpha}])$ according to (4.39) using the `getSubset` operation for SPZ as specified in Prop. 3.1.43.

Algorithm 7 Compute feasible solution for (4.50)

Require: Set $\mathcal{B} = \bigcup_{k=1}^{2n} \mathcal{PZ}_k$ enclosing the boundary, outer-approximation of the reachable set $\mathcal{R}^o_{\mathcal{X}_0}(t_f) = \langle c, G, G_I, E, id \rangle_{PZ}$.

Ensure: Set $\mathcal{C}$ satisfying Thm. 4.3.1.

1: $\underline{\alpha} \leftarrow -\mathbf{1},\ \overline{\alpha} \leftarrow \mathbf{1}$

2: $\mathcal{W} \leftarrow \langle c, G, [\,], G_I, E, id \rangle_{PZ}$

3: **for** $k \leftarrow 1$ to $2n$ **do**

4: $\quad f(y) = \mathbf{0} \leftarrow$ constraint from $\underline{\mathcal{W}}([\underline{\alpha}, \overline{\alpha}]) \cap \mathcal{PZ}_k$ (see (4.51))

5: $\quad \mathcal{I} \times \widehat{\mathcal{I}} \times \widehat{\mathcal{A}} \leftarrow \texttt{contract}(f(y), [\underline{\alpha}, \overline{\alpha}] \times [-\mathbf{1}, \mathbf{1}] \times [-\mathbf{1}, \mathbf{1}])$ (see Sec. 2.8)

6: $\quad [\underline{\alpha}, \overline{\alpha}] \leftarrow [\underline{\alpha}, \overline{\alpha}] \setminus \mathcal{I}$ (see Prop. 4.3.4)

7: **end for**

8: $\mathcal{C} \leftarrow \mathcal{W}$

9: **for** $i \leftarrow 1$ to p **do**

10: $\quad \mathcal{C} \leftarrow \texttt{getSubset}\big(\mathcal{C}, i, [\underline{\alpha}_{(i)}, \overline{\alpha}_{(i)}]\big)$ (see Prop. 3.1.43)

11: **end for**

It remains to show how we implement the set difference for intervals $[\underline{\alpha}, \overline{\alpha}] \setminus \mathcal{I}$ in Line 6 of Alg. 7:

Proposition 4.3.4. *(Set Difference Interval) Given two intervals $\mathcal{I}_1 = [l_1, u_1] \subset \mathbb{R}^n$ and $\mathcal{I}_2 = [l_2, u_2] \subset \mathbb{R}^n$ with $\mathcal{I}_2 \subseteq \mathcal{I}_1$, an inner-approximation of the set difference can be computed as $\mathcal{I}_1 \setminus \mathcal{I}_2 \supseteq [l, u]$ with*

$$\forall i \in \{1, \ldots, n\}: \quad [l_{(i)}, u_{(i)}] = \begin{cases} [l_{1(i)}, u_{1(i)}], & i \neq i^* \\ [l_{1(i)}, l_{2(i)}), & (i = i^*) \wedge (\Delta l_{(i)} \geq \Delta u_{(i)}) \\ (u_{2(i)}, u_{1(i)}], & \textit{otherwise} \end{cases}, \tag{4.52}$$

where

$$i^* = \underset{i \in \{1,\ldots,n\}}{\arg\max}\ \max\big(\underbrace{l_{2(i)} - l_{1(i)}}_{\Delta l_{(i)}}, \underbrace{u_{1(i)} - u_{2(i)}}_{\Delta u_{(i)}}\big).$$

The half-open intervals $[l_{1(i)}, l_{2(i)})$ and $(u_{2(i)}, u_{1(i)}]$ can be implemented by subtracting and adding a small offset. The computational complexity with respect to the dimension n is $\mathcal{O}(n)$.

Proof. Since the set difference is defined as $\mathcal{I}_1 \setminus \mathcal{I}_2 = \{s \mid s \in \mathcal{I}_1 \wedge s \notin \mathcal{I}_2\}$ according to (2.9), it holds that

$$\Big(\big([l, u] \subseteq \mathcal{I}_1\big) \wedge \big([l, u] \cap \mathcal{I}_2 = \emptyset\big)\Big) \Leftrightarrow \big([l, u] \subseteq \mathcal{I}_1 \setminus \mathcal{I}_2\big).$$

We therefore have to show that $[l, u] \subseteq \mathcal{I}_1$ and $[l, u] \cap \mathcal{I}_2 = \emptyset$. For multi-dimensional intervals we have

$$\big([l, u] \subseteq [l_1, u_1]\big) \Leftrightarrow \big(\forall i \in \{1, \ldots, n\}:\ [l_{(i)}, u_{(i)}] \subseteq [l_{1(i)}, u_{1(i)}]\big) \tag{4.53}$$

and

$$\big([l,u] \cap [l_2,u_2] = \emptyset\big) \Leftrightarrow \big(\exists i \in \{1,\dots,n\} :\ [l_{(i)},u_{(i)}] \cap [l_{2(i)},u_{2(i)}] = \emptyset\big). \tag{4.54}$$

With $[l,u]$ defined as in (4.52), it holds that $[l_{(i)},u_{(i)}] = [l_{1(i)},u_{1(i)}]$ for all dimensions $i \in \{1,\dots,n\} \setminus i^*$. Moreover, for dimension i^* we have $[l_{1(i^*)},l_{2(i^*)}) \subseteq [l_{1(i^*)},u_{1(i^*)}]$ and $(u_{2(i^*)},u_{1(i^*)}] \subseteq [l_{1(i^*)},u_{1(i^*)}]$ since $[l_2,u_2] \subseteq [l_1,u_1]$, so that $[l,u] \subseteq \mathcal{I}_1$ holds according to (4.53). For dimension i^* we furthermore have $[l_{1(i^*)},l_{2(i^*)}) \cap [l_{2(i^*)},u_{2(i^*)}] = \emptyset$ and $(u_{2(i^*)},u_{1(i^*)}] \cap [l_{2(i^*)},u_{2(i^*)}] = \emptyset$, so that $[l,u] \cap \mathcal{I}_2 = \emptyset$ holds according to (4.54).

Complexity: Computation of the vectors $\Delta l = l_2 - l_1$ and $\Delta u = u_1 - u_2$ requires $2n$ subtractions and therefore has complexity $\mathcal{O}(n)$. Moreover, finding the index i^* requires $2n$ comparisons of scalar numbers, which has complexity $\mathcal{O}(n)$. The overall complexity is therefore $\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$. □

We demonstrate the computation of the set difference for intervals by an example:

Example 4.3.5. *For the intervals* $\mathcal{I}_1 = [0,6] \times [0,4]$ *and* $\mathcal{I}_2 = [1,3] \times [2,3]$ *we obtain with Prop. 4.3.4*

$$\mathcal{I}_1 \setminus \mathcal{I}_2 \supseteq \left[\begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 6 \\ 4 \end{bmatrix} \right].$$

The sets $\mathcal{I}_1$, $\mathcal{I}_2$, *and* $\mathcal{I}_1 \setminus \mathcal{I}_2$ *are visualized in Fig. 4.15.*

Verification of Correctness

After computing the set $\mathcal{C}$ using Alg. 7, it remains to verify that $\mathcal{C}$ is a valid inner-approximation of the reachable set. For this, we introduce the time-inverted dynamics

$$\dot{x}(t) = -f(x(t)). \tag{4.55}$$

Using (4.55), we formulate the following theorem:

Theorem 4.3.6. *Let* $\xi(t_f,x_0)$ *denote the solution to* (4.55) *at the final time* t_f *for the initial point* $x(0) = x_0$. *If the SPZ* $\mathcal{C} = \langle c,G,[\,],E,id\rangle_{PZ}$ *computed with Alg. 7 satisfies*

$$\xi(t_f,c) \in \mathcal{X}_0,$$

it holds that $\mathcal{C}$ *is a valid inner-approximation of the reachable set* $\mathcal{C} \subseteq \mathcal{R}_{\mathcal{X}_0}(t_f)$.

Proof. According to Thm. 4.3.1, a set $\mathcal{C}$ is a valid inner-approximation of the reachable set if $\mathcal{C}$ is connected, the intersection $\mathcal{C} \cap \mathcal{B} = \emptyset$ is empty, and the intersection $\mathcal{C} \cap \mathcal{R}_{\mathcal{X}_0}(t_f) \neq \emptyset$ is not empty. Since $\mathcal{C}$ is represented as a SPZ and all SPZs are connected, it holds that $\mathcal{C}$ is connected. Moreover, since Alg. 7 computes a set which corresponds to a feasible solution for the optimization problem in (4.50), $\mathcal{C} \cap \mathcal{B} = \emptyset$ holds. Finally, if condition $\xi(t_f,c) \in \mathcal{X}_0$ is satisfied, it holds that the constant offset c of the SPZ $\mathcal{C}$ is contained in the exact reachable set $c \in \mathcal{R}_{\mathcal{X}_0}(t_f)$, which proves that $\mathcal{C} \cap \mathcal{R}_{\mathcal{X}_0}(t_f) \neq \emptyset$. □

For formal correctness, validated integration methods [142] have to be used for the simulation of the time-inverted dynamics.

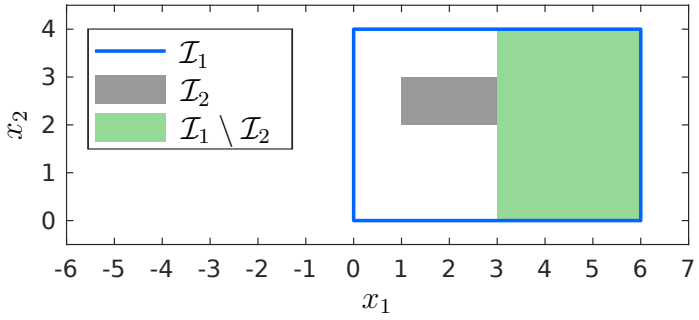

Figure 4.15: Visualization of the set difference calculated with Prop. 4.3.4 for the intervals from Example 4.3.5.

Let us finally demonstrate the computation of an inner-approximation with an example:

Example 4.3.7. *We consider the system*

$$\begin{aligned}\dot{x}_1 &= 0.5x_2 + 5\\ \dot{x}_2 &= x_1\big(0.5 - x_1(0.05 + 0.005x_2)\big) + 5\end{aligned}$$

together with the initial set

$$\mathcal{X}_0 = [-1,1]\times[-1,1] = \left\langle \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, [\,], \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, [1\ 2]\right\rangle_{PZ}$$

and a time horizon of $t_f = 1\text{s}$. *Computation of an outer-approximation for the reachable set with the conservative polynomialization algorithm in Alg. 6 yields the SPZ*

$$\mathcal{R}^o_{\mathcal{X}_0}(t_f) = \left\langle \begin{bmatrix}6.39\\5.60\end{bmatrix}, \begin{bmatrix}1.06 & 0.50 & -0.02 & -0.01\\0.08 & 0.92 & -0.07 & -0.06\end{bmatrix}, \begin{bmatrix}0.05 & 0\\0 & 0.04\end{bmatrix}, \begin{bmatrix}1 & 0 & 2 & 1\\0 & 1 & 0 & 1\end{bmatrix}, [1\ 2]\right\rangle_{PZ}$$

$$\begin{aligned}= \Bigg\{ &\begin{bmatrix}6.39\\5.60\end{bmatrix} + \begin{bmatrix}1.06\\0.08\end{bmatrix}\alpha_1 + \begin{bmatrix}0.50\\0.92\end{bmatrix}\alpha_2 - \begin{bmatrix}0.02\\0.07\end{bmatrix}\alpha_1^2 - \begin{bmatrix}0.01\\0.06\end{bmatrix}\alpha_1\alpha_2\\ &+ \begin{bmatrix}0.05\\0\end{bmatrix}\beta_1 + \begin{bmatrix}0\\0.04\end{bmatrix}\beta_2 \;\Big|\; \alpha_1,\alpha_2,\beta_1,\beta_2 \in [-1,1]\Bigg\}.\end{aligned}$$

According to (4.48), *the factor domain* $\mathcal{D}$ *corresponding to the boundary* $\partial\mathcal{X}_0$ *of the initial set is*

$$\begin{aligned}\mathcal{D} = &\underbrace{\big\{[\alpha_1\ \alpha_2]^T \mid \alpha_1 = 1,\ \alpha_2 \in [-1,1]\big\}}_{\mathcal{F}_{1,1}} \cup \underbrace{\big\{[\alpha_1\ \alpha_2]^T \mid \alpha_1 = -1,\ \alpha_2 \in [-1,1]\big\}}_{\mathcal{F}_{1,2}} \cup\\ &\underbrace{\big\{[\alpha_1\ \alpha_2]^T \mid \alpha_1 \in [-1,1],\ \alpha_2 = 1\big\}}_{\mathcal{F}_{2,1}} \cup \underbrace{\big\{[\alpha_1\ \alpha_2]^T \mid \alpha_1 \in [-1,1],\ \alpha_2 = -1\big\}}_{\mathcal{F}_{2,2}}.\end{aligned}$$

With the domain $\mathcal{D}$ *from above, a set* $\mathcal{B} = \bigcup_{k=1}^{2n} \mathcal{PZ}_k$ *that encloses the boundary* $\partial\mathcal{R}_{\mathcal{X}_0}$ *of the reachable set can be computed with the reachable subset approach. Exemplary, we*

obtain for $\mathcal{PZ}_2$ the SPZ

$$\mathcal{PZ}_2 = \underbrace{\mathcal{R}^o_{\mathcal{X}_0}(t_f)}_{\rfloor}(\mathcal{F}_{2,1}, [1\ 2]) = \underbrace{\mathcal{R}^o_{\mathcal{X}_0}(t_f)}_{\rfloor}(1,2)$$

$$= \left\{ \begin{bmatrix}6.39\\5.60\end{bmatrix} + \begin{bmatrix}1.06\\0.08\end{bmatrix}\alpha_1 + \begin{bmatrix}0.50\\0.92\end{bmatrix}\alpha_2 - \begin{bmatrix}0.02\\0.07\end{bmatrix}\alpha_1^2 - \begin{bmatrix}0.01\\0.06\end{bmatrix}\alpha_1\alpha_2 + \begin{bmatrix}0.05\\0\end{bmatrix}\beta_1 + \begin{bmatrix}0\\0.04\end{bmatrix}\beta_2 \;\middle|\; \alpha_2 = 1,\ \alpha_1, \beta_1, \beta_2 \in [-1,1] \right\}$$

$$= \left\{ \begin{bmatrix}6.89\\6.52\end{bmatrix} + \begin{bmatrix}1.05\\0.02\end{bmatrix}\alpha_1 - \begin{bmatrix}0.02\\0.07\end{bmatrix}\alpha_1^2 + \begin{bmatrix}0.05\\0\end{bmatrix}\beta_1 + \begin{bmatrix}0\\0.04\end{bmatrix}\beta_2 \;\middle|\; \alpha_1, \beta_1, \beta_2 \in [-1,1] \right\}.$$

After computing the set $\mathcal{B}$ enclosing the boundary, we apply Alg. 7 to calculate a set $\mathcal{C}$ that does not intersect $\mathcal{B}$. Execution of Alg. 7 using the parallel linearization approach for contraction (see Sec. 2.8) yields the following values for the four iterations of the loop in lines 3-7 of Alg. 7:

$$\begin{aligned}
&\textit{Iteration 1:} && \mathcal{I} = \left[\begin{bmatrix}0.84\\-1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix} \right], && [\underline{\alpha}, \overline{\alpha}] = \left[\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}0.84\\1\end{bmatrix} \right] \\
&\textit{Iteration 2:} && \mathcal{I} = \left[\begin{bmatrix}-1\\0.85\end{bmatrix}, \begin{bmatrix}0.84\\1\end{bmatrix} \right], && [\underline{\alpha}, \overline{\alpha}] = \left[\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}0.84\\0.85\end{bmatrix} \right] \\
&\textit{Iteration 3:} && \mathcal{I} = \left[\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-0.84\\0.85\end{bmatrix} \right], && [\underline{\alpha}, \overline{\alpha}] = \left[\begin{bmatrix}-0.84\\-1\end{bmatrix}, \begin{bmatrix}0.84\\0.85\end{bmatrix} \right] \\
&\textit{Iteration 4:} && \mathcal{I} = \left[\begin{bmatrix}-0.84\\-1\end{bmatrix}, \begin{bmatrix}0.84\\-0.84\end{bmatrix} \right], && [\underline{\alpha}, \overline{\alpha}] = \left[\begin{bmatrix}-0.84\\-0.84\end{bmatrix}, \begin{bmatrix}0.84\\0.85\end{bmatrix} \right],
\end{aligned}$$

which results in the set

$$\mathcal{C} = \texttt{getSubset}\Big(\texttt{getSubset}\big(\mathcal{W}, 1, [-0.84, 0.84]\big), 2, [-0.84, 0.85]\Big)$$

$$= \left\{ \begin{bmatrix}6.39\\5.60\end{bmatrix} + \begin{bmatrix}1.06\\0.08\end{bmatrix}\alpha_1 + \begin{bmatrix}0.50\\0.92\end{bmatrix}\alpha_2 - \begin{bmatrix}0.02\\0.07\end{bmatrix}\alpha_1^2 - \begin{bmatrix}0.01\\0.06\end{bmatrix}\alpha_1\alpha_2 \;\middle|\; \alpha_1 \in [-0.84, 0.84],\ \alpha_2 \in [-0.84, 0.85] \right\}$$

$$= \left\{ \begin{bmatrix}6.40\\5.60\end{bmatrix} + \begin{bmatrix}0.89\\0.07\end{bmatrix}\widehat{\alpha}_1 + \begin{bmatrix}0.42\\0.78\end{bmatrix}\widehat{\alpha}_2 - \begin{bmatrix}0.01\\0.05\end{bmatrix}\widehat{\alpha}_1^2 - \begin{bmatrix}0.01\\0.04\end{bmatrix}\widehat{\alpha}_1\widehat{\alpha}_2 \;\middle|\; \widehat{\alpha}_1, \widehat{\alpha}_2 \in [-1,1] \right\}.$$

Finally, verification with simulation of the time-inverted dynamics according to Thm. 4.3.6 proves that $\mathcal{R}^i_{\mathcal{X}_0}(t_f) = \mathcal{C}$ is a valid inner-approximation of the reachable set. The resulting sets from this example are visualized in Fig. 4.16.

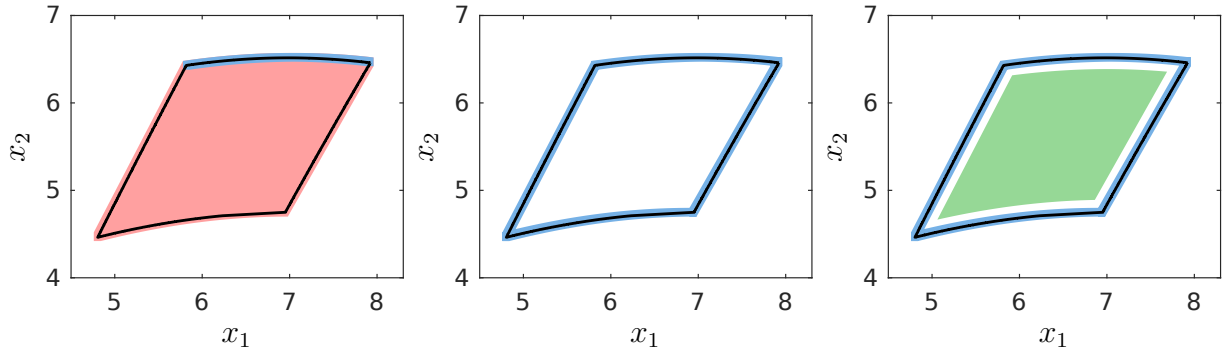

Figure 4.16: Visualization of the results from Example 4.3.7, where the outer-approximation $\mathcal{R}^o_{\mathcal{X}_0}(t_f)$ (red), the boundary of the exact reachable set $\partial\mathcal{R}^o_{\mathcal{X}_0}(t_f)$ (black), and the set $\mathcal{PZ}_2$ (blue) are shown on the left, the boundary enclosure $\mathcal{B}$ (blue) is depicted in the middle, and the resulting inner-approximation $\mathcal{R}^i_{\mathcal{X}_0}(t_f) = \mathcal{C}$ (green) is shown on the right.

4.3.3 Extension to Uncertain Inputs

We now extend the approach presented in Sec. 4.3.2 to nonlinear systems with uncertain inputs $\dot{x}(t) = f(x(t), u(t))$ as defined in Def. 2.3.2. The extension to systems with inputs is based on the following theorem:

Theorem 4.3.8. *Given a nonlinear system $\dot{x}(t) = f(x(t), u(t))$, an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$, and a set of uncertain inputs $\mathcal{U} \subset \mathbb{R}^m$, the reachable set due to constant inputs $\mathcal{R}_{\mathcal{X}_0,const}(t)$ is a subset of the reachable set due to time-varying inputs $\mathcal{R}_{\mathcal{X}_0}(t)$:*

$$\begin{aligned}\mathcal{R}_{\mathcal{X}_0,const}(t) &:= \left\{\xi(t, x_0, u(\cdot)) \;\middle|\; x_0 \in \mathcal{X}_0,\ u(0) \in \mathcal{U},\ \partial u(t)/\partial t = 0\right\} \\ &\subseteq \left\{\xi(t, x_0, u(\cdot)) \;\middle|\; x_0 \in \mathcal{X}_0,\ \forall \tau \in [0, t]:\ u(\tau) \in \mathcal{U}\right\} = \mathcal{R}_{\mathcal{X}_0}(t),\end{aligned}$$

where $\xi(t, x_0, u(\cdot))$ denotes the solution to $\dot{x}(t) = f(x(t), u(t))$ for an initial state $x(0) = x_0$ and the input trajectory $u(\cdot)$. The theorem holds for all times $t \in \mathbb{R}_{\geq 0}$.

Proof. Since time-varying inputs include constant inputs, the reachable set due to time-varying inputs contains the reachable set due to constant inputs. □

According to Thm. 4.3.8, we can compute an inner-approximation $\mathcal{R}^i_{\mathcal{X}_0}(t_f)$ of the reachable set due to time-varying inputs by calculating an inner-approximation of the reachable set due to constant inputs $\mathcal{R}^i_{\mathcal{X}_0}(t_f) \subseteq \mathcal{R}_{\mathcal{X}_0,const}(t_f) \subseteq \mathcal{R}_{\mathcal{X}_0}(t_f)$. With the extended system dynamics

$$\begin{bmatrix}\dot{x}(t) \\ \dot{u}(t)\end{bmatrix} = \begin{bmatrix}f(x(t), u(t)) \\ \mathbf{0}\end{bmatrix}, \tag{4.56}$$

any nonlinear system $\dot{x}(t) = f(x(t), u(t))$ with constant inputs can be equivalently represented as an autonomous system without inputs. Consequently, we first calculate an inner-approximation of the reachable set $\mathcal{R}^i_{\widehat{\mathcal{X}}_0}(t_f)$ for the autonomous system in (4.56) starting from the initial set $\widehat{\mathcal{X}}_0 = \mathcal{X}_0 \times \mathcal{U}$ using the approach presented in Sec. 4.3.2. An inner-approximation of the reachable set due to time-varying inputs $\mathcal{R}^i_{\mathcal{X}_0}(t_f)$ is then obtained by projecting $\mathcal{R}^i_{\widehat{\mathcal{X}}_0}(t_f)$ to the original state space:

$$\mathcal{R}^i_{\mathcal{X}_0}(t_f) = [I_n\ \mathbf{0}] \otimes \mathcal{R}^i_{\widehat{\mathcal{X}}_0}(t_f),$$

where we compute the linear map using Prop. 3.1.18. Especially for long time horizons t_f, the inner-approximation of the reachable set due to time-varying inputs with the reachable set due to constant inputs might become quite inaccurate. One simple extension to improve the accuracy is to instead use the reachable set due to piecewise constant inputs as an inner-approximation. It seems that a straightforward approach to realize this would be to successively apply the method described above for all time intervals with constant inputs. However, the final reachable set after the first time interval, which we would use as the initial set for the second time interval, is a SPZ. Since for SPZs it is yet unclear how to compute the boundary, it would therefore not be possible to apply the approach from Sec. 4.3.2 for the remaining time intervals. To solve this issue, we instead realize piecewise constant inputs by using a different extended system dynamics for each time interval with constant inputs

$$\forall t \in [(i-1) \cdot t_f/M, i \cdot t_f/M] : \quad \begin{bmatrix} \dot{x}(t) \\ \dot{u}_1(t) \\ \vdots \\ \dot{u}_M(t) \end{bmatrix} = \begin{bmatrix} f(x(t), u_i(t)) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

together with the initial set $\widehat{\mathcal{X}}_0 = \mathcal{X}_0 \times \mathcal{U} \times \ldots \times \mathcal{U}$, where $M \in \mathbb{N}$ is the number of time intervals with constant inputs. Since in this case the extended initial set $\widehat{\mathcal{X}}_0$ already contains the inputs $u_1(t), \ldots, u_M(t) \in \mathbb{R}^m$ for all time intervals, we can simply apply the approach in Sec. 4.3.2 to compute an inner-approximation of the reachable set. The extension to uncertain inputs presented here can equivalently be used to compute inner-approximations of reachable sets for nonlinear systems with uncertain parameter.

4.3.4 Computational Complexity

We now derive the computational complexity of our novel approach for calculating inner-approximations with respect to the system dimension n. Computation of an outer-approximation of the reachable set using the conservative polynomialization algorithm in Alg. 6 using SPZs has complexity $\mathcal{O}(n^5)$ according to Sec. 4.1.4. For the remainder of this section, let p, h, and q denote the number of dependent factors, the number of dependent generators, and the number of independent generators of the SPZ representing the outer-approximation.

Calculating an enclosure $\mathcal{B}$ of the boundary using the reachable subset approach requires according to (4.49) the calculation of $2n$ extended evaluation functions $\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}\big(1, id_{(i)}\big)$ and $\underline{\lfloor \mathcal{R}^o_{\mathcal{X}_0}(t_f) \rfloor}\big(-1, id_{(i)}\big)$ using Prop. 4.2.19. Computation of each extended evaluation function has complexity $\mathcal{O}(nw + n^2 \log(n))$ according to Prop. 4.2.19, where w is the length of the parameter vector. In our case the parameter vectors 1 and -1 are both scalars, so that $w = 1$. Consequently, the calculation of $\mathcal{B}$ has complexity $2n \cdot \mathcal{O}(n^2 \log(n)) = \mathcal{O}(n^3 \log(n))$.

Next, we consider the calculation of an inner-approximation $\mathcal{C}$ using Alg. 7. The first for-loop in lines 3-7 of Alg. 7 has complexity $2n \cdot (\mathcal{O}(\texttt{contract}) + \mathcal{O}(p))$ since the loop consists of $2n$ iterations, the contraction in Line 5 has complexity $\mathcal{O}(\texttt{contract})$, and the set difference for the p-dimensional intervals in Line 6 has complexity $\mathcal{O}(p)$ according to

Table 4.5: Computational complexity for calculating inner-approximations of reachable sets with respect to the dimension $n \in \mathbb{N}$ for the different contractors presented in Sec. 2.8.

Forward-backward	**Extremal functions**	**Parallel linearization**
$\mathcal{O}(n^5)$	$\mathcal{O}(n^5)$	$\mathcal{O}(n^{5.5})$

Prop. 4.3.4. The complexity for different contractors is specified in Tab. 2.5, where in our case the n constraints $f(y) = \mathbf{0}$ defined by the function $f(y)$ in (4.51) used for the contraction have $p + \widehat{p} + \widehat{q}$ variables and each subfunction $f_{(i)}(y)$ consists of $e = 2ph + h + 2\widehat{p}\widehat{h} + \widehat{h} + 2\widehat{q} + 1$ elementary operations, with $\widehat{p} \leq p$, $\widehat{h} \leq h$, and $\widehat{q} \leq q$ denoting the number of dependent factors, the number of dependent generators, and the number of independent generators of the SPZs $\mathcal{PZ}_k$ obtained from $\mathcal{R}^o_{\mathcal{X}_0}(t_f)$ using the reachable subset approach. The second for-loop in lines 9-11 of Alg. 7 consists of p iterations, where in each iteration the `getSubset` operation for SPZs is executed once. Since `getSubset` has complexity $\mathcal{O}(n^3 \log(n))$ according to Prop. 3.1.43, the overall complexity of the for-loop is $p \cdot \mathcal{O}(n^3 \log(n)) = \mathcal{O}(pn^3 \log(n))$.

Finally, for verification of correctness we have to perform one validated simulation. Since all reachability algorithms can equivalently be used for validated simulation, we could again use the conservative polynomialization algorithm for this, so that verification using validated simulation does not increase the overall computational complexity of our approach. Specialized validated simulation techniques, however, are usually computationally much more efficient.

Summarizing the computational complexities for the single parts yields an overall complexity of

$$\underbrace{\mathcal{O}(n^5)}_{\mathcal{R}^o_{\mathcal{X}_0}(t_f)} + \underbrace{\mathcal{O}(n^3 \log(n))}_{\mathcal{B}} + \underbrace{2n \cdot \mathcal{O}(\texttt{contract}) + \mathcal{O}(np) + \mathcal{O}(pn^3 \log(n))}_{\text{Alg. 7}},$$

which is $\mathcal{O}(n^5) + n \cdot \mathcal{O}(\texttt{contract})$ using Assumption 3.1.3. The overall complexities for different contractors are listed in Tab. 4.5. The extension to uncertain inputs described in Sec. 4.3.3 increases the dimension of the system to $n + m$, which according to Assumption 4.1.3 does not increase the computational complexity. Our approach for computing inner-approximations consequently only has polynomial complexity with respect to the system dimension and is therefore well suited for high-dimensional systems.

4.3.5 Numerical Examples

We now demonstrate the performance of our novel method for computing inner-approximations of reachable sets on several benchmark systems. To evaluate the precision of the computed inner-approximation, we use the minimum width ratio γ_{min} from [116, Sec. VI] defined as

$$\gamma_{min} = \min_{v \in \mathcal{V}} \frac{|\gamma_i(v)|}{|\gamma_o(v)|}$$

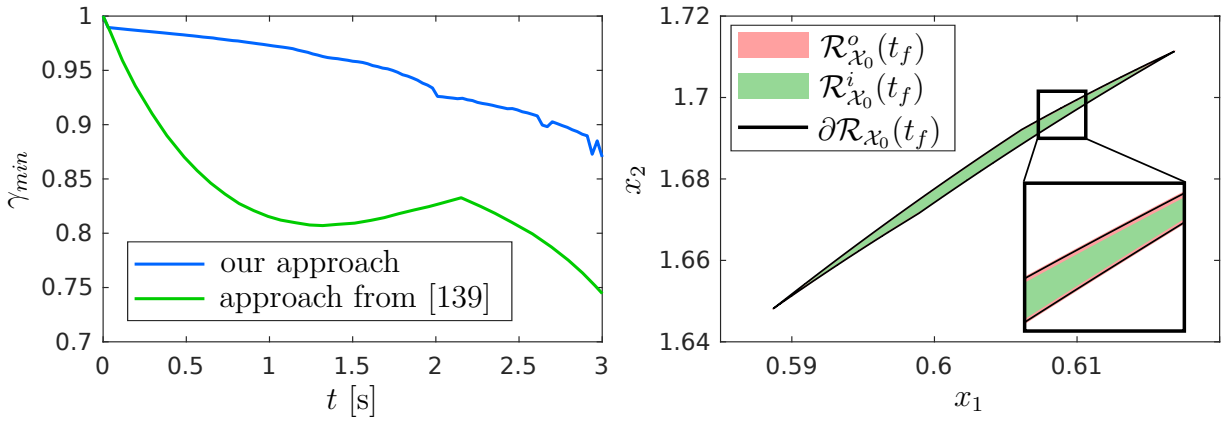

Figure 4.17: The precision of the computed inner-approximations for the Brusselator system in (4.57) over time is shown on the left, where the results for the method in [139] are taken from [139, Fig. 2]. The resulting inner-approximation and outer-approximation of the reachable set at the final time $t_f = 3$s computed with our approach are visualized on the right.

with

$$\gamma_i(v) = \max_{x \in \mathcal{R}^i_{\mathcal{X}_0}(t_f)} v^T x + \max_{x \in \mathcal{R}^i_{\mathcal{X}_0}(t_f)} -v^T x, \quad \gamma_o(v) = \max_{x \in \mathcal{R}^o_{\mathcal{X}_0}(t_f)} v^T x + \max_{x \in \mathcal{R}^o_{\mathcal{X}_0}(t_f)} -v^T x,$$

where we select the n axis-aligned unit-vectors as the set of vectors $\mathcal{V} \subset \mathbb{R}^n$ (see [116, Sec. VI]). For a ratio of $\gamma_{min} = 1$, the inner-approximation along the vectors $v \in \mathcal{V}$ is identical to the outer-approximation, whereas for a ratio of $\gamma_{min} = 0$, the inner-approximation is empty. Since γ_{min} cannot be computed exactly for general SPZs, we compute a tight under-approximation instead. Moreover, for all benchmarks we apply the nonlinear programming approach in (2.23) in combination with the parallel linearization contractor described in Sec. 2.8 for contraction.

