The Pennsylvania State University
The Graduate School

REACHABILITY ANALYSIS OF NONLINEAR AND HYBRID SYSTEMS USING HYBRID ZONOTOPES AND GRAPHS OF FUNCTIONS

A Dissertation in
Mechanical Engineering
by
Jacob A. Siefert

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2024

The dissertation of Jacob A. Siefert was reviewed and approved by the following:

Herschel C. Pangborn
Assistant Professor of Mechanical Engineering
Thesis Advisor

Sean B. Brennan
Professor of Mechanical Engineering

Constantino M. Lagoa
Professor of Electrical Engineering

Jack W. Langelaan
Professor of Aerospace Engineering

Brandon M. Hencey
Senior Mechanical Engineer, Air Force Research Laboratory

Robert F. Kunz
Professor of Mechanical Engineering
Associate Department Head for Graduate Programs

Abstract

Reachable sets are used to evaluate system performance and ensure constraint satisfaction in safety-critical applications while accounting for the effects of input, disturbance, and parameter uncertainties. However, many reachability approaches are not applicable to nonlinear systems or hybrid dynamics consisting of both continuous and discrete dynamics. Furthermore, both computational complexity and set representation complexity grow rapidly with system dimension and the duration over which sets are propagated. While computational complexity and set representation complexity can be reduced by over-approximating reachable sets, approximation techniques can suffer from significant error as over-approximations are propagated through the dynamics. This dissertation develops methods to enable efficient reachability analysis of hybrid and nonlinear systems using a novel set representation, the *hybrid zonotope*, with sets representing the mapping of functions called *graphs of functions*. Examples demonstrate how these methods can be applied to verify safety and performance of many classes of systems of engineering interest, including closed-loop systems with advanced controllers.

First, this dissertation proposes methods using graphs of functions, including set identities for calculating a set of outputs given a set of inputs, and vice versa. For reachability analysis of an autonomous system, this corresponds to one-step forward and backward reachability. The computational complexity and memory complexity of the proposed identities, when executed using hybrid zonotopes, motivates additional contributions. This includes techniques to represent and approximate graphs of functions as hybrid zonotopes by leveraging special ordered sets (SOS), and methods to improve scalability for constructing graphs of high-dimensional functions via functional decomposition.

Then, the proposed techniques are applied to perform reachability analysis of diverse classes of systems including mixed-logical dynamical (MLD) systems, linear systems in closed-loop with model predictive control (MPC), discrete hybrid automata (DHA), logical systems, neural nets, and nonlinear systems. Methods exploit the structure inherent to many of these classes of systems to generate a functional decomposition, which can be used to efficiently construct graphs of functions. The reachability techniques for each class are compared to existing state-of-the-art techniques using benchmark problems where applicable.

Lastly, the dissertation applies the techniques to set-valued state estimation (SVSE) and set-valued parameter identification (SVPI). Examples demonstrate how nonlinear measurement models can result in nonconvex and disjoint sets, and compare results to an idealized convex approach.

Numerical results demonstrate that proposed methods enable analyses which cannot be performed with other state-of-the-art tools, such as verification of large initial sets.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to recognize my advisor, Professor Herschel Pangborn, who always made time for thoughtful research discussions. His feedback was instrumental to my work as a graduate student, and his continued collaboration will undoubtedly contribute to my future work as a researcher. Professor Pangborn has also demonstrated compassion and patience throughout my doctoral studies, especially as my wife and I welcomed our first child. His combination of academic excellence and unfailing kindness played a key role in my positive graduate school experience.

In addition to Professor Pangborn, Professor Justin Koeln (University of Texas at Dallas), Professor Neera Jain (Purdue University) and Dr. Trevor Bird (formerly Purdue University) have contributed significantly to my graduate work through our hybrid zonotope collaboration. Our meetings have continuously generated fruitful research and thoughtful feedback. I'd especially like recognize Dr. Trevor Bird as a fellow graduate student and mentor. Mirroring a statement from the acknowledgements of his own dissertation, Trevor and I often solved problems using very different approaches. Trevor was always willing to explain his approach to me and I believe this greatly benefited my own understanding, and our co-authored papers. I'd also like to thank my committee members, Professor Sean Brennan, Professor Constantino Lagoa, Professor Jack Langelaan, and Dr. Brandon Hencey, for their guidance, support, and feedback throughout the course of my doctoral studies.

I'm proud to have worked alongside many gifted and hardworking students, including those from my bachelor's degree at the University of Maryland and my master's degree at the University of Minnesota (SKI-U-MAH), and my doctoral degree. This includes (from UMD) Kelly, J.W., Carey, (from UMN) Brian, Justinus, Aditya, Aleksander, Arpan, (and from PSU) Andrew I., Jason, Adim, Josh, Michael, Jacob, Madison, Dor, Nick, Ahmed M., Ahmed A., Haley, Shanthan, Shashank, Pete, and Troy. I would especially like to thank those with whom I spent many early mornings and late nights studying for qualifying exams, Seho, Changik, Garrett, and John, and those with whom I have collaborated closely on publications, Andrew T. and Jonah.

My family has been incredibly supportive. My parents, Ron and Julie, always believed in me, cared for me, and loved me. I truly cannot thank them enough. Together they have encouraged me to work hard and pursue my passions, taught me to treat others kindly, and equipped me with the skills necessary to be successful. My mother-in-law, Kathy, has also always believed in me since we met when I was in 7th grade. It took

my father-in-law, Scott, considerably longer acknowledge my existence, and so when he encouraged me to go back to school six years ago, it gave me the confidence I needed to fill out the applications. I'd also like to thank my sister, Tara, my sisters-in-law, Rachel and Angela (also Josh), and my close friends, Steven and James.

Most importantly, I would not have been able to accomplish anything without my wife and best friend, Allison. I am blessed to have met someone so kind, funny, brilliant, and patient as you. From seventh grade algebra homework, to results presented in this thesis, you have helped me with my studies. You are also an incredible mother to our daughter, Mae. To Mae, your smile is a source of motivation more powerful than rice puffs. I love you both.

Dedication

To my wife, Allison,
my daughter, Mae,
my parents, Ron and Julie, and
my parents-in-law, Scott and Kathy.

Mae, represented using two hybrid zonotopes.

Chapter 1
Introduction

1.1 Motivation and Background

Certifying safety and performance is critical to the development of many engineered systems, including autonomous vehicles, medical robotics, electrical grids, etc. [1–5]. Simulation over varied initial conditions, disturbances, and parameters allows for testing of different scenarios but may fail to generate worst-case behavior or reveal critical design flaws. Therefore, such simulations provide no guarantee of safety. Verification and falsification refer to the process of rigorously proving a statement, e.g., "the system will not achieve an unsafe state". Simulations can be used to falsify such a statement by counterexample if *any* simulated trajectory achieves an unsafe state, but in general, a finite number of simulations cannot be used to verify the safety of a system.

1.1.1 Reachability Analysis

Set-based reachability analysis generates the set of all states a system can achieve from a set of initial conditions for all admissible inputs, disturbances, and parameters. In lieu of calculating exact reachable sets, over- and inner-approximations of reachable sets can often be calculated with reduced complexity at the cost of conservatism. Both exact and approximate reachability analysis can be used to guarantee safety. For example, if an exact or over-approximated reachable set of a system does not intersect an unsafe set, then no individual trajectory can enter the unsafe set and therefore the system is safe. Similarly, exact and inner-approximations of reachable sets are often used to find invariant sets. Once entered, no trajectory will leave an invariant set; a property often exploited to guarantee recursive feasibility of receding-horizon control approaches like Model Predictive Control (MPC).

Several fundamental approaches for reachability analysis have been developed to provide guarantees of dynamic system behavior, each relying on different mathematical methods. Hamilton-Jacobi reachability calculates reachable sets by numerically solving nonlinear differential equations and is applicable to nonlinear dynamics including hybrid systems; however, its computational complexity scales exponentially with the system state dimension as partial differential equations are numerically solved by discretizing the state space. Techniques to reduce computational complexity decompose the system into subsystems based on its structure when possible [6–9].

Another approach for reachability analysis relies on generating barrier certificates to separate reachable sets from unsafe sets using a zero-sublevel set of a function [10, 11]. Barrier certificates are constructed such that no trajectory of the system can cross from negative to positive values of the barrier certificate function. Thus if the initial condition set is contained within negative values of the barrier certificate function, then the non-positive values of the barrier certificate function provide an over approximation of the reachable set. Deriving barrier certificates can be challenging, especially if the reachable set is close to an unsafe region.

This dissertation focuses on set propagation methods, in which a sequence of reachable sets, beginning with a set of initial conditions, are calculated using iterative set operations to propagate the system dynamics forward or backward in time [12]. Depending on the set representation used, set propagation methods have been shown to scale well with the state dimension, but may not be capable of *exactly* representing reachable sets.

1.1.2 Set Representations

Set propagation techniques and their performance are highly dependent upon the choice of set representation. Factors to consider include:

1. Reachable sets, the initial condition set and admissible input, disturbance, and parameter sets must be able to be represented or adequately approximated by the chosen set representation.

2. The set representation should be *closed* under operations used for set propagation. A class of sets is said to be closed under a given operation if the operation acting on that class results in a set of the same class.

3. The computational complexity of performing the required set operations and the complexity of the set representation itself should be considered, as this strongly

determines scalability given limited computational and memory resources.

Common operations used for propagation and analysis of reachable sets include linear transformations, Minkowski sums, Minkowski differences, generalized intersections, intersections with halfspaces, Cartesian products, complements, and unions. An overview of many commonly used set representations with their closure and complexity for various set operations can be found in [12, Table 1], and the relationship between various sets representations is depicted in [12, Figure 1][1].

Ellipsoids have been used effectively for reachability analysis, however of the set operations listed above they are closed only under linear transformation. Because ellipsoids are convex and symmetrical, for many system dynamics and applications they provide over-approximations of reachable sets rather than exact sets [13]. Polytopes, represented either as the intersection of halfspaces (H-rep) or the convex hull of vertices (V-rep), can represent asymmetrical convex sets. Both H-rep and V-rep are closed under the set operations listed above, with the exception of unions and complements. However, Minkowski sums exhibit exponential computational complexity in H-rep and generalized intersections are NP-hard in V-rep [14].

Polynomial zonotopes [15] and Taylor models [16] express an equivalent class of nonconvex sets related to polynomial functions, but use different representations. Their nonconvexity allows for reduced error when approximating nonlinear and hybrid dynamics as compared to convex methods. Both are closed for key set operations, with the exception of intersections. Constrained polynomial zonotopes [17] extend polynomial zonotopes to allow for closure under intersection and union operations, for which identities are known. Star sets [18] represent an even more general class of nonconvex sets and are closed under many set operations, however there are no closed-form identities for their unions and intersections.

Logical zonotopes [19] and their more generalized form, polynomial logical zonotopes [20], are set representations for reachability analysis of logical systems, e.g., Boolean functions. Polynomial logical zonotopes are closed under all fundamental Boolean functions.

The remainder of this section discusses zonotopes, constrained zonotopes, and hybrid zonotopes, the latter of which is the set representation used within this thesis. Numerical examples will be used to compare the reachability methods developed for hybrid zonotopes to the state-of-the-art.

[1]The set representation used in this thesis, namely the hybrid zonotope, is not included in the reference as it was proposed more recently.

Zonotopes are centrally symmetric convex polytopes. They are computationally efficient for linear transformations and Minkowski sums and scale well to high-dimensional state spaces. Introduced in 2016, constrained zonotopes [21] extend zonotopes by introducing linear equality constraints to the set definition. In doing so, constrained zonotopes can represent asymmetric polytopes (equivalent to H-rep and V-rep) while still enabling computationally efficient linear transformations, Minkowski sums, and generalized intersections. Due to their convexity, constrained zonotopes are not closed under unions or complements.

Hybrid zonotopes further extended constrained zonotopes by introducing binary factors into the set definition [22–24]. This allows them to represent non-convex sets and enables closure under unions and complements. Many operations that are closed and efficient for constrained zonotopes have been extended to hybrid zonotopes, including linear transformations, Minkowski sums, generalized intersections, intersections with halfspaces, and Cartesian products.

1.1.3 Set Propagation Techniques for Discrete-Time Hybrid Systems

Hybrid system models have found increased use in recent years due to their ability to capture mixed continuous and discrete dynamics, such as those exhibited by cyber-physical systems and logic-based controllers [25]. While they are incredibly expressive, hybrid system models are inherently complex to analyze. Basic properties of hybrid systems, such as stability and controllability, are not easily determined from the system model even in the case of linear hybrid systems [26, 27].

Set propagation techniques can be deployed to guarantee constraint satisfaction and performance of hybrid systems. The forward reachable sets of linear hybrid systems may be determined using a collection of convex sets by partitioning the state space into locations separated by guards and applying techniques developed for linear systems within each location [25, 28]. However, successive intersections with guards at each time step result in worst-case exponential growth in complexity, leading to computational-intractability for long time horizons. Over-approximation techniques like clustering-based methods reduce the complexity of sets at the cost of conservatism [29]. The extent of this conservatism is application-dependent and difficult to quantify [30].

Hybrid zonotopes have been shown to enable scalable closed-form solutions of forward reachable sets for broad classes of discrete-time linear hybrid systems. This includes Mixed Logical Dynamical (MLD) systems [22] and linear systems in closed loop with MPC [23], of which the explicit solution yields piecewise affine (PWA) control laws [31,32].

MLD systems are equivalent[2] to many discrete-time hybrid systems including Linear Complementarity (LC) systems, Extended Linear Complementarity (ELC) systems, PWA systems, max-min-plus-scaling (MMPS) systems, and Discrete Hybrid Automata (DHA) [31–33]. Tools for generating equivalent MLD systems, e.g., converting a DHA to an MLD system using HYSDEL [34], allow for forward reachability analysis of these equivalent systems by first converting to an MLD system. Previous work does not address reachability of the classes of systems directly using hybrid zonotopes or the ability to calculate backward reachable sets or invariant sets.

In general, backward reachable sets of linear hybrid systems are challenging to calculate. Even for *autonomous* hybrid systems, there may be many states that converge to a single state at the following time step, which can cause backwards reachable sets to grow quickly. This complexity is often compounded when considering disturbances, as analysis relies on computing or approximating Minkowski differences [35, 36]. Similar to finding forward reachable sets of hybrid systems, backward reachable sets may be found by considering collections of convex sets within each location, with worst-case exponential growth in set complexity with time [37, 38]. Backward reachability is often used to calculate invariant sets. Lacking scalable methods to calculate backward reachable sets under the PWA control laws of MPC, artificial constraints based on the invariant set under a simpler control law are introduced to ensure recursive feasibility, resulting in conservatism [31].