Comparison with other Approaches

We first compare our novel approach with the method in [139], which computes inner-approximations for the projection of the reachable set onto the coordinate axes. For the comparison, we use the Brusselator system

$$\begin{aligned} \dot{x}_1 &= 1 + x_1^2 x_2 - 2.5 x_1 \\ \dot{x}_2 &= 1.5 x_1 - x_1^2 x_2 \end{aligned} \tag{4.57}$$

from [143, Example 3.4.1] together with the initial set $\mathcal{X}_0 = [0.9, 1] \times [0, 0.1]$. The precision of the computed inner-approximation expressed by the minimum width ratio γ_{min} over time is shown in Fig. 4.17. It is clearly visible that the results for our approach are much tighter, even though we compute a full inner-approximation and not just an inner-approximation of the projection. Fig. 4.17 additionally visualizes the inner-approximation and outer-approximation from our approach at the final time to provide an impression of the achieved accuracy.

Next, we compare our approach with the method in [116], which computes non-convex inner-approximations represented as polynomial level sets. For the comparison we use the

Table 4.6: Dimension n, time horizon t_f, and reference to the dynamic equations for the benchmarks used for the comparison in Tab. 4.7.

Benchmark	Dimension	Time Horizon	Reference
Brusselator	2	3	[143, Example 3.4.1]
Jet engine	2	4	[143, Example 3.3.9]
Rössler	3	1.5	[143, Example 3.4.3]
Lotka-Volterra	4	1	[143, Example 5.2.3]
Biological system	7	0.2	[143, Example 5.2.4]

Table 4.7: Computation time in seconds and precision γ_{min} of the calculated inner-approximations for the benchmarks in Tab. 4.6, where we compare our approach with the method in [116]. The results for the method in [116] are taken from [116, Tab. 1], where the computation times for the method in [116] are measured on the machine used by the authors of [116].

Benchmark	Our Approach		Approach in [116]	
	time	γ_{min}	time	γ_{min}
Brusselator	64	**0.87**	**55**	0.7
Jet engine	**48**	**0.82**	56	0.8
Rössler	**32**	**0.73**	165	0.5
Lotka-Volterra	**238**	0.32	297	**0.4**
Biological system	**82**	**0.89**	632	0.25

benchmarks from [116, Sec. VI], which are listed in Tab. 4.6. For our approach we focused on initial sets given as intervals, whereas in [116] initial sets are simplices or ellipsoids. We therefore use interval enclosures of the simplices given in [116, Sec. VI] as initial sets and compare our approach with the results from the method in [116] for initial sets given as simplices (see [116, Tab. 1]). The results in Tab. 4.7 demonstrate that for most benchmarks our novel approach is both faster and more precise than the method in [116], even though we use larger initial sets. Especially for high-dimensional systems, our approach exhibits superior performance.

Uncertain Inputs

So far we only considered examples of autonomous systems. To demonstrate the performance of our approach for systems with inputs, we extend the electromechanical oscillator from [18, Example 3] with an uncertain input $u \in \mathbb{R}$ bounded by the set $\mathcal{U} = [-0.2, 0.2]$:

$$\begin{aligned} \dot{x}_1 &= x_2 + (x_2 - 2.8)u \\ \dot{x}_2 &= 0.2 - 0.7\sin(x_1) - 0.05x_2. \end{aligned} \tag{4.58}$$

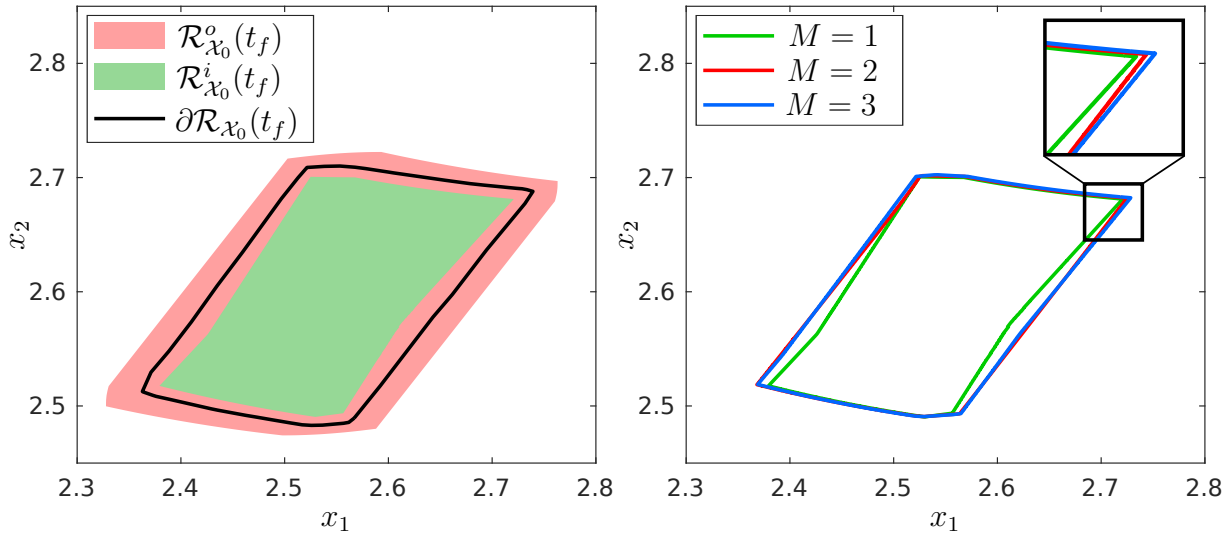

Figure 4.18: Inner-approximation of the final reachable set for the electromechanical oscillator in (4.58) calculated using constant inputs (left) and using piecewise constant inputs with different numbers of constant segments M (right).

Table 4.8: Computation time in seconds and precision γ_{min} of the inner-approximation $\mathcal{R}^i_{\mathcal{X}_0}(t_f)$ for the electromechanical oscillator in (4.58) calculated using piecewise constant inputs with different numbers of constant segments M.

Number of Segments	**Computation Time**	**Precision $\gamma_{\min}$**
$M = 1$	46	0.78
$M = 2$	127	0.82
$M = 3$	311	0.83

Moreover, we consider the initial set $\mathcal{X}_0 = [-0.1, 0.1] \times [2.9, 3.1]$ and the time horizon $t_f = 0.9$s. We first compute an inner-approximation of the reachable set with the method described in Sec. 4.3.3 using constant inputs. The results are visualized on the left side of Fig. 4.18. Next, we use piecewise constant inputs to compute an inner-approximation, where we investigate the effect of changing the number constant segments M. The results shown in Fig. 4.18 and in Tab. 4.8 clearly demonstrate that increasing M improves the precision of the computed inner-approximation. However, increasing M also prolongs the computation time as shown in Tab. 4.8.

4.4 Reachability Analysis for Hybrid Systems with Nonlinear Guard Sets

So far we considered reachability analysis for systems with purely continuous dynamics only. In the real world, however, most cyber-physical systems exhibit mixed continuous and discrete dynamics due to the interplay between physics and digital control. Such hybrid systems can be modeled by hybrid automata. In this section[5], we introduce a novel method for computing a tight outer-approximation of the reachable set for the very general case of hybrid systems with nonlinear guard sets. Again we use SPZs to represent reachable sets since they are well suited for handling both, the continuous dynamics and the discrete transitions of a hybrid automaton.

The structure of this section is as follows: We first summarize the current state of the art for reachability analysis of hybrid systems in Sec. 4.4.1. Next, in Sec. 4.4.2, we explain the basic procedure that we apply to compute outer-approximations of reachable sets for hybrid systems. In the main part in Sec. 4.4.3 we then present our novel approach for handling the discrete transitions in hybrid system reachability analysis. Finally, the computational complexity is derived in Sec. 4.4.5 and we demonstrate the performance of our novel approach on several benchmark systems in Sec. 4.4.6.

4.4.1 State of the Art

Reachability analysis for hybrid systems consists of two parts: 1) Computing the reachable set for the continuous dynamics and 2) handling the discrete transitions of the hybrid automaton, which requires to compute intersections between the reachable set and the guard sets. As demonstrated in Sec. 4.1.1, there exist many sophisticated approaches for calculating outer-approximations of reachable sets for continuous systems, so that the main challenge in reachability analysis for hybrid systems is to calculate the intersections between the reachable set and the guard sets.

For guards sets represented by polyhedra or hyperplanes, several methods for intersection computation have been developed. A straightforward approach is to compute the intersection between the reachable set and the guard sets geometrically: The method in [7] calculates the intersection between reachable sets represented by support functions and the guard sets by solving several convex minimization problems. To avoid an explosion in computation time, the sets resulting from partial intersections are often unified by convex hulls [7, 145]. Since the computation of convex hulls is computationally demanding, many approaches unify partial intersections by simpler sets or completely avoid the unification: In [31], the union of the partial intersections is enclosed by bundles of parallelotopes. The work in [146] shows how the intersection between multiple zonotopes and a hyperplane can be efficiently enclosed by a template polyhedron. The tool HyLAA [50] reduces the over-approximation resulting from the unification by applying a backtracking scheme that splits previously computed reachable sets. For guard sets represented by hyperplanes one can apply the method in [113], which avoids the need for unification completely by scaling the system dynamics in such a way that only the reachable set for one time interval intersects

[5] This section is based on [144].

the guard set. For high-dimensional systems not only unification, but also the geometric computation of intersections is computationally expensive. The technique in [147] avoids both, unification and geometric intersection computation, by directly mapping the reachable set onto the hyperplane that represents the guard set. Hybrid systems with guards sets represented by polyhedra or hyperplanes are also supported by many reachability tools: Geometric intersection computation is applied by the tools Flow* [44], HyDRA [148], HyLAA [50], JuliaReach [110], and SpaceEx [145]. Moreover, CORA [1] implements the methods [31], [146], and [147].

Currently, only few approaches exist for a more general class of hybrid systems which model guard sets as nonlinear level sets. For some simple cases, nonlinear guard sets can be enclosed by multiple polytopes, which makes it possible to apply the methods in [7] and [31]. The approach in [43] uses the constraints imposed by the guard intersection to contract the set of initial states, which then yields a Taylor model that encloses the intersection with the guard set. Another strategy is to determine the time interval in which the reachable set intersects the guard set followed by using the whole reachable set for the time interval as an over-approximation of the intersection with the guard. While this technique works well for validated integration methods that enclose only a single trajectory rather than a set of trajectories [142], it is often too conservative for reachability analysis. To compute tight enclosures of reachable sets, the approach in [39] uses a technique similar to [142], but additionally creates partitions in time until a user-defined precision is achieved; however, propagating the reachable sets for all partitions is computationally expensive. For this reason, [149] improves the approach in [39] by unifying the resulting sets from all partitions with an enclosing interval. Since interval enclosures result in large over-approximation errors, the approach in [114] unifies parallel sets with an enclosing zonotope instead. However, since [114] requires the computation of zonotope vertices, the approach has exponential complexity with respect to the system dimension. A tool that explicitly supports nonlinear guard sets is Ariadne [106], which uses the method in [150] that is based on the determination of hitting times to calculate the intersections with guard sets.

4.4.2 Reachability Analysis for Hybrid Systems

We now describe the basic procedure that we apply to compute tight outer-approximations for reachable sets of hybrid systems modeled by hybrid automata as defined in Def. 2.3.3. In particular, we consider the very general case of hybrid automata with nonlinear continuous dynamics $\dot{x}(t) = f_k(x(t), u(t))$, $k = 1, \ldots, p$, nonlinear reset functions $r_i(x)$, $i = 1, \ldots, q$, invariant sets $\mathcal{Y}_k$, $k = 1, \ldots, p$ represented by nonlinear level sets $\mathcal{Y}_k = \langle y_k(x), \leq \rangle_{LS} = \{x \mid y_k(x) \leq \mathbf{0}\}$, and guard sets $\mathcal{G}_i$, $i = 1, \ldots, q$ represented by degenerate nonlinear level sets $\mathcal{G}_i = \langle g_i(x), = \rangle_{LS} = \{x \mid g_i(x) = 0\}$, where $f_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $r_i : \mathbb{R}^n \to \mathbb{R}^n$, $y_k : \mathbb{R}^n \to \mathbb{R}^{o_k}$, and $g_i : \mathbb{R}^n \to \mathbb{R}$ are Lipschitz continuous functions. To compute an outer-approximation $\mathcal{R}^o([0, t_f])$ of the reachable set for the time horizon $t \in [0, t_f]$, we apply the procedure visualized in Fig. 4.19:

(1) Given an initial discrete state $\sigma_0 \in \{1, \ldots, p\}$ and an initial set $\mathcal{X}_0 \subseteq \mathcal{Y}_{\sigma_0}$, we first compute an outer-approximation of the reachable set for the continuous dynamics $\dot{x}(t) = f_{\sigma_0}(x(t), u(t))$ with the conservative polynomialization algorithm in Alg. 6

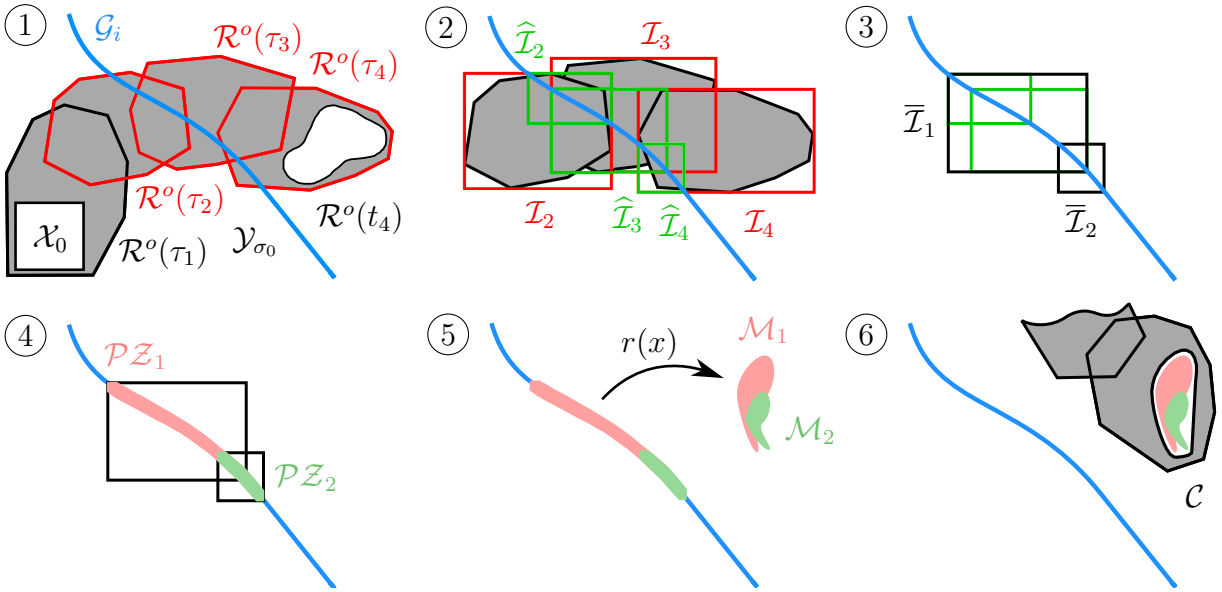

Figure 4.19: Visualization of the procedure applied to calculate outer-approximations of reachable sets for hybrid systems.

using SPZs until the reachable set is completely located outside the current invariant set $\mathcal{Y}_{\sigma_0}$ or the final time t_f is reached. In addition, we determine the time intervals τ_j for which the corresponding reachable set $\mathcal{R}^o(\tau_j)$ intersects the guard set $\mathcal{G}_i$ of a transition $\mathscr{T}_i = \langle \mathcal{G}_i, r_i(x), \sigma_0, d_i \rangle_T$. Our approach also works when only the reachable set within an invariant is propagated, which in some cases significantly reduces the conservatism, but is computationally more expensive.

(2) To obtain a rough over-approximation of the guard intersection using computationally cheap methods, we first enclose each reachable set intersecting the guard set with an interval $\mathcal{I}$. Afterward, we contract the resulting intervals so that they tightly enclose the intersection with the guard set.

(3) Next, to avoid the computationally expensive parallel propagation of reachable sets, we enclose the union of the contracted intervals $\widehat{\mathcal{I}}$ by a single interval $\overline{\mathcal{I}}$. We introduce an upper bound μ for the number of intervals that are unified to reduce the conservatism. For the example shown in Fig. 4.19, a value of $\mu = 2$ is used.

(4) We tightly enclose the intersection of the guard set with the previously-obtained interval $\overline{\mathcal{I}}$ by a SPZ $\mathcal{PZ}$.

(5) Afterward, we apply the reset function $r_i(x)$ to the previously obtained SPZ $\mathcal{PZ}$.

(6) Due to the upper bound μ, we might obtain parallel sets, which we unify by a single set $\mathcal{C}$ to avoid propagating several sets in parallel. The reason for using the upper bound μ in Step 3 followed by the unification of parallel sets in Step 6 is that the unification in Step 6 significantly increases the representation size of the resulting set if many parallel sets are unified. With the early partial unification in Step 3, we

avoid this issue. Finally, we repeat the described procedure for the target mode d_i of the transition $\mathscr{T}_i$ using $\mathcal{C}$ as the new initial set.

For the case that the reachable set intersects several guard sets, we compute the intersection separately for each guard set using the presented approach. If the computation of the contracted intervals in Step 2 is adapted appropriately, this does not lead to large over-approximations. While we here use the conservative polynomialization algorithm to compute reachable sets for the continuous dynamics of the single modes, the above procedure for hybrid system reachability analysis can be combined with arbitrary other reachability algorithms for continuous systems due to its modular design.

4.4.3 Discrete Transitions

We now describe the single steps of the procedure in Sec. 4.4.2 in detail. Let us begin with the detection of intersections between the reachable set and the invariant or guard sets as required in Step 1.

Intersection Detection

Our approach for detecting intersections between the reachable set and the invariant set is based on the following proposition:

Proposition 4.4.1. *Given a set $\mathcal{S} \subset \mathbb{R}^n$ and a level set $\mathcal{LS} = \langle w(x), \leq \rangle_{LS} \subset \mathbb{R}^n$ with $w : \mathbb{R}^n \to \mathbb{R}^o$, it holds that*

$$\big(\exists i \in \{1, \ldots, o\} : \; l_{(i)} > 0\big) \Rightarrow \big(\mathcal{S} \cap \mathcal{LS} = \emptyset\big),$$

where the lower bound $l \in \mathbb{R}^o$ and the upper bound $u \in \mathbb{R}^o$ are determined by range bounding

$$[l, u] = \texttt{bound}(w(x), \mathcal{S})$$

as defined in Def. 2.7.1.

Proof. According to the definition of the range bounding operation in Def. 2.7.1 we have

$$\big(l_{(i)} > 0\big) \overset{\text{Def. 2.7.1}}{\Leftrightarrow} \Big(\min_{x \in \mathcal{S}} \; w_{(i)}(x) > 0\Big) \Leftrightarrow \big(\forall x \in \mathcal{S} : \; w_{(i)}(x) > 0\big),$$

so that the intersection

$$\mathcal{LS} \cap \mathcal{S} \overset{\text{Def. 2.2.9}}{=} \big\{x \in \mathcal{S} \;\big|\; w_{(1)}(x) \leq 0 \wedge \ldots \wedge \underbrace{w_{(i)}(x) \leq 0}_{\forall x \in \mathcal{S}: \; w_{(i)}(x) > 0} \wedge \ldots \wedge w_{(o)}(x) \leq 0\big\} = \emptyset$$

results in the empty set. □

Similarly, our method for detecting intersections between the reachable set and the guard sets is based on the following result:

Proposition 4.4.2. *Given a set $\mathcal{S} \subset \mathbb{R}^n$ and a level set $\mathcal{LS} = \langle w(x), = \rangle_{LS} \subset \mathbb{R}^n$ with $w : \mathbb{R}^n \to \mathbb{R}$, it holds that*

$$\big(\mathcal{S} \cap \mathcal{LS} \neq \emptyset\big) \Rightarrow \big(l \leq 0 \leq u\big),$$

where the lower bound $l \in \mathbb{R}$ and the upper bound $u \in \mathbb{R}$ are determined by range bounding

$$[l, u] = \texttt{bound}(w(x), \mathcal{S})$$

as defined in Def. 2.7.1.

Proof. Since the intersection between $\mathcal{LS}$ and $\mathcal{S}$ is

$$\mathcal{LS} \cap \mathcal{S} \overset{\text{Def. 2.2.9}}{=} \big\{x \in \mathcal{S} \;\big|\; w(x) = 0\big\}, \tag{4.59}$$

we have

$$\big(\mathcal{S} \cap \mathcal{LS} \neq \emptyset\big) \overset{(4.59)}{\Rightarrow} \big(\exists x \in \mathcal{S} : \; w(x) = 0\big) \Rightarrow \left(\min_{x \in \mathcal{S}} \; w(x) \leq 0 \leq \max_{x \in \mathcal{S}} \; w(x)\right) \Rightarrow \big(l \leq 0 \leq u\big),$$

where we exploited that $[\min_{x \in \mathcal{S}} w(x), \max_{x \in \mathcal{S}} w(x)] \subseteq [l, u]$ holds according to the definition of range bounding in Def. 2.7.1. □

To check if the reachable set $\mathcal{R}^o(t_j)$ at time t_j is located outside of the current invariant set $\mathcal{Y}_{\sigma_0} = \langle y_{\sigma_0}(x), \leq \rangle_{LS}$, we first apply range bounding to compute

$$[l, u] = \texttt{bound}(y_{\sigma_0}(x), \mathcal{R}^o(t_j)),$$

and then use Prop. 4.4.1 to test if $\mathcal{R}^o(t_j) \cap \mathcal{Y}_{\sigma_0} = \emptyset$. Moreover, to determine the time intervals τ_j in which the reachable set $\mathcal{R}^o(\tau_j)$ intersects the guard set $\mathcal{G}_i = \langle g_i(x), = \rangle_{LS}$, we iterate over all time intervals τ_j and use range bounding to compute

$$[l, u] = \texttt{bound}\big(g_i(x), \mathcal{R}^o(\tau_j)\big)$$

in order to test if $\mathcal{R}^o(\tau_j) \cap \mathcal{G}_i \neq \emptyset$ using Prop. 4.4.2. The conservatism of intersection detection solely depends on the range bounding technique used (see Sec. 2.7). The simplest and fastest method is to enclose the reachable set by an interval as described in Sec. 3.1.4 and then use interval arithmetic for range bounding. A more accurate but also computational demanding approach is to convert the SPZ representing the reachable set to a Taylor model according to Prop. 3.1.13 followed by using Taylor models for range bounding.

Interval Contraction

Next, in Step 2 of the procedure in Sec. 4.4.2 we enclose the time interval reachable sets $\mathcal{R}^o(\tau_j)$ that intersect the guard set $\mathcal{G}_i$ with intervals

$$\mathcal{I}_j = \texttt{interval}(\mathcal{R}^o(\tau_j))$$

using the `interval` operation for SPZs as defined in Sec. 3.1.4. To reduce the conservatism, we reduce the size of the interval enclosure by applying a contractor as defined in Def. 2.8.1 to the constraint $g_i(x) = 0$ that defines the guard set $\mathcal{G}_i = \langle g_i(x), = \rangle_{LS}$:

$$\widehat{\mathcal{I}}_j = \texttt{contract}(g_i(x), \mathcal{I}_j).$$

Interval Unification

In Step 3 of the procedure in Sec. 4.4.2 we unite the contracted intervals $\widehat{\mathcal{I}}_j$ using the following proposition:

Proposition 4.4.3. *(Union Interval) Given two intervals $\mathcal{I}_1 = [l_1, u_1] \subset \mathbb{R}^n$ and $\mathcal{I}_2 = [l_2, u_2] \subset \mathbb{R}^n$, their union can be enclosed by the interval*

$$\mathcal{I}_1 \cup \mathcal{I}_2 \subseteq \underbrace{\left[\begin{bmatrix}\min(l_{1(1)}, l_{2,(1)})\\ \vdots \\ \min(l_{1(n)}, l_{2,(n)})\end{bmatrix}, \begin{bmatrix}\max(u_{1(1)}, u_{2,(1)})\\ \vdots \\ \max(u_{1(n)}, u_{2,(n)})\end{bmatrix}\right]}_{\mathcal{I}}.$$

The computational complexity with respect to the dimension n is $\mathcal{O}(n)$.

Proof. Since it holds for all dimensions $i \in \{1, \ldots, n\}$ that

$$[l_{1(i)}, u_{1(i)}] \subseteq [\min(l_{1(i)}, l_{2,(i)}), \max(u_{1(i)}, u_{2,(i)})]$$

and

$$[l_{2(i)}, u_{2(i)}] \subseteq [\min(l_{1(i)}, l_{2,(i)}), \max(u_{1(i)}, u_{2,(i)})],$$

we have $\mathcal{I}_1 = [l_{1(1)}, u_{1(1)}] \times \ldots \times [l_{1(n)}, u_{1(n)}] \subseteq \mathcal{I}$ and $\mathcal{I}_2 = [l_{2(1)}, u_{2(1)}] \times \ldots \times [l_{2(n)}, u_{2(n)}] \subseteq \mathcal{I}$, which proves that $\mathcal{I}_1 \cup \mathcal{I}_2 \subseteq \mathcal{I}$.

Complexity: To construct the interval $\mathcal{I}$ we have to perform $2n$ comparisons of scalar numbers, which has complexity $\mathcal{O}(2n) = \mathcal{O}(n)$. □

As a heuristic, we unite the contracted intervals for consecutive time intervals τ_j until the upper bound μ is reached.

Guard Set Intersection

We now describe Step 4 of the procedure from Sec. 4.4.2. Our approach for calculating guard intersections is based on the novel finding that the intersection of an interval with a specific type of polynomial level set can be represented as a Taylor model, which can be converted to a SPZ using Prop. 3.1.13. We demonstrate this with an example:

Example 4.4.4. *The intersection between the interval $\mathcal{I} = [-1, 1] \times [0, 2]$ and the polynomial level set*

$$\mathcal{LS} = \left\{ x \in \mathbb{R}^2 \;\middle|\; x_{(2)} = 2x_{(1)}^2 \right\}$$

can be equivalently represented by the Taylor model $\mathcal{T}(x)$ defined as

$$\mathcal{I} \cap \mathcal{LS} = \left\{ \mathcal{T}(x) \;\middle|\; x \in \mathcal{I} \right\}, \quad \mathcal{T}(x) = \left\langle \begin{bmatrix} x_{(1)} \\ 2x_{(1)}^2 \end{bmatrix}, \emptyset, [-1, 1] \times [0, 2] \right\rangle_{TM},$$

which can be converted to the SPZ

$$\mathcal{PZ} = \left\{ \begin{bmatrix}1\\0\end{bmatrix} \alpha_1 + \begin{bmatrix}0\\2\end{bmatrix} \alpha_1^2 \;\middle|\; \alpha_1 \in [-1, 1] \right\}$$

using Prop. 3.1.13. The resulting sets are visualized on the left side of Fig. 4.20.

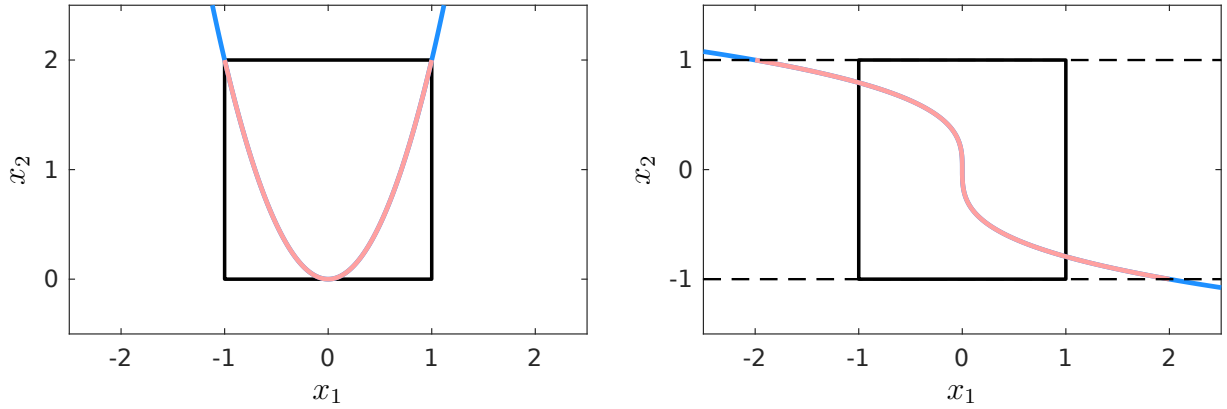

Figure 4.20: Visualization of the results from Example 4.4.4 (left) and Example 4.4.6 (right), where the level sets $\mathcal{LS}$ are depicted in blue, the intervals $\mathcal{I}$ are depicted in black, and the SPZs $\mathcal{PZ}$ enclosing the intersection are depicted in red.

This is generalized in the following theorem:

Theorem 4.4.5. *Given an interval $\mathcal{I} = [l, u] \subset \mathbb{R}^n$ and a level set*

$$\mathcal{LS} = \left\{ x \in \mathbb{R}^n \;\middle|\; x_{(k)} = p_k(x) \right\}, \quad k \in \{1, \ldots, n\}$$

defined by a polynomial function $p_k : \mathbb{R}^n \to \mathbb{R}$, their intersection $\mathcal{I} \cap \mathcal{LS}$ can be tightly enclosed by the Taylor model $\mathcal{T}(x)$ defined as

$$\mathcal{I} \cap \mathcal{LS} \subseteq \left\{ \mathcal{T}(x) \;\middle|\; x \in \mathcal{I} \right\}, \quad \mathcal{T}(x) = \Big\langle \underbrace{\begin{bmatrix} \begin{bmatrix} x_{(1)} & \ldots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \ldots & x_{(n)} \end{bmatrix}^T \end{bmatrix}}_{h(x)}, \emptyset, \mathcal{I} \Big\rangle_{TM}.$$

If the polynomial function $p_k(x)$ defining the level set $\mathcal{LS}$ satisfies

$$\min_{x \in \mathcal{I}} \; p_k(x) \geq l_{(k)} \quad \text{and} \quad \max_{x \in \mathcal{I}} \; p_k(x) \leq u_{(k)}, \tag{4.60}$$

the Taylor model $\mathcal{T}(x)$ exactly represents the intersection $\mathcal{I} \cap \mathcal{LS}$; otherwise, $\mathcal{T}(x)$ encloses the intersection $\mathcal{I} \cap \mathcal{LS}$.

Proof. The idea of the proof is to replace variable $x_{(k)}$ with the function $p_k(x)$ that defines the level set. If condition (4.60) is fulfilled, the intersection $\mathcal{I} \cap \mathcal{LS}$ can be equivalently expressed as

$$\mathcal{I} \cap \mathcal{LS} = \left\{ x \in \mathcal{I} \;\middle|\; x_{(k)} = p_k(x) \right\} \overset{(4.60)}{=}$$

$$\left\{ \begin{bmatrix} \begin{bmatrix} x_{(1)} & \ldots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \ldots & x_{(n)} \end{bmatrix}^T \end{bmatrix} \;\middle|\; x \in \mathcal{I} \right\} = \left\{ \mathcal{T}(x) \;\middle|\; x \in \mathcal{I} \right\}.$$

If condition (4.60) is not fulfilled, the calculated Taylor model encloses the intersection $\mathcal{I} \cap \mathcal{LS}$ since

$$\big\{p_k(x) \;\big|\; x \in \mathcal{I}\big\} \supseteq [l_{(k)}, u_{(k)}]$$

and $\mathcal{I} = [l, u]$. □

Let us demonstrate the over-approximation error for the case that condition (4.60) is not fulfilled with an example:

Example 4.4.6. *The intersection between the interval* $\mathcal{I} = [-1, 1] \times [-1, 1]$ *and the polynomial level set*

$$\mathcal{LS} = \Big\{x \in \mathbb{R}^2 \;\Big|\; x_{(1)} = -2x_{(2)}^3\Big\}$$

can be enclosed by the Taylor model $\mathcal{T}(x)$ *defined as*

$$\mathcal{I} \cap \mathcal{LS} = \big\{\mathcal{T}(x) \;\big|\; x \in \mathcal{I}\big\}, \quad \mathcal{T}(x) = \left\langle \begin{bmatrix} -2x_{(2)}^3 \\ x_{(2)} \end{bmatrix}, \emptyset, [-1, 1] \times [-1, 1] \right\rangle_{TM}$$

according to Thm. 4.4.5, which can be converted to the SPZ

$$\mathcal{PZ} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha_1 - \begin{bmatrix} 0 \\ 2 \end{bmatrix} \alpha_1^3 \;\middle|\; \alpha_1 \in [-1, 1] \right\}$$

using Prop. 3.1.13. Since we have

$$\min_{x_{(2)} \in [-1,1]} -2x_{(2)}^3 = -2 \not\geq -1 \quad \textit{and} \quad \max_{x_{(2)} \in [-1,1]} -2x_{(2)}^3 = 2 \not\leq 1$$

condition (4.60) *is violated, so that* $\mathcal{T}(x)$ *and* $\mathcal{PZ}$ *represent an enclosure instead of the exact intersection. This is visualized on the right side of Fig. 4.20. For this example the over-approximation error can be avoided by properly contracting the interval* $\mathcal{I}$ *prior to the computation of the intersection, which motivates the contraction in Step 2 of the procedure in Sec. 4.4.2*

Based on Thm. 4.4.5, we now show how the intersection between a level set $\mathcal{LS} = \langle w(x), = \rangle_{LS}$ defined by a non-polynomial function $w : \mathbb{R}^n \to \mathbb{R}$ and an interval $\mathcal{I}$ can be tightly enclosed with a Taylor model. We distinguish the case where the equality constraint $w(x) = 0$ is symbolically solvable for one variable $x_{(k)}$ from the case where it is not. The constraint $w(x) = 0$ is symbolically solvable for the variable $x_{(k)}$ if the set $\{x \mid w(x) = 0\}$ can be equivalently represented as

$$\big\{x \;\big|\; x_{(k)} = \widehat{w}(x)\big\} \;\text{ with }\; \frac{\partial \widehat{w}(x)}{\partial x_{(k)}} = 0, \tag{4.61}$$

where $\partial \widehat{w}(x)/\partial x_{(k)} = 0$ implies that $\widehat{w}(x)$ does not depend on $x_{(k)}$ for all x. We demonstrate this with an example:

Example 4.4.7. *The guard set*

$$\mathcal{LS} = \Big\{x \in \mathbb{R}^2 \;\Big|\; \underbrace{x_{(2)}x_{(1)} + \sin(x_{(1)})}_{w(x)} = 0\Big\}$$

can be equivalently represented as

$$\mathcal{LS} = \left\{ x \in \mathbb{R}^2 \;\middle|\; x_{(2)} = \underbrace{-\frac{\sin(x_{(1)})}{x_{(1)}}}_{\widehat{w}(x)} \right\}$$

since the constraint $w(x) = 0$ *is symbolically solvable for* $x_{(2)}$.

We first consider the case where the equality constraint is symbolically solvable for one variable:

Proposition 4.4.8. *Given an interval* $\mathcal{I} \subset \mathbb{R}^n$*, a level set* $\mathcal{LS} = \langle w(x), = \rangle_{LS} \subset \mathbb{R}^n$ *with* $w : \mathbb{R}^n \to \mathbb{R}$ *which can be equivalently represented as* $\mathcal{LS} = \left\{ x \mid x_{(k)} = \widehat{w}(x) \right\}$ *with* $\partial \widehat{w}(x)/\partial x_{(k)} = 0$*, and the Taylor order* $\kappa \in \mathbb{N}$*, the intersection* $\mathcal{I} \cap \mathcal{LS}$ *can be tightly enclosed by the Taylor model* $\mathcal{T}(x)$ *defined as*

$$\mathcal{I} \cap \mathcal{LS} \subseteq \left\{ \mathcal{T}(x) \mid x \in \mathcal{I} \right\}, \quad \mathcal{T}(x) = \left\langle \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} \mathbf{0} \\ l \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ u \\ \mathbf{0} \end{bmatrix} \end{bmatrix}, \mathcal{I} \right\rangle_{TM},$$

where the polynomial function $p_k(x)$ *represents the polynomial part of the Taylor series expansion of the function* $\widehat{w}(x)$

$$\widehat{w}(x) \in \underbrace{\sum_{i=0}^{\kappa} \frac{\left((x - x^*)^T \nabla\right)^i \widehat{w}(x^*)}{i!}}_{p_k(x)} \oplus \mathcal{L}(x) \tag{4.62}$$

at the expansion point $x^* = \texttt{center}(\mathcal{I})$*. The lower bound* $l \in \mathbb{R}$ *and upper bound* $u \in \mathbb{R}$ *enclose the Lagrange remainder* $\mathcal{L}(x)$ *given as*

$$\forall x \in \mathcal{I} : \quad \mathcal{L}(x) = \left\{ \frac{\left((x - x^*)^T \nabla\right)^{\kappa+1} \widehat{w}(\widehat{x})}{(\kappa+1)!} \;\middle|\; \widehat{x} = x^* + \lambda(x - x^*),\ \lambda \in [0,1] \right\} \subseteq [l, u] \tag{4.63}$$

by an interval $[l, u]$.

Proof. With the Taylor series expansion of $\widehat{w}(x)$ in (4.62) the intersection between $\mathcal{I}$ and the $\mathcal{LS}$ can be formulated as

$$\mathcal{I} \cap \mathcal{LS} = \left\{ x \in \mathcal{I} \mid x_{(k)} = \widehat{w}(x) \right\} \overset{(4.62)}{\subseteq} \left\{ x \in \mathcal{I} \mid x_{(k)} \in p_k(x) \oplus [l, u] \right\}$$

$$= \left\{ \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ x_{(k)} \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix} \;\middle|\; x \in \mathcal{I},\ x_{(k)} \in p_k(x) \oplus [l, u] \right\}$$

$$= \left\{ \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ s \\ \mathbf{0} \end{bmatrix} \;\middle|\; x \in \mathcal{I},\ s \in [l, u] \right\} \overset{\text{Def. 2.2.8}}{=} \left\{ \mathcal{T}(x) \mid x \in \mathcal{I} \right\},$$

which results in the Taylor model $\mathcal{T}(x)$. □

If the equality constraint is solvable for multiple variables we choose the variable that results in the tightest enclosure of the intersection $\mathcal{I} \cap \mathcal{G}$. We demonstrate the computation of the intersection according to Prop. 4.4.8 with an example:

Example 4.4.9. *We consider the level set*

$$\mathcal{LS} = \left\{ x \in \mathbb{R}^2 \;\middle|\; e^{x_{(1)}} + 0.2\, x_{(1)}^2 - x_{(2)} - 1 = 0 \right\} = \left\{ x \in \mathbb{R}^2 \;\middle|\; x_{(2)} = \underbrace{e^{x_{(1)}} + 0.2\, x_{(1)}^2 - 1}_{\widehat{w}(x)} \right\}$$

and the interval $\mathcal{I} = [-2, -1] \times [-0.5, 0]$. *Computation of a Taylor model that encloses the intersection* $\mathcal{I} \cap \mathcal{LS}$ *according to Prop. 4.4.8 using a Taylor series of order* $\kappa = 2$ *yields*

$$\mathcal{T}(x) = \left\langle \begin{bmatrix} x_{(1)} \\ -0.1911 + 0.5579\, x_{(1)} + 0.3116\, x_{(1)}^2 \end{bmatrix}, \left[\begin{bmatrix} 0 \\ -0.0077 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.0077 \end{bmatrix} \right], \mathcal{I} \right\rangle_{TM},$$

which can be equivalently represented by the SPZ

$$\mathcal{PZ} = \left\{ \begin{bmatrix} -1.5 \\ -0.3269 \end{bmatrix} + \begin{bmatrix} 0.5 \\ -0.1884 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 0.0779 \end{bmatrix} \alpha_1^2 + \begin{bmatrix} 0 \\ 0.0077 \end{bmatrix} \beta_1 \;\middle|\; \alpha_1, \beta_1 \in [-1, 1] \right\}$$

using Prop. 3.1.13. The resulting sets are visualized on the left side of Fig. 4.21.

Next, we consider the case where the equality constraint is not symbolically solvable for one variable:

Proposition 4.4.10. *Given an interval* $\mathcal{I} \subset \mathbb{R}^n$, *a level set* $\mathcal{LS} = \langle w(x), = \rangle_{LS} \subset \mathbb{R}^n$ *with* $w : \mathbb{R}^n \to \mathbb{R}$, *and the Taylor order* $\kappa \in \mathbb{N}$, *the intersection* $\mathcal{I} \cap \mathcal{LS}$ *can be tightly enclosed by the Taylor model* $\mathcal{T}(x)$ *defined as*

$$\mathcal{I} \cap \mathcal{LS} \subseteq \left\{ \mathcal{T}(x) \;\middle|\; x \in \mathcal{I} \right\}, \quad \mathcal{T}(x) = \left\langle \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix}, \left[\begin{bmatrix} \mathbf{0} \\ l \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ u \\ \mathbf{0} \end{bmatrix} \right], \mathcal{I} \right\rangle_{TM},$$

where the polynomial function $p_k(x)$ *is obtained by splitting the function*

$$\frac{1}{a_{(k)}} \left(-w(x^*) + a^T x^* - \sum_{\substack{i=1 \\ i \neq k}}^{n} a_{(i)} x_{(i)} - \sum_{i=2}^{\kappa} \frac{\left((x - x^*)^T \nabla \right)^i w(x^*)}{i!} \right) := p_k(x) + \widehat{p}(x) \quad (4.64)$$

with

$$a = \left. \frac{\partial w(x)}{\partial x} \right|_{x^*}, \quad x^* = \texttt{center}(\mathcal{I}), \quad \frac{\partial p_k(x)}{\partial x_{(k)}} = 0$$

into one part $\widehat{p}(x)$ *containing the variable* $x_{(k)}$ *and one part* $p_k(x)$ *that does not contain the variable* $x_{(k)}$. *The lower bound* $l \in \mathbb{R}$ *and upper bound* $u \in \mathbb{R}$ *enclose* $\widehat{p}(x)$ *and the Lagrange remainder* $\mathcal{L}(x)$

$$\forall x \in \mathcal{I} : \quad \widehat{p}(x) \oplus \frac{(-1)}{a_{(k)}} \underbrace{\left\{ \frac{\left((x - x^*)^T \nabla \right)^{\kappa+1} w(\widehat{x})}{(\kappa+1)!} \;\middle|\; \widehat{x} = x^* + \lambda(x - x^*),\ \lambda \in [0, 1] \right\}}_{\mathcal{L}(x)} \subseteq [l, u] \quad (4.65)$$

with an interval $[l, u]$.

Proof. Using a Taylor series expansion with order κ

$$w(x) \in \sum_{i=0}^{\kappa} \frac{\left((x-x^*)^T\nabla\right)^i\ w(x^*)}{i!} \oplus \mathcal{L}(x) =$$
$$w(x^*) - a^T x^* + a_{(k)}x_{(k)} + \sum_{\substack{i=1\\ i\neq k}}^{n} a_{(i)}x_{(i)} + \sum_{i=2}^{\kappa} \frac{\left((x-x^*)^T\nabla\right)^i\ w(x^*)}{i!} \oplus \mathcal{L}(x)$$

of the function $w(x)$ at the expansion point x^*, the equality constraint $w(x) = 0$ can be solved for

$$\forall x \in \mathcal{I}: \quad x_{(k)} \in$$
$$\frac{1}{a_{(k)}}\left(- w(x^*) + a^T x^* - \sum_{\substack{i=1\\ i\neq k}}^{n} a_{(i)}x_{(i)} - \sum_{i=2}^{\kappa} \frac{\left((x-x^*)^T\nabla\right)^i\ w(x^*)}{i!}\right) \oplus \frac{(-1)}{a_{(k)}}\mathcal{L}(x) \tag{4.66}$$
$$\overset{(4.64)}{=} p_k(x) + \widehat{p}(x) \oplus \frac{(-1)}{a_{(k)}}\mathcal{L}(x) \overset{(4.65)}{\subseteq} p_k(x) \oplus [l,u],$$

so that the intersection between the interval $\mathcal{I}$ and the level set $\mathcal{LS}$ can be formulated as

$$\mathcal{I} \cap \mathcal{LS} = \left\{x \in \mathcal{I} \;\middle|\; w(x) = 0\right\} \overset{(4.66)}{\subseteq} \left\{x \in \mathcal{I} \;\middle|\; x_{(k)} \in p_k(x) \oplus [l,u]\right\}$$
$$= \left\{ \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ x_{(k)} \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix} \;\middle|\; x \in \mathcal{I},\ x_{(k)} \in p_k(x) \oplus [l,u] \right\}$$
$$= \left\{ \begin{bmatrix} \begin{bmatrix} x_{(1)} & \dots & x_{(k-1)} \end{bmatrix}^T \\ p_k(x) \\ \begin{bmatrix} x_{(k+1)} & \dots & x_{(n)} \end{bmatrix}^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ s \\ \mathbf{0} \end{bmatrix} \;\middle|\; x \in \mathcal{I},\ s \in [l,u] \right\} \overset{\text{Def. 2.2.8}}{=} \left\{\mathcal{T}(x) \;\middle|\; x \in \mathcal{I}\right\},$$

which results in the Taylor model $\mathcal{T}(x)$. □

We select the variable $x_{(k)}$ for which the constraint is solved such that we achieve the tightest enclosure of $\mathcal{I} \cap \mathcal{LS}$, where we have to ensure that $\partial w(x)/\partial x_{(k)}|_{x^*} \neq 0$ to avoid divisions by zero. If $\partial w(x)/\partial x_{(k)}|_{x^*} = 0$ for all variables $x_{(k)}$, $k = 1, \dots, n$, we use a different expansion point x^* for the Taylor series. In rare cases the computed Taylor model becomes very large due to the obtained over-approximation. To increase the robustness of our approach, we substitute each dimension of the calculated Taylor model for which the width of the interval remainder of the Taylor model is larger than the width of the domain $\mathcal{I}$ by $\mathcal{I}$.

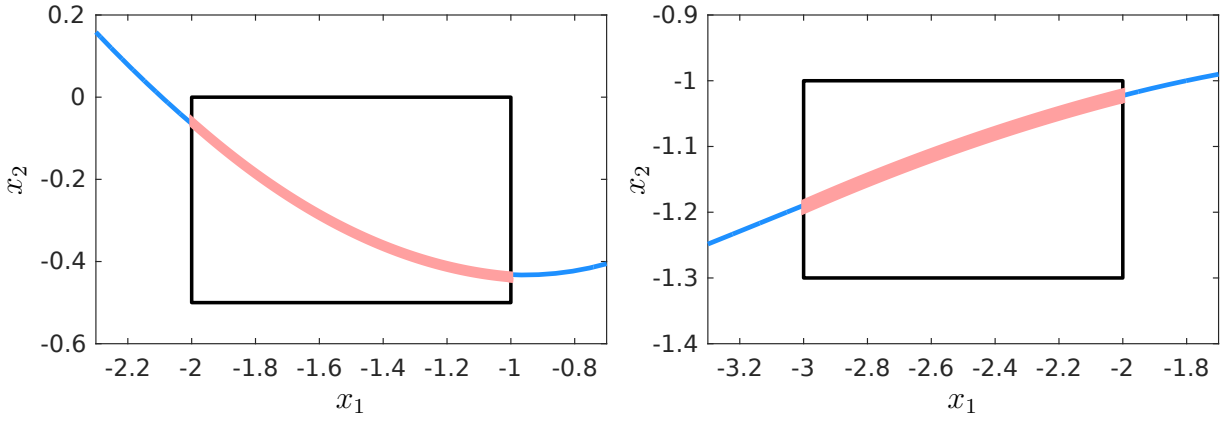

Figure 4.21: Visualization of the results from Example 4.4.9 (left) and Example 4.4.11 (right), where the level sets $\mathcal{LS}$ are depicted in blue, the intervals $\mathcal{I}$ are depicted in black, and the SPZs $\mathcal{PZ}$ enclosing the intersection are depicted in red.

We demonstrate the computation of the intersection for the case where the equality constraint is not solvable for one variable with an example:

Example 4.4.11. *We consider the level set*

$$\mathcal{LS} = \left\{ x \in \mathbb{R}^2 \;\middle|\; 0.2\big(\sin(x_{(1)})x_{(2)} + \cos(x_{(2)})x_{(1)}\big) - x_{(2)} - 1 = 0 \right\}$$

and the interval $\mathcal{I} = [-3,-2] \times [-1.3,-1]$. *Computation of a Taylor model enclosing the intersection according to Prop. 4.4.10 using a Taylor series of order* $\kappa = 2$ *yields*

$$\mathcal{T}(x) = \left\langle \begin{bmatrix} x_{(1)} \\ -0.9481 - 0.0496\,x_{(1)} - 0.0437\,x_{(1)}^2 \end{bmatrix}, \begin{bmatrix} \begin{bmatrix} 0 \\ -0.0072 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.0087 \end{bmatrix} \end{bmatrix}, \mathcal{I} \right\rangle_{TM},$$

which can be equivalently represented by the SPZ

$$\mathcal{PZ} = \left\{ \begin{bmatrix} -2.5 \\ -1.0964 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.0844 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ -0.0109 \end{bmatrix} \alpha_1^2 + \begin{bmatrix} 0 \\ 0.0079 \end{bmatrix} \beta_1 \;\middle|\; \alpha_1, \beta_1 \in [-1,1] \right\}$$

using Prop. 3.1.13. The resulting sets are visualized on the right side of Fig. 4.21.

Based on Prop. 4.4.8 and Prop. 4.4.10 we can tightly enclose the intersection between the united intervals $\overline{\mathcal{I}}$ from Step 3 of the procedure in Sec. 4.4.2 and the guard set $\mathcal{G}_i$ by a Taylor model, which we convert to a SPZ $\mathcal{PZ}$ using Prop. 3.1.13. What remains is to show how we compute an interval enclosure of the Lagrange remainder $\mathcal{L}(x)$ in (4.63) and (4.65) with an interval. We consider Taylor order $\kappa = 2$ for simplicity, which is often sufficient to achieve a tight enclosure. For the Lagrange remainder in (4.63) we obtain

$$\begin{aligned} \forall x \in \mathcal{I}: \; \mathcal{L}(x) &\overset{(4.63)}{=} \left\{ \frac{\big((x-x^*)^T\nabla\big)^3\,\widehat{w}(\widehat{x})}{3!} \;\middle|\; \widehat{x} = x^* + \lambda(x-x^*),\; \lambda \in [0,1] \right\} \\ &\subseteq \frac{1}{6}\left\{ \big((x-x^*)^T\nabla\big)^3\,\widehat{w}(\widehat{x}) \;\middle|\; x, \widehat{x} \in \mathcal{I} \right\} \overset{(2.10)}{\subseteq} \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-x^*)\big) = [l,u], \end{aligned} \tag{4.67}$$

where

$$\boldsymbol{\mathcal{D}} = \texttt{bound}\big(\nabla^3 \widehat{w}(x), \mathcal{I}\big)$$

is computed using range bounding as introduced in Sec. 2.7. The Lagrange remainder in (4.65) can be enclosed in a similar way, where we substitute the function $\widehat{w}(x)$ with the function $w(x)$.

Reset Function

In Step 5 of the procedure described in Sec. 4.4.2, we apply the reset function $r_i(x)$ to the SPZs $\mathcal{PZ}$ that enclose the intersection with the guard set. This corresponds to calculating the set

$$\mathcal{M} = \big\{ r_i(x) \;\big|\; x \in \mathcal{PZ} \big\}. \tag{4.68}$$

Since the set $\mathcal{M}$ as defined in (4.68) cannot be calculated exactly, we compute a tight enclosure instead. For this, we first abstract the reset function $r_i(x)$ by a Taylor series expansion of order κ_r

$$r_{i(j)}(x) \in \sum_{k=0}^{\kappa_r} \frac{\big((x-x^*)^T\nabla\big)^k r_{i(j)}(x^*)}{k!} \oplus \mathcal{L}_{r(j)}(x), \quad j = 1, \ldots, n, \tag{4.69}$$

around the expansion point $x^* = \texttt{center}(\mathcal{PZ})$, where `center` returns the constant offset of a SPZ and

$$\mathcal{L}_{r(j)}(x) = \left\{ \frac{\big((x-x^*)^T\nabla\big)^{\kappa_r+1} r_{i(j)}(\widehat{x})}{(\kappa_r+1)!} \;\middle|\; \widehat{x} = x^* + \lambda(x-x^*),\ \lambda \in [0,1] \right\} \tag{4.70}$$

is the Lagrange remainder. We focus at this point on the case with Taylor order $\kappa_r = 2$ for simplicity since the extension to higher orders is straightforward. For Taylor order $\kappa_r = 2$, the Taylor series expansion in (4.69) becomes

$$\begin{aligned} r_{i(j)}(x) \in\, & r_{i(j)}(x^*) + \underbrace{\left.\frac{\partial r_{i(j)}(x)}{\partial x}\right|_{x^*}}_{A_{(j,\cdot)}} (x - x^*) \\ & + \frac{1}{2}(x-x^*)^T \underbrace{\left.\frac{\partial^2 r_{i(j)}(x)}{\partial x^2}\right|_{x^*}}_{Q_j} (x-x^*) \oplus \mathcal{L}_{r(j)}(x), \quad j = 1, \ldots, n. \end{aligned} \tag{4.71}$$

Using (4.71) and the following enclosure of the Lagrange remainder

$$\forall x \in \mathcal{PZ}: \quad \mathcal{L}_r(x) = \mathcal{L}_{r(1)}(x) \times \ldots \times \mathcal{L}_{r(n)}(x) \overset{(4.70)}{=}$$

$$\left\{ s \;\middle|\; s_{(j)} = \frac{\big((x-x^*)^T\nabla\big)^3 r_{i(j)}(\widehat{x})}{3!},\ \widehat{x} = x^* + \lambda(x-x^*),\ \lambda \in [0,1] \right\} \overset{\mathcal{PZ} \subseteq \mathcal{I}_r}{\subseteq} \tag{4.72}$$

$$\frac{1}{6}\left\{ s \;\middle|\; s_{(j)} = \big((x-x^*)^T\nabla\big)^3 r_{(j)}(\widehat{x}),\ x, \widehat{x} \in \mathcal{I}_r \right\} \overset{(2.10)}{\subseteq} \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}_r, \mathcal{I}_r \oplus (-x^*)\big)$$

with

$$\mathcal{I}_r = \texttt{interval}(\mathcal{PZ}), \quad \boldsymbol{\mathcal{D}}_r = \texttt{bound}\big(\nabla^3 r_i(x), \mathcal{I}_r\big),$$

we calculate a tight enclosure of the set $\mathcal{M}$ in (4.68) as

$$\mathcal{M} = \big\{ r_i(x) \;\big|\; x \in \mathcal{PZ} \big\}$$

$$\overset{(4.71)}{\subseteq} \left\{ r_i(x^*) + A(x-x^*) + \frac{1}{2}\begin{bmatrix} (x-x^*)^T Q_1 (x-x^*) \\ \vdots \\ (x-x^*)^T Q_n (x-x^*) \end{bmatrix} \oplus \mathcal{L}_r(x) \;\middle|\; x \in \mathcal{PZ} \right\}$$

$$\overset{(4.72)}{\subseteq} r_i(x^*) \oplus \big(A \otimes (\mathcal{PZ} \oplus (-x^*))\big) \boxplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{PZ} \oplus (-x^*)\big) \oplus \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}_r, \mathcal{I}_r \oplus (-x^*)\big)$$

with $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$, where the linear map is computed using Prop. 3.1.18, the quadratic map is computed using Prop. 3.1.31, the exact addition $\boxplus$ is computed using Prop. 3.1.20, Minkowski sums are calculated using Prop. 3.1.19, the interval enclosure $\mathcal{I}_r = \texttt{interval}(\mathcal{PZ})$ is calculated as described in Sec. 3.1.4, and $\boldsymbol{\mathcal{D}}_r = \texttt{bound}(\nabla^3 r_i(x), \mathcal{I}_r)$ is calculated using range bounding as described in Sec. 2.7.

Unification

Finally, in the last step of the procedure in Sec. 4.4.2, we unite parallel sets $\mathcal{M}_j$, $j = 1, \ldots, \mu$. For this, we compute the linear combination on SPZs using Prop. 3.1.26. As for the conversion from V-representation to Z-representation in Alg. 4, we use a hierarchical approach to unite all sets since this minimizes the representation size of the resulting SPZ $\mathcal{C}$:

$$\mathcal{C} = comb\Big(\;\ldots\; comb\big(comb(\mathcal{M}_1, \mathcal{M}_2), comb(\mathcal{M}_3, \mathcal{M}_4)\big) \;\ldots\; \Big).$$

If the desired zonotope order is exceeded during unification, one can additionally reduce the representation size using the `reduce` operation in Prop. 3.1.39.

4.4.4 Preventing Endless Loops for Identity Resets

One problem with the procedure for reachability analysis of hybrid systems introduced in Sec. 4.4.2 is that it can potentially result in endless loops before reaching the final time. Especially for hybrid automata with identity reset functions $r(x) = x$ this issue occurs quite often due to the missing translation by the reset function. In this section, we introduce a solution to prevent these endless loops. Let us first demonstrate the problem on the example of a hybrid automaton with two modes $\mathscr{H} = \langle \mathbf{F}, \mathbf{Y}, \mathbf{T} \rangle_{HA}$, where

$$\mathscr{H} = \Big\langle \big(f_1(x(t), u(t)), f_2(x(t), u(t))\big), (\mathcal{Y}_1, \mathcal{Y}_2), (\mathscr{T}_1, \mathscr{T}_2) \Big\rangle_{HA}$$

with transitions

$$\mathscr{T}_1 = \langle \mathcal{G}, r(x), 1, 2 \rangle_T, \quad \mathscr{T}_2 = \langle \mathcal{G}, r(x), 2, 1 \rangle_T, \quad r(x) = x$$

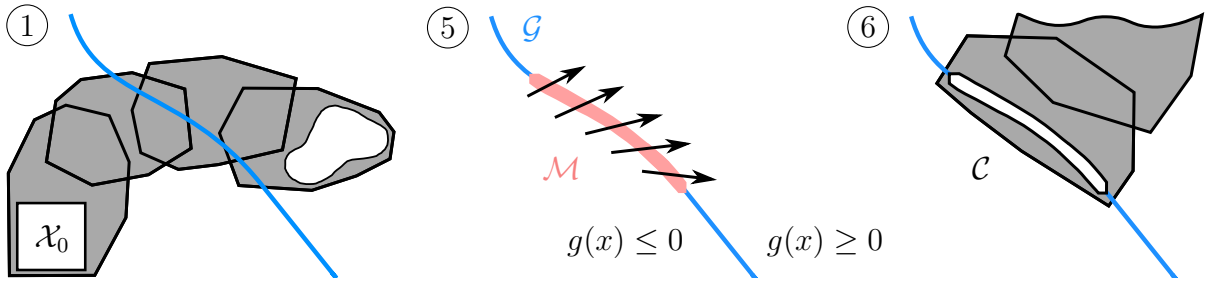

Figure 4.22: Visualization of the steps 1, 5, and 6 from the procedure in Sec. 4.4.2 for a system with identity resets, where the flow of the system is depicted with black arrows.

and invariant and guard sets

$$\mathcal{Y}_1 = \{x \mid g(x) \leq 0\}, \quad \mathcal{Y}_2 = \{x \mid g(x) \geq 0\}, \quad \mathcal{G} = \{x \mid g(x) = 0\}.$$

Applying the procedure in Sec. 4.4.2 with $\mu = 1$ to compute the reachable set for an initial set $\mathcal{X}_0$ and an initial mode $\sigma_0 = 1$ results in the situation visualized in Fig. 4.22: Since the reachable set intersects the guard set $\mathcal{G}$ of transition $\mathscr{T}_1$, we compute a SPZ $\mathcal{PZ}$ that encloses the intersection between the reachable set and the guard set. Due to the identity reset and $\mu = 1$, the initial set $\mathcal{C} = \mathcal{M} = \mathcal{PZ}$ for continuous reachability analysis in mode 2 is identical to $\mathcal{PZ}$. Now, due to the over-approximation in $\mathcal{PZ}$ and in the reachable set enclosure, we have that the first time interval reachable set $\mathcal{R}^o(\tau_1)$ for mode 2 will again intersect the guard set $\mathcal{G}$, so that the system takes the transition $\mathscr{T}_2$ back to mode 1. This process will continue in an endless loop of jumps between mode 1 and mode 2 since the algorithm is not able to make any progress in time.

From Fig. 4.22 we can deduce that the flow of the system in mode 2 defined by the differential equation $\dot{x}(t) = f_2(x(t), u(t))$ actually points from mode 1 to mode 2, so that it is impossible to get back from mode 2 to mode 1. To formalize this observation, we take the total derivative of the function $g(x)$ that defines the guard set with respect to time, which yields

$$\frac{dg(x(t))}{dt} = \frac{\partial g(x(t))}{\partial x} \cdot \frac{\partial x(t)}{\partial t} = \frac{\partial g(x(t))}{\partial x} \cdot f_2(x(t), u(t)), \tag{4.73}$$

where we inserted the system dynamics $\partial x(t)/\partial t = \dot{x}(t) = f_2(x(t), u(t))$. Clearly, it holds that it is impossible that the system gets from mode 2 back to mode 1 if the total derivative is greater than 0 for all inputs $u(t) \in \mathcal{U}$ and all points inside the initial set $\mathcal{C}$ for mode 2:

$$\min_{\substack{x(t)\in\mathcal{C} \\ u(t)\in\mathcal{U}}} \frac{dg(x(t))}{dt} \overset{(4.73)}{=} \min_{\substack{x(t)\in\mathcal{C} \\ u(t)\in\mathcal{U}}} \underbrace{\frac{\partial g(x(t))}{\partial x} \cdot f_2(x(t), u(t))}_{h(z)} > 0, \tag{4.74}$$

where $z = [x(t)^T \; u(t)^T]^T$. To prevent endless loops, we can then simply use range bounding to efficiently check if the criterion in (4.74) is satisfied or not:

$$(l > 0) \Rightarrow \left(\min_{\substack{x(t)\in\mathcal{C} \\ u(t)\in\mathcal{U}}} \frac{dg(x(t))}{dt} > 0 \right), \quad [l, u] = \texttt{bound}(h(z), \mathcal{C} \times \mathcal{U}).$$

For this we can either enclose the SPZ $\mathcal{C} \times \mathcal{U}$ with an interval as described in Sec. 3.1.4 or convert it to a Taylor model using Prop. 3.1.12 before we apply one of the range bounding techniques in Sec. 2.7.