Logical systems represent a subset of DHA[3], and reachability can be performed using the techniques developed for MLD systems [22]. Recent work using logical zonotopes and polynomial logical zonotopes has provided methods for reachability of logical systems that do not require conversion to an equivalent MLD [19, 20]. Previous work using hybrid zonotopes did not address direct reachability of logical systems, i.e., without first converting to an MLD.

1.1.4 Set Propagation Techniques for Nonlinear Systems

This dissertation addresses nonlinear system in closed loop with PWA controllers, e.g., neural networks using Rectified Linear Unit (ReLU) activation functions. Discussion herein focuses on the four state-of-the-art reachability tools (CORA [39], JuliaReach [40],

[2]See [33] for additional assumptions required for LC→MLD, ELC→MLD, MLD→PWA, and PWA→MLD.

[3]DHA consist of an event generator (EG), finite state machine (FSM), mode selector (MS), and switched affine system (SAS). Logical systems can be represented using an FSM only.

NNV [41], POLAR [42])[4] included in the Artificial Intelligence and Neural Network Control Systems category of the Applied veRification for Continuous and Hybrid systems (ARCH) competitions in 2022 [43] and 2023 [44]. In later chapters, benchmark problems from [43, 44] are used to demonstrate and compare the proposed methods to these state-of-the-art tools.

COntinuous Reachability Analyzer (CORA) [45, 46] uses polynomial zonotopes to approximate the input-output relationship of activation functions and nonlinear dynamics. Over-approximated reachable sets are calculated by efficient mappings via the over-approximated activation functions of the neural network and nonlinear functions associated with the plant dynamics. JuliaReach [47] converts structured zonotopes to and from Taylor models, and then performs reachability using structured zonotopes for the neural network and Taylor models for the plant, to efficiently construct reachable sets for the closed-loop system. The Neural Network Verification (NNV) tool [41] uses star sets for efficient computation of exact or approximated reachable sets through neural networks, and utilizes CORA for nonlinear plant dynamics. The POLynomial ARithmetic-base (POLAR) framework [42] computes functional over-approximations of the flowmap. POLAR uses univariate Bernstein polynomial abstractions to address non-differentiable activation functions.

In most cases, each of these state-of-the-art tools performs well for the benchmark examples in [43, 44], though it is noted that the initial sets for the benchmark problems are relatively small. Large initial sets pose computational challenges as stated in a review of set propagation techniques in [12], which includes techniques used by these tools: *"Several challenging research problems remain to be addressed in the field, such as handling large initial sets for nonlinear systems and many guard intersections in hybrid systems. Both aspects are especially relevant when verifying systems involving neural networks."* To avoid these challenges, large initial sets may be partitioned, though for systems beyond a few dimensions the partitioning itself introduces worst-case exponential complexity.

1.1.5 Set-Based Methods for State Estimation and Parameter Identification

In addition to safety verification as discussed above, set-based methods can also be used for state estimation and parameter identification. In contrast to classical approaches for

[4]Many of these tools contain multiple methods to address different classes of systems, however a more detailed discussion of each of their capabilities falls beyond the scope of this dissertation.

estimation and identification, such as Kalman filters, set-based methods do not require knowledge or assumptions about stochastic properties of uncertainties. Instead, set-based methods only require enclosures that bound uncertainties [48, 49]. Numerous techniques for linear system dynamics are available in the literature, e.g., [21, 50], however developing set-based approaches for hybrid and nonlinear systems has been identified as an open area for study [51, 52]. A key challenge is that most set representations capable of accurately representing the non-convexities of these systems are either computationally expensive or not closed under required set operations. Methods that rely solely on convex sets and linearization can yield conservative results with significant approximation error.

1.2 Contributions

This thesis proposes fundamental theoretic contributions to set-based methods for reachability analysis and demonstrates their utility for verification, state estimation, and parameter identification of hybrid and nonlinear systems in open-loop and in closed-loop with advanced controllers. Applications to benchmark systems illustrate the impact of new methods as compared to the state-of-the-art. The key research contributions are as follows.

1. *Reachability of discrete-time linear hybrid systems:* Enable scalable and efficient methods for forward and backward reachability analysis of
   - Mixed Logical Dynamical (MLD) systems,
   - linear systems in closed-loop with Model Predictive Control (MPC),
   - Discrete Hybrid Automata (DHA),
   - logical systems.
2. *Reachability of nonlinear systems with neural network controllers:* Reduce computational complexity, memory complexity, and error of over-approximated forward reachability analysis of nonlinear systems in closed-loop with neural network controllers.
3. *State Estimation and Parameter Identification:* Reduce approximation error for set-valued state estimation and set-valued parameter identification in the case of highly nonlinear models.

The above contributions are achieved using novel techniques that combine graphs of functions, hybrid zonotopes, special-ordered set approximations, and functional decomposition. A MATLAB-based toolbox for hybrid zonotopes, named zonoLAB [53], has also been developed, although it does not yet include all the capabilities proposed in this thesis.

1.3 Document Organization

The remainder of this thesis is structured as follows. Chapter 2 provides preliminaries including introducing hybrid zonotopes, set operations, and conversion from vertex representation. Chapter 3 provides the general framework using graphs of functions which are constructed using functional decomposition and special ordered sets, and for which input-output identities are produced. Chapter 4 proposes forward and backward reachability analysis techniques specific to several classes of hybrid and nonlinear systems and provides comparison to the state-of-the-art using benchmark examples. Chapter 5 extends the results from Chapter 3 and Chapter 4 for state estimation and parameter identification of highly nonlinear models and provides comparison to an idealized approach using convex sets. Chapter 6 concludes the thesis. Figure 1.1 depicts the high-level connections between the key concepts within each chapter.

The thesis attempts to places results in an order convenient for the reader. In doing so, contributions of this thesis and its associated publications are often amongst existing contributions from literature by other authors. In addition to clearly citing each theoretical and numerical result that is published, each section indicates which results are considered contributions of this thesis.

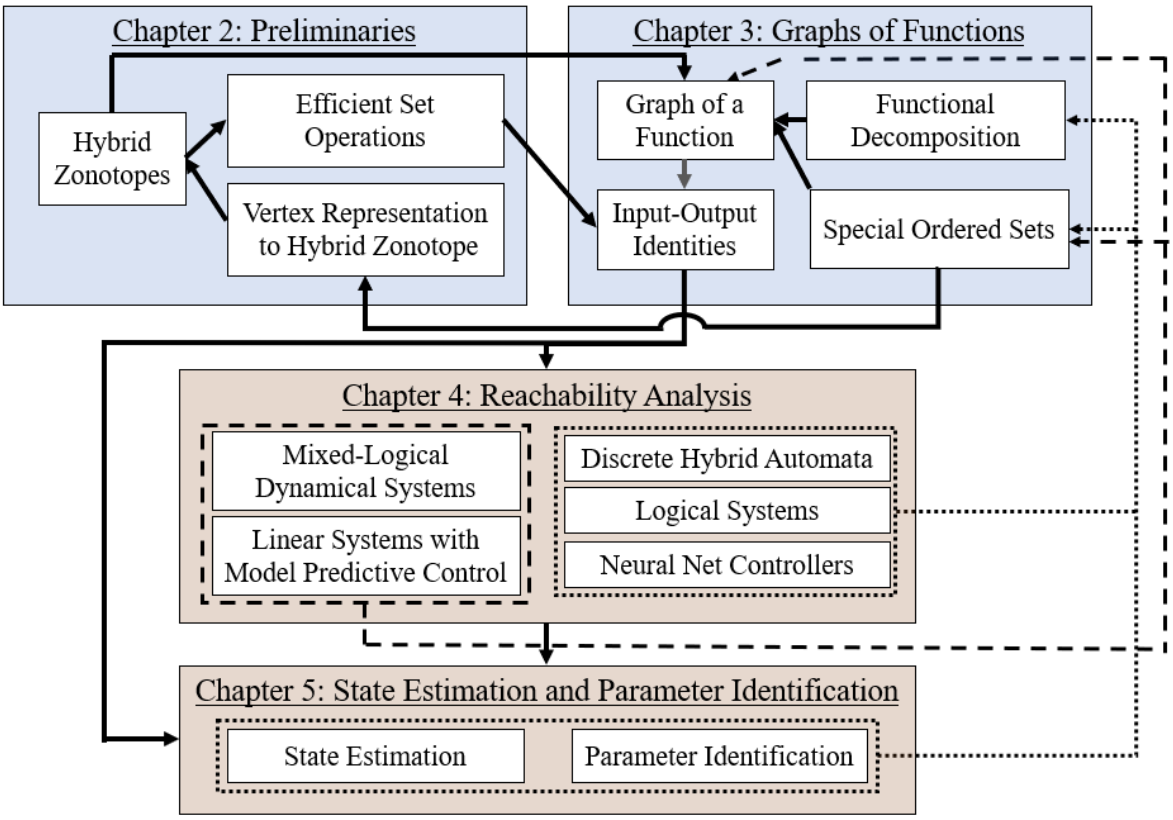


Figure 1.1: Chapter Overview and Key Concepts: Solid lines indicate the flow of theoretical concepts and dashed lines are used to indicate how specific system models and problems are posed in terms of the fundamental theoretical framework in Chapter 2 and Chapter 3.

Chapter 2
Preliminaries

2.1 Notation

Vectors and scalars are denoted by lowercase letters, e.g., $x \in \mathbb{R}^n$. Matrices are denoted by uppercase letters, e.g., $G \in \mathbb{R}^{n \times n_g}$. The element in the i^{th} row and j^{th} column of a matrix G is denoted $G_{(i,j)}$. The i^{th} row and j^{th} column are denoted $G_{(i,\cdot)}$ and $G_{(\cdot,j)}$, respectively.

Sets are denoted by uppercase calligraphic letters[1], e.g., $\mathcal{Z} \subset \mathbb{R}^n$, and uppercase Greek letters, e.g., $\Phi \subset \mathbb{R}^n$. The topological boundary of a set is denoted by $\partial\mathcal{Z}$ and its interior by $\mathcal{Z}^\circ$. The closure of a set is denoted by $\overline{\mathcal{Z}}$ such that $\overline{\mathcal{Z}}$ includes both the interior and boundary of $\mathcal{Z}$. Subscripts are used to distinguish between properties that are defined for multiple sets, e.g., $n_{g,z}$ describes the complexity g of the representation of $\mathcal{Z}$ while $n_{g,w}$ describes the complexity g of the representation of $\mathcal{W}$. The n-dimensional unit hypercube is denoted by

$$\mathcal{B}_\infty^n = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq 1\} \ , \tag{2.1}$$

and the n-dimensional constrained unit hypercube is denoted by

$$\mathcal{B}_\infty^n(A, b) = \{x \in \mathbb{R}^n : \|x\|_\infty \leq 1, \ A\xi = b\} \ . \tag{2.2}$$

The set of all n-dimensional binary vectors is denoted by $\{-1, 1\}^n$, e.g.,

$$\{-1, 1\}^2 = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\} \ . \tag{2.3}$$

[1] $\mathcal{O}$ is an exception, reserved for computational complexity.

The concatenation of two column vectors to a single column vector is denoted by $(\xi_1\ \xi_2) = [\xi_1^T\ \xi_2^T]^T$ and $\mathrm{diag}(x)$ yields the diagonal and square matrix with diagonal elements corresponding to the elements of $x \in \mathbb{R}^n$, e.g.,

$$\mathrm{diag}(x) = \begin{bmatrix} x_1 e_1 \\ \vdots \\ x_n e_n \end{bmatrix}, \tag{2.4}$$

where e_i denotes the i^{th} row of the identity matrix.

Bolded $\mathbf{1}$ and $\mathbf{0}$ denote matrices of all 1 and 0 elements, respectively, and $\mathbf{I}$ denotes the identity matrix. Subscripts are used to identify the dimension of these matrices when not easily deduced from context. The interval between two scalars a and b is denoted $\begin{bmatrix} a, & b \end{bmatrix} \subset \mathbb{R}^1$. A comma is used to distinguish from the matrix $\begin{bmatrix} a & b \end{bmatrix} \in \mathbb{R}^{1\times 2}$.

2.2 Set Operations

This section provides definitions for set operations used throughout the thesis. Given the sets $\mathcal{Z}$, $\mathcal{W}$, $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$, and matrix $R \in \mathbb{R}^{m\times n}$, we define the following set operations:

$$R\mathcal{Z} = \{Rz \mid z \in \mathcal{Z}\}\ , \tag{2.5a}$$
$$\mathcal{Z} \oplus \mathcal{W} = \{z + w \mid z \in \mathcal{Z},\ w \in \mathcal{W}\}\ , \tag{2.5b}$$
$$\mathcal{Z} \ominus \mathcal{W} = \{x \in \mathbb{R}^n \mid x + w \in \mathcal{Z},\ \forall w \in \mathcal{W}\}\ , \tag{2.5c}$$
$$\mathcal{Z} \cap_R \mathcal{Y} = \{z \in \mathcal{Z} \mid Rz \in \mathcal{Y}\}\ , \tag{2.5d}$$
$$\mathcal{Z} \cup \mathcal{W} = \{x \in \mathbb{R}^n \mid x \in \mathcal{Z} \vee x \in \mathcal{W}\}\ , \tag{2.5e}$$
$$\overline{\mathcal{Z}^c} = \{x \in \mathbb{R}^n \mid x \notin \mathcal{Z}^\circ\}\ , \tag{2.5f}$$
$$\mathcal{C}_{\mathcal{X}}(\mathcal{Z}) = \{x \in \mathcal{X} \mid x \notin \mathcal{Z}^\circ\}\,, \tag{2.5g}$$
$$Z \times \mathcal{Y} = \{(z, y) \mid z \in \mathcal{Z},\ y \in \mathcal{Y}\}\,. \tag{2.5h}$$

The linear mapping of $\mathcal{Z}$ by R is given by (2.5a), the Minkowski sum of $\mathcal{Z}$ and $\mathcal{W}$ is given by (2.5b), the Minkowski difference of $\mathcal{W}$ from $\mathcal{Z}$ is given by (2.5c), the generalized intersection of $\mathcal{Z}$ and $\mathcal{Y}$ under R is given by (2.5d), the standard intersection for $R = \mathbf{I}$ is denoted by $\cap$, the union of $\mathcal{Z}$ and $\mathcal{W}$ is given by (2.5e), the closure of the complement of $\mathcal{Z}$ is given by (2.5f), the closure of the complement of $\mathcal{Z}$ defined over the set $\mathcal{X}$ is given by (2.5g), and the Cartesian product is given by (2.5h).

2.3 Set Representations

This section provides a review of several set representations with emphasis on *closure* and complexity of set representations under key set operations defined by (2.5).

Definition 2.1 (Closure of Sets) *A class of sets is said to be "closed" if a set operation acting on that class results in a set of the same class.*

Complexity regarding sets and their operations are discussed in two distinct ways, referred to as "memory complexity" and "computational complexity".

1. *Memory complexity* refers to the number of variables required to represent a set in a particular set representation, e.g., an n-dimensional hypercube can be represented by 2^n vertices in V-rep. It is also important to consider *memory complexity growth* under set operations as a function of the memory complexities of argument sets.

2. *Computational complexity* refers to how the bound on worst-case run time of a given computation scales as a function of the memory complexities of arguments. For example, the computational complexity of vector addition in $\mathbb{R}^n$ is $\mathcal{O}(n)$, meaning that the computation time of vector addition scales linearly with the state dimension. Computational complexities of set operations are found by summing the computational complexities of the required calculations. As is common practice, coefficients and lower-order terms are often dropped for conciseness. A review of computational complexity can be found in [54].

For a more comprehensive review of set representations, the reader is directed to [12, Table 1] and [21, 22].

2.3.1 Ellipsoids, Convex Polytopes, and Zonotopes

Definition 2.2 (Ellipsoid) *Let $\mathcal{E}, \mathcal{P}, \mathcal{Z} \subset \mathbb{R}^n$. The set $\mathcal{E}$ is an ellipsoid if there exists $Q \in \mathbb{R}^{n \times n}$ and $c \in \mathbb{R}^n$ such that*

$$\mathcal{E} = \{Q\xi + c \mid \xi \in \mathcal{B}_2^n\} \ . \tag{2.6}$$

Ellipsoids are computationally efficient under linear mapping (2.5a) with complexity $\mathcal{O}(\max(mn^2, m^2n))$ [55, Section 2.2.1] but are not closed under several key operations including Minkowski sum (2.5b) and generalized intersection (2.5d).

Definition 2.3 (Polytope - Vertex Representation) *A set $\mathcal{P} \in \mathbb{R}^n$ is a convex polytope if and only if it is bounded and $\exists\ V \in \mathbb{R}^{n \times n_v}$ with $n_v < \infty$ such that*

$$\mathcal{P} = \left\{ V\lambda \mid \lambda_j \geq 0\ ,\ \forall j \in \{1, ..., n_v\},\ \mathbf{1}^T\lambda = 1 \right\}\ .$$

When a convex polytope is represented using a collection of vertices as in Definition 2.3, it is said to be in *vertex*-representation (V-rep). Polytopes can be equivalently defined by the intersection of halfspaces as follows.

Definition 2.4 (Polytope - Halfspace Representation) *A set $\mathcal{P} \in \mathbb{R}^n$ is a convex polytope if and only if it is bounded and $\exists\ H \in \mathbb{R}^{n_h \times n}$ and $f \in \mathbb{R}^{n_h}$ such that*

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

When a polytope is represented by a collection of halfspace constraints, it is said to be in *halfspace*-representation (H-rep). Convex polytopes are closed under linear mapping (2.5a), Minkowski sum (2.5b), and generalized intersection (2.5d). The complexity of performing Minkowski sum in H-rep is $\mathcal{O}(2^n)$ [56, Table 1] while generalized intersection in vertex representation is NP-hard due to required facet and vertex enumeration, limiting its use in higher dimensions [12, 14].

Definition 2.5 (Zonotope) *The set $\mathcal{Z}$ is a zonotope if there exists $G \in \mathbb{R}^{n \times n_g}$ and $c \in \mathbb{R}^n$ such that*

$$\mathcal{Z} = \{G\xi + c \mid \xi \in \mathcal{B}_\infty^{n_g}\}\ .$$

Zonotopes are computationally efficient under linear mappings with complexity $\mathcal{O}(mn^2)$ [57, Table 1] and Minkowski sums with complexity $\mathcal{O}(n)$ [57, Table 1] but are not closed under generalized intersection as they are limited to representing centrally symmetric sets.

2.3.2 Constrained Zonotopes

Definition 2.6 (Constrained Zonotope) *[21, Definition 3] Let $\mathcal{Z}_h \subset \mathbb{R}^n$. The set $\mathcal{Z}_h$ is a constrained zonotope if there exists $G \in \mathbb{R}^{n \times n_g}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{n_c \times n_g}$, and $b \in \mathbb{R}^{n_c}$ such that*

$$\mathcal{Z}_c = \{G\xi + c \mid \xi \in \mathcal{B}_\infty^{n_g},\ A\xi = b\}\ . \tag{2.7}$$

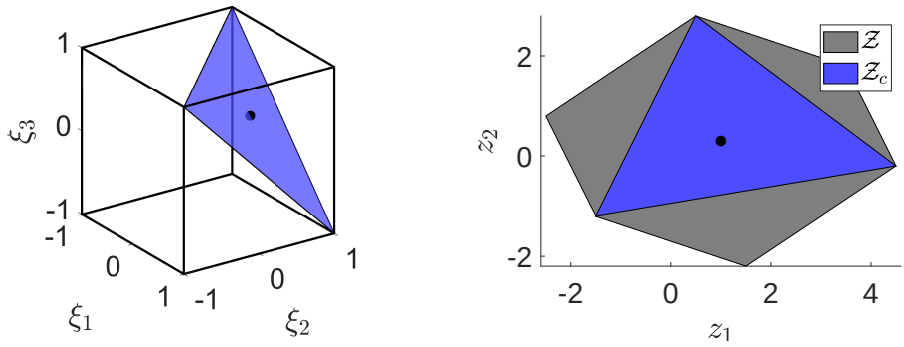
The constrained zonotope is given in *constrained generator representation* (CG-rep) and the shorthand notation of $\mathcal{Z}_c = \langle G, c, A, b \rangle \subset \mathbb{R}^n$ is used to denote the set given by

(2.7). While zonotopes must be centrally symmetric and are therefore not equivalent to convex polytopes, the addition of linear equality constraints allows constrained zonotopes to represent any convex polytope while providing efficient identities for linear mapping with complexity $\mathcal{O}(mn^2)$, Minkowski sum with complexity $\mathcal{O}(n)$, and generalized intersection with complexity $\mathcal{O}(n)$ [12, Table 1].

Example 2.1 *[21] Consider the constrained zonotope* $\mathcal{Z}_c = \{G_z, c_z, A_z, b_z\} \subset \mathbb{R}^2$ *given by*

$$\mathcal{Z}_c = \left\langle \begin{bmatrix} 1.5 & -1.5 & 0.5 \\ 1 & 0.5 & -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, 1 \right\rangle. \tag{2.8}$$

From the definition of the constrained zonotope, it is clear that the zonotope defined by $\mathcal{Z} = \{G_z, c_z\} \subset \mathbb{R}^2$ *will satisfy* $\mathcal{Z}_c \subseteq \mathcal{Z}$*, because affine mappings preserve set containment and a constrained unit hypercube is a subset of a hypercube. This is depicted in Fig 2.1.*



(a) (b)

Figure 2.1: Example of a constrained zonotope.
(a) Constrained unit hypercube $\mathcal{B}^3_\infty(A, b)$. (b) Constrained zonotope $\mathcal{Z}_c = \{G_z, c_z, A_z, b_z\}$, generated as the affine image of the constrained unit hypercube. This is plotted over zonotope $\mathcal{Z} = \{G_z, c_z\}$, generated as the affine image of the entire unit hypercube [21, Figure 1]

2.3.3 Hybrid Zonotopes

2.3.3.1 Introduction

The following sections provide preliminaries for the hybrid zonotope set representation used throughout this thesis, including its definition, set operation identities, and set conversion identities. The sections contain results from the literature, many of which are included in [58]. In addition, the following contributions by the author are presented.

1. Proposition 2.5 presents an identity to reduce memory complexity growth when computing unions of many hybrid zonotopes.

2. Proposition 2.6 presents an identity for computing the Minkowski difference of a hybrid zonotope with a zonotope.

3. Theorem 2.1 provides an efficient identity for converting a collection of V-rep polytopes into a hybrid zonotope. The memory complexity of the resulting hybrid zonotope grows linearly both with the total number of vertices across all polytopes and with the number of polytopes.

Remark 2.1 *Also note the important work proposed in [22] and later extended in [58] on exact order reduction techniques for hybrid zonotopes, which is not reproduced in this thesis.*

2.3.3.2 Definition

Definition 2.7 (Hybrid Zonotope) [22, Definition 3] *The set $\mathcal{Z}_h \subset \mathbb{R}^n$ is a hybrid zonotope if there exists $G^c \in \mathbb{R}^{n\times n_g}$, $G^b \in \mathbb{R}^{n\times n_b}$, $c \in \mathbb{R}^n$, $A^c \in \mathbb{R}^{n_c\times n_g}$, $A^b \in \mathbb{R}^{n_c\times n_b}$, and $b \in \mathbb{R}^{n_c}$ such that*

$$\mathcal{Z}_h = \left\{ \begin{bmatrix} G^c & G^b \end{bmatrix} \begin{bmatrix} \xi^c \\ \xi^b \end{bmatrix} + c \;\middle|\; \begin{matrix} \begin{bmatrix} \xi^c \\ \xi^b \end{bmatrix} \in \mathcal{B}_\infty^{n_g} \times \{-1,1\}^{n_b}, \\ \begin{bmatrix} A^c & A^b \end{bmatrix} \begin{bmatrix} \xi^c \\ \xi^b \end{bmatrix} = b \end{matrix} \right\}. \tag{2.9}$$

A hybrid zonotope is the union of 2^{n_b} constrained zonotopes corresponding to the possible combinations of binary factors $\xi^b \in \{-1,1\}^{n_b}$. The hybrid zonotope is given in *hybrid constrained generator representation* (HCG-rep) and the shorthand notation of $\mathcal{Z}_h = \langle G^c, G^b, c, A^c, A^b, b\rangle \subset \mathbb{R}^n$ is used to denote the set given by (2.9). Continuous and binary *generators* refer to the columns of G^c and G^b, respectively. By inspection, a hybrid zonotope with no binary factors is a constrained zonotope, $\mathcal{Z}_c = \langle G, c, A, b\rangle \subset \mathbb{R}^n$, and a hybrid zonotope with no binary factors and no constraints is a zonotope, $\mathcal{Z} = \langle G, c\rangle \subset \mathbb{R}^n$.

The hybrid zonotope consists of a continuous portion equivalent to that of the constrained zonotope shifted by contributions from a discrete, finite set. This is depicted in the following example.

Example 2.2 *[58] Consider the hybrid zonotope $\mathcal{Z}_h = \left\{G^c, G^b, c, A^c, A^b, b\right\} \subset \mathbb{R}^2$ given by*

$$\mathcal{Z}_h = \left\langle \begin{bmatrix} 1.5 & -1.5 & 0.5 \\ 1 & 0.5 & -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, 1, 1 \right\rangle, \tag{2.10}$$

and shown in Fig. 2.2 where a single binary factor having $G^b = \mathbf{1}$ *and* $A^b = 1$ *is added to the constrained zonotope from a previous example* (2.8).

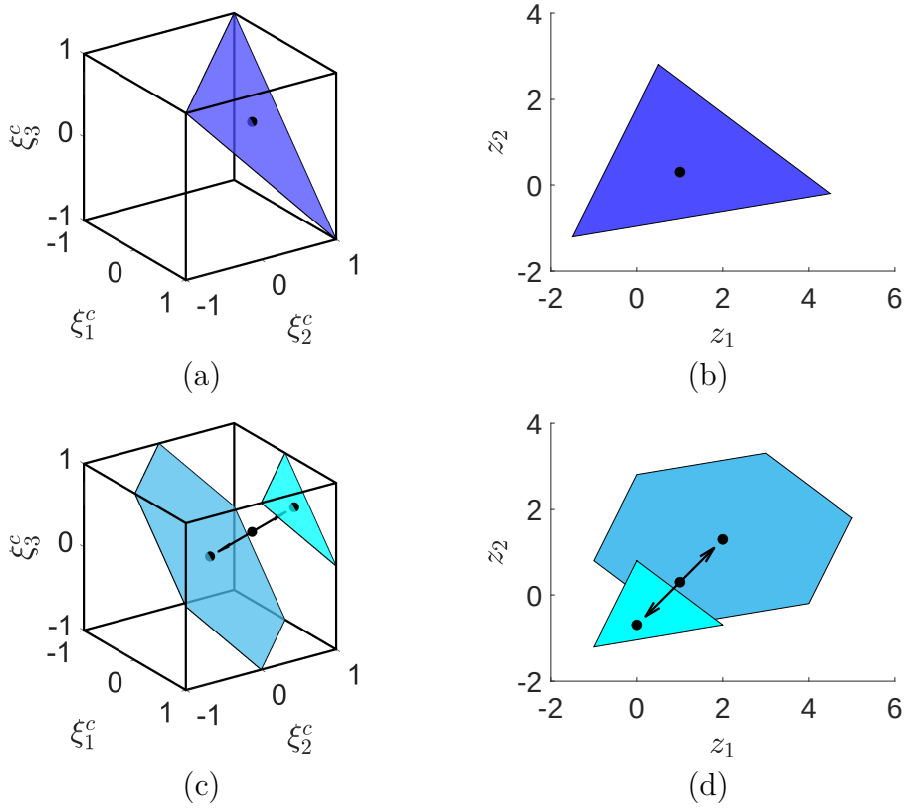


Figure 2.2: Example from [58] of generating a hybrid zonotope by adding one binary factor to the constrained zonotope (2.8) for $G^b = \mathbf{1}$ and $A^b = 1$.
(a) Constrained unit hypercube $\mathcal{B}^3_\infty(A^c, b)$. (b) Constrained zonotope taken as the affine image of (a), $\mathcal{Z}_c = G^c\mathcal{B}^3_\infty(A^c, b) \oplus c$. (c) Adding one binary factor to the constrained unit hypercube results in two possible shifts in the hyperplane, $A^c\xi^c = b - A^b\xi^b_1$ and $A^c\xi^c = b - A^b\xi^b_2$, one for each entry of the discrete set $\xi^b_i \in \{-1, 1\}$. (d) Hybrid zonotope taken as the affine image of (c) with shifted centers, $\mathcal{Z}_h = G^c\mathcal{B}^3_\infty(A^c, b - A^b\xi^b_1) \oplus (c + G^b\xi^b_1) \cup G^c\mathcal{B}^3_\infty(A^c, b - A^b\xi^b_2) \oplus (c + G^b\xi^b_2)$.