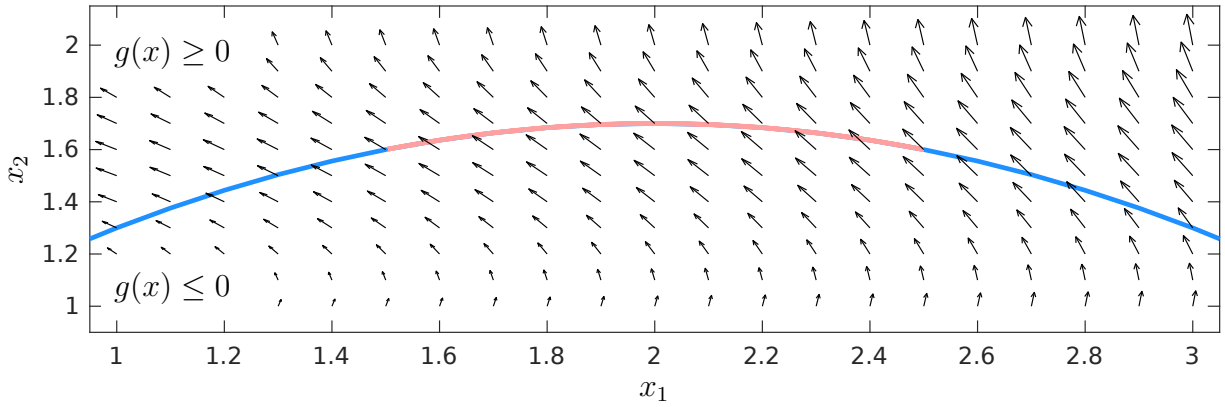

Figure 4.23: Visualization of Example 4.4.12, where the flow function is depicted by black arrows, the guard set $\mathcal{G}$ is shown in blue, and the SPZ $\mathcal{PZ}$ is shown in red.

Let us finally demonstrate our approach for preventing endless loops with an example:

Example 4.4.12. *We consider the flow function*

$$\begin{bmatrix} \dot{x}_{(1)} \\ \dot{x}_{(2)} \end{bmatrix} = \underbrace{\begin{bmatrix} \sin(3\,x_{(2)}) \\ 0.5\,x_{(1)}x_{(2)} \end{bmatrix}}_{f(x,u)},$$

the guard set

$$\mathcal{G} = \{x \in \mathbb{R}^2 \mid \underbrace{4\,x_{(1)}^2 - 16\,x_{(1)} - 1 + 10\,x_{(2)}}_{g(x)} = 0\},$$

and the SPZ

$$\mathcal{PZ} = \left\{ \begin{bmatrix} 2 \\ 1.7 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \alpha_1 - \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \alpha_1^2 \;\middle|\; \alpha_1 \in [-1, 1] \right\},$$

which are visualized in Fig. 4.23. For the total derivative with respect to time we obtain according to (4.73)

$$\begin{aligned} \frac{dg(x)}{dt} = \frac{\partial g(x)}{\partial x} \cdot f(x,u) &= \begin{bmatrix} (8\,x_{(1)} - 16) & 10 \end{bmatrix} \begin{bmatrix} \sin(3\,x_{(2)}) \\ 0.5\,x_{(1)}x_{(2)} \end{bmatrix} \\ &= (8\,x_{(1)} - 16)\sin(3\,x_{(2)}) + 5\,x_{(1)}x_{(2)}. \end{aligned}$$

Using Taylor models for range bounding yields

$$\texttt{bound}\big((8\,x_{(1)} - 16)\sin(3\,x_{(2)}) + 5\,x_{(1)}x_{(2)}, \mathcal{PZ}\big) = [15.9, 17.1] > 0,$$

which proves that the guard set can only be crossed in one direction.

4.4.5 Computational Complexity

We now derive the computational complexity of our novel approach with respect to the system dimension n. For this, we require the following assumption:

Assumption 4.4.13. *Given a hybrid automaton with*

- *invariant sets* $\mathcal{Y}_k = \langle y_k(x), \leq \rangle_{LS}$ *with* $y_k : \mathbb{R}^n \to \mathbb{R}^{o_k}$, $k = 1, \ldots, p$,
- *guard sets* $\mathcal{G}_i = \langle g_i(x), = \rangle_{LS}$ *with* $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, q$,
- *reset functions* $r_i : \mathbb{R}^n \to \mathbb{R}^n$, $i = 1, \ldots, q$,

where $e \in \mathbb{N}_0$ *denotes the maximum number of elementary operations required for the evaluation of* $g_i(x)$ *and one subfunction* $y_{k(j)}(x)$, $j = 1, \ldots, o_k$ *or* $r_{i(j)}(x)$, $j = 1, \ldots, n$, *we assume for the derivation of the computational complexity that*

$$e = c_e n, \quad \forall k \in \{1, \ldots, p\} : \; o_k = c_o n$$

with constants $c_e, c_o \in \mathbb{R}_{\geq 0}$. *In addition, we assume that taking the derivative of* $y_{k(j)}(x)$, $g_i(x)$, *and* $r_{i(j)}(x)$ *only changes the number of required elementary operations by a constant factor.*

It is easy to see that Assumption 4.4.13 does not always hold. If, for example, $y_{k(j)}(x)$, $g_i(x)$, or $r_{i(j)}(x)$ is a non-sparse quadratic function, we have $e = 0.5(n^2 + n)$, so that Assumption 4.4.13 is violated. In practice, however, Assumption 4.4.13 is satisfied for most hybrid systems. In addition to the above assumption, we consider Taylor order $\kappa = 2$ for computation of the enclosing Taylor models and Taylor order $\kappa_r = 2$ for enclosing the nonlinear mapping defined by the reset function. Moreover, we use interval arithmetic for range bounding. Since the number of time intervals for reachability analysis as well as the upper bound for the number of parallel intervals μ do not depend on the system dimension, we have to perform all computations on a constant number of parallel sets. For the derivation of the computational complexity it is therefore sufficient to consider the computations on a single set only.

Let us now derive the computational complexity for the single steps of the procedure in Sec. 4.4.2. For computing an outer-approximation of the reachable for the continuous dynamics in Step 1, we use the conservative polynomialization algorithm in Alg. 6, which has complexity $\mathcal{O}(n^5)$ according to Sec. 4.1.4. To detect when the reachable set is fully located outside the invariant set and to select the time intervals in which the reachable set intersects the guard set, we first enclose all time point reachable sets $\mathcal{R}^o(t_j)$ and time interval reachable sets $\mathcal{R}^o(\tau_j)$ with intervals as described in Sec. 3.1.4. According to Tab. 3.3, computing an interval enclosure of a SPZ has complexity $\mathcal{O}(n^3)$ if interval arithmetic is used for range bounding. In addition, the complexity of intersection detection using Prop. 4.4.1 and Prop. 4.4.2 is $o_k \cdot \mathcal{O}(\texttt{bound}) + \mathcal{O}(\texttt{bound}) = o_k \cdot \mathcal{O}(\texttt{bound})$, which is $o_k \cdot \mathcal{O}(e)$ according to Tab. 2.4 if interval arithmetic is used for range bounding.

In Step 2, we can reuse the interval enclosures computed in Step 1, so that the computational complexity is identical to the complexity of contraction $\mathcal{O}(\texttt{contract})$, where the complexity for different contractors is summarized in Tab. 2.5.

Computing the union of two n-dimensional intervals using Prop. 4.4.3 as done in Step 3 has complexity $\mathcal{O}(n)$ according to Prop. 4.4.3.

In Step 4, we first compute a Taylor model that encloses the intersection with the guard set using Prop. 4.4.8 or Prop. 4.4.10 followed by a conversion of the Taylor model to a SPZ using Prop. 3.1.13. For both, Prop. 4.4.8 and Prop. 4.4.10, the part dominating the computational complexity is the enclosure of the Lagrange remainder $\mathcal{L}(x) \subseteq 1/6\ poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-x^*))$ with $\boldsymbol{\mathcal{D}} = \texttt{bound}(\nabla^3 \widehat{w}(x), \mathcal{I})$ according to (4.67). Since $\nabla^3 \widehat{w}(x)$ consists of n^3 scalar functions, construction of $\boldsymbol{\mathcal{D}}$ has complexity $\mathcal{O}(en^3)$ according to Tab. 2.4 if interval arithmetic is used for range bounding. In addition, computation of the cubic map $1/6\ poly(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-x^*))$ as defined in (2.10) using interval arithmetic has complexity $\mathcal{O}(n^3)$. Finally, the conversion of the resulting Taylor model to a SPZs has complexity $\mathcal{O}(1)$ according to Prop. 3.1.13.

During Step 5, we enclose the set $\mathcal{M}$ resulting from the nonlinear mapping of the reset function by $\mathcal{M} \subseteq r_i(x^*) \oplus (A \otimes (\mathcal{PZ} \oplus (-x^*))) \boxplus 1/2\ sq(\mathcal{Q}, \mathcal{PZ} \oplus (-x^*)) \oplus 1/6\ poly(\boldsymbol{\mathcal{D}}_r, \mathcal{I}_r \oplus (-x^*))$ with $\mathcal{I}_r = \texttt{interval}(\mathcal{PZ})$ and $\boldsymbol{\mathcal{D}}_r = \texttt{bound}(\nabla^3 r_i(x), \mathcal{I}_r)$. Computation of the linear map $A \otimes (\mathcal{PZ} \oplus (-x^*))$ has complexity $\mathcal{O}(n^3)$ according to Prop. 3.1.18, computation of the quadratic map $sq(\mathcal{Q}, \mathcal{PZ} \oplus (-x^*))$ has complexity $\mathcal{O}(n^3(n + \log(n))) = \mathcal{O}(n^4)$ according to Prop. 3.1.31, the exact addition $\boxplus$ has complexity $\mathcal{O}(n^2 \log(n))$ according to Prop. 3.1.20, the Minkowski sums have complexity $\mathcal{O}(n)$ according to Prop. 3.1.19, and the interval enclosure $\texttt{interval}(\mathcal{PZ})$ has complexity $\mathcal{O}(n^3)$ according to Tab. 3.3 if interval arithmetic is used for range bounding. Moreover, since $\nabla^3 r_i(x)$ consists of n^4 scalar functions, construction of $\boldsymbol{\mathcal{D}}_r$ has complexity $\mathcal{O}(en^4)$ according to Tab. 2.4 if interval arithmetic is used for range bounding. In addition, computation of the cubic map $1/6\ poly(\boldsymbol{\mathcal{D}}_r, \mathcal{I}_r \oplus (-x^*))$ as defined in (2.10) using interval arithmetic has complexity $\mathcal{O}(n^4)$.

Finally, in Step 6, we unite parallel sets with the linear combination for SPZs using Prop. 3.1.26, which has complexity $\mathcal{O}(n^2)$ according to Prop. 3.1.26. Adding the complexities for the single steps yields an overall complexity of

$$\underbrace{\mathcal{O}(n^5) + \mathcal{O}(n^3) + o_k \cdot \mathcal{O}(e)}_{\text{Step 1}} + \underbrace{\mathcal{O}(\texttt{contract})}_{\text{Step 2}} + \underbrace{\mathcal{O}(n)}_{\text{Step 3}} + \underbrace{\mathcal{O}(en^3) + \mathcal{O}(n^3) + \mathcal{O}(1)}_{\text{Step 4}}$$
$$+ \underbrace{\mathcal{O}(n^3) + \mathcal{O}(n^4) + \mathcal{O}(n^2 \log(n)) + \mathcal{O}(n) + \mathcal{O}(n^3) + \mathcal{O}(en^4) + \mathcal{O}(n^4)}_{\text{Step 5}} + \underbrace{\mathcal{O}(n^2)}_{\text{Step 6}},$$

which is $\mathcal{O}(n^5) + \mathcal{O}(\texttt{contract})$ using Assumption 4.4.13. Under consideration of Assumption 4.4.13, the most expensive contractor has complexity $\mathcal{O}(\texttt{contract}) = \mathcal{O}(n^{4.5})$ according to Tab. 2.5, so that contraction does not increase the computational complexity. In addition, also the check for preventing endless loops described in Sec. 4.4.4 has a complexity of less than $\mathcal{O}(n^5)$ and therefore does not increase the computational complexity either. Consequently, we obtain an overall complexity of $\mathcal{O}(n^5)$, which is dominated by the complexity of reachability analysis for the continuous dynamics and the enclosure of the Lagrange remainder for the Taylor series of the reset function.

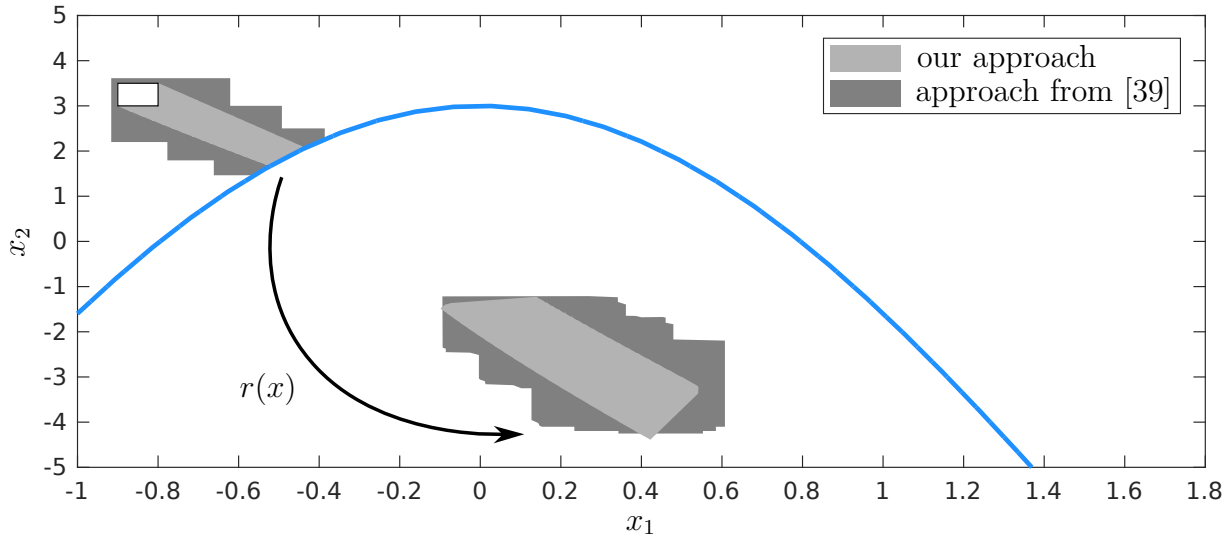

Figure 4.24: Comparison of the reachable set for the artificial hybrid system calculated with our approach and the approach in [39], where the initial set is depicted in white with a black border and the guard set is depicted in blue. The results for the approach from [39] are taken from [39, Fig. 3].

4.4.6 Numerical Examples

Finally, we demonstrate the performance of our novel approach on several benchmark systems. For all benchmarks we use interval arithmetic for intersection detection in Step 1 of the procedure in Sec. 4.4.2 and apply the forward-backward contractor for the contraction in Step 2. Moreover, we use a Taylor order of $\kappa = 2$ for computation of the guard intersection and a Taylor order of $\kappa_r = 1$ for computation of the reset mapping. Unless explicitly stated otherwise, we do not use an upper bound μ for the maximum number of intervals that are united in Step 3.

Artificial Hybrid System

First, we consider the 2-dimensional artificial hybrid system from [39, Sec. 6.1]. The system has one nonlinear guard set

$$\mathcal{G} = \left\{ x \in \mathbb{R}^2 \;\middle|\; \cos(x_{(1)}) - 0.1\, x_{(2)} - 0.7 = 0 \right\} \tag{4.75}$$

and an uncertain reset function

$$r(x) = \begin{bmatrix} -x_{(1)} \\ \nu\, x_{(2)} \end{bmatrix}, \quad \nu \in [-2.05, -2].$$

To implement the uncertain reset function, we introduce an auxiliary state $x_{(3)} \in [-2.05, -2]$ with dynamics $\dot{x}_{(3)} = 0$. We compare our novel method with the approach from [39]. Since the equality constraint that defines the guard set in (4.75) is symbolically solvable for one variable, we calculate the intersection with the guard set according to Prop. 4.4.8. The visualization of the reachable set in Fig. 4.24 shows that the reachable

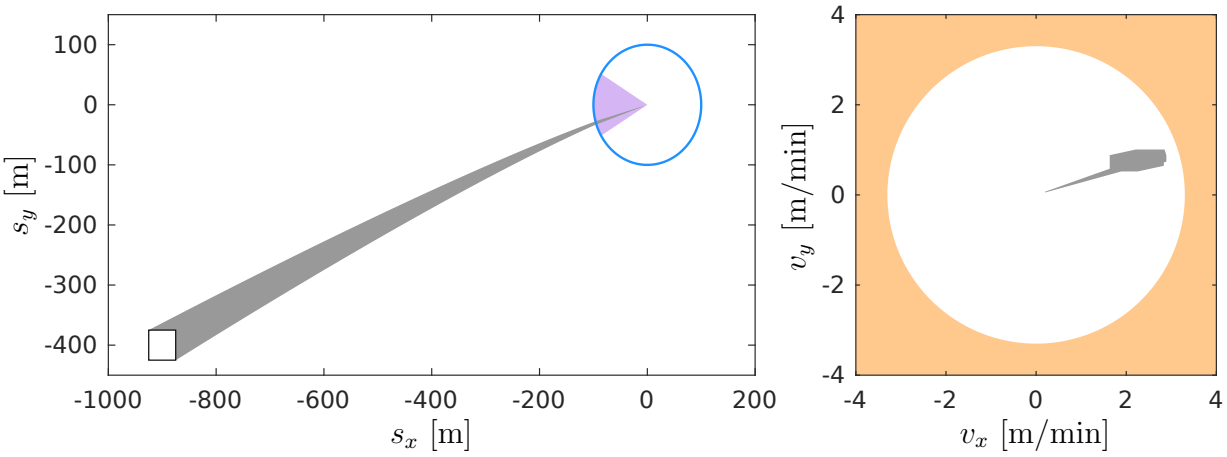

Figure 4.25: Reachable set for the spacecraft rendezvous benchmark (gray), where the initial set is depicted in white with a black border, the guard set is depicted in blue, the line-of-sight cone is depicted in purple, and the unsafe set defined by the velocity constraint is depicted in orange.

set computed with our method is much tighter. In addition, the computation time for our method is only 0.87 seconds, while the computation time for the approach in [39] is 26 seconds on their machine.

Spacecraft Rendezvous

Next, we again examine the spacecraft docking maneuver, which we already considered previously in Sec. 4.1.5. This time, however, we use the original formulation of the benchmark in [151], which contains a nonlinear guard set

$$\mathcal{G} = \left\{ [s_x \ s_y \ v_x \ v_y]^T \in \mathbb{R}^4 \;\middle|\; s_x^2 + s_y^2 - (100\text{m})^2 = 0 \right\}, \tag{4.76}$$

where the four system states are the planar positions s_x, s_y and corresponding velocities v_x, v_y of the spacecraft. Unlike in the previously considered version of the benchmark, we now also only have the two discrete modes *approaching* and *rendezvous attempt*. The system starts in mode *approaching* from the initial set $s_x \in [-925, -875]$m, $s_y \in [-425, -375]$m, $v_x = 0$m/min, and $v_y = 0$m/min. If the spacecraft is at a distance of 100m from the space station, the system transitions into mode *rendezvous attempt* where a different controller is applied. This transition is modeled by the guard set $\mathcal{G}$ in (4.76). As in Sec. 4.1.5, the specifications for the benchmark are that in mode *rendezvous attempt* the spacecraft stays inside the line-of-sight cone and the absolute velocity stays below 3.3m/min. The considered time horizon is $t_f = 200$min.

We compare our method to the approach in [31], which is implemented in CORA [1]. To handle discrete transitions, this approach first computes the intersection geometrically and then encloses the union of all partial intersections with bundles of parallelotopes. Since [31] is only applicable for guard sets represented by polytopes or hyperplanes, we enclose the nonlinear guard set in (4.76) with 10 equally sized polytopes. Because the equality constraint defining the guard set in (4.76) is not symbolically solvable for one variable, we

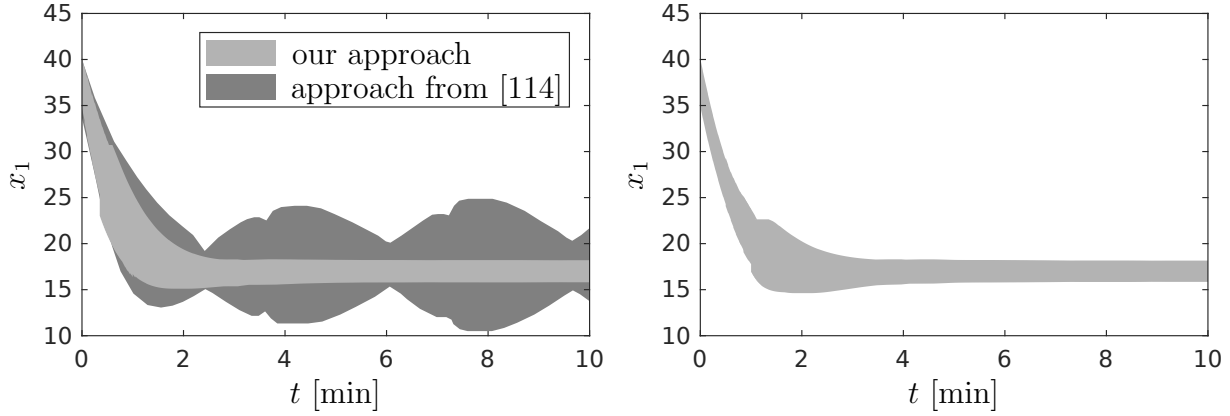

Figure 4.26: Comparison of the reachable set for a transcriptional regulator network with $N = 6$ genes (left) and $N = 12$ genes (right) calculated with our approach and the approach in [114]. The results for the approach in [114] are taken from [114, Fig. 11].

calculate the intersection with the guard set according to Prop. 4.4.10. Fig. 4.25 visualizes the results from our approach. The reachable sets computed with our approach and the approach in [31] both have similar accuracy and both satisfy the specifications. However, computing the guard intersection with the approach in [31] takes 4.96 seconds, whereas the computation of the guard intersection with our approach only takes 0.93 seconds. Moreover, for high-dimensional systems the computation time for enclosing nonlinear guards by polytopes might already be very large.

Transcriptional Regulator Network

The last benchmark is a transcriptional regulator network with N genes and system dimension $n = 2N$ [114, Sec. VIII.D] that we already examined in Sec. 4.1.5. In contrast to the purely continuous version considered previously, we now use the version with the artificial guard set

$$\mathcal{G} = \left\{ x \in \mathbb{R}^{2N} \;\middle|\; \sum_{i=1}^{2N} (x_{(i)} - 5)^5 - 75^2 = 0 \right\} \tag{4.77}$$

in [114, Sec. VIII.D]. We investigate the cases with $N = 6$ and $N = 12$ genes, which correspond to $n = 12$ and $n = 24$ system states, respectively. For the calculation of the reachable set with our novel approach, we use an upper bound of $\mu = 5$ for the case with $N = 6$ genes as well as an upper bound $\mu = 7$ for the case with $N = 12$ genes. Since the equality constraint that defines the guard set in (4.77) is not symbolically solvable for one variable, we calculate the intersection with the guard set according to Prop. 4.4.10.

We compare our method with the approach from [114]. The visualization of the results in Fig. 4.26 shows that the reachable set computed with our approach is much tighter for the case with $N = 6$ genes. In addition, the computation time for our approach is only 9.2 seconds, while the computation time for the approach in [114] is 130 seconds on their machine (see [114, Tab. IV]). For the case with $N = 12$ genes, the approach from [114] is not applicable due to a memory overflow (see [114, Tab. IV]). However, with our novel approach we can calculate the reachable set in only 35.6 seconds (see Fig. 4.26).

4.5 Summary

In this chapter we introduced novel approaches for reachability analysis of nonlinear continuous and hybrid systems, all of which use sparse polynomial zonotopes to represent reachable sets. Moreover, all approaches we presented only have polynomial complexity with respect to the system dimension and are therefore well suited for the verification of high-dimensional cyber-physical systems.

For computing outer-approximations of reachable sets for nonlinear continuous systems using the conservative polynomialization algorithm, significant improvements can be achieved with sparse polynomial zonotopes. In particular, sparse polynomial zonotopes store sets very compactly, they preserve dependencies, and they are closed under all set operations required for the conservative polynomialization algorithm. Consequently, with sparse polynomial zonotopes one can improve both, the computation time and the accuracy of the algorithm, compared to zonotopes and the non-sparse representation of polynomial zonotopes in [47], as we demonstrated on several numerical examples.

In addition, by utilizing dependency preservation of sparse polynomial zonotopes, we proved that reachability analysis using the conservative polynomialization algorithm preserves dependencies between the initial states and the reachable states. Based on this fundamental result, we presented a novel method for the efficient extraction of reachable subsets. As we demonstrated on multiple numerical examples, this novel reachable subset approach results in significant speed-ups for many applications, such as safety falsification, optimization over reachable sets, and motion-primitive-based control.

While outer-approximations of reachable sets are mainly used to verify that specifications are satisfied, inner-approximation are useful to prove that specifications are violated. Based on the conservative polynomialization algorithm and our reachable subset approach, we introduced a novel method for computing non-convex inner-approximations of reachable sets for nonlinear continuous systems. Since we extract inner-approximations directly from precomputed outer-approximations of reachable sets, our method is computationally very efficient, as we showed with several numerical examples.

For reachability analysis of hybrid systems, the main challenge is to handle the discrete transitions between different modes, since this requires the computation of intersections between the reachable set and the guard sets. We developed a novel approach for the very general case of hybrid systems with guard sets represented by nonlinear level sets, which tightly encloses these intersections by sparse polynomial zonotopes. Since our approach unifies reachable sets before computing the intersections, the computational demanding propagation of parallel sets can be avoided. Moreover, we additionally presented a novel strategy for preventing endless loops during reachability analysis, which is a common problem for many reachability algorithms for hybrid system. Performance evaluation on multiple benchmarks demonstrated that our novel approach achieves very good results with regard to accuracy and computation time compared to other state of the art methods.

Chapter 5

Selected Applications

While in Sec. 4 we showed in detail how the advantageous properties of SPZs can be exploited to improve reachability analysis of nonlinear continuous and hybrid systems, we have so far not considered any applications for CPZs and the Z-representation of polytopes. For this reason, we present in this section several applications for these two novel set representations.

5.1 Applications for Constrained Polynomial Zonotopes

We begin with applications for CPZs, where we consider set-based observers and program verification using inductive invariants in particular.

5.1.1 Set-Based Observers

For many systems it is not possible to measure all system states. In order to control such systems one therefore often uses an observer that estimates the values for the non-measurable states. While estimates are sufficient for systems that are not safety-critical, for safety-critical systems one requires a set-based observer which returns a set that is guaranteed to contain the actual system state rather than an estimate. In this section, we introduce a novel set-based observer for nonlinear discrete-time systems, which is based on CPZs.

Let us first briefly review the current state of the art. Most approaches for set-based observers consider linear discrete-time systems. These observers can be extended to handle nonlinear systems by applying conservative linearization [152, Sec. II.D]. Set-based observers can be grouped into three main categories: intersection-based observers, set-propagation observers, and interval observers. Intersection-based observers [26, 32, 36, 153] consist of a prediction step and a correction step, where in the prediction step reachability analysis is used to compute the set of possible states based on an uncertain system model. Next, in the correction step, this set is intersected with a strip defined by the current uncertain measurement. Set-propagation observers [27, 154], on the other hand, avoid intersections by combining the concept of a Luenberger observer [155] with reachability analysis. Finally, interval observers [156–158] enclose the set of possible states with an

interval, where two different observers are used to estimate the upper bound and the lower bound of the interval. A comparison of different set-based observers is provided in [159].

The novel observer we introduce here is an intersection-based observer. CPZs are well suited for this kind of observers since they are closed under polynomial maps and intersections, so that both, the propagation and the correction step can be calculated with high accuracy. We consider general nonlinear discrete-time systems defined as

$$\begin{aligned} x_{j+1} &= f(x_j, u) \\ y_j &= h(x_j) + v, \end{aligned} \tag{5.1}$$

where $x_j \in \mathbb{R}^n$ and $y_j \in \mathbb{R}^a$ are the vectors of system states and system outputs at time step $j \in \mathbb{N}_0$, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the vector of uncertain inputs, and $v \in \mathcal{V} \subset \mathbb{R}^a$ is the vector of measurement errors. While many observers explicitly distinguish between known inputs and uncertain inputs, we here consider the case with uncertain inputs only for simplicity since including known inputs is straightforward. Both, the dynamic function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and the measurement function $h : \mathbb{R}^n \to \mathbb{R}^a$ in (5.1) are Lipschitz continuous. Moreover, we consider the case where the set of uncertain inputs is a zonotope $\mathcal{U} = \langle c_z, G_z \rangle_Z$ and the set of measurement errors is a symmetric interval centered at the origin $\mathcal{V} = [-d, d]$. Given the system in (5.1), an initial set $\mathcal{X}_0$ containing the initial state $x_0 \in \mathcal{X}_0$, and a sequence of measurements $\widehat{y}_j$, the goal of a set-based observer is to compute a sequence of sets $\mathcal{X}_j$ that is consistent with the system dynamics and the measurements and as tight as possible. To achieve this, intersection-based observers compute in each time step j the set

$$\mathcal{X}_{j+1} = \big\{ f(x_j, u) \;\big|\; x_j \in \mathcal{X}_j,\ u \in \mathcal{U} \big\} \cap \big\{ x \in \mathbb{R}^n \;\big|\; \widehat{y}_{j+1} - h(x) \in \mathcal{V} \big\}, \tag{5.2}$$

where the nonlinear map $\{f(x_j, u) \mid x_j \in \mathcal{X}_j,\ u \in \mathcal{U}\}$ corresponds to the prediction step and the intersection with $\{x \in \mathbb{R}^n \mid \widehat{y}_{j+1} - h(x) \in \mathcal{V}\}$ corresponds to the correction step. Since for general nonlinear systems it is not possible to compute the set in (5.2) exactly, we compute a tight enclosure instead using CPZs.