In a second example, a "chessboard" set is used to demonstrate how hybrid zonotopes make use of binary factors to represent collections of polytopes with significantly lower memory complexity than if represented using convex polytopes.

Example 2.3 *Consider the hybrid zonotope* $\mathcal{Z}_h = \left\{G^c, G^b, c, A^c, A^b, b\right\} \subset \mathbb{R}^2$ *given by*

$$\mathcal{Z}_h = \left\langle \begin{bmatrix} \frac{1}{8} & 0 \\ 0 & \frac{1}{8} \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 0 & 0 & \frac{1}{8} \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \emptyset, \emptyset, \emptyset \right\rangle, \tag{2.11}$$

and shown in Fig. 2.3. The set is the union of 32 *zonotopes, each corresponding to* $\left\langle \begin{bmatrix} \frac{1}{8} & 0 \\ 0 & \frac{1}{8} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\rangle$ *shifted by various combinations of binary generators. The fifth binary generator encodes coupling between the dimensions, which gives the checkered pattern. To represent this set as a collection of convex sets would require 32 convex sets, whereas a single hybrid zonotope represents the set compactly with only 7 generators (2 continuous, 5 binary) and no constraints.*

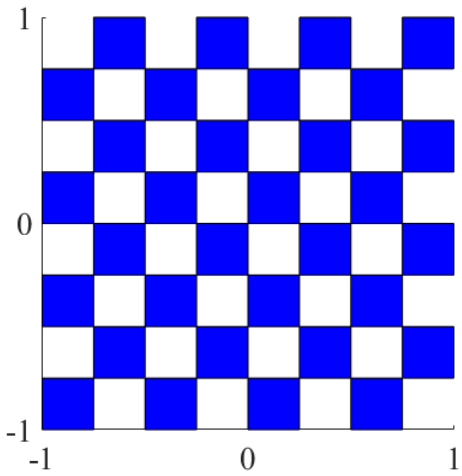


Figure 2.3: "Chessboard" set given by the compact representation of the hybrid zonotope defined in (2.11).
The chessboard could be represented as a collection of 32 V-rep polytopes with 4 vertices per polytope, as a collection of 32 H-rep polytopes with 4 halfspace inequalities per polytope, or as a collection of 32 G-rep/CG-rep polytopes with 2 generators per polytope.

Example 2.3 provides a simple example of the memory complexity advantages of hybrid zonotopes, though it may not yet be clear how to extend hybrid zonotopes beyond shifting convex sets in a structured way. Later results including conversion of collections of V-rep polytopes into a hybrid zonotope and utilizing functional decomposition will further highlight the expressiveness of hybrid zonotopes.

2.3.3.3 Analyses

This section includes point containment, emptiness, and support function sampling that are commonly used to analyze reachable sets for the purposes of verification and falsification. This section specifically discusses these analyses in the context of hybrid zonotopes.

Proposition 2.1 (Point Containment) [58, Proposition 3.2.8]
For any $\mathcal{Z}_h = \left\langle G^c, G^b, c, A^c, A^b, b \right\rangle \subset \mathbb{R}^n$,

$$z \in \mathcal{Z}_h \iff \left\{ \|\xi^c\|_\infty \leq 1 \,,\, \xi^b \in \{-1,1\}^{n_b} \;\middle|\; \begin{bmatrix} G^c & G^b \\ A^c & A^b \end{bmatrix} \begin{bmatrix} \xi^c \\ \xi^b \end{bmatrix} = \begin{bmatrix} z - c \\ b \end{bmatrix} \right\} \neq \emptyset \,. \quad (2.12)$$

Proof *Omitted for brevity. See [58, Proposition 3.2.8].*

Proposition 2.2 (Emptiness) [58, Proposition 3.2.8]
For any $\mathcal{Z}_h = \left\langle G^c, G^b, c, A^c, A^b, b \right\rangle \subset \mathbb{R}^n$,

$$\mathcal{Z}_h \neq \emptyset \iff \left\{ \|\xi^c\|_\infty \leq 1 \,,\, \xi^b \in \{-1,1\}^{n_b} \;\middle|\; A^c\xi^c + A^b\xi^b = b \right\} \neq \emptyset \,. \quad (2.13)$$

Proof *Omitted for brevity. See [58, Proposition 3.2.8].*

For hybrid zonotopes, checking for point containment and emptiness can be seen from Proposition 2.1 and Proposition 2.13 to require solution of mixed-integer linear feasibility programs (MILPs).

Calculating support functions is another set analysis tool commonly used to evaluate sets in the directions of constraint functions and/or to construct convex over-approximations. The support function is defined generally as follows.

Definition 2.8 (Support Function) *The support function of a set* $\mathcal{Z} \subset \mathbb{R}^n$ *in the direction* l *is*

$$\rho_{\mathcal{X}}(l) = \max \left\{ l^T x \;\middle|\; x \in \mathcal{X} \right\} \,, \quad (2.14)$$

and it holds that $\mathcal{X} \subset \mathcal{H}_l^-$ *for the supporting halfspace*

$$\mathcal{H}_l^- = \left\{ z \in \mathbb{R}^n \;\middle|\; l^T z \leq \rho_{\mathcal{Z}_h}(l) \right\} \,. \quad (2.15)$$

The support function of a hybrid zonotope $\mathcal{Z}_h = \left\langle G^c, G^b, c, A^c, A^b, b \right\rangle \subset \mathbb{R}^n$ can be evaluated by solving the MILP,

$$\rho_{\mathcal{Z}_h}(l) = \max \left\{ l^T(G^c\xi^c + G^b\xi^b + c) \;\middle|\; \begin{array}{c} A^c\xi^c + A^b\xi^b = b \,, \\ \|\xi^c\|_\infty \leq 1 \,,\, \xi^b \in \{-1,1\}^{n_b} \end{array} \right\} \,. \quad (2.16)$$

The support function evaluated for each direction l_1, l_2,...l_N is denoted

$$\rho_X(L) = \begin{bmatrix} \rho_X(l_1) \\ \vdots \\ \rho_X(l_N) \end{bmatrix}, \tag{2.17}$$

where $L = \begin{bmatrix} l_1 & l_2 & \cdots & l_N \end{bmatrix}$.

Remark 2.2 *The analysis of hybrid zonotopes is significantly more computationally expensive than that for convex sets, e.g., constrained zonotopes require solving linear programs (LPs) rather than MILPs. However, hybrid zonotopes admit closed and efficient identities for nonconvex sets in many instances where convex approximation would incur unacceptable error, as is demonstrated using numerical examples in Chapter 4 and Chapter 5. Furthermore, structure inherent in the hybrid zonotope representation such as underlying binary trees can be analyzed and leveraged by MILP solvers like GUROBI [59] to increase computational speed.*

2.3.3.4 Set Operations

Hybrid zonotopes have been shown to be closed and have identities under linear mapping, Minkowski sum, generalized intersection, generalized halfspace intersection, Minkowski difference, union, Cartesian product, and bounded complements. Identities for each are presented herein, with the exception of bounded complements [24, Section IV].

2.3.3.4.1 Linear Mapping, Minkowski Sum, Generalized Intersection, and Generalized Halfspace Intersection

The identities for linear mapping, Minkowski sum, and generalized intersection of constrained zonotopes from [21, Proposition 1] can be extended to hybrid zonotopes by accounting for the additional binary constraint, as proven in [22, Proposition 7].

Proposition 2.3 (Linear Mapping, Minkowski Sum, Generalized Intersection, and Generalized Halfspace Intersection) [22, Proposition 1]

For any

$$\mathcal{Z}_h = \langle G_z^c, G_z^b, c_z, A_z^c, A_z^b, b_z \rangle , \tag{2.18}$$

$$\mathcal{W}_h = \langle G_w^c, G_w^b, c_w, A_w^c, A_w^b, b_w \rangle \subset \mathbb{R}^n , \tag{2.19}$$

$$\mathcal{Y}_h = \langle G_y^c, G_y^b, c_y, A_y^c, A_y^b, b_y \rangle \subset \mathbb{R}^m , \tag{2.20}$$

$$R \in \mathbb{R}^{m \times n}, \text{ and} \tag{2.21}$$

$$\mathcal{H}^- = \{x \in \mathbb{R}^m \mid h^T x \leq f\}, \tag{2.22}$$

the following identities hold:

$$R\mathcal{Z}_h = \left\langle RG_z^c, RG_z^b, Rc_z, A_z^c, A_z^b, b_z \right\rangle, \tag{2.23}$$

$$\mathcal{Z}_h \oplus \mathcal{W}_h = \left\langle \begin{bmatrix} G_z^c & G_w^c \end{bmatrix}, \begin{bmatrix} G_z^b & G_w^b \end{bmatrix}, c_z + c_w, \begin{bmatrix} A_z^c & \mathbf{0} \\ \mathbf{0} & A_w^c \end{bmatrix}, \begin{bmatrix} A_z^b & \mathbf{0} \\ \mathbf{0} & A_w^b \end{bmatrix}, \begin{bmatrix} b_z \\ b_w \end{bmatrix} \right\rangle, \tag{2.24}$$

$$\mathcal{Z}_h \cap_R \mathcal{Y}_h = \left\langle \begin{bmatrix} G_z^c & \mathbf{0} \end{bmatrix}, \begin{bmatrix} G_z^b & \mathbf{0} \end{bmatrix}, c_z, \begin{bmatrix} A_z^c & \mathbf{0} \\ \mathbf{0} & A_y^c \\ RG_z^c & -G_y^c \end{bmatrix}, \begin{bmatrix} A_z^b & \mathbf{0} \\ \mathbf{0} & A_y^b \\ RG_z^b & -G_y^b \end{bmatrix}, \begin{bmatrix} b_z \\ b_y \\ c_y - Rc_z \end{bmatrix} \right\rangle, \tag{2.25}$$

$$\begin{aligned} \mathcal{Z}_h \cap_R \mathcal{H}^- &= \left\langle \begin{bmatrix} G_z^c & \mathbf{0} \end{bmatrix}, G_z^b, c_z, \begin{bmatrix} A_z^c & \mathbf{0} \\ h^T RG_z^c & \frac{d_m}{2} \end{bmatrix}, \begin{bmatrix} A_z^b \\ h^T RG_z^b \end{bmatrix}, \begin{bmatrix} b_z \\ f - h^T Rc_z - \frac{d_m}{2} \end{bmatrix} \right\rangle, \\ d_m &= f - h^T Rc_z + \sum_{i=1}^{n_{g,z}} |h^T Rg_z^{(c,i)}| + \sum_{i=1}^{n_{b,z}} |h^T Rg_z^{(b,i)}|. \end{aligned} \tag{2.26}$$

Proof *Omitted for brevity. See [22, Proposition 7].*

The time complexity of linear mappings given by (2.23) is $\mathcal{O}(mn(n_g + n_b))$. That of Minkowski sums given by (2.24) is $\mathcal{O}(n)$, and that of generalized intersections given by (2.25) is $\mathcal{O}(mn(n_g + n_b))$ and $\mathcal{O}(n)$ when $R = \mathbf{I}_n$. The time complexity of generalized halfspace intersections given by (2.26) is $\mathcal{O}(mn(n_g + n_b)$ and $\mathcal{O}(n(n_g + n_b))$ for $R = \mathbf{I}_n$.

2.3.3.4.2 Unions of Hybrid Zonotopes

The closure and identity for the union of two hybrid zonotopes is as follows.

Proposition 2.4 (Union) [24, Proposition 1]

For any

$$\mathcal{Z}_h = \langle G_z^c, G_z^b, c_z, A_z^c, A_z^b, b_z \rangle \subset \mathbb{R}^n, \text{ and} \tag{2.27}$$

$$\mathcal{W}_h = \langle G_w^c, G_w^b, c_w, A_w^c, A_w^b, b_w \rangle \subset \mathbb{R}^n, \tag{2.28}$$

define the vectors $\hat{G}^b \in \mathbb{R}^n$, $\hat{c} \in \mathbb{R}^n$, $\hat{A}_z^b \in \mathbb{R}^{n_{c,z}}$, $\hat{b}_z \in \mathbb{R}^{n_{c,z}}$, $\hat{A}_w^b \in \mathbb{R}^{n_{c,w}}$, *and* $\hat{b}_w \in \mathbb{R}^{n_{c,w}}$, *such that*

$$\begin{bmatrix} I & I \\ -I & I \end{bmatrix} \begin{bmatrix} \hat{G}^b \\ \hat{c} \end{bmatrix} = \begin{bmatrix} G_w^b \mathbf{1} + c_z \\ G_z^b \mathbf{1} + c_w \end{bmatrix},$$

$$\begin{bmatrix} -I & I \\ I & I \end{bmatrix} \begin{bmatrix} \hat{A}_z^b \\ \hat{b}_z \end{bmatrix} = \begin{bmatrix} b_z \\ -A_z^b \mathbf{1} \end{bmatrix},$$
$$\begin{bmatrix} -I & I \\ I & I \end{bmatrix} \begin{bmatrix} \hat{A}_w^b \\ \hat{b}_w \end{bmatrix} = \begin{bmatrix} -A_w^b \mathbf{1} \\ b_w \end{bmatrix}.$$

Then the union of $\mathcal{Z}_h$ and $\mathcal{W}_h$ is the hybrid zonotope

$$\mathcal{Z}_h \cup \mathcal{W}_h = \langle G_u^c, G_u^b, c_u, A_u^c, A_u^b, b_u \rangle \subset \mathbb{R}^n, \text{ where}$$

$$\begin{aligned} G_u^c &= \begin{bmatrix} G_z^c & G_w^c & \mathbf{0} \end{bmatrix}, G_u^b = \begin{bmatrix} G_z^b & G_w^b & \hat{G}^b \end{bmatrix}, c_u = \hat{c}, \\ A_u^c &= \begin{bmatrix} A_z^c & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_w^c & \mathbf{0} \\ & A_3^c & I \end{bmatrix}, A_u^b = \begin{bmatrix} A_z^b & \mathbf{0} & \hat{A}_z^b \\ \mathbf{0} & A_w^b & \hat{A}_w^b \\ & A_3^b & \end{bmatrix}, b_u = \begin{bmatrix} \hat{b}_z \\ \hat{b}_w \\ b_3 \end{bmatrix}, \\ A_3^c &= \begin{bmatrix} I & \mathbf{0} \\ -I & \mathbf{0} \\ \mathbf{0} & I \\ \mathbf{0} & -I \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, A_3^b = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \frac{1}{2}\mathbf{1} \\ \mathbf{0} & \mathbf{0} & \frac{1}{2}\mathbf{1} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{2}\mathbf{1} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{2}\mathbf{1} \\ \frac{1}{2}I & \mathbf{0} & \frac{1}{2}\mathbf{1} \\ -\frac{1}{2}I & \mathbf{0} & \frac{1}{2}\mathbf{1} \\ \mathbf{0} & \frac{1}{2}I & -\frac{1}{2}\mathbf{1} \\ \mathbf{0} & -\frac{1}{2}I & -\frac{1}{2}\mathbf{1} \end{bmatrix}, b_3 = \begin{bmatrix} \frac{1}{2}\mathbf{1} \\ \frac{1}{2}\mathbf{1} \\ \frac{1}{2}\mathbf{1} \\ \frac{1}{2}\mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \end{aligned} \tag{2.29}$$

Proof *The proof is given in [24, Proposition 1] and is omitted here for brevity.*

Note that the number of constraints grows rapidly when Proposition 2.4 is used iteratively, e.g., $\bigcup_{i=1}^{N} \mathcal{W}_i$, as the number of additional continuous generators and constraints in (2.29) is dependent on the size of the argument sets [58, Equation 3.32], resulting in increasing memory complexity growth in these variables with each iteration. Only one binary generator is added per union operation. An alternative identity for unions that is advantageous when taking the union of many hybrid zonotopes is presented in Proposition 2.5. It is important to note that (2.29) is used to execute the union in (2.30).

Proposition 2.5 (Efficient Union of Many Hybrid Zonotopes)
Consider the hybrid zonotopes given by

$$\mathcal{Z}_{h,i} = \langle G_{z,i}^c, G_{z,i}^b, c_{z,i}, A_{z,i}^c, A_{z,i}^b, b_{z,i} \rangle \subset \mathbb{R}^n, \quad \forall\, i \in \{1, 2, ..., N\}.$$

Their union is given by

$$\mathcal{U}_i = \left(\begin{bmatrix} \boldsymbol{I}_n \\ \mathbf{0} \end{bmatrix} \mathcal{Z}_{h,i} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \right) \cup \left\{ \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \right\}, \tag{2.30}$$

$$\bigcup_{i=1}^{N} Z_{h,i} = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_{n,1} \end{bmatrix} \left(\left(\bigoplus_{i=1}^{N} \mathcal{U}_i \right) \cap \begin{bmatrix} \mathbf{0}_{1,n} & 1 \end{bmatrix} \left\{ 1 \right\} \right). \tag{2.31}$$

Proof *By applying the set operation definitions for linear transformations and unions,* (2.30) *yields*

$$\mathcal{U}_i = \left\{ \begin{bmatrix} z_i \\ b_i \end{bmatrix} \middle| \begin{array}{c} b_i \in \{0,1\}, \\ z_i \in \begin{cases} \mathcal{Z}_i, & \text{if } b_i = 1 \\ \{\mathbf{0}\}, & \text{if } b_i = 0 \end{cases} \end{array} \right\}. \tag{2.32}$$

Thus the right side of (2.31) *gives*

$$\left\{ \sum_{i=1}^{N} z_i \middle| \begin{array}{c} b_i \in \{0,1\}, \ \forall i \in \{1,2,...,N\}, \\ \sum_{i=1}^{N} b_i = 1, \\ z_i \in \begin{cases} \mathcal{Z}_i, & \text{if } b_i = 1 \\ \{\mathbf{0}\}, & \text{if } b_i = 0 \end{cases}, \quad \forall i \in \{1,2,...,N\} \end{array} \right\}. \tag{2.33}$$

Use of the additive identity and reasoning from the constraints on b_i that there exists exactly one i such that $b_i = 1$ completes the proof. □

When executed with hybrid zonotopes, the memory complexity of unions by (2.31) is proportional to the total complexity of all $\mathcal{Z}_i$. Example 2.4 demonstrates key concepts associated with the identity (2.31) and Example 2.5 compares the growth in memory complexity to iterative use of the identity given by (2.29).

Example 2.4 *Consider the four hybrid zonotopes given by*

$$\mathcal{Z}_1 = \left\langle \boldsymbol{I}_2, \emptyset, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \emptyset, \emptyset, \emptyset \right\rangle,$$
$$\mathcal{Z}_2 = \left\langle \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \emptyset, \begin{bmatrix} -3 \\ -3 \end{bmatrix}, \emptyset, \emptyset, \emptyset \right\rangle,$$
$$\mathcal{Z}_3 = \left\langle \begin{bmatrix} -\frac{1}{2} & 0 & 0 \\ -2 & -\frac{1}{2} & 0 \end{bmatrix}, \emptyset, \begin{bmatrix} -\frac{5}{2} \\ \frac{5}{2} \end{bmatrix}, \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}, \emptyset, 1 \right\rangle,$$
$$\mathcal{Z}_4 = \left\langle \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ 2 & \frac{1}{2} & 0 \end{bmatrix}, \emptyset, \begin{bmatrix} \frac{5}{2} \\ -\frac{5}{2} \end{bmatrix}, \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}, \emptyset, 1 \right\rangle,$$

and plotted in Figure 2.4(a). The hybrid zonotope resulting using the union identity (2.31) *is plotted in Figure 2.4(b), though significant insight into the identity is provided by plotting an intermediate result shown in (c),* $\bigoplus_{i=1}^{4} \mathcal{U}_i$ *with the third dimension corresponding to how many sets are "chosen". The plot demonstrates the Minkowski sum of combinations of* $\mathcal{Z}_1$*,* $\mathcal{Z}_2$*,* $\mathcal{Z}_3$*, and* $\mathcal{Z}_4$*. We briefly adopt standard "n-choose-r" notation* $\binom{N}{r}$*, where* $N = 4$ *corresponds to the total number of hybrid zonotopes. Each value in the third dimension of Figure 2.4(c) relates to* r*, e.g., for* $r = 1$ *the Minkowski sum where only one set* $\mathcal{Z}_i$ *is chosen, and the remaining sets are the origin, which corresponds to the union. This gives insight into the generalized intersection in* (2.31) *with* $\{1\}$*. For* $r = 2$ *there are six convex sets plotted which correspond to* $\binom{4}{2}$ *potential Minkowski sums of two sets from four possible sets, given by*

$$\mathcal{Z}_1 \oplus \mathcal{Z}_2, \quad \mathcal{Z}_1 \oplus \mathcal{Z}_3, \quad \mathcal{Z}_1 \oplus \mathcal{Z}_4, \quad \mathcal{Z}_2 \oplus \mathcal{Z}_3, \quad \mathcal{Z}_2 \oplus \mathcal{Z}_4, \quad \text{and} \quad \mathcal{Z}_3 \oplus \mathcal{Z}_4 \,.$$

Example 2.5 *In this example, consider* N *zonotopes* $\mathcal{Z}_i \in \mathbb{R}^2$ *,* $\forall\ i \in \{1, 2, ..., N\}$*, each with two generators, i.e.,* $n_{g,i} = 2$*. The union*

$$Z_u = \bigcup_{i=1}^{N} \mathcal{Z}_i \,, \tag{2.34}$$

is calculated using two methods for $N \in \{2, 3, ..., 20\}$*. Method 1 (M1) iterates over* (2.29) *and Method 2 (M2) calculates the union via* (2.31)*. Resulting memory complexities are plotted in Figure 2.5 demonstrating the memory complexity advantages of Method 2.*

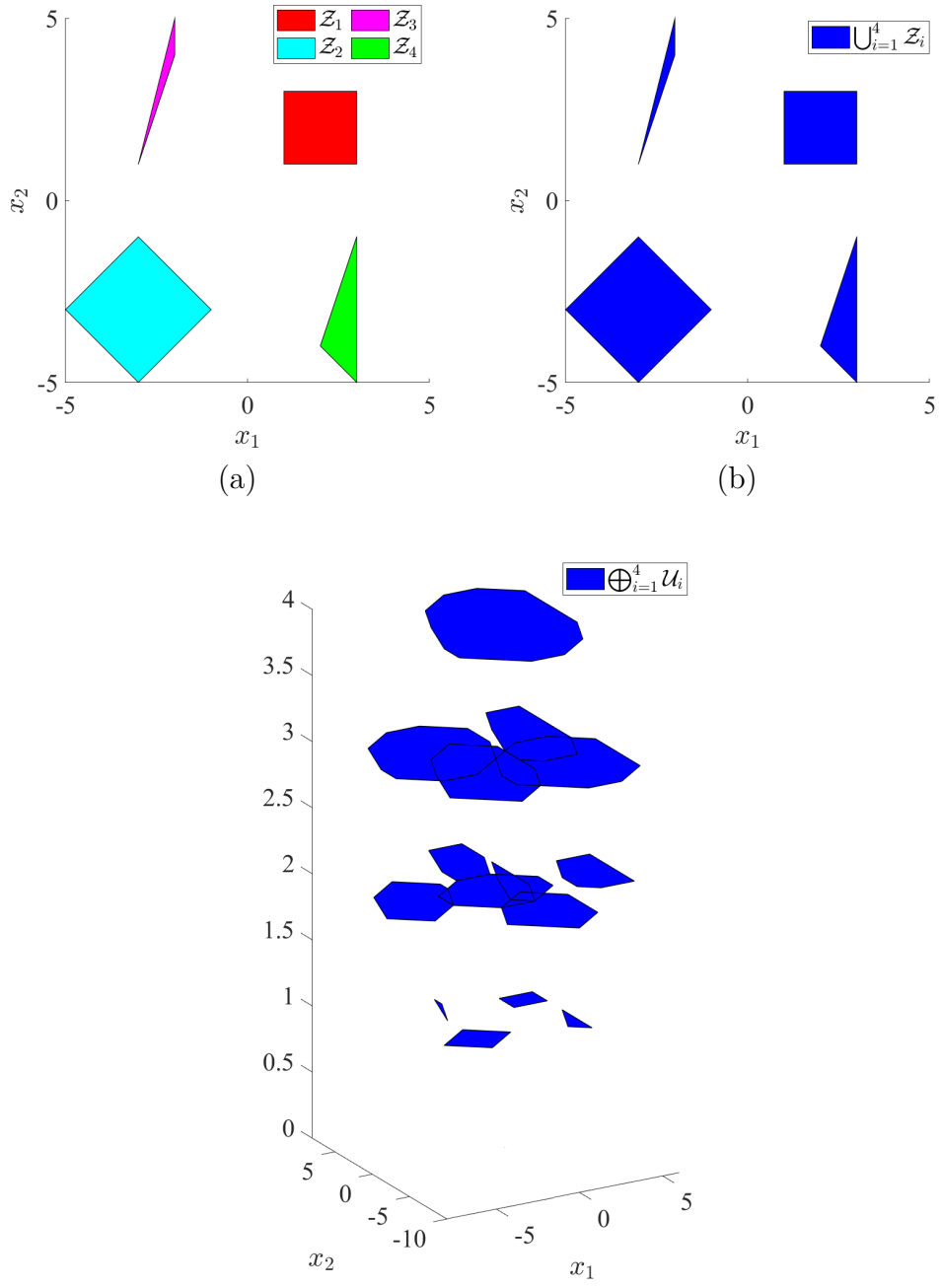
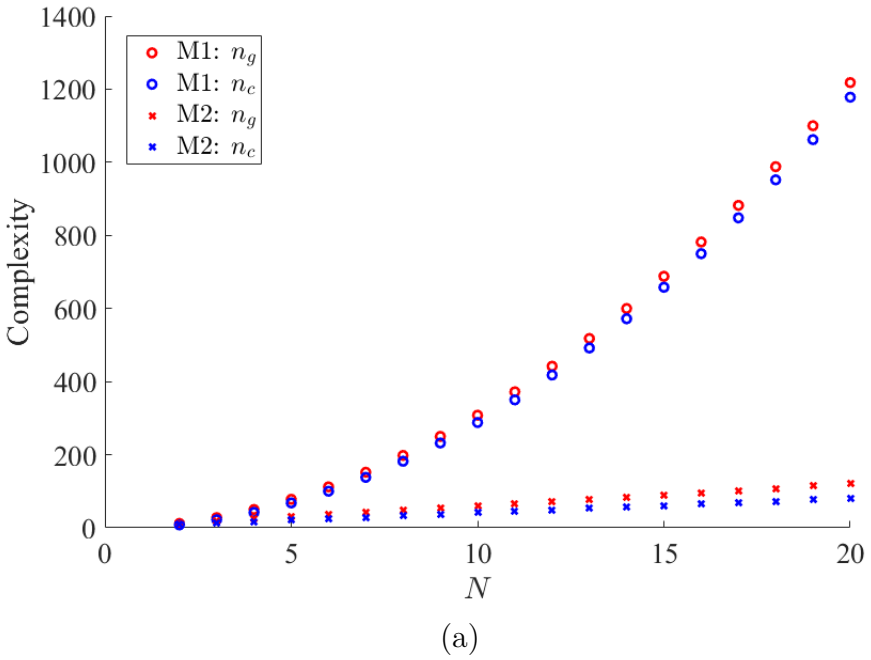
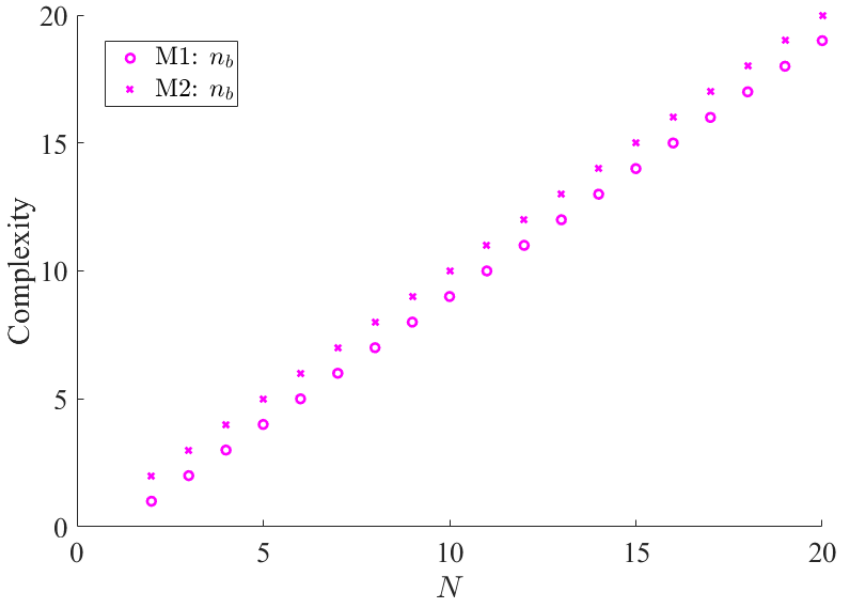


Figure 2.4: Example of supplemental union of hybrid zonotopes using (2.31). (b) shows the resulting hybrid zonotope from the union of the sets $\mathcal{Z}_i$, $\forall i \in \{1, 2, 3, 4\}$ plotted in (a) using the identity given in (2.31). (c) plots an intermediate set which relates to all $\binom{4}{0}$, $\binom{4}{1}$, $\binom{4}{2}$, $\binom{4}{3}$, $\binom{4}{4}$ combinations for the Minkowski sum of $\mathcal{Z}_i$, $\forall i \in \{1, 2, 3, 4\}$.