Let us first consider the prediction step, where we use $z_j = [x_j^T\ u^T]^T$ to concisely formulate the system dynamics as $x_{j+1} = f(z_j)$. Enclosing the dynamic function with a Taylor series expansion of order 2 at the expansion point $z^* = [c^T\ c_z^T]$ with $c \in \mathbb{R}^n$ representing the constant offset of the CPZ $\mathcal{X}_j$ yields

$$\begin{aligned} f_{(i)}(z_j) \in f_{(i)}(z^*) &+ \underbrace{\left.\frac{\partial f_{(i)}(z_j)}{\partial z_j}\right|_{z^*}}_{A_{(i,\cdot)}} (z_j - z^*) \\ &+ \frac{1}{2}(z_j - z^*)^T \underbrace{\left.\frac{\partial^2 f_{(i)}(z_j)}{\partial z_j^2}\right|_{z^*}}_{Q_i} (z_j - z^*) \oplus \mathcal{L}_{(i)}(z_j), \quad i = 1, \ldots, n, \end{aligned} \tag{5.3}$$

where the Lagrange remainder $\mathcal{L}(z_j) = \mathcal{L}_{(1)}(z_j) \times \ldots \times \mathcal{L}_{(n)}(z_j)$ is enclosed by

$$\forall z_j \in \mathcal{Z}_j : \quad \mathcal{L}(z_j) =$$

$$\left\{ s \;\middle|\; s_{(i)} = \frac{\left((z_j - z^*)^T \nabla\right)^3 f_{(i)}(\widehat{z})}{3!},\ \widehat{z} = z^* + \lambda(z_j - z^*),\ \lambda \in [0,1] \right\} \overset{\mathcal{Z}_j \subseteq \mathcal{I}}{\subseteq} \tag{5.4}$$

$$\frac{1}{6}\left\{ s \;\middle|\; s_{(i)} = \left((z_j - z^*)^T \nabla\right)^3 f_{(i)}(\widehat{z}),\ z_j, \widehat{z} \in \mathcal{I} \right\} \overset{(2.10)}{\subseteq} \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*)\big)$$

with

$$\mathcal{Z}_j = \mathcal{X}_j \times \mathcal{U}, \quad \mathcal{I} = \texttt{interval}(\mathcal{Z}_j), \quad \boldsymbol{\mathcal{D}} = \texttt{bound}\big(\nabla^3 f(z_j), \mathcal{I}\big).$$

Using (5.3) and (5.4), we can calculate a tight enclosure of the nonlinear map from the prediction step as follows:

$$\big\{ f(x_j, u) \mid x_j \in \mathcal{X}_j,\ u \in \mathcal{U} \big\} = \big\{ f(z_j) \mid z_j \in \mathcal{Z}_j \big\} =$$

$$\overset{(5.3)}{\subseteq} \left\{ f(z^*) + A(z_j - z^*) + \frac{1}{2} \begin{bmatrix} (z_j - z^*)^T Q_1 (z_j - z^*) \\ \vdots \\ (z_j - z^*)^T Q_n (z_j - z^*) \end{bmatrix} \oplus \mathcal{L}(z_j) \;\middle|\; z_j \in \mathcal{Z}_j \right\} \tag{5.5}$$

$$\overset{(5.4)}{\subseteq} f(z^*) \oplus \big(A \otimes (\mathcal{Z}_j \oplus (-z^*))\big) \boxplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}_j \oplus (-z^*)\big) \oplus \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*)\big)$$

with $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$. The Cartesian product $\mathcal{Z}_j = \mathcal{X}_j \times \mathcal{U}$ is computed using Prop. 3.2.19, where we first convert the zonotope $\mathcal{U}$ to a CPZ as described in Sec. 3.2.3. Moreover, the linear map $A \otimes (\mathcal{Z}_j \oplus (-z^*))$ is computed using Prop. 3.2.16, the quadratic map $sq(\mathcal{Q}, \mathcal{Z}_j \oplus (-z^*))$ is computed using Prop. 3.2.22, the exact addition $\boxplus$ is computed using Prop. 3.2.18, Minkowski sums are calculated using Prop. 3.2.17, the interval enclosure $\mathcal{I} = \texttt{interval}(\mathcal{Z}_j)$ is calculated as described in Sec. 3.2.4, and $\boldsymbol{\mathcal{D}} = \texttt{bound}(\nabla^3 f(z_j), \mathcal{I})$ is calculated using range bounding as described in Sec. 2.7.

Next, we consider the correction step. Since the set of measurement errors $\mathcal{V}$ is an interval, we can equivalently represent the set $\{x \in \mathbb{R}^n \mid \widehat{y}_{j+1} - h(x) \in \mathcal{V}\}$ as a level set:

$$\big\{ x \in \mathbb{R}^n \;\big|\; \widehat{y}_{j+1} - h(x) \in \mathcal{V} \big\} \overset{\mathcal{V} = [-d, d]}{=} \big\{ x \in \mathbb{R}^n \;\big|\; -d \leq \widehat{y}_{j+1} - h(x) \leq d \big\} =$$

$$\big\{ x \in \mathbb{R}^n \;\big|\; (\widehat{y}_{j+1} - h(x) - d \leq \mathbf{0}) \wedge (-\widehat{y}_{j+1} + h(x) - d \leq \mathbf{0}) \big\} \tag{5.6}$$

$$\overset{\text{Def. } 2.2.9}{=} \left\langle \begin{bmatrix} \widehat{y}_{j+1} - h(x) - d \\ -\widehat{y}_{j+1} + h(x) - d \end{bmatrix}, \leq \right\rangle_{LS}.$$

The intersection required in the correction step can therefore simply be implemented by applying Prop. 3.2.24. By combining the prediction step in (5.5) and the correction step

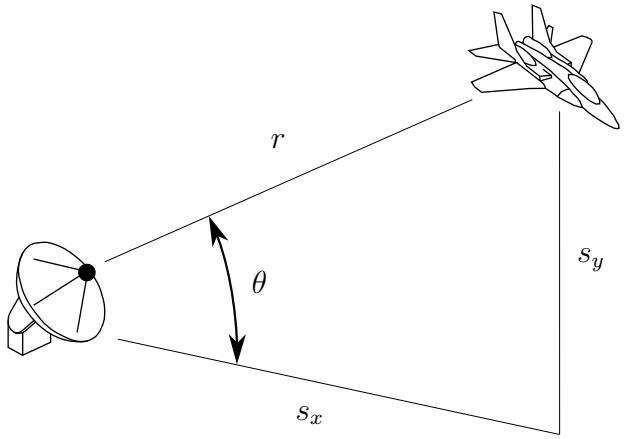

Figure 5.1: Visualization of the airplane tracking benchmark from Example 5.1.1.

in (5.6), we finally obtain for (5.2):

$$\begin{aligned}\mathcal{X}_{j+1} \subseteq \Big(f(z^*) \oplus \big(A \otimes (\mathcal{Z}_j \oplus (-z^*))\big) \boxplus \frac{1}{2} sq\big(\mathcal{Q}, \mathcal{Z}_j \oplus (-z^*)\big) \\ \oplus \frac{1}{6} poly\big(\boldsymbol{\mathcal{D}}, \mathcal{I} \oplus (-z^*)\big)\Big) \cap \left\langle \begin{bmatrix} \widehat{y}_{j+1} - h(x) - d \\ -\widehat{y}_{j+1} + h(x) - d \end{bmatrix}, \leq \right\rangle_{LS}.\end{aligned} \tag{5.7}$$

According to Tab. 3.9, many of the set operations on CPZs required for the observer in (5.7) increase the representation size. To keep the computation time small, we therefore additionally have to perform constraint reduction and order reduction using Prop. 3.2.29 and Prop. 3.2.31 in each time step. It is also straightforward to extend the observer in (5.7) to nonlinear continuous-time systems, since we can simply use the conservative polynomialization algorithm in Alg. 6 for the prediction step. Let us finally demonstrate our CPZ-based observer with an example:

Example 5.1.1. *We consider the system visualized in Fig. 5.1, which consists of a radar dish that aims to determine the position of an airplane. Since the radar dish automatically adjusts its pose to follow the signal it receives from the airplane, it is sufficient to consider the two-dimensional plane spanned by the radar dish and the airplane. We model the dynamic behavior of the airplane as*

$$\begin{aligned} v_{j+1} &= v_j + \Delta t\, u_1 \\ \phi_{j+1} &= \phi_j + \Delta t\, u_2 \\ s_{x,j+1} &= s_{x,j} + \Delta t\, v_j\, \cos(\phi_j) \\ s_{y,j+1} &= s_{y,j} + \Delta t\, v_j\, \sin(\phi_j), \end{aligned} \tag{5.8}$$

where the system state $x_j = [v_j\ \phi_j\ s_{x,j}\ s_{y,j}]^T$ consists of the airplane velocity v_j, the airplane orientation ϕ_j, and the x and y positions of the airplane's center of mass $s_{x,j}$ and $s_{y,j}$. The uncertain inputs $u = [u_1\ u_2]^T$ are bounded by the set $\mathcal{U} = [-0.5, 0.5]\mathrm{m/s}^2 \times [-0.02, 0.02]\mathrm{rad/s}$. The radar dish cannot measure the position of the airplane directly, but instead measures the distance r_j and the bearing θ_j. For the measurement function, we

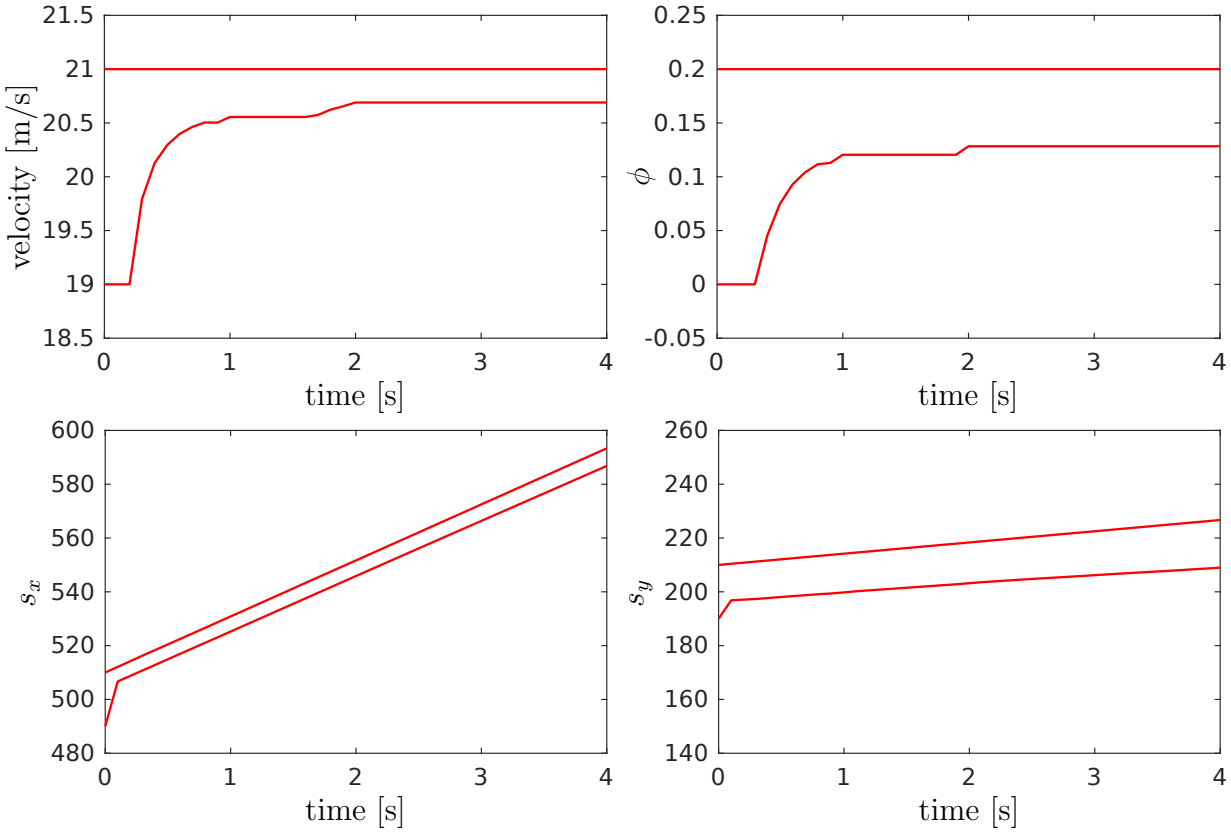

Figure 5.2: Bounds for the system states of the airplane from Example 5.1.1 over time computed with the CPZ-based observer in (5.7).

therefore obtain

$$r_j = \sqrt{s_{x,j}^2 + s_{y,j}^2} + v_1$$
$$\theta_j = \tan^{-1}\left(\frac{s_{y,j}}{s_{x,j}}\right) + v_2,$$

where $y_j = [r_j \ \theta_j]^T$ *is the vector of system outputs. The measurement error* $v = [v_1 \ v_2]^T$ *is bounded by the set* $\mathcal{V} = [-0.1, 0.1]\text{m} \times [-0.05, 0.05]\text{rad}$. *Moreover, we consider the initial set* $\mathcal{X}_0 = [19, 21]\text{m/s} \times [0, 0.2]\text{rad} \times [490, 510]\text{m} \times [190, 210]\text{m}$ *and we run the observer with a sampling rate of* $\Delta t = 0.1\text{s}$. *To obtain a sequence of measurements* $\widehat{y}_j$, *we simulate the dynamic system in* (5.8) *for a random initial state using randomly sampled disturbances* u *and measurement errors* v. *The results shown in Fig. 5.2 demonstrate that our CPZ-based observer successfully reduces the initial uncertainty for most system states. The computation time for a time horizon of* 4 *seconds is only* 3.346 *seconds, so that the observer actually runs in real-time.*

5.1.2 Program Verification using Inductive Invariants

Up to now we focused on the verification of the physical behavior of cyber-physical systems. However, nowadays this kind of systems are often controlled by large and complicated computer programs. To guarantee safety, it is therefore crucial to also verify the correctness of the software. One common approach that is often used for verification of computer programs are inductive invariants. Given a program and a set of possible initial values for the program inputs, an inductive invariant is a set containing all possible values that the program variables may take during program execution. The limits on the variables defined by the inductive invariant can for example be used to prove safety, verify that the program does not run into singular points like, e.g., division by zero, or deduce bounds for the maximum occurring rounding errors. In this section we demonstrate that CPZs are well suited for the computation of tight inductive invariants.

Let us first provide an overview of the current state of the art. A common technique to obtain an inductive invariant is to compute the reachable set of the computer program with set-based propagation [29,34,48], where [29] uses zonotopes, [34] uses zonotopes with polytopic domains, and [48] uses quadratic zonotopes to represent the invariant. Other approaches compute inductive invariants represented as polynomial level sets through optimization [56,57,160], where [56] applies Sum-of-Squares techniques, [57] uses polynomial optimization problems, and [160] exploits a Gröbner basis to reduce invariant computation to a constraint satisfaction problem. Yet another method is to compute inductive invariants represented by polynomial level sets based on counterexamples [161–163].

With CPZs, we can compute tight inductive invariants using set-based propagation. As an example we consider the following MATLAB function:

```
function [x,y] = program(x,y)
   for i = 1:3
      if y >= x^2
         x = x - 0.5*y;
         y = 1.2*y;
      else
         x = x*y;
         y = -0.8*y^2;
      end
   end
end
```
(5.9)

The inputs `x` and `y` to the function are bounded by $\texttt{x} \in [0,1]$ and $\texttt{y} \in [0,1]$. To compute an inductive invariant for the function in (5.9), we require the set operations linear map for the assignments `x = x - 0.5*y` and `y = 1.2*y`, quadratic map for the assignments `x = x*y` and `y = -0.8*y^2`, intersection for evaluating the condition of the if-statement, and union to unite the sets for the two branches of the if-else-statement as well as the sets for the 3 iterations of the for-loop. Since CPZs are closed under all these operations, we can compute the exact inductive invariant of the function in (5.9). For this, let us denote the set at the end of the i-th iteration of the for-loop by $\mathcal{X}^{(i)}$, where $\mathcal{X}^{(0)} = [0,1] \times [0,1]$ is

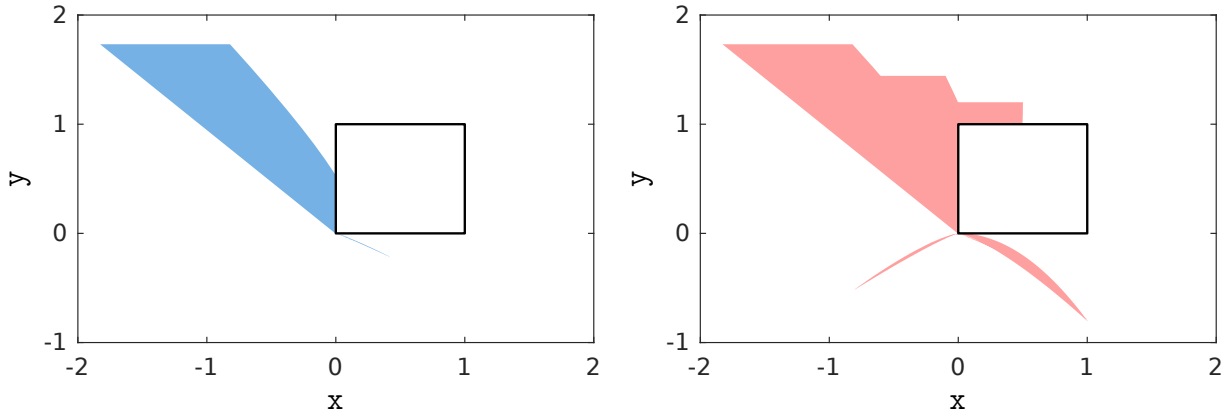

Figure 5.3: Visualization of the final set $\mathcal{X}^{(3)}$ (left) and the inductive invariant $\mathcal{INV}$ (right) for the computer program in (5.9), where the input set is shown in white with a black border.

the input set. Then the values that satisfy the condition `y >= x^2` of the if-statement are given by the set $\mathcal{X}^{(i)} \cap \mathcal{LS}_1$ and the values that take the else-branch are given by the set $\mathcal{X}^{(i)} \cap \mathcal{LS}_2$, where the level sets $\mathcal{LS}_1$ and $\mathcal{LS}_2$ are defined as

$$\mathcal{LS}_1 = \langle x_{(1)}^2 - x_{(2)}, \leq \rangle_{LS}, \quad \mathcal{LS}_2 = \langle -x_{(1)}^2 + x_{(2)}, \leq \rangle_{LS}.$$

Consequently, the set $\mathcal{X}^{(i+1)}$ resulting from the propagation of the set $\mathcal{X}^{(i)}$ through the body of the for-loop can be computed as

$$\mathcal{X}^{(i+1)} = \big(A \otimes (\mathcal{X}^{(i)} \cap \mathcal{LS}_1)\big) \cup sq(\mathcal{Q}, \mathcal{X}^{(i)} \cap \mathcal{LS}_2) \tag{5.10}$$

with

$$A = \begin{bmatrix} 1 & -0.5 \\ 0 & 1.2 \end{bmatrix}, \quad \mathcal{Q} = \{Q_1, Q_2\}, \quad Q_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 0 & 0 \\ 0 & -0.8 \end{bmatrix},$$

where the intersections with the level sets are calculated using Prop. 3.2.24, the linear map is calculated using Prop. 3.2.16, the quadratic map is calculated using Prop. 3.2.22, and the union is calculated using Prop. 3.2.25. After repeating the set-propagation in (5.10) for all 3 iterations of the for-loop, the inductive invariant can be computed as

$$\mathcal{INV} = \mathcal{X}^{(0)} \cup \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \mathcal{X}^{(3)}.$$

The overall computation of the inductive invariant using CPZs takes 5.56 seconds and the results are visualized in Fig. 5.3. The CPZ-based approach presented here can be easily extended to programs with infinite loops by using the containment check for CPZs described in Sec. 3.2.6.

5.2 Applications for the Z-Representation of Polytopes

For the Z-representation of polytopes, we discuss the applications *range bounding on polytopic domains* and *generalized barycentric coordinates* in detail.

5.2.1 Range Bounding on Polytopic Domains

In this section we demonstrate how the Z-representation of polytopes can be utilized to improve the results for range bounding on polytopic domains. Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a domain $\mathcal{D} \subset \mathbb{R}^n$, the range bounding operation as defined in Def. 2.7.1 aims to determine tight bounds of the function values on the domain:

$$\texttt{bound}(f(x), \mathcal{D}) \supseteq \left[\min_{x\in\mathcal{D}} f(x),\ \max_{x\in\mathcal{D}} f(x)\right]. \tag{5.11}$$

Clearly, in order to use interval arithmetic for range bounding, it is required that the domain $\mathcal{D}$ is an interval. If this is not the case one can enclose $\mathcal{D}$ by an interval, which however potentially results in very conservative bounds due to the over-approximation. Similarly, for affine arithmetic it is required that the domain $\mathcal{D}$ is an affine object, which is equivalent to a zonotope. For range bounding using Taylor models, $\mathcal{D}$ has to be represented as a Taylor model.

In this section, we consider the range bounding problem $\texttt{bound}(f(x), \mathcal{P})$, where the domain $\mathcal{P} = \langle c, G, \mathbf{E}\rangle_Z$ is a polytope in Z-representation. Using the definition of the Z-representation in Def. 3.3.1

$$\mathcal{P} = \langle c, G, \mathbf{E}\rangle_Z \overset{\text{Def. 3.3.1}}{=} \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha_{\mathbf{E}_{(i,k)}} \in [-1, 1] \right\}, \tag{5.12}$$

we have

$$\max_{x\in\mathcal{P}} f(x) \overset{(5.12)}{=} \max_{\alpha\in[-\mathbf{1},\mathbf{1}]} \underbrace{f\left(c + \sum_{i=1}^{h} \left(\prod_{k=1}^{m_i} \alpha_{\mathbf{E}_{(i,k)}} \right) G_{(\cdot,i)} \right)}_{\widehat{f}(\alpha)},$$

where $\alpha = [\alpha_1\ \ldots\ \alpha_p]^T$. Since the same relation holds for the minimum, we obtain

$$\texttt{bound}(f(x), \mathcal{P}) = \texttt{bound}\big(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}]\big) \tag{5.13}$$

according to (5.11). Using the Z-representation of polytopes, any range bounding problem $\texttt{bound}(f(x), \mathcal{P})$ with a polytopic domain $\mathcal{P}$ can consequently be reformulated as an equivalent range bounding problem $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}])$ with an interval domain $[-\mathbf{1}, \mathbf{1}]$. The reformulation therefore avoids the enclosure of the polytopic domain with an interval or an affine object, and consequently often results in significantly tighter bounds as we demonstrate with an example:

Example 5.2.1. *We consider the range bounding problem* $\texttt{bound}(f(x),\mathcal{P})$ *for the function*

$$f(x) = x_{(1)}\big(3 - x_{(1)}\big) + x_{(2)}\big(2 - x_{(2)}\big) - 3.25 + 4\cos(x_{(1)})\sin(x_{(2)})$$

and the polytope in Z-representation

$$\mathcal{P} = \left\langle \begin{bmatrix} 0 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ -0.5 & 1.5 & -0.5 \end{bmatrix}, \left(1, 2, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \right\rangle_Z$$

$$= \left\{ \begin{bmatrix} 0 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} \alpha_{(1)} + \begin{bmatrix} 0 \\ 1.5 \end{bmatrix} \alpha_{(2)} + \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} \alpha_{(1)}\,\alpha_{(2)} \;\middle|\; \alpha \in [-\mathbf{1}, \mathbf{1}] \right\},$$

which are visualized in Fig. 5.4. According to (5.13)*, this can be equivalently formulated as the range bounding problem* $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}])$ *with*

$$\widehat{f}(\alpha) = \alpha_{(2)}^2\big(1.5\,\alpha_{(1)} - 1.25\,\alpha_{(1)}^2 - 2.25\big) - \alpha_{(1)}^2\big(2.5\,\alpha_{(2)} + 1.25\big) + \alpha_{(2)}\big(3\,\alpha_{(1)} + 4.5\big) +$$
$$1.5\,\alpha_{(1)} - 4.5 + 4\cos(\alpha_{(1)} + \alpha_{(1)}\alpha_{(2)})\sin(-0.5 - 0.5\,\alpha_{(1)} + 1.5\,\alpha_{(2)} - 0.5\,\alpha_{(1)}\alpha_{(2)}).$$

To use interval arithmetic for the original range bounding problem $\texttt{bound}(f(x),\mathcal{P})$*, we first have to enclose the polytope* $\mathcal{P}$ *by the interval* $\mathcal{P} \subseteq \mathcal{I} = [-2, 2] \times [-2, 2]$*. Consequently, we obtain the quite conservative bounds* $\texttt{bound}(f(x),\mathcal{I}) = [-25.25, 18.75]$*. On the other hand, for the equivalent range bounding problem* $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}])$ *we can apply interval arithmetic directly, so that we obtain the tighter bounds* $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}]) = [-26.25, 9.75]$*. By using Taylor models for range bounding, we even obtain* $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}]) = [-14.888, 1.410]$*, which is very close to the exact result* $[-14.887, 1.409]$*.*

If Taylor models are used for range bounding one can also simply convert the polytope $\mathcal{P}$ in Z-representation to a Taylor model using Prop. 3.3.12 and Prop. 3.1.13. This will give an identical result compared to using the reformulated range bounding problem $\texttt{bound}(\widehat{f}(\alpha), [-\mathbf{1}, \mathbf{1}])$ and is often easier to implement.

5.2.2 Generalized Barycentric Coordinates

Barycentric coordinates are originally defined for simplices, where the coordinates express each point inside a simplex as a linear combination of the simplex vertices. Generalized barycentric coordinates extend this concept to general polytopes: Given a polytope $\mathcal{P} = \langle [v_1 \;\ldots\; v_s] \rangle_V \subset \mathbb{R}^n$ and a point $x \in \mathcal{P}$ inside the polytope, the generalized barycentric coordinates $\theta_i(x)$, $i = 1, \ldots, s$ of the point must satisfy

$$x = \sum_{i=1}^{s} \theta_i(x)\, v_i, \quad \sum_{i=1}^{s} \theta_i(x) = 1, \quad \forall i \in \{1, \ldots, s\} : \; \theta_i(x) \geq 0. \tag{5.14}$$

For polytopes with more than $n + 1$ vertices, the generalized barycentric coordinates are not unique. One exemplary application for barycentric coordinates is control theory, where optimal controllers for the polytope vertices are constructed offline and barycentric coordinates are used during online application to obtain the controller for the measured state by

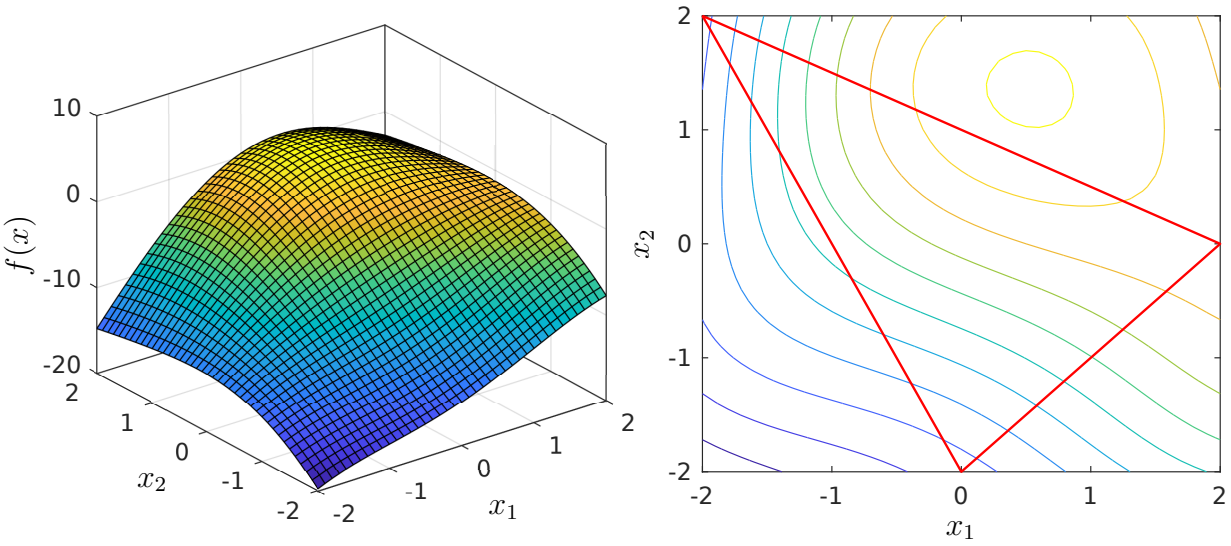

Figure 5.4: Visualization of the function $f(x)$ (left) and the polytope $\mathcal{P}$ (red, right) from Example 5.2.1, where the figure on the right additionally shows isolines of the function $f(x)$.

interpolation [127, 164, 165]. Given a point $x \in \mathcal{P}$, its generalized barycentric coordinates can be determined with linear programming. However, since for the above-mentioned control applications linear programming is often still too slow for real-time applicability, the approach in [124] derives a closed-form expression for generalized barycentric coordinates.