(a)



(b)

Figure 2.5: Memory complexity growth comparison of two methods to compute unions of hybrid zonotopes.
M1 uses iterative unions by (2.29) and yields quadratic growth in n_g and n_c. M2 calculates unions via (2.31), which results in linear growth in n_g and n_c. Both M1 and M2 result in linear growth in n_b.

2.3.3.4.3 Minkowski Difference with a Zonotope

The identity for a Minkowski difference of a zonotope from a hybrid zonotope is given in **Proposition 2.6**.

Proposition 2.6 (Minkowski Difference with Zonotope) [60, Proposition 1]
For any $\mathcal{Z}_h = \langle G_z^c, G_z^b, c_z, A_z^c, A_z^b, b_z \rangle$ *and* $\mathcal{W} = \langle G_w, c_w \rangle \subset \mathbb{R}^n$ *for* $G_w = [g_w^{(1)} \dots g_w^{(n_{g,w})}]$, *the Minkowski difference* $\mathcal{Z}_d = \mathcal{Z}_h \ominus \mathcal{W}$ *is a hybrid zonotope computed by the recursion:*

$$\mathcal{Z}_{int}^{(0)} = \mathcal{Z}_h - c_w \,, \tag{2.35a}$$

$$\mathcal{Z}_{int}^{(i)} = \left(\mathcal{Z}_{int}^{(i-1)} + g_w^{(i)}\right) \cap \left(\mathcal{Z}_{int}^{(i-1)} - g_w^{(i)}\right) \,, \tag{2.35b}$$

$$\mathcal{Z}_d = \mathcal{Z}_{int}^{(n_{g,w})} \,. \tag{2.35c}$$

Proof *The proof mirrors that for Minkowski difference of zonotopes [35, Theorem 1]. There, only the subtrahend* $\mathcal{W}$ *is specified as a zonotope, while the minuend* $\mathcal{Z}_h$ *is an arbitrary set. The hybrid zonotope's closure under Minkowski sums and intersections allows the recursion* (2.35) *to be generated through a finite number of set operations.* □

Time complexity of Minkowski differences given by Proposition 2.6, dominated by vector addition, is $\mathcal{O}(nn_{g,w})$. Complexity of the resulting set, $\mathcal{Z}_d$, is $n_{g,d} = 2^{n_{g,w}} n_{g,h}$, $n_{b,d} = 2^{n_{g,w}} n_{b,h}$, and $n_{c,d} = 2^{n_{g,w}} n_{c,h} + n n_{g,w}$. These can be derived by applying the Minkowski sum and intersection operation results of [22].

2.3.3.5 Exact Set Representation Conversions

This section overviews methods to exactly convert between common polytope representations (H-rep, V-rep, CG-rep) and hybrid zonotopes. Many sets are naturally and efficiently expressed as a collection of polytopes, e.g., polytopic obstacle maps in two dimensions [61]. While hybrid zonotopes can compactly represent complex[2] sets and have efficient identities for many set operations, it can be challenging to *construct* a hybrid zonotope to represent a complex set of interest. This section presents methods to convert sets represented as a collection polytopes to a hybrid zonotope.

2.3.3.5.1 Collections of CG-rep Polytopes to HCG-rep Conversion from a single CG-rep to HCG-rep is trivial, i.e.,

$$\langle G^c, c, A^c, b \rangle = \langle G^c, \emptyset, c, A^c, \emptyset, b \rangle \,. \tag{2.36}$$

[2] unions of exponential numbers of polytopes

Once each CG-rep polytope is converted to a hybrid zonotope, a single hybrid zonotope can be obtained either by iteration of (2.29) or by the identity given in (2.31). Computational advantages of the latter are discussed in Example 2.5 and preceding text.

2.3.3.5.2 Collections of H-rep Polytopes to HCG-rep

The conversion from H-rep to HCG-rep first generates an interval, represented in HCG-rep for each H-rep polytope such that each H-rep polytope is contained within its corresponding interval. Then halfspace intersections (2.26) are enforced on each interval to generate an equivalent HCG-representation for each H-rep polytope. Finally a single hybrid zonotope can be obtained either by iteration of (2.29) or by the identity given in (2.31). Computational advantages of the latter are discussed in Example 2.5 and preceding text.

The process to convert an H-rep polytope to HCG-rep, which is a natural extension of [21, Theorem 1], is now summarized for the H-rep polytope

$$\mathcal{H} = \left\{ x \;\middle|\; \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n_h} \end{bmatrix} x \leq \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n_h} \end{bmatrix} \right\} \subset \mathbb{R}^n \,. \tag{2.37}$$

Single H-rep Polytope to HCG-rep

1. Generate a set $\mathcal{P}$ such that $\mathcal{H} \subseteq \mathcal{P}$. This can be achieved by sampling the support functions

$$\begin{aligned} \bar{x} &= \quad \rho_{\mathcal{H}}(\ \ \mathbf{I}_n) \\ \underline{x} &= -\rho_{\mathcal{H}}(-\mathbf{I}_n) \end{aligned}$$

   and generating the zonotope

$$\mathcal{P} = \left\langle \mathrm{diag}\Big(\frac{\bar{x} - \underline{x}}{2}\Big),\ \frac{\bar{x} + \underline{x}}{2} \right\rangle \,, \tag{2.38}$$

   which by construction is the interval hull of $\mathcal{H}$. Note that sampling the support functions to generate $\bar{x}$ and $\underline{x}$ requires solving $2n$ linear programs.

2. Trivial conversion to HCG-rep is given by

$$\mathcal{P} = \left\langle \mathrm{diag}\big(\frac{\bar{x} - \underline{x}}{2}\big),\ \emptyset,\ \frac{\bar{x} + \underline{x}}{2},\ \emptyset,\ \emptyset,\ \emptyset \right\rangle . \tag{2.39}$$

2.3.3.5.3 Collections of V-rep Polytopes to HCG-rep

Theorem 2.1 (V-rep Collection to Hybrid Zonotope) [62, Theorem 5]

A set $\mathcal{S}$ consisting of the union of N V-rep polytopes, $\mathcal{S} = \cup_{i=1}^{N} \mathcal{P}_i$, with a total of n_v vertices can be represented as a hybrid zonotope with memory complexity

$$\begin{aligned} n_g &= 2n_v \,, \\ n_b &= N \,, \\ n_c &= n_v + 2 \,. \end{aligned} \tag{2.40}$$

Proof *Define the vertex matrix $V = [v_1, \ldots, v_{n_v}] \in \mathbb{R}^{n \times n_v}$ and construct a corresponding incidence matrix $M \in \mathbb{R}^{n_v \times N}$ with entries $M_{(j,i)} \in \{0, 1\}$, $\forall\, i, j$, such that*

$$\mathcal{P}_i = \left\{ V\lambda \;\middle|\; \lambda_j \in \begin{cases} [0\ 1], & \text{if } j \in \{k \mid M_{(k,i)} = 1\} \\ \{0\}, & \text{if } j \in \{k \mid M_{(k,i)} = 0\} \end{cases}, \ \mathbf{1}_{n_v}^T \lambda = 1 \right\} . \tag{2.41}$$

Define the hybrid zonotope

$$\mathcal{Q} = \frac{1}{2} \left\langle \begin{bmatrix} \mathbf{I}_{n_v} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_N \end{bmatrix}, \begin{bmatrix} \mathbf{1}_{n_v} \\ \mathbf{1}_N \end{bmatrix}, \begin{bmatrix} \mathbf{1}_{n_v}^T \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{1}_N^T \end{bmatrix}, \begin{bmatrix} 2 - n_v \\ 2 - N \end{bmatrix} \right\rangle ,$$

and the polyhedron $\mathcal{H} = \{h \in \mathbb{R}^{n_v} \mid h \leq \mathbf{0}\}$, and let

$$\mathcal{D} = \mathcal{Q} \cap_{[\mathbf{I}_{n_v} \ -M]} \mathcal{H} \,. \tag{2.42}$$

Then the set $\mathcal{S}$ is equivalently given by the hybrid zonotope

$$\mathcal{Z}_S = \begin{bmatrix} V & \mathbf{0} \end{bmatrix} \mathcal{D} \,. \tag{2.43}$$

By direct application of set operation identities provided in [22, Section 3.2], it can be shown that $\mathcal{Z}_\mathcal{S}$ has the complexity given by (2.40). The remainder of the proof shows equivalency of $\mathcal{Z}_\mathcal{S}$ and $\mathcal{S}$. For any $(\lambda, \delta) \in \mathcal{D}$ there exists some $(\xi^c, \xi^b) \in \mathcal{B}_\infty^{n_v} \times \{-1, 1\}^N$

such that $\mathbf{1}_{n_v}^T \xi^c = 2 - n_v$, $\mathbf{1}_N^T \xi^b = 2 - N$, $\lambda = 0.5\xi^c + 0.5\mathbf{1}_{n_v}$, $\delta = 0.5\xi^b + 0.5\mathbf{1}_N$, *and* $\lambda - M\delta \in \mathcal{H} \implies \lambda \leq M\delta$. *Thus* $\lambda \in [0,1]^{n_v}$, $\delta \in \{0,1\}^N$, $\sum_{i=1}^{n_v} \lambda_i = 1$, *and* $\sum_{i=1}^{N} \delta_i = 1$ *results in* $\delta_i = 1 \implies \delta_{j \neq i} = 0$. *Let* $\delta_i = 1$, *then* $\lambda \leq M\delta$ *enforces* $\lambda_j \in [0,1]$, $\forall\ j \in \{k \mid M_{(k,i)} = 1\}$ *and* $\lambda_j = 0\ \forall\ j \in \{k \mid M_{(k,i)} = 0\}$. *Therefore given any* $z \in \mathcal{Z}_S$ *corresponding to* $\delta_i = 1$,

$$z = \sum \lambda_j v_j\ ,\ \forall\ j \in \{k \mid M_{(k,i)} = 1\}\ , \tag{2.44}$$

thus $z \in \mathcal{P}_i \subseteq \mathcal{S}$ *and* $\mathcal{Z}_S \subseteq\ \mathcal{S}$.

Conversely, given any $x \in \mathcal{S}$, $\exists\ i$ *such that* $x \in \mathcal{P}_i = \sum \lambda_j v_j\ ,\ \forall\ j \in \{k \mid M_{(k,i)} = 1\}$. *This is equivalent to* (2.44), *therefore* $x \in \mathcal{Z}_\mathcal{S}$, $\mathcal{S} \subseteq \mathcal{Z}_\mathcal{S}$, *and* $\mathcal{Z}_\mathcal{S} = \mathcal{S}$. □

Applying Theorem 2.1 results in complexity $\mathcal{O}(n(n_v + N)^2)$, both improving the computational complexity and memory complexity of representing a collection of V-rep polytopes as a single hybrid zonotope as compared to pre-existing methods. The pre-existing approach would be to convert each polytope from V-rep to H-rep to HCG-rep, and then take iterative unions using the method given in [24]. However, this would involve greater computational complexity as conversion between V-rep and H-rep alone has worst-case exponential computational complexity [21], and produces a set with memory complexity that scales quadratically with the number of polytopes. Furthermore Theorem 2.1 allows for the intuitive representation of sets as hybrid zonotopes through the use of the vertex matrix and incidence matrix, as demonstrated in Example 2.6.

Example 2.6 *Consider a vertex matrix* $V = [v_1, v_2, v_3]$ *consisting of the vertices of a triangle. The set of vertices, the set of points along the edges of the triangle, and the convex hull of the vertices can be found as a hybrid zonotope using Theorem 2.1 and the respective incidence matrices*

$$M_{vertices} = \boldsymbol{I}_3\ ,\ M_{edges} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}\ ,\ M_\Delta = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\ .$$

2.3.3.6 Approximate Set Representation Conversion

Set complexity often grows as set operations are performed, as can be seen for all set operations in Section 2.3.3.4 except for linear mappings. One method to curb growth in memory complexity is take a convex over-approximation of the set. First we define an interval over-approximation.

2.3.3.6.1 Interval Over-approximation of HCG-rep Consider the hybrid zonotope $\mathcal{Z}_h = \langle G_z^c, G_z^b, c_z, A_z^c, A_z^b, b_z \rangle$.

1. Generate a set $\mathcal{P}$ such that $\mathcal{Z}_h \subseteq \mathcal{P}$. This can be achieved by sampling the support functions

$$\begin{aligned} \bar{x} = &\quad \rho_{\mathcal{Z}}(\ \ \mathbf{I}_n) \\ \underline{x} = &-\rho_{\mathcal{Z}}(-\mathbf{I}_n) \end{aligned}$$

   and generating the zonotope

$$\mathcal{P} = \left\langle \text{diag}\Big(\frac{\bar{x} - \underline{x}}{2}\Big),\ \frac{\bar{x} + \underline{x}}{2} \right\rangle ,$$

   which by construction is the interval hull of $\mathcal{Z}$. Note that sampling the support functions to generate $\bar{x}$ and $\underline{x}$ requires solving $2n$ MILPs.

2. Trivial conversion to HCG-rep is given by

$$\mathcal{P} = \left\langle \text{diag}\Big(\frac{\bar{x} - \underline{x}}{2}\Big),\ \emptyset,\ \frac{\bar{x} + \underline{x}}{2},\ \emptyset,\ \emptyset,\ \emptyset \right\rangle . \tag{2.45}$$

2.3.3.6.2 Polytope Over-approximation of HCG-rep More generally, convex approximations for arbitrary support function directions $L = [l_1, ..., l_N]$ are generated as follows. Consider the hybrid zonotope $\mathcal{Z}_h = \langle G_z^c, G_z^b, c_z, A_z^c, A_z^b, b_z \rangle$.