In this section we show that a closed-form expression for generalized barycentric coordinates $\theta_i(\alpha)$ with respect to the factors α of the Z-representation is obtained automatically during the conversion of a polytope in V-representation to Z-representation using Alg. 4. The Z-representation of the polytope can therefore be equivalently represented as

$$\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V = \left\{ \sum_{i=1}^{s} \theta_i(\alpha)\, v_i \;\middle|\; \alpha \in [-\mathbf{1}, \mathbf{1}] \right\}, \tag{5.15}$$

where the functions $\theta_i(\alpha)$ satisfy

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \sum_{i=1}^{s} \theta_i(\alpha) = 1 \quad \text{and} \quad \forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \theta_1(\alpha), \dots, \theta_s(\alpha) \geq 0.$$

For the conversion from V-representation to Z-representation, Alg. 4 first represents all polytope vertices in Z-representation and then successively unites all vertices using the convex hull. To prove that the resulting Z-representation of the polytope can be equivalently represented with generalized barycentric coordinates as in (5.15), we use mathematical induction:

Base case: Initially, all polytopes $\mathcal{P}_j = \langle v_j, [\,], \emptyset \rangle_Z$ with $j = 1, \dots, s$ considered by Alg. 4 represent a single polytope vertex v_j and can therefore be equivalently represented as in (5.15) with a single generalized barycentric coordinate $\theta(\alpha) = 1$:

$$\mathcal{P}_j = \{ v_j \} = \{ \theta(\alpha)\, v_j \mid \alpha \in [-\mathbf{1}, \mathbf{1}] \}.$$

Induction hypothesis: We assume that we have two polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$ that can be equivalently represented with generalized barycentric coordinates as in (5.15):

$$\begin{aligned}\mathcal{P}_1 &= \left\langle [v_{1,1} \ \dots \ v_{1,s_1}] \right\rangle_V = \left\{ \sum_{i=1}^{s_1} \theta_{1,i}(\alpha_1)\, v_{1,i} \;\middle|\; \alpha_1 \in [-\mathbf{1}, \mathbf{1}] \right\} \\ \mathcal{P}_2 &= \left\langle [v_{2,1} \ \dots \ v_{2,s_2}] \right\rangle_V = \left\{ \sum_{i=1}^{s_2} \theta_{2,i}(\alpha_2)\, v_{2,i} \;\middle|\; \alpha_2 \in [-\mathbf{1}, \mathbf{1}] \right\},\end{aligned} \tag{5.16}$$

where the functions $\theta_{1,i}(\alpha_1)$ and $\theta_{2,i}(\alpha_2)$ satisfy

$$\begin{aligned}&\forall \alpha_1 \in [-\mathbf{1}, \mathbf{1}]: \quad \sum_{i=1}^{s_1} \theta_{1,i}(\alpha_1) = 1, \quad \forall \alpha_1 \in [-\mathbf{1}, \mathbf{1}]: \quad \theta_{1,1}(\alpha_1), \dots, \theta_{1,s_1}(\alpha_1) \geq 0 \\ &\forall \alpha_2 \in [-\mathbf{1}, \mathbf{1}]: \quad \sum_{i=1}^{s_2} \theta_{2,i}(\alpha_2) = 1, \quad \forall \alpha_2 \in [-\mathbf{1}, \mathbf{1}]: \quad \theta_{2,1}(\alpha_2), \dots, \theta_{2,s_2}(\alpha_2) \geq 0.\end{aligned} \tag{5.17}$$

Induction step: Finally, we show that unification of the two polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$ from the induction hypothesis using the convex hull as done by Alg. 4 results in a polytope that can be equivalently represented with generalized barycentric coordinates as in (5.15):

$$conv(\mathcal{P}_1, \mathcal{P}_2) = comb(\mathcal{P}_1, \mathcal{P}_2) \overset{(2.11)}{=}$$

$$\left\{ \underbrace{\frac{1}{2}(1+\lambda)}_{\widehat{\theta}_1(\lambda)} p_1 + \underbrace{\frac{1}{2}(1-\lambda)}_{\widehat{\theta}_2(\lambda)} p_2 \;\middle|\; p_1 \in \mathcal{P}_1,\ p_2 \in \mathcal{P}_2,\ \lambda \in [-1, 1] \right\} \overset{(5.16)}{=}$$

$$\left\{ \widehat{\theta}_1(\lambda) \left(\sum_{i=1}^{s_1} \theta_{1,i}(\alpha_1)\, v_{1,i} \right) + \widehat{\theta}_2(\lambda) \left(\sum_{i=1}^{s_2} \theta_{2,i}(\alpha_2)\, v_{2,i} \right) \;\middle|\; \alpha_1, \alpha_2 \in [-\mathbf{1}, \mathbf{1}],\ \lambda \in [-1, 1] \right\} =$$

$$\left\{ \sum_{i=1}^{s_1+s_2} \theta_i(\alpha)\, v_i \;\middle|\; \alpha \in [-\mathbf{1}, \mathbf{1}] \right\},$$

where $\alpha = [\alpha_1^T \ \alpha_2^T \ \lambda]^T$ and

$$\theta_i(\alpha) = \begin{cases} \widehat{\theta}_1(\lambda)\, \theta_{1,i}(\alpha_1), & i \leq s_1 \\ \widehat{\theta}_2(\lambda)\, \theta_{2,i-s_1}(\alpha_2), & \text{otherwise} \end{cases}, \quad v_i = \begin{cases} v_{1,i}, & i \leq s_1 \\ v_{2,i-s_1}, & \text{otherwise} \end{cases}. \tag{5.18}$$

We exploited that the convex hull and the linear combination are identical since $\mathcal{P}_1$ and $\mathcal{P}_2$ are both convex sets. The resulting functions $\theta_i(\alpha)$ are valid generalized barycentric coordinates since they satisfy

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \sum_{i=1}^{s_1+s_2} \theta_i(\alpha) \overset{(5.18)}{=} \widehat{\theta}_1(\lambda) \underbrace{\left(\sum_{i=1}^{s_1} \theta_{1,i}(\alpha_1) \right)}_{\overset{(5.17)}{=} 1} + \widehat{\theta}_2(\lambda) \underbrace{\left(\sum_{i=1}^{s_2} \theta_{2,i}(\alpha_2) \right)}_{\overset{(5.17)}{=} 1}$$

$$= \widehat{\theta}_1(\lambda) + \widehat{\theta}_2(\lambda) = \frac{1}{2}(1+\lambda) + \frac{1}{2}(1-\lambda) = 1$$

and

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}],\ \forall i \in \{1, \dots, s_1\}: \quad \theta_i(\alpha) \overset{(5.18)}{=} \widehat{\theta}_1(\lambda)\, \theta_{1,i}(\alpha_1) = \underbrace{0.5(1+\lambda)}_{\geq 0} \underbrace{\theta_{1,i}(\alpha_1)}_{\overset{(5.17)}{\geq} 0} \geq 0,$$

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}],\ \forall i \in \{1, \dots, s_2\}: \quad \theta_{s_1+i}(\alpha) \overset{(5.18)}{=} \widehat{\theta}_2(\lambda)\, \theta_{2,i}(\alpha_2) = \underbrace{0.5(1-\lambda)}_{\geq 0} \underbrace{\theta_{2,i}(\alpha_2)}_{\overset{(5.17)}{\geq} 0} \geq 0,$$

which concludes the proof.

Let us demonstrate generalized barycentric coordinates for the Z-representation with an example:

Example 5.2.2. *We consider the polytope*

$$\mathcal{P} = \left\langle \begin{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} & \begin{bmatrix} -2 \\ 2 \end{bmatrix} & \begin{bmatrix} 0 \\ -2 \end{bmatrix} \end{bmatrix} \right\rangle_V$$

in V-representation. With the approach described in this section, $\mathcal{P}$ can be equivalently represented as

$$\mathcal{P} = \left\{ \underbrace{\frac{1}{4}\big(1+\alpha_{(1)}\big)\big(1+\alpha_{(2)}\big)}_{\theta_1(\alpha)} \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \underbrace{\frac{1}{4}\big(1-\alpha_{(1)}\big)\big(1+\alpha_{(2)}\big)}_{\theta_2(\alpha)} \begin{bmatrix} -2 \\ 2 \end{bmatrix} + \underbrace{\frac{1}{2}\big(1-\alpha_{(2)}\big)}_{\theta_3(\alpha)} \begin{bmatrix} 0 \\ -2 \end{bmatrix} \;\middle|\; \alpha \in [-\mathbf{1}, \mathbf{1}] \right\},$$

where we constructed the closed-form expressions for the generalized barycentric coordinates $\theta_1(\alpha)$, $\theta_2(\alpha)$, and $\theta_3(\alpha)$ during conversion of $\mathcal{P}$ to Z-representation using Alg. 4.

5.3 Summary

In this chapter we presented selected applications for constrained polynomial zonotopes and the Z-representation of polytopes, all of which are closely related to formal verification and safe control of cyber-physical systems.

Since constrained polynomial zonotopes are closed under polynomial maps and intersections, they are an excellent fit for set-based observers that rely on intersection computation since both, the prediction and the correction step, can be calculated with high accuracy. Using constrained polynomial zonotopes we therefore presented a novel set-based observer for nonlinear discrete-time systems with nonlinear measurement functions. As we demonstrated with a numerical example, our observer successfully reduces the uncertainty on the system state and is fast enough to run in real-time.

Constrained polynomial zonotopes are also well suited for the verification of computer programs using inductive invariants. In particular, due to the advantageous properties of constrained polynomial zonotopes, nonlinear assignments, intersections with conditions of if-statements, and the unification of sets from different loop-iterations can often be computed exactly. As we showed for an exemplary program, one consequently often obtains the exact invariant or a very tight enclosure.

With the Z-representation of polytopes, any range bounding problem with a polytopic domain can be transformed to an equivalent range bounding problem with an interval domain. Since this transformation circumvents the need to enclose the domain by an interval, one often obtains significantly tighter bounds, as we demonstrated for an exemplary range bounding problem.

The Z-representation of polytopes also has a natural connection to generalized barycentric coordinates, which are, among other things, very useful for many control applications. In particular, closed-form expressions for the generalized barycentric coordinates with respect to the factors of the Z-representation are obtained automatically during conversion from V-representation to Z-representation.

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

In this thesis, we introduced the novel set representations sparse polynomial zonotopes, constrained polynomial zonotopes, and the Z-representation of polytopes as extensions to polynomial zonotopes. Moreover, we developed new approaches for reachability analysis of nonlinear continuous and hybrid systems and presented several additional applications for our novel set representations.

Extensions of Polynomial Zonotopes

We first introduced a novel sparse representation of polynomial zonotopes, which represents polynomial zonotopes much more compact than the previous non-sparse representation. Sparse polynomial zonotopes can represent non-convex sets and are closed under the set operations linear map, Minkowski sum, Cartesian product, convex hull and quadratic map, for all of which we derived closed-form expressions. Moreover, we showed how any interval, zonotope, bounded polytope and Taylor model can be equivalently represented as a sparse polynomial zonotope, which further substantiates the relevance of this novel set representation. In addition, sparse polynomial zonotopes preserve dependencies, which is advantageous for many applications.

Next, we presented constrained polynomial zonotopes, which extend sparse polynomial zonotopes by adding polynomial equality constraints for the dependent factors. This results in a set representation that is closed under all relevant set representations including intersection and union. Furthermore, in addition to the set representations that can be represented as sparse polynomial zonotopes, constrained polynomial zonotopes can also represent ellipsoids. For both, sparse polynomial zonotopes and constrained polynomial zonotopes, all relevant set operations only have polynomial complexity with respect to the dimension and we presented efficient techniques for reducing the representation size. Overall, these novel set representations are therefore well suited to handle the complexity and nonlinearity of high-dimensional cyber-physical systems.

As a third set representation, we finally introduced the Z-representation of polytopes, which stores polynomial zonotopes that represent polytopes very compactly. An analysis of the representation size showed that for polytopes that are similar to zonotopes, the Z-representation is more compact than the V-representation and the H-representation. Again

we derived closed-form expressions for set operations on the Z-representation and provided algorithms for the conversion between V-representation and Z-representation.

Reachability Analysis

One major application for the novel set representations presented in this thesis, and in particular sparse polynomial zonotopes, is reachability analysis. We first demonstrated the benefits of sparse polynomial zonotopes with respect to accuracy and computation time for reachability analysis of nonlinear continuous systems using the conservative polynomialization approach. Moreover, we proved that with sparse polynomial zonotopes the relations between initial states and reachable states are preserved, which results in significant speed-ups for many applications, such as safety falsification, optimization over reachable sets, and safe control. We then exploited this dependency preservation to introduce a novel method for computing tight non-convex inner-approximations of reachable sets for nonlinear continuous systems, which can, for example, be used to disprove specifications. Finally, we presented a novel approach for reachability analysis of hybrid systems with nonlinear guard sets. Again, we used sparse polynomial zonotopes to represent the reachable sets since they are well suited for handling both, the continuous dynamics as well as the discrete transitions of a hybrid system. Due to the computational efficiency of sparse polynomial zonotopes, all reachability algorithms presented in this thesis only have polynomial complexity with respect to the system dimension and are therefore well-suited for the verification of high-dimensional systems.

Selected Applications

Despite reachability analysis, our novel set representations can also be used for many other applications. For constrained polynomial zonotopes, we discussed set-based observers and program verification using inductive invariants in detail. Set-based observers require the evaluation of nonlinear maps in the prediction step and the computation of intersections with level sets in the correction step. Both operations can be tightly enclosed by constrained polynomial zonotopes. Similarly, for the computation of inductive invariants, one requires nonlinear maps for nonlinear assignments, intersections for the conditions of if-statements, and unions to unify sets from different paths and loop-iterations. Again, with constrained polynomial zonotopes it is straightforward to handle all of these operations. For the Z-representation of polytopes, we examined range bounding on polytopic domains and generalized barycentric coordinates, two problems that have high relevance for many applications.

6.2 Future Directions

Finally, we provide an outlook about possible future extensions of the research presented in this thesis.

Extensions of Polynomial Zonotopes

Since many operations on sparse polynomial zonotopes and constrained polynomial zonotopes increase the representation size, good order reduction methods are crucial for their effective use. As we demonstrated by several numerical examples, the reduction techniques that we introduced in this thesis work quite well. However, there is still room for improvement. Especially for sparse polynomial zonotopes an order reduction method that compensates the removed generators by prolonging the remaining dependent generator instead of introducing new independent generators would be very advantageous, since computations on the dependent generators are exact, while computations on the independent generators are over-approximative. Another major improvement for sparse polynomial zonotopes and constrained polynomial zonotopes would be the development of a more efficient containment check, since the containment checks that we presented in Sec. 3.1.6 and Sec. 3.2.6 have exponential complexity and are therefore restricted to low-dimensional sets. One promising way to achieve this would be the derivation of a closed-form expression for converting a sparse polynomial zonotope or a constrained polynomial zonotope to a polynomial level set, since for level sets containment checks are cheap and straightforward to implement. For the Z-representation of polytopes, a major focus of future research should be the minimization of the representation size when converting from polytope V-representation to Z-representation. Potential strategies for achieving this could be to either represent the polytope as a union of zonotopes since zonotopes can be compactly represented in Z-representation, or to exploit dependencies during unification of the vertices using the convex hull.

Reachability Analysis

All common reachability tools and algorithms including the ones presented here in Chapter 4 have the disadvantage that the tightness of the computed reachable set heavily depends on the correct tuning of user-defined parameters, such as time step size and zonotope order. Since the tuning usually has to be done by experts, this currently prevents the broad use of reachability analysis in industry. A major focus of future research on reachability analysis should therefore be the fully automated tuning of these parameters. While there already exists some recent work in this direction [166, 167], none of these approaches can provide any guarantees on the tightness of the computed reachable set. The overall goal should be the development of an algorithm that takes a maximum allowed distance between the exact reachable set and the computed enclosure as an input and then automatically tunes the parameters in such a way that this maximum allowed distance is met. Clearly, a requirement for this is a reachability algorithm that actually converges to the exact reachable set if the parameters are refined accordingly. While this is quite easy to achieve when splitting the reachable set is explicitly considered [168], it is much harder if computation-

ally expensive splits should be avoided. A first step toward a fully automated reachability algorithm with tightness guarantees would therefore be to either prove convergence for an existing algorithm, such as the conservative polynomialization algorithm in Sec. 4.1.2, or to develop a new algorithm that converges without splitting. With reachability algorithms that are able to compute inner-approximations and outer-approximations of reachable sets with arbitrary precision, one could even go one step further and build an automated verifier which automatically increases the precision until the specifications can either be proven or disproven. While this is straightforward for simple specifications defined by forbidden sets, it is much more challenging for temporal logic specifications, where it is not yet clear if these verification problems are even decidable.

Selected Applications

While we only discussed a few specific applications for constrained polynomial zonotopes and the Z-representation in Chapter 5, there of course exist many more, especially for constrained polynomial zonotopes: A topic of increasing interest is the verification of artificial intelligence, where it is often necessary to propagate sets through large neural networks. If the network consists of neurons with ReLU activation functions, this requires the set operations intersection and union. On the other hand, if sigmoid or hyperbolic tangent activation functions are used, one obtains strongly nonlinear maps. Since constrained polynomial zonotopes are closed under intersection as well as union and can enclose the sets resulting from nonlinear maps tightly using a Taylor series expansion, they seem to be well suited for neural network verification. While in Sec. 4.4 we presented an approach based on sparse polynomial zonotopes for hybrid system reachability analysis for computational reasons, in the future, constrained polynomial zonotopes could potentially be used to handle discrete transitions in hybrid systems without any over-approximation. For each discrete transition we first compute the intersection of the guard set with all time interval reachable sets intersecting the guard and then unite the sets for all time intervals. Since constrained polynomial zonotopes are closed under intersection and union, we can compute the exact set. Yet another application for constrained polynomial zonotopes is the verification of temporal logic specifications since the *reachset temporal logic* approach in [169] requires a set representation that is closed under intersection and for which one can efficiently check if a set is contained in a union of sets. Finally, due to their expressiveness, constrained polynomial zonotopes are advantageous for all algorithms where a mixture of different set representations has to be handled. The intersection of a zonotope and an ellipsoid, for example, can be easily computed using constrained polynomial zonotopes.

Appendices

Appendix A

High-Order Polynomial Maps

We now show that every higher order polynomial map as defined in (2.10) can be computed using a sequence of quadratic maps:

Proposition A.1. *(Polynomial Map) Given sets* $\mathcal{S}_1, \ldots, \mathcal{S}_o \subset \mathbb{R}^n$ *and a set of coefficients* $\mathcal{A} = \{a^{(i)}_{j_1,\ldots,j_o} \in \mathbb{R} \mid i \in \{1,\ldots,w\},\ j_1,\ldots,j_o \in \{1,\ldots,n\}\}$*, the polynomial map of order* $o \in \mathbb{N}_{\geq 2}$ *defined by the multiplication of the sets* $\mathcal{S}_1, \ldots, \mathcal{S}_o$ *and* $\mathcal{A}$ *is*

$$poly(\mathcal{A}, \mathcal{S}_1, \ldots, \mathcal{S}_o) = sq(\mathcal{M}_1, \mathcal{S}_1, \mathcal{Y}_1) \boxplus \ldots \boxplus sq(\mathcal{M}_n, \mathcal{S}_1, \mathcal{Y}_n),$$

with

$$\begin{aligned}
\forall j_1, \ldots, j_{o-2} \in \{1, \ldots, n\} : \\
\mathcal{Y}_{j_1} &= sq(\mathcal{M}_1, \mathcal{S}_2, \mathcal{Y}_{j_1,1}) \boxplus \ldots \boxplus sq(\mathcal{M}_n, \mathcal{S}_2, \mathcal{Y}_{j_1,n}), \\
\mathcal{Y}_{j_1,j_2} &= sq(\mathcal{M}_1, \mathcal{S}_3, \mathcal{Y}_{j_1,j_2,1}) \boxplus \ldots \boxplus sq(\mathcal{M}_n, \mathcal{S}_3, \mathcal{Y}_{j_1,j_2,n}), \\
&\vdots \\
\mathcal{Y}_{j_1,\ldots,j_{o-3}} &= sq(\mathcal{M}_1, \mathcal{S}_{o-2}, \mathcal{Y}_{j_1,\ldots,j_{o-3},1}) \boxplus \ldots \boxplus sq(\mathcal{M}_n, \mathcal{S}_{o-2}, \mathcal{Y}_{j_1,\ldots,j_{o-3},n}), \\
\mathcal{Y}_{j_1,\ldots,j_{o-2}} &= sq(\mathcal{Q}_{j_1,\ldots,j_{o-2}}, \mathcal{S}_{o-1}, \mathcal{S}_o),
\end{aligned}$$

where

$$\begin{aligned}
\forall i \in \{1, \ldots, n\},\ \forall j_1, \ldots, j_{o-2} \in \{1, \ldots, n\} : \\
\mathcal{M}_i = \{M_{i,1}, \ldots, M_{i,n}\}, \quad \mathcal{Q}_{j_1,\ldots,j_{o-2}} = \left\{Q^{(1)}_{j_1,\ldots,j_{o-2}}, \ldots, Q^{(n)}_{j_1,\ldots,j_{o-2}}\right\}, \\
Q^{(i)}_{j_1,\ldots,j_{o-2}} = \begin{bmatrix} a^{(i)}_{j_1,\ldots,j_{o-2},1,1} & \cdots & a^{(i)}_{j_1,\ldots,j_{o-2},1,n} \\ \vdots & \ddots & \vdots \\ a^{(i)}_{j_1,\ldots,j_{o-2},n,1} & \cdots & a^{(i)}_{j_1,\ldots,j_{o-2},n,n} \end{bmatrix},
\end{aligned}$$

and $M_{i,j} \in \mathbb{R}^{n \times n}$ *denotes an all-zero matrix where the* j*-th entry in the* i*-th row is identical to* 1*.*

Proof. The definition of the polynomial map in (2.10) can be equivalently formulated as

$$poly(\mathcal{A}, \mathcal{S}_1, \ldots, \mathcal{S}_o) \overset{(2.10)}{=} \Big\{ x \;\Big|\; x_{(i)} = \sum_{j_1=1}^{n} \ldots \sum_{j_o=1}^{n} a^{(i)}_{j_1,\ldots,j_o} \cdot s_{1(j_1)} \cdot \ldots \cdot s_{o(j_o)}, \\ s_1 \in \mathcal{S}_1, \ldots, s_o \in \mathcal{S}_o,\ i = 1, \ldots, w \Big\}$$

$$= \Big\{ x \;\Big|\; s_1 \in \mathcal{S}_1, \ldots, s_o \in \mathcal{S}_o,\ i = 1, \ldots, w, \\ x_{(i)} = \sum_{j_1=1}^{n} s_{1(j_1)} \cdot \underbrace{\Big(\sum_{j_2=1}^{n} s_{2(j_2)} \cdot \ldots \cdot \underbrace{\Big(\sum_{j_{o-1}=1}^{n} \sum_{j_o=1}^{n} a^{(i)}_{j_1,\ldots,j_o} \cdot s_{o-1(j_{o-1})} \cdot s_{o(j_o)} \Big)}_{y_{j_1,\ldots,j_{o-2}(i)}} \ldots \Big)}_{y_{j_1(i)}} \Big\}$$

$$= \Big\{ x \;\Big|\; x_{(i)} = \sum_{j_1=1}^{n} s_{1(j_1)} \cdot y_{j_1(i)},\ s_1 \in \mathcal{S}_1, \ldots, s_o \in \mathcal{S}_o,\ i = 1, \ldots, w, \\ \underbrace{y_{j_1(i)} = \sum_{j_2=1}^{n} s_{2(j_2)} \cdot y_{j_1,j_2(i)}}_{y_{j_1} \in \mathcal{Y}_{j_1}}, \ldots, \underbrace{y_{j_1,\ldots,j_{o-3}(i)} = \sum_{j_{o-2}=1}^{n} s_{o-2(j_{o-2})} \cdot y_{j_1,\ldots,j_{o-2}(i)}}_{y_{j_1,\ldots,j_{o-3}} \in \mathcal{Y}_{j_1,\ldots,j_{o-3}}}, \\ \underbrace{y_{j_1,\ldots,j_{o-2}(i)} = \sum_{j_{o-1}=1}^{n} \sum_{j_o=1}^{n} a^{(i)}_{j_1,\ldots,j_o} \cdot s_{o-1(j_{o-1})} \cdot s_{o(j_o)}}_{y_{j_1,\ldots,j_{o-2}(i)} \in \mathcal{Y}_{j_1,\ldots,j_{o-2}} = sq(\mathcal{Q}_{j_1,\ldots,j_{o-2}}, \mathcal{S}_{o-1}, \mathcal{S}_o)} \Big\}$$

$$= sq(\mathcal{M}_1, \mathcal{S}_1, \mathcal{Y}_1) \boxplus \ldots \boxplus sq(\mathcal{M}_n, \mathcal{S}_1, \mathcal{Y}_n),$$

so that the polynomial map can be computed as a sequence of quadratic maps and exact additions. □

When computing the polynomial map using Prop. A.1, it is crucial that the implementation of the quadratic map for the corresponding set representation preserves the dependencies between different sets, since otherwise one potentially obtains very large over-approximations.

Appendix B

Proof for the Union of Constrained Polynomial Zonotopes

In this appendix, we prove Prop. 3.2.25 for computing the union of two CPZs. As a prerequisite for the proof, we introduce the `constrDom` operation:

Definition B.1. *Given a constraint defined by the constraint generator matrix $A \in \mathbb{R}^{m \times q}$, the constraint vector $b \in \mathbb{R}^m$, and the constraint exponent matrix $R \in \mathbb{N}_0^{p \times q}$,* `constrDom` *returns the set of values satisfying the constraint:*

$$\texttt{constrDom}(A, b, R) = \left\{ \alpha \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b,\ \alpha_k \in [-1, 1] \right\},$$

where $\alpha = [\alpha_1 \ \dots \ \alpha_p]^T$.

Using Def. B.1, it is straightforward to see that the following identity holds:

$$\begin{aligned} &\left\langle c, G, E, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & R_2 \end{bmatrix} \right\rangle_{CPZ} = \\ &\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_1} \left(\prod_{k=1}^{p} \alpha_k^{R_{1(k,i)}} \right) A_{1(\cdot,i)} = b_1,\ \alpha \in \mathcal{D} \right\}, \end{aligned} \tag{B.1}$$

where $\mathcal{D} = \texttt{constrDom}(A_2, b_2, R_2)$ and $\alpha = [\alpha_1 \ \dots \ \alpha_p]^T$. As another prerequisite, we introduce the following lemma:

Lemma B.2. *Given a constant offset $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times h}$, an exponent matrix $E \in \mathbb{N}_0^{p \times h}$, and two domains $\mathcal{D}_1, \mathcal{D}_2 \subseteq [-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$, it holds that*

$$\begin{aligned} &\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha \in (\mathcal{D}_1 \cup \mathcal{D}_2) \right\} = \\ &\underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha \in \mathcal{D}_1 \right\}}_{\mathcal{CPZ}_1} \cup \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha \in \mathcal{D}_2 \right\}}_{\mathcal{CPZ}_2}, \end{aligned}$$

where $\alpha = [\alpha_1 \ \dots \ \alpha_p]^T$.

Proof. Using the definition of CPZs in Def. 3.2.1 and the definition of the union in (2.8) we obtain

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha \in (\mathcal{D}_1 \cup \mathcal{D}_2) \right\} =$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \alpha \in \mathcal{D}_1 \vee \alpha \in \mathcal{D}_2 \right\} \overset{\text{Def. 3.2.1}}{=}$$

$$\{x \mid x \in \mathcal{CPZ}_1 \vee x \in \mathcal{CPZ}_2\} \overset{(2.8)}{=} \mathcal{CPZ}_1 \cup \mathcal{CPZ}_2,$$

which concludes the proof. □

Before we begin with the proof, let us first recap the result from Prop. 3.2.25. According to Prop. 3.2.25, the union of two CPZs $\mathcal{CPZ}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1 \rangle_{CPZ} \subset \mathbb{R}^n$ and $\mathcal{CPZ}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2 \rangle_{CPZ} \subset \mathbb{R}^n$ is

$$\begin{aligned} \mathcal{CPZ}_1 \cup \mathcal{CPZ}_2 = \Bigg\langle & \underbrace{0.5(c_1 + c_2)}_{c}, \underbrace{\begin{bmatrix} 0.5(c_1 - c_2) & G_1 & G_2 \end{bmatrix}}_{G}, \underbrace{\begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & E_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & E_2 \end{bmatrix}}_{E}, \\ & \underbrace{\begin{bmatrix} \widehat{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \overline{A} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & A_1 & \mathbf{0} & -0.5\ b_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & A_2 & 0.5\ b_2 \end{bmatrix}}_{A}, \underbrace{\begin{bmatrix} \widehat{b} \\ \overline{b} \\ 0.5\ b_1 \\ 0.5\ b_2 \end{bmatrix}}_{b}, \underbrace{\begin{bmatrix} \widehat{R} & \overline{R} & \begin{bmatrix} \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & 0 \\ R_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & R_2 & \mathbf{0} \end{bmatrix} \end{bmatrix}}_{R} \Bigg\rangle_{CPZ}, \end{aligned} \tag{B.2}$$

where

$$\widehat{A} = 1, \;\; \widehat{b} = 1, \;\; \widehat{R} = \begin{bmatrix} 1 \\ 1 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \;\; H = \begin{bmatrix} \begin{bmatrix} 2 & \dots & 2 \end{bmatrix} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \begin{bmatrix} 2 & \dots & 2 \end{bmatrix} \\ 2I_{p_2} & \dots & 2I_{p_2} \end{bmatrix},$$

$$\overline{A} = \begin{bmatrix} 1 & -1 & \frac{1}{2p_1}\mathbf{1} & -\frac{1}{2p_1}\mathbf{1} & -\frac{1}{2p_2}\mathbf{1} & -\frac{1}{2p_2}\mathbf{1} & -\frac{1}{4p_1p_2}\mathbf{1} & \frac{1}{4p_1p_2}\mathbf{1} \end{bmatrix}, \;\; \overline{b} = 0,$$

$$\overline{R} = \begin{bmatrix} \begin{bmatrix} 1 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 2I_{p_1} & 2I_{p_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 2I_{p_2} & 2I_{p_2} \end{bmatrix} & \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ H \end{bmatrix} & \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ H \end{bmatrix} \end{bmatrix}.$$

The outline of the proof is as follows: We first show in Sec. B.1 that the constraints in (B.2) restrict the values for the factors α_k to the domain $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ corresponding to

the union of two domains $\mathcal{D}_1$ and $\mathcal{D}_2$. Afterward, in Sec. B.2, we apply Lemma B.2 to express the resulting CPZ $\langle c, G, E, A, b, R\rangle_{CPZ}$ in (B.2) as $\langle c, G, E, A, b, R\rangle_{CPZ} = \mathcal{S}_1 \cup \mathcal{S}_2$, where $\mathcal{S}_1, \mathcal{S}_2$ represent the sets corresponding to the domains $\mathcal{D}_1, \mathcal{D}_2$, respectively. Finally, we show in Sec. B.3 that $\mathcal{S}_1 = \mathcal{CPZ}_1$ and $\mathcal{S}_2 = \mathcal{CPZ}_2$ holds, which concludes the proof.

B.1 Domain defined by the Constraints

First, we show that the constraints in (B.2) restrict the values for the factors α_k to the domain $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$. The matrices $\widehat{A}, \widehat{R}$ and the vector $\widehat{b}$ in (B.2) define the constraint

$$\alpha_1 \alpha_2 = 1. \tag{B.3}$$

The only solutions for (B.3) within the domain $\alpha_1 \in [-1, 1]$, $\alpha_2 \in [-1, 1]$ are the two points $\alpha_1 = 1$, $\alpha_2 = 1$ and $\alpha_1 = -1$, $\alpha_2 = -1$. The constraint (B.3) therefore restricts the values for the factors α_k to the domain

$$\begin{aligned}\widehat{\mathcal{D}} &= \texttt{constrDom}(\widehat{A}, \widehat{b}, \widehat{R}) \\ &= \underbrace{1 \times 1 \times [-1, 1] \times \ldots \times [-1, 1]}_{\widehat{\mathcal{D}}_1} \cup \underbrace{-1 \times -1 \times [-1, 1] \times \ldots \times [-1, 1]}_{\widehat{\mathcal{D}}_2}.\end{aligned} \tag{B.4}$$

The matrices $\overline{A}, \overline{R}$ and the vector $\overline{b}$ in (B.2) define the constraint

$$\begin{aligned}\sum_{i=1}^{\overline{q}} \left(\prod_{k=1}^{p_1+p_2+2} \alpha_k^{\overline{R}_{(k,i)}} \right) \overline{A}_{(\cdot,i)} &= \alpha_1 - \alpha_2 + \frac{1}{2} f_1(\alpha) - \frac{1}{2} \alpha_1 f_1(\alpha) - \frac{1}{2} f_2(\alpha) \\ &\quad - \frac{1}{2} \alpha_1 f_2(\alpha) - \frac{1}{4} f_1(\alpha) f_2(\alpha) + \frac{1}{4} \alpha_1 f_1(\alpha) f_2(\alpha) = \\ &= \underbrace{\left(1 + \alpha_1 + \frac{1}{2} f_1(\alpha)(1 - \alpha_1)\right) \left(1 - \frac{1}{2} f_2(\alpha)\right) - \alpha_2 - 1}_{g(\alpha)} = 0 = \overline{b},\end{aligned} \tag{B.5}$$

where

$$f_1(\alpha) = \frac{1}{p_1} \sum_{k=3}^{p_1+2} \alpha_{(k)}^2, \quad f_2(\alpha) = \frac{1}{p_2} \sum_{k=p_1+3}^{p_1+p_2+2} \alpha_{(k)}^2, \tag{B.6}$$

with $\overline{q}$ denoting the number of columns of matrix $\overline{A}$ and $\alpha = [\alpha_1 \ \ldots \ \alpha_{p_1+p_2+2}]^T$. Let $\overline{\mathcal{D}} = \texttt{constrDom}(\overline{A}, \overline{b}, \overline{R})$ be the restricted domain for the factor values corresponding to the constraint $g(\alpha) = 0$ in (B.5). Then the factor domain $\mathcal{D}$ for the combination of the constraints defined by $\widehat{A}, \widehat{b}, \widehat{R}$ and $\overline{A}, \overline{b}, \overline{R}$ is

$$\begin{aligned}\mathcal{D} &= \texttt{constrDom}\left(\begin{bmatrix} \widehat{A} & 0 \\ 0 & \overline{A} \end{bmatrix}, \begin{bmatrix} \widehat{b} \\ \overline{b} \end{bmatrix}, \begin{bmatrix} \widehat{R} & \overline{R} \end{bmatrix} \right) \\ &= \texttt{constrDom}(\widehat{A}, \widehat{b}, \widehat{R}) \cap \texttt{constrDom}(\overline{A}, \overline{b}, \overline{R}) = \widehat{\mathcal{D}} \cap \overline{\mathcal{D}} \overset{\text{(B.4)}}{=} \underbrace{(\widehat{\mathcal{D}}_1 \cap \overline{\mathcal{D}})}_{\mathcal{D}_1} \cup \underbrace{(\widehat{\mathcal{D}}_2 \cap \overline{\mathcal{D}})}_{\mathcal{D}_2}.\end{aligned} \tag{B.7}$$

To compute the domain $\mathcal{D}_1 = \widehat{\mathcal{D}}_1 \cap \overline{\mathcal{D}}$ in (B.7) we insert the values $\alpha_1 = 1$, $\alpha_2 = 1$ from $\widehat{\mathcal{D}}_1$ into $g(\alpha) = 0$. This yields $f_2(\alpha) = 0$ according to (B.5), which is only satisfiable for $\alpha_{p_1+3} = 0, \ldots, \alpha_{p_1+p_2+2} = 0$. Moreover, inserting the values $\alpha_1 = -1$, $\alpha_2 = -1$ from $\widehat{\mathcal{D}}_2$ into $g(\alpha) = 0$ yields according to (B.5)

$$\underbrace{f_1(\alpha)\left(1 - \frac{1}{2}\underbrace{f_2(\alpha)}_{\in[0,1]}\right)}_{\in[0.5,1]} = 0, \tag{B.8}$$

which is only satisfiable for $f_1(\alpha) = 0$. Since the constraint $f_1(\alpha) = 0$ is only satisfiable for $\alpha_3 = 0, \ldots, \alpha_{p_1+2} = 0$ according to (B.6), the constraint $g(\alpha) = 0$ is consequently also only satisfiable for $\alpha_3 = 0, \ldots, \alpha_{p_1+2} = 0$. In summary, the domain for the combination of the constraint $\alpha_1\alpha_2 = 1$ in (B.3) and the constraint $g(\alpha) = 0$ in (B.5) is therefore

$$\begin{aligned}\mathcal{D} &= \texttt{constrDom}\Big(\underbrace{\begin{bmatrix}\widehat{A} & 0\\ 0 & \overline{A}\end{bmatrix}}_{A_U}, \underbrace{\begin{bmatrix}\widehat{b}\\ \overline{b}\end{bmatrix}}_{b_U}, \underbrace{\begin{bmatrix}\widehat{R} & \overline{R}\end{bmatrix}}_{R_U}\Big) \overset{(B.7)}{=} \underbrace{(\widehat{\mathcal{D}}_1 \cap \overline{\mathcal{D}})}_{\mathcal{D}_1} \cup \underbrace{(\widehat{\mathcal{D}}_2 \cap \overline{\mathcal{D}})}_{\mathcal{D}_2} \\ &= \underbrace{\left\{ \begin{bmatrix}1 & 1 & \alpha_3 & \ldots & \alpha_{p_1+2} & \mathbf{0}\end{bmatrix}^T \mid \alpha_3, \ldots, \alpha_{p_1+2} \in [-1,1]\right\}}_{\mathcal{D}_1} \cup \\ &\quad \underbrace{\left\{ \begin{bmatrix}-1 & -1 & \mathbf{0} & \alpha_{p_1+3} & \ldots & \alpha_{p_1+p_2+2}\end{bmatrix}^T \mid \alpha_{p_1+3}, \ldots, \alpha_{p_1+p_2+2} \in [-1,1]\right\}}_{\mathcal{D}_2},\end{aligned} \tag{B.9}$$

which enables us to apply Lemma B.2 in the next section.