1. Construct an interval approximation via (2.45).

2. Sample the support function in the desired directions, i.e.,

$$f = \rho_{\mathcal{Z}_h}(L) .$$

3. Generate the polyhedron

$$\mathcal{H} = \Big\{ x \mid L^T x \leq f \Big\} .$$

4. The convex over-approximation $\bar{\mathcal{Z}}_h$ is given by

$$\bar{\mathcal{Z}}_h = P \cap \mathcal{H} . \tag{2.46}$$

Chapter 3
Graphs of Functions

This chapter presents results for graphs of functions. Section 3.1 defines graphs of functions, presents identities to calculate a set of outputs that are achievable for a function given a set of inputs, and vice versa. These are fundamental methods used in this thesis to compute reachable sets in Chapter 4 and perform set-valued state estimation and parameter identification in Chapter 5. The promising computational complexity and memory complexity growth of the input-output identities when sets are represented using hybrid zonotopes motivates following sections. Section 3.2 provides several methods to construct hybrid zonotope graphs of functions using special ordered sets (SOS). Section 3.3 presents methods to extend results in Section 3.2 to sets in higher dimensions using functional decomposition and Section 3.4 provides several hybrid zonotope representations for several common functions.

3.1 Definition and Input-Output Identities

Consider a set-valued mapping $\phi : \mathrm{D}_p \to \mathrm{D}_q$ which assigns each element $p \in \mathrm{D}_p \subset \mathbb{R}^{n_p}$ to a subset of $\mathrm{D}_q \subset \mathbb{R}^{n_q}$. A graph of a function is defined as the set of ordered pairs associated with ϕ and the domain D_p as follows.

Definition 3.1 (Graph of a function) *Given a set-valued mapping $\phi : \mathrm{D}_p \to \mathrm{D}_q$, the graph of the function ϕ, denoted Φ, is defined as*

$$\Phi \equiv \left\{ \begin{bmatrix} p \\ q \end{bmatrix} \;\middle|\; \begin{array}{c} q \in \phi(p) \\ p \in \mathrm{D}_p \end{array} \right\}. \tag{3.1}$$

A graph of the function $q = \sin(p)$ for $\mathrm{D}_p = \left[0\,,\ \ 2\pi\right]$ is shown in Figure 3.1, along with the sets D_p and D_q.

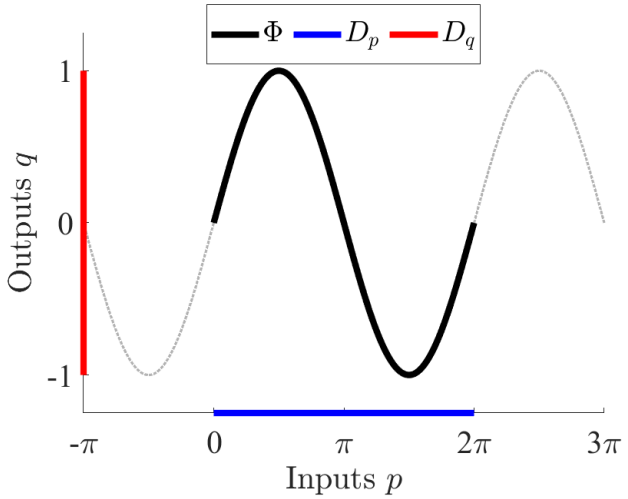


Figure 3.1: Example graph of a function for $q = \sin(p)$.
The sets D_p and D_q are also shown projected onto the axes and $\sin(p)$ is plotted over a larger domain than D_p.

Using graphs of functions, the set of outputs given a set of inputs, and vice versa, are given by the identities[1] presented in Theorem 3.1 and Theorem 3.2, respectively.

Theorem 3.1 (Input Set to Output Set) [63, Theorem 2]

If the set of inputs satisfies $\mathcal{P} \subseteq \mathrm{D}_p$, then the set of outputs, defined as

$$\mathcal{Q} \equiv \{q \mid q \in \phi(p),\ p \in \mathcal{P}\}\ , \tag{3.2}$$

can be calculated using the graph of the function ϕ, $\Phi = \{(x, \phi(x)) | x \in \mathrm{D}_p\}$, as

$$\mathcal{Q} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{n_q} \end{bmatrix} \left(\Phi \cap \begin{bmatrix} \boldsymbol{I}_{n_p} & \mathbf{0} \end{bmatrix} \mathcal{P} \right)\ . \tag{3.3}$$

Proof *Direct application of the definitions of generalized intersection and linear transformation to the right side of* (3.3) *yields*

$$\begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{n_q} \end{bmatrix} \left(\Phi \cap \begin{bmatrix} \boldsymbol{I}_{n_p} & \mathbf{0} \end{bmatrix} \mathcal{P} \right) = \{q \mid q \in \phi(p),\ p \in \mathcal{P} \cap \mathrm{D}_p\}\ .$$

By the assumption $\mathcal{P} \subseteq \mathrm{D}_p \implies \mathcal{P} \cap \mathrm{D}_p = \mathcal{P}$. □

[1] Similar identities specific to reachable sets of hybrid systems were first presented by the author in [60]. Later they were used for nonlinear reachability in [62]. In [63] they were generalized to input-output identities as presented here.

The assumption that $\mathcal{P} \subseteq \mathrm{D}_p$ is not restrictive as Φ can be constructed for a domain of interest D_p.

Continuing the example above, Figure 3.2(a) demonstrates how (3.3) can be used to calculate the set of outputs $\mathcal{Q}$ given a set of inputs $\mathcal{P} = \left[\frac{5\pi}{6}\ ,\ \ \frac{7\pi}{6}\right]$. If (3.3) is used when the condition that $\mathcal{P} \subseteq \mathrm{D}_p$ in Theorem 3.1 is *not* met, e.g., $\mathcal{P} = \left[\frac{3\pi}{2}\ ,\ \ 3\pi\right] \not\subseteq \mathrm{D}_p$, the identity yields the set of outputs corresponding to the set $\mathcal{P} \cap \mathrm{D}_p$. An example of this is shown in Figure 3.2(b) and the proof is the first line of the proof for Theorem 3.1.

Theorem 3.2 (Output Set to Input Set) [63, Theorem 3]

The set of inputs $\mathcal{P}_{\mathrm{D}_p}$ within a region of interest D_p and consistent with the output set $\mathcal{Q}$, defined by

$$\mathcal{P}_{\mathrm{D}_p} \equiv \{p \mid q \in \phi(p),\ q \in \mathcal{Q},\ p \in \mathrm{D}_p\}\ , \tag{3.4}$$

can be calculated using the graph of the function ϕ, $\Phi = \{(x, \phi(x)) | x \in \mathrm{D}_p\}$, as

$$\mathcal{P}_{\mathrm{D}_p} = \begin{bmatrix} \boldsymbol{I}_{n_p} & \mathbf{0} \end{bmatrix} \left(\Phi \cap \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{n_q} \end{bmatrix} \mathcal{Q} \right)\ . \tag{3.5}$$

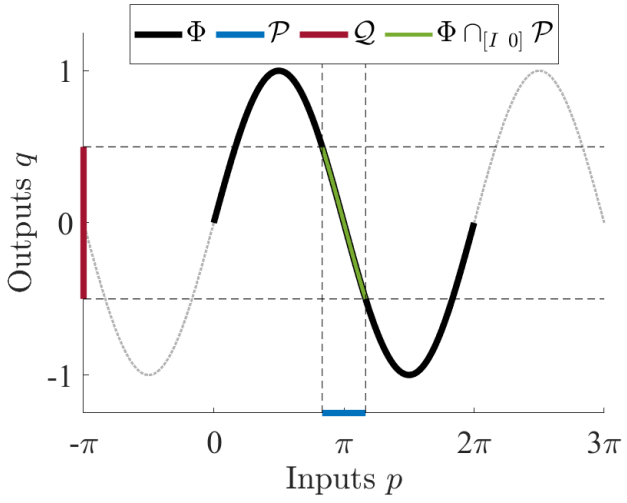
Proof *By direct application of the definitions of generalized intersection and linear transformation, the right side of* (3.5) *yields* (3.4). □

The condition on the input set of (3.4) that $p \in \mathrm{D}_p$ is not restrictive as Φ can be constructed for a domain of interest D_p. An example of calculating an input set, given an output set and graph of a function is depicted in Figure 3.3.

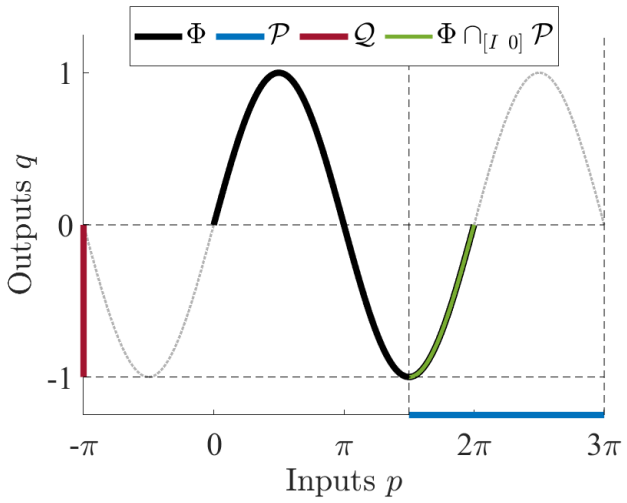
3.1.1 Set Representations for Graph of Function Input-Output Identities

In this section, set representations are considered for use with the input-output identities presented as (3.3) and (3.5). The identities in (3.3) and (3.5) use linear transformations and generalized intersections, therefore the chosen set representation should be closed and ideally computationally efficient and scalable for these operations. Furthermore, the set representation must be able to represent the graph of functions of interest. When the function is nonlinear, it becomes important that the set representation can capture nonconvexities and potential discontinuities inherent to its graph.

In addition to hybrid zonotopes, other set representations such as sublevel sets, constrained polynomial zonotopes [17], and polynomial logical zonotopes [20] exhibit



(a)



(b)

Figure 3.2: Example of input set to output set via graph of a function (3.3) for $q = \sin(x)$. (a) The identity (3.3) is used to calculate the set of outputs $\mathcal{Q}$ achievable given a set of inputs $\mathcal{P} = \left[\frac{5\pi}{6}\,,\ \ \frac{7\pi}{6}\right]$. The condition that $\mathcal{P} \subseteq \mathrm{D}_p$ in Theorem 3.1 is met as $\left[\frac{5\pi}{6}\,,\ \ \frac{7\pi}{6}\right] \subset \mathrm{D}_p = \left[0\,,\ \ 2\pi\right]$, shown in Figure 3.1. (b) If (3.3) is used when the condition that $\mathcal{P} \subseteq \mathrm{D}_p$ in Theorem 3.1 is *not* met, e.g., $\mathcal{P} = \left[\frac{3\pi}{2}\,,\ \ 3\pi\right] \not\subseteq \mathrm{D}_p$, the identity yields the set of outputs for the intersection $\mathcal{P} \cap \mathrm{D}_p$. Grey dashed lines correspond to the generalized intersection and linear transformations in (3.3)

some/all of these properties for various functions. Sublevel sets are expressive, but their general representation often makes order reduction and analysis challenging. Constrained polynomial zonotopes appear to be a viable set representation for the input-output

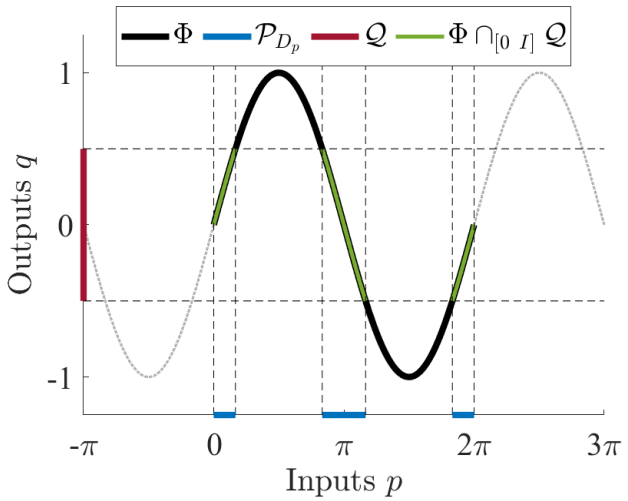


Figure 3.3: Example of output set to input set via graph of a function (3.5) for $q = \sin(x)$. Note that the resulting set of inputs $\mathcal{P}_{\mathrm{D}_p}$ is both nonconvex and disjoint. Grey dashed lines correspond to the generalized intersection and linear transformations in (3.5).

identities but cannot exactly represent PWA functions of interest to this thesis. Logical zonotopes are limited to use with logical systems. The interested reader is referred to [12, Table 1] and references therein for alternative set representations.

3.1.1.1 Hybrid Zonotopes for Graph of Function Input-Output Identities

The remainder of this thesis uses hybrid zonotopes for three reasons. Firstly, the use of hybrid zonotopes to execute the identities given in (3.3) and (3.5) results in linear growth in memory complexity. Specifically, the identities in (3.3) and (3.5) of Theorem 3.1 and Theorem 3.2, respectively, result in memory complexity

Theorem 3.1	**Theorem 3.2**
$n_{g,q} = n_{g,p} + n_{g,\phi}$,	$n_{g,p} = n_{g,q} + n_{g,\phi}$,
$n_{b,q} = n_{b,p} + n_{b,\phi}$,	$n_{b,p} = n_{b,q} + n_{b,\phi}$,
$n_{c,q} = n_{c,p} + n_{c,\phi} + n$,	$n_{c,p} = n_{c,q} + n_{c,\phi} + n$.

Secondly, the computational complexities of (3.3) and (3.5) scale linearly with n_p and n_q. Thirdly, later sections will show how piecewise-affine approximations of nonlinear functions, such as SOS approximations and neural net approximations, can be represented exactly and efficiently using hybrid zonotopes.

A fundamental challenge is that a given set representation and its operations can

only achieve computation of *exact* input and output sets for a limited class of functions [64]. It follows that graphs of functions can only be *exactly* computed for a limited class of functions. To obtain formal guarantees for a broader class of functions, *over-approximations* of graphs of functions can be computed instead. To this end, Corollary 3.1 extends the previous results by considering the effect of using an over-approximation of the graph of a function.

Corollary 3.1 (Effect of Using an Over-Approximated Graph of a Function) *For the identities* (3.3) *and* (3.5) *provided by Theorem 3.1 and Theorem 3.2, respectively, if* Φ *is replaced with an over-approximation* $\bar{\Phi}$*, then the identities will instead yield over-approximations of the left sides.*

Proof *Set containment is preserved under linear transformation and generalized intersection.* □

Corollary 3.1 motivates the following section on SOS approximations, which can be used to generate hybrid zonotope over-approximations of graphs of nonlinear functions. Furthermore, the classes of functions for which hybrid zonotopes can be implemented can now be formally stated.