B.2 Reformulation as Union of Sets

We now prove that the resulting CPZ $\langle c, G, E, A, b, R\rangle_{CPZ}$ in (B.2) defines the union of two sets $\mathcal{S}_1$ and $\mathcal{S}_2$. Using the domain $\mathcal{D}$ as defined in (B.9) and introducing

$$A_L = \begin{bmatrix}A_1 & \mathbf{0} & -0.5\ b_1\\ \mathbf{0} & A_2 & 0.5\ b_2\end{bmatrix}, \quad b_L = \begin{bmatrix}0.5\ b_1\\ 0.5\ b_2\end{bmatrix}, \quad R_L = \begin{bmatrix}\mathbf{0} & \mathbf{0} & 1\\ \mathbf{0} & \mathbf{0} & 0\\ R_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & R_2 & \mathbf{0}\end{bmatrix}, \tag{B.10}$$

the resulting CPZ $\langle c, G, E, A, b, R\rangle_{CPZ}$ in (B.2) can be equivalently represented as

$$\left\langle c, G, E, \underbrace{\begin{bmatrix}\widehat{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0\\ \mathbf{0} & \overline{A} & \mathbf{0} & \mathbf{0} & 0\\ \mathbf{0} & \mathbf{0} & A_1 & \mathbf{0} & -0.5\ b_1\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & A_2 & 0.5\ b_2\end{bmatrix}}_{A}, \underbrace{\begin{bmatrix}\widehat{b}\\ \overline{b}\\ 0.5\ b_1\\ 0.5\ b_2\end{bmatrix}}_{b}, \underbrace{\left[\begin{array}{cc|ccc}\widehat{R} & \overline{R} & \multicolumn{3}{c}{\multirow{2}{*}{$\begin{bmatrix}\mathbf{0} & \mathbf{0} & 1\\ \mathbf{0} & \mathbf{0} & 0\\ R_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & R_2 & \mathbf{0}\end{bmatrix}$}}\\ & & & & \end{array}\right]}_{R} \right\rangle_{CPZ}$$

$$\overset{(B.9),(B.10)}{=} \left\langle c, G, E, \begin{bmatrix}A_U & \mathbf{0}\\ \mathbf{0} & A_L\end{bmatrix}, \begin{bmatrix}b_U\\ b_L\end{bmatrix}, \begin{bmatrix}R_U & R_L\end{bmatrix} \right\rangle_{CPZ}$$

$$\overset{\text{(B.1),(B.9)}}{=} \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_L} \left(\prod_{k=1}^{p} \alpha_k^{R_{L(k,i)}} \right) A_{L(\cdot,i)} = b_L,\ \alpha \in \mathcal{D} \right\} \tag{B.11}$$

$$\overset{\substack{\text{Lemma B.2,} \\ \mathcal{D}=\mathcal{D}_1 \cup \mathcal{D}_2}}{=} \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_L} \left(\prod_{k=1}^{p} \alpha_k^{R_{L(k,i)}} \right) A_{L(\cdot,i)} = b_L,\ \alpha \in \mathcal{D}_1 \right\}}_{\mathcal{S}_1}$$

$$\cup \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_L} \left(\prod_{k=1}^{p} \alpha_k^{R_{L(k,i)}} \right) A_{L(\cdot,i)} = b_L,\ \alpha \in \mathcal{D}_2 \right\}}_{\mathcal{S}_2},$$

where $p = p_1 + p_2 + 2$, $q_L = q_1 + q_2 + 1$, and $\alpha = [\alpha_1\ \ldots\ \alpha_p]^T$.

B.3 Equivalence of Sets

It remains to show that $\mathcal{S}_1 = \mathcal{CPZ}_1$ and $\mathcal{S}_2 = \mathcal{CPZ}_2$. Inserting the definition of the domain $\mathcal{D}_1$ in (B.9) into the definition of the set $\mathcal{S}_1$ in (B.11) yields

$$\mathcal{S}_1 \overset{\text{(B.11)}}{=} \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_L} \left(\prod_{k=1}^{p} \alpha_k^{R_{L(k,i)}} \right) A_{L(\cdot,i)} = b_L,\ \alpha \in \mathcal{D}_1 \right\} \overset{\text{(B.9)}}{=}$$

$$\left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_L} \left(\prod_{k=1}^{p} \alpha_k^{R_{L(k,i)}} \right) A_{L(\cdot,i)} = b_L,\ \alpha_1, \alpha_2 = 1, \right.$$
$$\left. \alpha_3, \ldots, \alpha_{p_1+2} \in [-1, 1],\ \ \alpha_{p_1+3}, \ldots, \alpha_{p_1+p_2+2} = 0 \right\} \overset{\text{(B.2),(B.10)}}{=}$$

$$\left\{ \underbrace{0.5(c_1 + c_2) + 0.5(c_1 - c_2)\alpha_1}_{\overset{\alpha_1=1}{=} c_1} + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_{2+k}^{E_{1(k,i)}} \right) G_{1(\cdot,i)} + \sum_{i=1}^{h_2} \underbrace{\left(\prod_{k=1}^{p_2} \alpha_{2+p_1+k}^{E_{2(k,i)}} \right)}_{=0} G_{2(\cdot,i)} \;\middle|\; \right.$$
$$\sum_{i=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_{2+k}^{R_{1(k,i)}} \right) A_{1(\cdot,i)} = \underbrace{0.5b_1 + 0.5b_1\alpha_1}_{\overset{\alpha_1=1}{=} b_1},\ \sum_{i=1}^{q_2} \underbrace{\left(\prod_{k=1}^{p_2} \alpha_{2+p_1+k}^{R_{2(k,i)}} \right)}_{=0} A_{2(\cdot,i)} = \underbrace{0.5b_2 - 0.5b_2\alpha_1}_{\overset{\alpha_1=1}{=} \mathbf{0}},$$
$$\left. \alpha_1, \alpha_2 = 1,\ \ \alpha_3, \ldots, \alpha_{p_1+2} \in [-1, 1],\ \ \alpha_{p_1+3}, \ldots, \alpha_{p_1+p_2+2} = 0 \right\} =$$

$$\underbrace{\left\{ c_1 + \sum_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_{2+k}^{E_{1(k,i)}} \right) G_{1(\cdot,i)} \;\middle|\; \sum_{i=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_{2+k}^{R_{1(k,i)}} \right) A_{1(\cdot,i)} = b_1,\ \alpha_3, \ldots, \alpha_{p_1+2} \in [-1,1] \right\}}_{=\mathcal{CPZ}_1}.$$

The proof that $\mathcal{S}_2 = \mathcal{CPZ}_2$ is similar to the proof for $\mathcal{S}_1$ and therefore omitted at this point.

Appendix C

Dependency Preservation for Sparse Polynomial Zonotopes

To prove the correctness of the reachable subset approach presented in Sec. 4.2, we need to show that the implementations of the linear map, the Minkowski sum with a zonotope, the Cartesian product with a zonotope, the quadratic map, order reduction, and restructuring on SPZs as specified in Sec. 3.1 are dependency-preserving. While we already demonstrated in Prop. 4.2.13 that the Minkowski sum with a zonotope is dependency-preserving, we now provide the proofs for the remaining set operations. We begin with the linear map:

Proposition C.1. *(Dependency Preservation Linear Map) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ *and a matrix* $M \in \mathbb{R}^{w \times n}$*, it holds that the implementation of the linear map* $M \otimes \mathcal{PZ}$ *in Prop. 3.1.18 is dependency-preserving.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad M \otimes \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) \subseteq \underline{\lfloor M \otimes \mathcal{PZ} \rfloor}(\alpha). \tag{C.1}$$

Inserting the evaluation function for SPZs as defined in Def. 4.2.10 and the implementation of the linear map in Prop. 3.1.18 into (C.1) yields

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad M \otimes \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha)$$

$$\overset{\text{Def. } 4.2.10}{=} M\left(c + \sum_{i=1}^{h}\left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}}\right) G_{(\cdot,i)}\right) \oplus \left(M \otimes \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\}\right)$$

$$\overset{(2.1)}{=} Mc + \sum_{i=1}^{h}\left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}}\right) MG_{(\cdot,i)} \oplus \left\{ \sum_{j=1}^{q} \beta_j MG_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\}$$

$$\overset{\text{Def. } 4.2.10}{=} \underline{\lfloor \langle Mc, MG, MG_I, E, id \rangle_{PZ} \rfloor}(\alpha) \overset{\text{Prop. } 3.1.18}{=} \underline{\lfloor M \otimes \mathcal{PZ} \rfloor}(\alpha),$$

which is identical to (C.1) and therefore concludes the proof. □

Next, we consider the Cartesian product:

Proposition C.2. *(Dependency Preservation Cartesian Product) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ *and a zonotope* $\mathcal{Z} = \langle c_z, G_z \rangle_Z \subset \mathbb{R}^n$*, it holds that the implementation of the Cartesian product* $\mathcal{PZ} \times \mathcal{Z}$ *in Prop. 3.1.22 is dependency-preserving.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) \times \mathcal{Z} \subseteq \underline{\lfloor \mathcal{PZ} \times \mathcal{Z} \rfloor}(\alpha). \tag{C.2}$$

Inserting the evaluation function for SPZs as defined in Def. 4.2.10 and the implementation of the Cartesian product in Prop. 3.1.22 into (C.2) yields

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) \times \mathcal{Z} \overset{\text{Def. 4.2.10}}{=} \left(c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \langle \mathbf{0}, G_I \rangle_Z \right) \times \mathcal{Z}$$

$$\overset{(2.4)}{=} \begin{bmatrix} c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \\ \mathbf{0} \end{bmatrix} \oplus \left\{ \sum_{j=1}^{q} \beta_j \begin{bmatrix} G_{I(\cdot,j)} \\ \mathbf{0} \end{bmatrix} \;\middle|\; \beta_j \in [-1, 1] \right\}$$
$$\oplus \left\{ \sum_{j=1}^{l} \beta_{q+j} \begin{bmatrix} \mathbf{0} \\ G_{z(\cdot,j)} \end{bmatrix} \;\middle|\; \beta_{q+j} \in [-1, 1] \right\}$$

$$\overset{\text{Def. 4.2.10}}{=} \underline{\left\lfloor \left\langle \begin{bmatrix} c \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} G \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} G_I & \mathbf{0} \\ \mathbf{0} & G_z \end{bmatrix}, E, id \right\rangle_{PZ} \right\rfloor}(\alpha) \overset{\text{Prop. 3.1.22}}{=} \underline{\lfloor \mathcal{PZ} \times \mathcal{Z} \rfloor}(\alpha),$$

which is identical to (C.2) and therefore concludes the proof. □

For the quadratic map we focus on the case $sq(\mathcal{Q}, \mathcal{PZ})$ with a single SPZ $\mathcal{PZ}$ only since the general case involving two different SPZs is not required for the conservative polynomialization algorithm:

Proposition C.3. *(Dependency Preservation Quadratic Map) Given a SPZ* $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ *and a discrete set of matrices* $\mathcal{Q} = \{Q_1, \ldots, Q_w\}$ *with* $Q_i \in \mathbb{R}^{n \times n}$*,* $i = 1, \ldots, w$*, it holds that the implementation of the quadratic map* $sq(\mathcal{Q}, \mathcal{PZ})$ *in Prop. 3.1.31 is dependency-preserving.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad sq\big(\mathcal{Q}, \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha)\big) \subseteq \underline{\lfloor sq(\mathcal{Q}, \mathcal{PZ}) \rfloor}(\alpha). \tag{C.3}$$

Inserting the evaluation function for SPZs as defined in Def. 4.2.10 and the implementation of the quadratic map in Prop. 3.1.31 into (C.3) yields

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] : \quad sq\big(\mathcal{Q}, \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha)\big)$$

$$\overset{\text{Def. } 4.2.10}{=} sq\Big(\mathcal{Q}, c + \sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)} \oplus \Big\{\sum_{j=1}^{q}\beta_j G_{I(\cdot,j)} \;\Big|\; \beta_j \in [-1,1]\Big\}\Big)$$

$$\overset{(2.6)}{=} \begin{bmatrix} \Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big)^T Q_1\, c \\ \vdots \\ \Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big)^T Q_w\, c \end{bmatrix} + \begin{bmatrix} c^T Q_1\Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big) \\ \vdots \\ c^T Q_w\Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big) \end{bmatrix}$$

$$+ \begin{bmatrix} \Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big)^T Q_1\Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big) \\ \vdots \\ \Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big)^T Q_w\Big(\sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big) \end{bmatrix} \oplus$$

$$\underbrace{\Big\{x \;\Big|\; x_{(i)} = c^T Q_i\, c + \Big(c + \sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big)^T Q_i\Big(\sum_{j=1}^{q}\beta_j G_{I(\cdot,j)}\Big) + \Big(\sum_{j=1}^{q}\beta_j G_{I(\cdot,j)}\Big)^T Q_i\Big(c + \sum_{i=1}^{h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\Big)G_{(\cdot,i)}\Big) + \Big(\sum_{j=1}^{q}\beta_j G_{I(\cdot,j)}\Big)^T Q_i\Big(\sum_{j=1}^{q}\beta_j G_{I(\cdot,j)}\Big),\ \beta_j \in [-1,1],\ i = 1,\dots,w\Big\}}_{\overset{\text{Prop. } 3.1.31}{\subseteq} \langle c_z, G_z\rangle_Z} \tag{C.4}$$

$$\subseteq \sum_{i=1}^{h^2+2h}\Big(\prod_{k=1}^{p}\alpha_{(k)}^{\breve{E}_{(k,i)}}\Big)\breve{G}_{(\cdot,i)} \oplus \langle c_z, G_z\rangle_Z$$

$$\overset{\text{Def. } 4.2.10}{=} \underbrace{\lfloor\langle c_z, \breve{G}, G_z, \breve{E}, id\rangle_{PZ}\rfloor}(\alpha) \overset{\text{Prop. } 3.1.31}{=} \underbrace{\lfloor sq(\mathcal{Q}, \mathcal{PZ})\rfloor}(\alpha),$$

where the zonotope $\langle c_z, G_z\rangle_Z$ is computed according to (3.25) and the matrices $\breve{G}$ and $\breve{E}$ are defined as

$$\breve{G} = \begin{bmatrix}\widehat{G}_1 & \widehat{G}_2 & \overline{G}_1 & \dots & \overline{G}_h\end{bmatrix}, \quad \breve{E} = \begin{bmatrix}E & E & \overline{E}_1 & \dots & \overline{E}_h\end{bmatrix}$$

with

$$\forall j \in \{1,\dots,h\}:$$

$$\overline{E}_j = E + E_{(\cdot,j)} \cdot \mathbf{1}, \quad \overline{G}_j = \begin{bmatrix} G_{(\cdot,j)}^T Q_1 G \\ \vdots \\ G_{(\cdot,j)}^T Q_w G \end{bmatrix}, \quad \widehat{G}_1 = \begin{bmatrix} c^T Q_1^T G \\ \vdots \\ c^T Q_w^T G \end{bmatrix}, \quad \widehat{G}_2 = \begin{bmatrix} c^T Q_1 G \\ \vdots \\ c^T Q_w G \end{bmatrix}.$$

Since (C.4) is identical to (C.3), it holds that the implementation of the quadratic map on SPZs in Prop. 3.1.31 is dependency-preserving. □

For order reduction of SPZs we focus on the case where Girard's method is used for zonotope order reduction for simplicity since considering other zonotope order reduction methods is straightforward. Moreover, we require the following result about order reduction of zonotopes:

Lemma C.4. *Given two zonotopes, $\mathcal{Z}_1 = \langle c_1, G_1 \rangle_Z \subset \mathbb{R}^n$, $\mathcal{Z}_2 = \langle c_2, G_2 \rangle_Z \subset \mathbb{R}^n$, and a desired zonotope order $\rho_d \geq 1$, it holds that*

$$\mathcal{Z}_1 \oplus \texttt{reduce}(\mathcal{Z}_2, \rho_d) \subseteq \texttt{reduce}(\mathcal{Z}_1 \oplus \mathcal{Z}_2, \rho_d)$$

if Girard's method is used for zonotope order reduction.

Proof. From the description of zonotope order reduction using Girard's method in Sec. 2.6, we obtain

$$\begin{aligned}
&\mathcal{Z}_1 \oplus \texttt{reduce}(\mathcal{Z}_2, \rho_d) \overset{(2.19)}{=} \mathcal{Z}_1 \oplus \left\langle c_2, \left[G_{2(\cdot,\mathcal{K})} \ \texttt{diag}\Big(\sum_{i \in \mathcal{H}} |G_{2(\cdot,i)}| \Big) \right] \right\rangle_Z \\
&= \left\langle c_1 + c_2, \left[G_1 \ G_{2(\cdot,\mathcal{K})} \ \texttt{diag}\Big(\sum_{i \in \mathcal{H}} |G_{2(\cdot,i)}| \Big) \right] \right\rangle_Z \\
&\overset{\substack{\mathcal{H} \subseteq \mathcal{H}_2 \\ \mathcal{K}_2 \subseteq \mathcal{K}}}{\subseteq} \left\langle c_1 + c_2, \left[G_{1(\cdot,\mathcal{K}_1)} \ G_{2(\cdot,\mathcal{K}_2)} \ \texttt{diag}\Big(\sum_{i \in \mathcal{H}_1} |G_{1(\cdot,i)}| + \sum_{i \in \mathcal{H}_2} |G_{2(\cdot,i)}| \Big) \right] \right\rangle_Z \\
&\overset{(2.19)}{=} \texttt{reduce}(\mathcal{Z}_1 \oplus \mathcal{Z}_2, \rho_d),
\end{aligned} \tag{C.5}$$

where the transformation in line 3 results in an enclosure since all generators that are selected for reduction in the case $\texttt{reduce}(\mathcal{Z}_2, \rho_d)$ are selected for reduction in the case $\texttt{reudce}(\mathcal{Z}_1 \oplus \mathcal{Z}_2, \rho_d)$, too. The sets

$$\mathcal{K} = \{o_{1(1)}, \ldots, o_{1(b)}\}, \quad \mathcal{H} = \{o_{1(b+1)}, \ldots, o_{1(l_1)}\}, \quad \mathcal{K}_1 = \{o_{2(1)}, \ldots, o_{2(b)}\} \cap \{1, \ldots, l_1\},$$

$$\mathcal{H}_1 = \{o_{2(b+1)}, \ldots, o_{2(l_1+l_2)}\} \cap \{1, \ldots, l_1\}, \quad \mathcal{K}_2 = \{o_{2(1)} - l_1, \ldots, o_{2(b)} - l_1\} \cap \{1, \ldots, l_2\},$$

$$\mathcal{H}_2 = \{o_{2(b+1)} - l_1, \ldots, o_{2(l_1+l_2)} - l_1\} \cap \{1, \ldots, l_2\}$$

with $b = \lfloor n(\rho - 1) \rfloor$ denoting the number of generators that are not reduced are determined from the sorted generators

$$||G_{2(\cdot,o_{1(1)})}||_1 - ||G_{2(\cdot,o_{1(1)})}||_\infty \geq \ldots \geq ||G_{2(\cdot,o_{1(l_1)})}||_1 - ||G_{2(\cdot,o_{1(l_1)})}||_\infty$$

$$||\overline{G}_{(\cdot,o_{2(1)})}||_1 - ||\overline{G}_{(\cdot,o_{2(1)})}||_\infty \geq \ldots \geq ||\overline{G}_{(\cdot,o_{2(l_1+l_2)})}||_1 - ||\overline{G}_{(\cdot,o_{2(l_1+l_2)})}||_\infty,$$

where $\overline{G} = [G_1 \ G_2]$ and $o_1 \in \mathbb{N}^{l_2}$, $o_2 \in \mathbb{N}^{l_1+l_2}$ store the indices of the sorted generators. It remains to show that $\mathcal{H} \subseteq \mathcal{H}_2$ and $\mathcal{K}_2 \subseteq \mathcal{K}$ as used in (C.5) holds. Since the number of generator that are not reduced is identical for both cases $\texttt{reduce}(\mathcal{Z}_2, \rho_d)$ and $\texttt{reduce}(\mathcal{Z}_1 \oplus \mathcal{Z}_2, \rho_d)$, it holds that $|\mathcal{K}| = |\mathcal{K}_1 \cup \mathcal{K}_2| = b$. Consequently, because the matrix $\overline{G}$ contains

all generators in G_2 plus the generators in G_1 which potentially have a larger norm than the generators in G_2, we have

$$||G_{(\cdot,o_{2(b)})}||_1 - ||G_{(\cdot,o_{2(b)})}||_\infty \geq ||G_{2(\cdot,o_{1(b)})}||_1 - ||G_{2(\cdot,o_{1(b)})}||_\infty,$$

so that $\mathcal{K}_2 \subseteq \mathcal{K}$. Since $\mathcal{K}_2 \cup \mathcal{H}_2 = \mathcal{K} \cup \mathcal{H} = \{1, \ldots, l_2\}$, it therefore holds that $\mathcal{H} \subseteq \mathcal{H}_2$. □

Using Lemma C.4, we now prove that order reduction of SPZs is dependency-preserving:

Proposition C.5. *(Dependency Preservation Order Reduction) Given a SPZ $\mathcal{PZ} = \langle c, G, G_I, E, id \rangle_{PZ} \subset \mathbb{R}^n$ and a desired zonotope order $\rho_d \geq 1$, it holds that the implementation of* $\texttt{reduce}(\mathcal{PZ}, \rho_d)$ *in Prop. 3.1.39 is dependency-preserving if Girard's method is used for zonotope order reduction.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{reduce}(\underline{\lfloor \mathcal{PZ} \rfloor}(\alpha), \rho_d) \subseteq \underline{\lfloor \texttt{reduce}(\mathcal{PZ}, \rho_d) \rfloor}(\alpha, id),$$

where we use the extended definition of the evaluation function in Def. 4.2.11 since the `reduce` operation potentially decreases the number of dependent factors due to the removal of dependent factors that do not occur anymore after reduction. For the proof, we require the sets of indices $\mathcal{H}_1$, $\mathcal{H}_2$, $\overline{\mathcal{H}}_1$, $\overline{\mathcal{H}}_2$, $\mathcal{K}$, and $\overline{\mathcal{K}}$ defined by the sorted generators

$$\begin{aligned} &\underbrace{||G_{I(\cdot,o_{1(1)})}||_2 \geq \ldots \geq ||G_{I(\cdot,o_{1(b)})}||_2}_{G_{I(\cdot,i)},\ i \in \overline{\mathcal{H}}_1} \geq \underbrace{||G_{I(\cdot,o_{1(b+1)})}||_2 \geq \ldots \geq ||G_{I(\cdot,o_{1(q)})}||_2}_{G_{I(\cdot,i)},\ i \in \mathcal{H}_1} \\ &\underbrace{||\overline{G}_{(\cdot,o_{2(1)})}||_2 \geq \ldots \geq ||\overline{G}_{(\cdot,o_{2(b)})}||_2}_{G_{(\cdot,i)},\ i \in \overline{\mathcal{K}} \text{ and } G_{I(\cdot,i)},\ i \in \overline{\mathcal{H}}_2} \geq \underbrace{||\overline{G}_{(\cdot,o_{2(b+1)})}||_2 \geq \ldots \geq ||\overline{G}_{(\cdot,o_{2(h+q)})}||_2}_{G_{(\cdot,i)},\ i \in \mathcal{K} \text{ and } G_{I(\cdot,i)},\ i \in \mathcal{H}_2}, \end{aligned} \tag{C.6}$$

where $\overline{G} = [G\ G_I]$, $b = \lfloor n(\rho_d - 1) \rfloor$ denotes the number of generators that are not reduced, and $o_1 \in \mathbb{N}^q$, $o_2 \in \mathbb{N}^{h+q}$ store the indices of the sorted generators. For $\texttt{reduce}(\underline{\lfloor \mathcal{PZ} \rfloor}(\alpha), \rho_d)$ we then obtain

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{reduce}(\underline{\lfloor \mathcal{PZ} \rfloor}(\alpha), \rho_d) \overset{\text{Def. 4.2.10}}{=}$$

$$\texttt{reduce}\left(\left\langle c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)}, [\,], G_I, [\,], [\,] \right\rangle_{PZ}, \rho_d\right) \overset{\text{Prop. 3.1.39}}{\subseteq}$$

$$\left\langle c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)}, [\,], [G_{I(\cdot,\overline{\mathcal{H}}_1)}\ G_{z,1}], [\,], [\,] \right\rangle_{PZ} \overset{\text{Prop. 3.1.39}}{=} \tag{C.7}$$

$$c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)} \oplus \langle \mathbf{0}, G_{I(\cdot,\overline{\mathcal{H}}_1)} \rangle_Z \oplus \underbrace{\texttt{reduce}\big(\langle \mathbf{0}, G_{I(\cdot,\mathcal{H}_1)} \rangle_Z, 1\big)}_{\langle \mathbf{0}, G_{z,1} \rangle_Z} \subseteq$$

$$\sum_{i\in\overline{\mathcal{K}}}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)}\oplus\underbrace{\texttt{zonotope}\big(\langle c,G_{(\cdot,\mathcal{K})},[\,],E_{(\cdot,\mathcal{K})},id\rangle_{PZ}\big)}_{\mathcal{Z}}$$
$$\oplus\langle\mathbf{0},G_{I(\cdot,\overline{\mathcal{H}}_1)}\rangle_Z\oplus\texttt{reduce}\big(\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_1)}\rangle_Z,1\big),$$

where we exploited for the transformation in the last line that $\overline{\mathcal{K}}\cup\mathcal{K}=\{1,\ldots,h\}$ holds according to (C.6). Next, we apply Lemma C.4 to obtain a relation between $\texttt{reduce}(\lfloor\underline{\mathcal{PZ}}\rfloor(\alpha),\rho_d)$ and $\lfloor\underline{\texttt{reduce}(\mathcal{PZ},\rho_d)}\rfloor(\alpha,id)$. For this, we first have to show that all independent generators that are reduced in the case $\texttt{reduce}(\lfloor\underline{\mathcal{PZ}}\rfloor(\alpha),\rho_d)$ are also reduced in the case $\lfloor\underline{\texttt{reduce}(\mathcal{PZ},\rho_d)}\rfloor(\alpha,id)$, which corresponds to the relation $\mathcal{H}_1\subseteq\mathcal{H}_2$. Since the matrix $\overline{G}$ contains all generators in G_I plus the generators in G which potentially have a larger Euclidean norm than the generators in G_I, we have

$$||\overline{G}_{(\cdot,o_{2(b)})}||_2\geq||G_{I(\cdot,o_{1(b)})}||_2,$$

which proves that $\overline{\mathcal{H}}_2\subseteq\overline{\mathcal{H}}_1$ according to (C.6). Because $\overline{\mathcal{H}}_1\cup\mathcal{H}_1=\overline{\mathcal{H}}_2\cup\mathcal{H}_2=\{1,\ldots,q\}$, we then have that $\overline{\mathcal{H}}_2\subseteq\overline{\mathcal{H}}_1$ implies $\mathcal{H}_1\subseteq\mathcal{H}_2$. Moreover, $\mathcal{H}_1\subseteq\mathcal{H}_2$ and $\overline{\mathcal{H}}_2\subseteq\overline{\mathcal{H}}_1$ implies that there exist a set $\mathcal{M}$ satisfying $\mathcal{H}_2=\mathcal{H}_1\cup\mathcal{M}$ and $\overline{\mathcal{H}}_1=\overline{\mathcal{H}}_2\cup\mathcal{M}$, so that we obtain for (C.7)

$$\forall\alpha\in[-\mathbf{1},\mathbf{1}]:\quad\texttt{reduce}(\lfloor\underline{\mathcal{PZ}}\rfloor(\alpha),\rho_d)\overset{\text{(C.7)}}{=}$$

$$\sum_{i\in\overline{\mathcal{K}}}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)}\oplus\mathcal{Z}\oplus\langle\mathbf{0},G_{I(\cdot,\overline{\mathcal{H}}_1)}\rangle_Z\oplus\texttt{reduce}\big(\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_1)}\rangle_Z,1\big)\overset{\overline{\mathcal{H}}_1=\overline{\mathcal{H}}_2\cup\mathcal{M}}{=}$$

$$\sum_{i\in\overline{\mathcal{K}}}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)}\oplus\mathcal{Z}\oplus\langle\mathbf{0},G_{I(\cdot,\overline{\mathcal{H}}_2)}\rangle_Z\oplus\langle\mathbf{0},G_{I(\cdot,\mathcal{M})}\rangle_Z \tag{C.8}$$
$$\oplus\texttt{reduce}\big(\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_1)}\rangle_Z,1\big)\overset{\text{Lemma C.4}}{\subseteq}$$

$$\sum_{i\in\overline{\mathcal{K}}}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)}\oplus\langle\mathbf{0},G_{I(\cdot,\overline{\mathcal{H}}_2)}\rangle_Z\oplus\texttt{reduce}\big(\underbrace{\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_1)}\rangle_Z\oplus\langle\mathbf{0},G_{I(\cdot,\mathcal{M})}\rangle_Z}_{\mathcal{H}_2=\mathcal{H}_1\cup\mathcal{M}}\oplus\mathcal{Z},1\big).$$
$$\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_2)}\rangle_Z$$

Finally, using the definitions of the evaluation function and the extended evaluation function for SPZs in Def. 4.2.10 and Def. 4.2.11, we can show that (C.8) is identical to $\lfloor\underline{\texttt{reduce}(\mathcal{PZ},\rho_d)}\rfloor(\alpha,id)$:

$$\forall\alpha\in[-\mathbf{1},\mathbf{1}]:\quad\texttt{reduce}(\lfloor\underline{\mathcal{PZ}}\rfloor(\alpha),\rho_d)\overset{\text{(C.8)}}{=}$$

$$\underbrace{\sum_{i\in\overline{\mathcal{K}}}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)}\oplus\langle\mathbf{0},G_{I(\cdot,\overline{\mathcal{H}}_2)}\rangle_Z}_{\lfloor\langle\mathbf{0},G_{(\cdot,\overline{\mathcal{K}})},G_{I(\cdot,\overline{\mathcal{H}}_2)},E_{(\cdot,\overline{\mathcal{K}})},id\rangle_{PZ}\rfloor(\alpha)}\oplus\underbrace{\texttt{reduce}\big(\langle\mathbf{0},G_{I(\cdot,\mathcal{H}_2)}\rangle_Z\oplus\mathcal{Z},1\big)}_{\langle c_{z,2},G_{z,2}\rangle_Z}\overset{\text{Def. 4.2.10}}{=}$$

$$\underbrace{\big\langle c_{z,2},G_{(\cdot,\overline{\mathcal{K}})},[G_{I(\cdot,\overline{\mathcal{H}}_2)}\ G_{z,2}],E_{(\cdot,\overline{\mathcal{K}})},id\big\rangle_{PZ}}(\alpha)\overset{\text{Def. 4.2.11}}{=}$$

$$\underbrace{\big\langle c_{z,2},G_{(\cdot,\overline{\mathcal{K}})},[G_{I(\cdot,\overline{\mathcal{H}}_2)}\ G_{z,2}],E_{(\mathcal{N},\overline{\mathcal{K}})},id_{(\mathcal{N})}\big\rangle_{PZ}}(\alpha,id)\overset{\text{Prop. 3.1.39}}{=}\underbrace{\texttt{reduce}(\mathcal{PZ},\rho_d)}(\alpha,id),$$

where the set $\mathcal{N}$ that removes all-zero rows from the exponent matrix is defined as in (3.33). □

Finally, we consider the `restructure` operation:

Proposition C.6. *(Dependency Preservation Restructuring) Given a SPZ* $\mathcal{PZ}=\langle c,G,G_I,E,id\rangle_{PZ}\subset\mathbb{R}^n$ *and an upper bound for the number of dependent factors* $p_d\geq n$*, it holds that the implementation of* `restructure`$(\mathcal{PZ},p_d)$ *in Prop. 3.1.41 is dependency-preserving.*

Proof. According to the definition of dependency preservation in Def. 4.2.4, we have to show that

$$\begin{aligned}\forall\alpha\in[-\mathbf{1},\mathbf{1}]:\ &\texttt{restructure}(\underbrace{\mathcal{PZ}}(\alpha),p_d)\\&\subseteq\underbrace{\texttt{restructure}(\mathcal{PZ},p_d)}\big(\alpha,id\big),\end{aligned}\tag{C.9}$$

where we have to use the extended definition of the evaluation function in Def. 4.2.4 since the `restructure` operation increases the number of dependent factors. The `restructure` operation on SPZs as specified in Prop. 3.1.41 is computed with the case distinction

$$\texttt{restructure}(\mathcal{PZ},p_d)=\begin{cases}\mathcal{PZ}_1, & p\leq p_d-n\\ \mathcal{PZ}_2, & \text{otherwise}\end{cases},\tag{C.10}$$

which depends on the desired number of dependent factors p_d. Since the SPZ $\underbrace{\mathcal{PZ}}(\alpha)$ in (C.9) has no dependent factors ($p=0$), we obtain according to (C.10) $\texttt{restructure}(\underbrace{\mathcal{PZ}}(\alpha),p_d)=\underbrace{\mathcal{PZ}_1}(\alpha)$. For $\underbrace{\texttt{restructure}(\mathcal{PZ},p_d)}\big(\alpha,id\big)$ in (C.9), however, both cases in (C.10) can occur. We first consider the case $p\leq p_d-n$ in (C.10), so that we obtain for (C.9)

$$\forall\alpha\in[-\mathbf{1},\mathbf{1}]:\ \ \texttt{restructure}(\underbrace{\mathcal{PZ}}(\alpha),p_d)$$

$$\overset{\text{Def. 4.2.10}}{=}\texttt{restructure}\left(\left\langle c+\sum_{i=1}^{h}\left(\prod_{k=1}^{p}\alpha_{(k)}^{E_{(k,i)}}\right)G_{(\cdot,i)},[\,],G_I,[\,],[\,]\right\rangle_{PZ},p_d\right)$$

$$\overset{\text{Prop. 3.1.41}}{=} \left\langle c_z + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)}, G_z, [\,], I_n, \texttt{uniqueID}(n) \right\rangle_{PZ}$$

$$= \underbrace{c_z + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{(k,i)}} \right) G_{(\cdot,i)}}_{\lfloor \langle c_z, G, [\,], E, id \rangle_{PZ} \rfloor(\alpha)} \oplus \left\{ \sum_{i=1}^{n} G_{z(\cdot,i)} \breve{\alpha}_{(i)} \;\middle|\; \breve{\alpha} \in [-\mathbf{1}, \mathbf{1}] \right\} \tag{C.11}$$

$$\overset{\text{Def. 4.2.11}}{=} \underbrace{\left\langle c_z, [G\ G_z], [\,], \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_n \end{bmatrix}, [id\ \texttt{uniqueID}(n)] \right\rangle_{PZ}}(\alpha, id)$$

$$\overset{\text{Prop. 3.1.41}}{=} \underline{\texttt{restructure}(\mathcal{PZ}_1, p_d)}(\alpha, id) \overset{p \leq p_d - n}{=} \underline{\texttt{restructure}(\mathcal{PZ}, p_d)}(\alpha, id),$$

where $\langle c_z, G_z \rangle_Z = \texttt{reduce}(\langle c, G_I \rangle_Z, 1)$. Next, we consider the case $p > p_d - n$ in (C.10), for which dependent factors are removed from $\mathcal{PZ}_1$ to satisfy the upper bound p_d. Consequently, we obtain for (C.9)

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}]: \quad \texttt{restructure}(\underline{\mathcal{PZ}}(\alpha), p_d)$$

$$\overset{\text{(C.11)}}{=} \underbrace{\underbrace{\left\langle c_z, [G\ G_z], [\,], \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_n \end{bmatrix}, [id\ \texttt{uniqueID}(n)] \right\rangle_{PZ}}_{\lfloor \langle c_z, G_1, [\,], E_1, id_1 \rangle_{PZ} \rfloor(\alpha, id)}(\alpha, id)}$$

$$\overset{\text{Def. 4.2.11}}{=} \left\{ c_z + \sum_{i=1}^{h+n} \left(\prod_{k=1}^{p} \alpha_{(k)}^{E_{1(k,i)}} \right) \left(\prod_{k=p+1}^{p+n} \breve{\alpha}_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} \;\middle|\; \breve{\alpha}_k \in [-1, 1] \right\}$$

$$\subseteq \left\{ \sum_{i \in \overline{\mathcal{H}}} \left(\prod_{k \in \mathcal{M}} \alpha_{(k)}^{E_{1(k,i)}} \right) \left(\prod_{k \in \mathcal{F}} \breve{\alpha}_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} \;\middle|\; \breve{\alpha}_k \in [-1, 1] \right\} \oplus \underbrace{\texttt{zonotope}\big(\langle c_z, G_{1(\cdot,\mathcal{H})}, [\,], E_{1(\cdot,\mathcal{H})}, id_1 \rangle_{PZ}\big)}_{\langle \overline{c}_z, \overline{G}_z \rangle_Z}$$

$$\overset{\text{Def. 4.2.11}}{=} \underline{\left\langle \overline{c}_z, G_{1(\cdot,\overline{\mathcal{H}})}, \overline{G}_z, E_{1(\overline{\mathcal{N}},\overline{\mathcal{H}})}, id_{1(\overline{\mathcal{N}})} \right\rangle_{PZ}}(\alpha, id)$$

$$\overset{\text{Prop. 3.1.41}}{=} \underline{\texttt{restructure}(\mathcal{PZ}_2, p_d)}(\alpha, id) \overset{p > p_d - n}{=} \underline{\texttt{restructure}(\mathcal{PZ}, p_d)}(\alpha, id),$$

where

$$\mathcal{M} = \{1, \ldots, p\} \cap \overline{\mathcal{N}}, \quad \mathcal{F} = \{p+1, \ldots, p+n\} \cap \overline{\mathcal{N}},$$

and the sets of indices $\overline{\mathcal{N}}$, $\mathcal{H}$, and $\overline{\mathcal{H}}$ are defined as in Prop. 3.1.41. □

Bibliography

[1] M. Althoff. An introduction to CORA 2015. In *Proc. of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[2] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer Science & Business Media, 2001.

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

[4] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. In *Proc. of the IFAC World Congress*, pages 8966–8971, 2008.

[5] M. Althoff and G. Frehse. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *Proc. of the International Conference on Decision and Control*, pages 7439–7446, 2016.

[6] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. Eliminating spurious transitions in reachability with support functions. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 149–158, 2015.

[7] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *Proc. of the International Conference on Analysis and Design of Hybrid Systems*, pages 94–101, 2012.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[9] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *Transactions on Automatic Control*, 52(1):26–38, 2007.

[10] D. Bertsekas and I. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *Transactions on Automatic Control*, 16(2):117–128, 1971.

[11] B. Legat, P. Tabuada, and R. M. Jungers. Computing controlled invariant sets for hybrid systems with applications to model-predictive control. In *Proc. of the International Conference on Analysis and Design of Hybrid Systems*, pages 193 – 198, 2018.

[12] T. Alamo, A. Cepeda, and D. Limon. Improved computation of ellipsoidal invariant sets for saturated control systems. In *Proc. of the International Conference on Decision and Control*, pages 6216–6221, 2005.

[13] B. Grünbaum. *Convex Polytopes*. Graduate Texts in Mathematics. Springer Science & Business Media, 2003.

[14] H. R. Tiwary. On the hardness of computing intersection, union and Minkowski sum of polytopes. *Discrete & Computational Geometry*, 40(3):469–479, 2008.

[15] H. R. Tiwary. On computing the shadows and slices of polytopes. *arXiv:0804.4150*, 2008, Preprint.

[16] V. Kaibel and M. E. Pfetsch. *Algebra, Geometry and Software Systems*, chapter Some Algorithmic Problems in Polytope Theory, pages 23–47. Springer Science & Business Media, 2003.

[17] Z. Han and B. H. Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 287–301, 2006.

[18] B. Xue, Z. She, and A. Easwaran. Under-approximating backward reachable sets by polytopes. In *Proc. of the International Conference on Computer Aided Verification*, pages 457–476, 2016.

[19] F. Anstett, G. Millérioux, and G. Bloch. Polytopic observer design for LPV systems based on minimal convex polytope finding. *Journal of Algorithms & Computational Technology*, 3(1):23–43, 2009.

[20] M. Rungger and P. Tabuada. Computing robust controlled invariant sets of linear systems. *Transactions on Automatic Control*, 62(7):3665–3670, 2017.

[21] M. A. Ben Sassi and A. Girard. Computation of polytopic invariants for polynomial dynamical systems using linear programming. *Automatica*, 48(12):3114–3121, 2012.

[22] M. Althoff, G. Frehse, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:369–395, 2020.

[23] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer Science & Business Media, 2012.

[24] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technical University of Munich, 2010.

[25] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 291–305, 2005.

[26] T. Alamo, J. M. Bravo, and E. F. Camacho. Guaranteed state estimation by zonotopes. In *Proc. of the International Conference on Decision and Control*, pages 5831–5836, 2003.

[27] C. Combastel. Zonotopes and Kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence. *Automatica*, 55:265–273, 2015.

[28] F. Gruber and M. Althoff. Computing safe sets of linear sampled-data systems. *Control Systems Letters*, 5(2):385–390, 2021.

[29] E. Goubault and S. Putot. A zonotopic framework for functional abstractions. *Formal Methods in System Design*, 47(3):302–360, 2015.

[30] L. H. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[31] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *Proc. of the International Conference on Decision and Control*, pages 6814–6821, 2011.

[32] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[33] A. S. Adimoolam and T. Dang. Using complex zonotopes for stability verification. In *Proc. of the American Control Conference*, pages 4269–4274, 2016.

[34] K. Ghorbal, E. Goubault, and S. Putot. A logical product approach to zonotope intersection. In *Proc. of the International Conference on Computer Aided Verification*, pages 212–226, 2010.

[35] E. K. Kostousova. State estimation for dynamic systems via parallelotopes optimization and parallel computations. *Optimization Methods and Software*, 9(4):269–306, 1998.

[36] L. Chisci, A. Garulli, and G. Zappa. Recursive state bounding by parallelotopes. *Automatica*, 32(7):1049–1055, 1996.

[37] L. Jaulin, M. Kieffer, and O. Didrit. *Applied Interval Analysis*. Springer Science & Business Media, 2006.

[38] A. Eggers, N. Ramdani, N. S. Nedialkov, and M. Fränzle. Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. *Software & Systems Modeling*, 14(1):121–148, 2012.

[39] N. Ramdani and N. S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162, 2010.

[40] A. El-Guindy, D. Han, and M. Althoff. Estimating the region of attraction via forward reachable sets. In *Proc. of the American Control Conference*, pages 1263–1270, 2017.

[41] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.

[42] K. Makino and M. Berz. Verified global optimization with Taylor model based range bounders. *Transactions on Computers*, 4(11):1611–1618, 2005.

[43] X. Chen, S. Sankaranarayanan, and E. Ábrahám. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of the Real-Time Systems Symposium*, pages 183–192, 2012.

[44] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of the International Conference on Computer-Aided Verification*, pages 258–263, 2013.

[45] K. Makino and M. Berz. Rigorous integration of flows and ODEs using Taylor models. In *Proc. of the International Conference on Symbolic Numeric Computation*, pages 79–84, 2009.

[46] M. Neher, K. R. Jackson, and N. S. Nedialkov. On Taylor model based integration of ODEs. *Journal on Numerical Analysis*, 45(1):236–262, 2007.

[47] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 173–182, 2013.

[48] A. Adjé, P.-L. Garoche, and A. Werey. Quadratic zonotopes. In *Proc. of the Asian Symposium on Programming Languages and Systems*, pages 127–145, 2015.

[49] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In *Proc. of the International Conference on Computer Aided Verification*, pages 477–494, 2016.

[50] S. Bak and P. S. Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 173–178, 2017.

[51] S. Bak and P. S. Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In *Proc. of the International Conference on Computer Aided Verification*, pages 401–420, 2017.

[52] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. *Transactions on Automatic Control*, 50(7):947–957, 2005.

[53] B. Xue, M. Fränzle, and N. Zhan. Under-approximating reach sets for polynomial continuous systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 51–60, 2018.

[54] W. Tan and A. Packard. Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming. *Transactions on Automatic Control*, 53(2):565–571, 2008.

[55] M. Korda, D. Henrion, and C. N. Jones. Convex computation of the maximum controlled invariant set for polynomial control systems. *Journal on Control and Optimization*, 52(5):2944–2969, 2014.

[56] A. Adjé, P.-L. Garoche, and V. Magron. Property-based polynomial invariant generation using sums-of-squares optimization. In *Proc. of the International Static Analysis Symposium*, pages 235–251, 2015.

[57] S. de Oliveira, S. Bensalem, and V. Prevosto. Synthesizing invariants by solving solvable loops. In *Proc. of the International Symposium on Automated Technology for Verification and Analysis*, pages 327–343, 2017.

[58] I. Bárány. A generalization of Carathéodory's theorem. *Discrete Mathematics*, 40(2):141–152, 1982.

[59] D. E. Knuth. *The Art of Computer Programming.* Addison-Wesley, 1997.

[60] V. Y. Pan and Z. Q. Chen. The complexity of the matrix eigenproblem. In *Proc. of the Annual Symposium on Theory of Computing*, pages 507–516, 1999.

[61] T. F. Chan. An improved algorithm for computing the singular value decomposition. *Transactions on Mathematical Software*, 8(1):72–83, 1982.

[62] Q. Du and J. E. Fowler. Low-complexity principal component analysis for hyperspectral image compression. *The International Journal of High Performance Computing Applications*, 22(4):438–448, 2008.

[63] M. Kallay. Convex hull algorithms for higher dimensions. 1981.

[64] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer Science & Business Media, 2012.

[65] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proc. of the Annual Symposium on Theory of Computing*, pages 302–311, 1984.

[66] Y. Ye and E. Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44(1-3):157–179, 1989.

[67] D. S. Watkins. *Fundamentals of Matrix Computations.* John Wiley & Sons, 2004.

[68] A.-K. Kopetzki, B. Schürmann, and M. Althoff. Methods for order reduction of zonotopes. In *Proc. of the International Conference on Decision and Control*, pages 5626–5633, 2017.

[69] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

[70] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[71] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[72] G. T. Cargo and O. Shisha. The Bernstein form of a polynomial. *Journal of Research of the National Bureau of Standards*, 70B(1):79–81, 1966.

[73] S. Ray and P. Nataraj. A matrix method for efficient computation of Bernstein coefficients. *Reliable Computing*, 17:40–71, 2012.

[74] M. Althoff, D. Grebenyuk, and N. Kochdumper. Implementation of Taylor models in CORA 2018. In *Proc. of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 145–173, 2018.

[75] G. Trombettoni, Y. Papegay, G. Chabert, and O. Pourtallier. A box-consistency contractor based on extremal functions. In *Proc. of the International Conference on Principles and Practice of Constraint Programming*, pages 491–498, 2010.

[76] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *Transactions on Automatic Control*, 66(9):4043–4058, 2021.

[77] M. Reimer. *Multivariate Polynomial Approximation*. Birkhäuser, 2012.

[78] O. Mullier, E. Goubault, M. Kieffer, and S. Putot. General inner approximation of vector-valued functions. *Reliable Computing*, 18:117–143, 2013.

[79] A. Goldsztejn and L. Jaulin. Inner approximation of the range of vector-valued functions. *Reliable Computing*, 14:1–23, 2010.

[80] N. Kochdumper and M. Althoff. Constrained polynomial zonotopes. To appear in: *Acta Informatica*, 2021.

[81] J. Garloff. The Bernstein algorithm. *Interval Computations*, 2(6):154–168, 1993.

[82] S. Kousik, B. Zhang, P. Zhao, and R. Vasudevan. Safe, optimal, real-time trajectory planning with a parallel constrained Bernstein algorithm. *Transactions on Robotics*, 37(3):815–830, 2020.

[83] M. Althoff. On computing the Minkowski difference of zonotopes. *arXiv:1512.02794*, 2015, Preprint.

[84] I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4:317–367, 1998.

[85] N. Kochdumper and M. Althoff. Representation of polytopes as polynomial zonotopes. *arXiv:1910.07271*, 2019, Preprint.

[86] A. Jeffrey and H. H. Dai. *Handbook of Mathematical Formulas and Integrals*. Elsevier, 2008.

[87] M. Padberg. *Linear Optimization and Extensions*. Springer Science & Business Media, 2013.

[88] L. Rade and B. Westergren. *Mathematics Handbook for Science and Engineering*. Springer Science & Business Media, 2013.

[89] E. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184, 1970.

[90] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: Computational complexity and applications to Gröbner bases. *Journal on Discrete Mathematics*, 6(2):246–269, 1993.

[91] N. Matringe, A. V. Moura, and R. Rebiha. Generating invariants for non-linear hybrid systems by linear algebraic methods. In *Proc. of the International Static Analysis Symposium*, pages 373–389, 2010.

[92] S. Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 221–230, 2010.

[93] H. Kong, S. Bogomolov, C. Schilling, Y. Jiang, and T. A. Henzinger. Safety verification of nonlinear hybrid systems based on invariant clusters. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 163–172, 2017.

[94] A. Chutinan and B. H. Krogh. Computational techniques for hybrid system verification. *Transactions on Automatic Control*, 48(1):64–75, 2003.

[95] T. Dang and D. Salinas. Image computation for polynomial dynamical systems using the Bernstein expansion. In *Proc. of the International Conference on Computer Aided Verification*, pages 219–232, 2009.

[96] P. S. Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In *Proc. of the International Conference on Embedded Software*, 2013, Article 26.

[97] F. Immler. Verified reachability analysis of continuous systems. In *Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 37–51, 2015.

[98] J. A. dit Sandretto and A. Chapoutot. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22:79–103, 2016.

[99] J. Hoefkens, M. Berz, and K. Makino. *Scientific Computing, Validated Numerics, Interval Methods*, chapter Verified High-Order Integration of DAEs and Higher-Order ODEs, pages 281–292. Springer Science & Business Media, 2013.

[100] N. Ramdani, N. Meslem, and Y. Candau. Reachability analysis of uncertain nonlinear systems using guaranteed set integration. In *Proc. of the IFAC World Congress*, pages 8972–8977, 2008.

[101] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *Transactions on Automatic Control*, 43(4):540–554, 1998.

[102] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 20–35, 2003.

[103] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[104] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of the International Conference on Decision and Control*, pages 4042–4048, 2008.

[105] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 11–19, 2010.

[106] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *International Journal of Robust and Nonlinear Control*, 24(4):699–724, 2014.

[107] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 68–82, 2015.

[108] J. A. dit Sandretto and A. Chapoutot. DynIbex: A differential constraint library for studying dynamical systems. In *International Conference on Hybrid Systems: Computation and Control*, 2016, Poster.

[109] F. Immler. Tool presentation: Isabelle/HOL for reachability analysis of continuous systems. In *Proc. of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 180–187, 2015.

[110] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. JuliaReach: A toolbox for set-based reachability. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

[111] F. Immler, M. Althoff, L. Benet, A. Chapoutot, X. Chen, M. Forets, L. Geretti, N. Kochdumper, D. P. Sanders, and C. Schilling. ARCH-COMP19 category report: Continuous and hybrid systems with nonlinear dynamics. In *Proc. of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 41–61, 2019.

[112] M. Althoff, S. Bak, M. Forets, G. Frehse, N. Kochdumper, R. Ray, C. Schilling, and S. Schupp. ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proc. of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 14–40, 2019.

[113] S. Bak, S. Bogomolov, and M. Althoff. Time-triggered conversion of guards for reachability analysis of hybrid automata. In *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*, pages 133–150, 2017.

[114] M. Maïga, N. Ramdani, L. Travé-Massuyè, and C. Combastel. A comprehensive method for reachability analysis of uncertain nonlinear hybrid systems. *Transactions on Automatic Control*, 61(9):2341–2356, 2015.

[115] N. Kochdumper, B. Schürmann, and M. Althoff. Utilizing dependencies to obtain subsets of reachable sets. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2020, Article 1.

[116] X. Chen, S. Sankaranarayanan, and E. Ábrahám. Under-approximate flowpipes for non-linear continuous systems. In *Proc. of the International Conference on Formal Methods in Computer-Aided Design*, pages 59–66, 2014.

[117] S. Bogomolov, G. Frehse, A. Gurung, D. Li, G. Martius, and R. Ray. Falsification of hybrid systems using symbolic reachability and trajectory splicing. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2019.

[118] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivančić, A.Gupta, and G. J. Pappas. Monte-Carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 211–220, 2010.

[119] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo. Two-layered falsification of hybrid systems guided by Monte Carlo tree search. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2894–2905, 2018.

[120] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 125–134, 2012.

[121] Y. S. R. Annapureddy and G. E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proc. of the Annual Conference of the IEEE Industrial Electronics Society*, pages 91–96, 2010.

[122] Y. S. R. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257, 2011.

[123] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Proc. of the International Conference on Computer Aided Verification*, pages 167–170, 2010.

[124] B. Schürmann and M. Althoff. Closed-form expressions of convex combinations for controller design. In *Proc. of the American Control Conference*, pages 2795–2801, 2016.

[125] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications. In *Proc. of the International Workshop on Formal Approaches to Software Testing and Runtime Verification*, pages 178–192, 2006.

[126] D. Heß, M. Althoff, and T. Sattel. Formal verification of maneuver automata for parameterized motion primitives. In *Proc. of the International Conference on Intelligent Robots and Systems*, pages 1474–1481, 2014.

[127] B. Schürmann and M. Althoff. Convex interpolation control with formal guarantees for disturbed and constrained nonlinear systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 121–130, 2017.

[128] B. Schürmann and M. Althoff. Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *Proc. of the American Control Conference*, pages 2522–2529, 2017.

[129] B. Schürmann and M. Althoff. Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. In *Proc. of the IFAC World Congress*, pages 11515–11522, 2017.

[130] N. Kochdumper and M. Althoff. Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems. In *Proc. of the International Conference on Decision and Control*, pages 2130–2137, 2020.

[131] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 257–271, 2006.

[132] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 202–214, 2000.

[133] A. Hamadeh and J. Goncalves. Reachability analysis of continuous-time piecewise affine systems. *Automatica*, 44(12):3189–3194, 2008.

[134] E. Goubault, O. Mullier, S. Putot, and M. Kieffer. Inner approximated reachability analysis. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 163–172, 2014.

[135] E. Goubault and S. Putot. Robust under-approximations and application to reachability of non-linear control systems with disturbances. *Control Systems Letters*, 4(4):928–933, 2020.

[136] M. Li, P. N. Mosaad, M. Fränzle, Z. She, and B. Xue. Safe over- and under-approximation of reachable sets for autonomous dynamical systems. In *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*, pages 252–270, 2018.

[137] B. Xue, M. Fränzle, and N. Zhan. Inner-approximating reachable sets for polynomial systems with time-varying uncertainties. *Transactions on Automatic Control*, 65(4):1468–1483, 2019.

[138] B. Xue, P. N. Mosaad, M. Fränzle, M. Chen, Y. Li, and N. Zhan. Safe over- and under-approximation of reachable sets for delay differential equations. In *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*, pages 281–299, 2017.

[139] E. Goubault and S. Putot. Forward inner-approximated reachability of non-linear continuous systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 1–10, 2017.

[140] E. Goubault and S. Putot. Inner and outer reachability for the verification of control systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 11–22, 2019.

[141] B. Xue, A. Easwaran, N.-J. Cho, and M. Fränzle. Reach-avoid verification for nonlinear systems based on boundary analysis. *Transactions on Automatic Control*, 62(7):3518–3523, 2016.

[142] N. S. Nedialkov and M. von Mohrenschildt. Rigorous simulation of hybrid dynamic systems with symbolic and interval methods. In *Proc. of the American Control Conference*, pages 140–147, 2002.

[143] X. Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.

[144] N. Kochdumper and M. Althoff. Reachability analysis for hybrid systems with nonlinear guard sets. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2020, Article 2.

[145] 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 *Proc. of the International Conference on Computer Aided Verification*, pages 379–395, 2011.

[146] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 215–228, 2008.

[147] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 45–54, 2012.

[148] S. Schupp, E. Abraham, I. B. Makhlouf, and S. Kowalewski. HyPro: A C++ library of state set representations for hybrid systems reachability analysis. In *Proc. of the NASA Formal Methods Symposium*, pages 288–294, 2017.

[149] M. Maïga, N. Ramdani, and L. Travé-Massuyès. A fast method for solving guard set intersection in nonlinear hybrid reachability. In *Proc. of the International Conference on Decision and Control*, pages 508–513, 2013.

[150] P. Collins, E. Bresolin, L. Geretti, and T. Villa. Computing the evolution of hybrid systems using rigorous function calculus. In *Proc. of the International Conference on Analysis and Design of Hybrid Systems*, pages 284–290, 2012.

[151] N. Chan and S. Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. In *Proc. of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 20–32, 2017.

[152] C. Combastel. A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. In *Proc. of the International Conference on Decision and Control*, pages 7228–7234, 2005.

[153] F. Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *Transactions on Automatic Control*, 13(1):22–28, 1968.

[154] Y. Wang, M. Zhou, V. Puig, G. Cembrano, and Z. Wang. Zonotopic fault detection observer with H_ performance. In *Proc. of the Chinese Control Conference*, pages 7230–7235, 2017.

[155] D. G. Luenberger. Observing the state of a linear system. *Transactions on Military Electronics*, 8(2):74–80, 1964.

[156] F. Mazenc and O. Bernard. Interval observers for linear time-invariant systems with disturbances. *Automatica*, 47(1):140–147, 2011.

[157] Y. Wang, D. M. Bevly, and B. Rajamani. Interval observer design for LPV systems with parametric uncertainty. *Automatica*, 60:79–85, 2015.

[158] W. Tang, Z. Wang, Y. Wang, T. Raïssi, and Y. Shen. Interval estimation methods for discrete-time linear time-invariant systems. *Transactions on Automatic Control*, 64(11):4717–4724, 2019.

[159] M. Althoff and J. J. Rath. Comparison of guaranteed state estimators for linear time-invariant systems. *Automatica*, 130, 2021, Article 109662.

[160] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Non-linear loop invariant generation using Gröbner bases. In *Proc. of the Annual Symposium on Principles of Programming Languages*, pages 318–329, 2004.

[161] P. Garg, C. Löding, P. Madhusudan, and D. Neider. ICE: A robust framework for learning invariants. In *Proc. of the International Conference on Computer Aided Verification*, pages 69–87, 2014.

[162] T. Nguyen, T. Antonopoulos, A. Ruef, and M. Hicks. Counterexample-guided approach to finding numerical invariants. In *Proc. of the Joint Meeting on Foundations of Software Engineering*, pages 605–615, 2017.

[163] A. Izycheva, E. Darulova, and H. Seidl. Counterexample- and simulation-guided floating-point loop invariant synthesis. In *Proc. of the International Static Analysis Symposium*, pages 156–177, 2020.

[164] H. Chaofan, Y. Lingyu, W. Zhenchao, S. Bin, and Z. Jing. Linear parameter-varying attitude controller design for a reusable launch vehicle during reentry. In *Proc. of the Chinese Guidance, Navigation and Control Conference*, pages 2723–2728, 2014.

[165] Dengying and Zhoujie. LPV H-infinity controller design for a wind power generator. In *Proc. of the International Conference on Robotics, Automation and Mechatronics*, pages 873–878, 2008.

[166] M. Wetzlinger, N. Kochdumper, and M. Althoff. Adaptive parameter tuning for reachability analysis of linear systems. In *Proc. of the International Conference on Decision and Control*, pages 5145–5152, 2020.

[167] M. Wetzlinger, A. Kulmburg, and M. Althoff. Adaptive parameter tuning for reachability analysis of nonlinear systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2021, Article 16.

[168] M. Rungger and M. Zamani. Accurate reachability analysis of uncertain nonlinear systems. In *Proc. of the International Conference on Hybrid Systems: Computation and Control*, pages 61–70, 2018.

[169] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. STL model checking of continuous and hybrid systems. In *Proc. of the International Symposium on Automated Technology for Verification and Analysis*, pages 412–427, 2016.