Corollary 3.2 (Exact Graph of a Function Using HCG-rep) *Hybrid zonotopes can exactly represent the graph of a function if and only if the graph of a function is the union of a finite number of polytopes.*

Proof *A hybrid zonotope can represent any polytope (Theorem 2.1 provides a constructive proof) and hybrid zonotopes are closed under union (Proposition 2.4), which implies that a hybrid zonotope can represent the graph of a function if the graph of the function is the union of a finite number of polytopes. If the graph of a function is not the union of a finite number of polytopes, then it can be reasoned that there does not exist a hybrid zonotope that can* exactly *represent the graph of the function. This is because a hybrid zonotope is the union of* 2^{n_b} *constrained zonotopes corresponding to each combination of binary factors, and constrained zonotopes are polytopes.* □

Corollary 3.3 (Over-Approximated Graph of a Function Using HCG-rep) *Hybrid zonotopes can* over-approximate *a graph of any function if and only if the graph of the function is bounded.*

Proof *A hybrid zonotope is the union of* 2^{n_b} *constrained zonotopes corresponding to each combination of binary factors, and constrained zonotopes are polytopes. Therefor*

hybrid zonotopes cannot represent unbounded graphs of function. Furthermore, if a set is bounded, then there exists a polytope that contains the set.[2] *Any polytope can be represented as a hybrid zonotope (Theorem 2.1 provides a constructive proof).* □

Corollary 3.2 and Corollary 3.3 are the provide necessary *and* sufficient conditions for using hybrid zonotopes to represent exact and over-approximated graphs of functions. Later sections will use of hybrid zonotopes to represent graphs of functions for several classes of hybrid and nonlinear systems wherein the ability to represent each class of system will be discussed in detail.

In Corollary 3.3, the restriction that the graph of the function applies to both the input and output space of the function, i.e., hybrid zonotopes cannot represent the graph of a function where D_p is unbounded, nor can they represent the graph of a function if $\exists x \in \mathrm{D}_p$ such that $\phi(x)$ is undefined.

While the proof of Corollary 3.3 uses the existence of a single polytope, in practice such an approximation often would incur lead to large approximation error. The following section on special ordered set approximation presents more sophisticated methods for over-approximating graphs of nonlinear functions.

3.2 Special Ordered Set Approximations

3.2.1 SOS Approximations Using Hybrid Zonotopes

Piecewise affine approximations, and related theory [30, 65], provide fundamental tools used for system identification [66, 67], control of nonlinear systems [38, 68, 69], and reachability of nonlinear systems [70, 71].

Special Ordered Set approximations, a type of PWA approximation, were originally developed to approximate solutions of nonlinear optimization programs by replacing nonlinear functions with piecewise-linear approximations [72]. Herein an SOS approximation is defined equivalently to the definition given in [73, Section 1.2]. An incidence matrix is introduced to mirror the structure given to collections of V-rep polytopes earlier in (2.41).

Definition 3.2 (SOS Approximation) *An SOS approximation $\mathcal{S}$ of a scalar-valued function $g(x) : \mathbb{R}^n \to \mathbb{R}$ is the union of N polytopes, i.e., $\mathcal{S} = \cup_{i=1}^{N} \mathcal{P}_i$. The collection of polytopes is defined by a vertex matrix $V = [v_1, \ldots, v_{n_v}] \in \mathbb{R}^{(n+1)\times n_v}$, where $v_i =$*

[2] e.g., the interval hull of the closure of the set

$(x_i, g(x_i))$, $\forall i$, *and a corresponding incidence matrix* $M \in \mathbb{R}^{n_v \times N}$ *with entries* $M_{(j,i)} \in \{0, 1\}$, $\forall\, i, j$, *such that*

$$\mathcal{P}_i = \left\{ V\lambda \;\middle|\; \lambda_j \in \begin{cases} \left[0\,, \quad 1\right], & \text{if } j \in \{k \mid M_{(k,i)} = 1\} \\ \{0\}, & \text{if } j \in \{k \mid M_{(k,i)} = 0\} \end{cases}, \; \mathbf{1}_{n_v}^T \lambda = 1 \right\}, \tag{3.6}$$

$$([\boldsymbol{I}_n \; \mathbf{0}]\mathcal{P}_i)^\circ \cap [\boldsymbol{I}_n \; \mathbf{0}]\mathcal{P}_j = \emptyset\,, \quad \forall i \neq j\,, \text{and} \tag{3.7}$$

$$\mathbf{1}^T M_{(\cdot,i)} \leq n + 1\,, \quad \forall\; i \in \{1, ..., N\}\,. \tag{3.8}$$

The set of points x_i, $i \in \{1, ..., n_v\}$ are referred to as the *sampling* of the first n dimensions. Each polytope $\mathcal{P}_i$ given by (3.6) is the convex hull of the vertices given by $V_{(\cdot,i)}$ corresponding to the index of all entries of $M_{(\cdot,i)}$ that equal 1. The constraint (3.7) enforces that none of the simplices' interiors "overlap" while allowing for sharing of topological boundaries. The constraint (3.8) enforces that $\mathcal{P}_i$ will be at most an n-dimensional simplex ($\mathbf{1}^T M_{(\cdot,i)} = n + 1$) and a lower dimensional simplex otherwise ($\mathbf{1}^T M_{(\cdot,i)} < n + 1$).

Example 3.1 *Consider* $y = \sin(x)$ *for* $x \in [-4, 4]$. *An SOS approximation with 11 evenly spaced breakpoints is given by vertex matrix*

$$V = \begin{bmatrix} -4 & -3.2 & -2.8 & \dots & 4 \\ \sin(-4) & \sin(-3.2) & \sin(-2.8) & \dots & \sin(4) \end{bmatrix}$$

and incidence matrix

$$M = \begin{bmatrix} \boldsymbol{I}_{10} \\ \mathbf{0}_{1\times 10} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{1\times 10} \\ \boldsymbol{I}_{10} \end{bmatrix}\,. \tag{3.9}$$

The first column of the incidence matrix

$$M_{(1,:)} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \end{bmatrix}^T$$

corresponds to one section of the SOS approximation for the domain $x \in [-4,\ -3.2]$, *which is a 1-dimensional simplex. The SOS approximation is plotted in Figure 3.4. Dotted vertical lines correspond to breakpoint locations and the domains between breakpoints correspond to the 10 columns of the incidence matrix* M (3.9).

Because SOS approximations are the union of a finite number of polytopes, they can be exactly represented as a hybrid zonotope. Furthermore, because SOS approximations

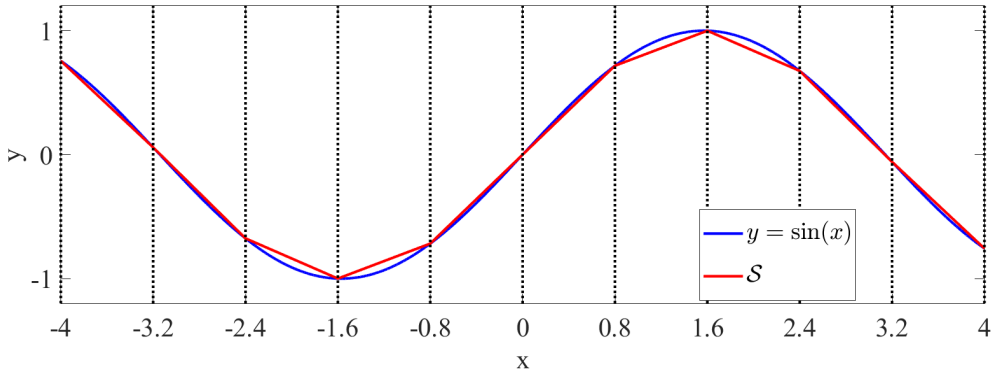


Figure 3.4: SOS approximation of $y = \sin(x)$ for $x \in \left[-4\,,\ \ 4\right]$ with 11 evenly spaced breakpoints.
Dotted vertical lines correspond to breakpoint locations and the domains between breakpoints correspond to the 10 columns of the incidence matrix M (3.9) and 10 1-dimensional simplices.

are defined by their breakpoints, they are conveniently written as a collection of polytopes in vertex representation, which can be efficiently converted to a hybrid zonotope using Theorem 2.1 with resulting memory complexity

$$n_g = 2n_v\,, \tag{3.10}$$
$$n_b = N\,, \tag{3.11}$$
$$n_c = n_v + 2\,. \tag{3.12}$$

3.2.2 Hybrid Zonotope Over-approximations for Graphs of Nonlinear Functions

By accounting for the worst-case error of the approximated graphs of functions, generated as SOS approximations or otherwise, hybrid zonotopes that *contain* graphs of nonlinear functions can be generated. The following result does not depend on the use of HCG-rep, though hybrid zonotopes have efficient identities for linear mapping and Minkowski summation used in (3.14).

Corollary 3.4 (Construct Envelope Given Error Bounds) *Given an approximation $\hat{h}(x)$ of a scalar-valued function $h(x) : \mathbb{R}^n \to \mathbb{R}$, its corresponding graph of a function $\hat{\mathcal{H}}$ defined over the domain $\mathrm{D}_{\hat{\mathcal{H}}}$, and an error bound given by the interval $[a, b]$ such that $h(x) \in [\hat{h}(x) + a,\ \hat{h}(x) + b]$ for all $x \in \mathrm{D}_{\hat{\mathcal{H}}}$, an over-approximation of the graph of $h(x)$*

over the domain $\mathrm{D}_{\hat{\mathcal{H}}}$

$$\mathcal{H} = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \;\middle|\; \begin{array}{c} y = h(x) \\ x \in \mathrm{D}_{\hat{\mathcal{H}}} \end{array} \right\} \tag{3.13}$$

is given by

$$\mathcal{H} \subseteq \hat{\mathcal{H}} \oplus \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \begin{bmatrix} a, & b \end{bmatrix} . \tag{3.14}$$

Proof *Defining the right side of* (3.14) *as* $\bar{\mathcal{H}}$*,*

$$\bar{\mathcal{H}} = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \;\middle|\; \begin{bmatrix} x \in \mathrm{D}_{\hat{\mathcal{H}}}, \\ y \in [\hat{h}(x) + a, \;\; \hat{h}(x) + b] \end{bmatrix} \right\}.$$

By assumption, $h(x) \in [\hat{h}(x) + a, \;\; \hat{h}(x) + b]$ *for all* $x \in \mathrm{D}_{\hat{\mathcal{H}}}$*.* □

3.2.2.1 Bounding SOS Approximation Error

Corollary 3.4 assumes that bounds $\begin{bmatrix} a, & b \end{bmatrix}$ on the approximation error are known. Generally, for a function $h(x)$ approximated over a domain by a PWA approximation $\bar{h}(x)$, error bounds can be posed as the mixed integer nonlinear programs

$$a = \min_{x \in \mathcal{X}} \bar{h}(x) - h(x) \tag{3.15}$$

$$b = \max_{x \in \mathcal{X}} \bar{h}(x) - h(x) \tag{3.16}$$

where $\mathcal{X}$ is the domain of the approximation. In general these programs are challenging to solve, though by decomposing functions into unary or binary functions, as will be proposed in Section 3.3, (3.15) and (3.16) only need to be solved for $x \in \mathbb{R}^1$ or $\mathbb{R}^2$. Additional properties such as whether $h(x)$ is continuous, differentiable, convex, or concave can significantly reduce the complexity of solving (3.15) and (3.16). Error bounds for affine approximations of some nonlinear functions can be found in [74, Chapter 3], e.g., for x^2 a closed-form solution to (3.15) is found by leveraging the convex and differentiable properties. In the case of PWA approximations, the process can be repeated for each partition of the domain and the worst-case error bounds should be used.

Example 3.2 demonstrates the construction of a hybrid zonotope over-approximation of the graph of a nonlinear function.

Example 3.2 *Consider the SOS approximation given in Example 3.1 of* $y = \sin(x), \quad x \in \begin{bmatrix} -4\,, & 4 \end{bmatrix}$. *Using error bounds provided in [73, Proposition 3.2] over each piece of the PWA approximation, it can be shown that* $\mathcal{E} = \begin{bmatrix} a, & b \end{bmatrix} = \begin{bmatrix} -0.0736, & 0.0736 \end{bmatrix}$ *provides adequate error bound, i.e.,*

$$h(x) \in \begin{bmatrix} \hat{h}(x) + a, & \hat{h}(x) + b \end{bmatrix}, \tag{3.17}$$

where $\hat{h}(x)$ *is the PWA approximation defined by linear interpolation between breakpoints. Thus, a hybrid zonotope enclosure of the graph of* $\sin(x)$ *for over the domain* $x \in \begin{bmatrix} -4, & 4 \end{bmatrix}$ *is created by representing* $\mathcal{S}$ *as a hybrid zonotope and applying Corollary 3.4. The resulting set is plotted in Figure 3.5*

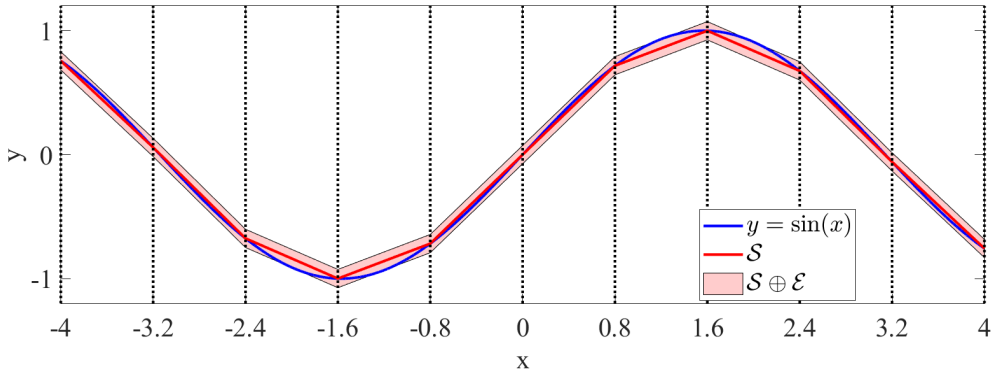


Figure 3.5: Hybrid zonotope over-approximation of the graph of the function $y = \sin(x)$. Dotted vertical lines correspond to breakpoint locations and the domains between breakpoints correspond to the 10 columns of the incidence matrix M (3.9). A hybrid zonotope over-approximation is generated by representing $\mathcal{S}$ using Theorem 2.1 and applying Corollary 3.4.

3.2.3 Unary SOS Construction

Due to the sampling of breakpoints in the input space, SOS approximations require additional tools to avoid exponential scaling with the number of inputs. This is addressed in Section 3.3 using functional decomposition methods, which allow for the composition of unary and binary functions to represent higher-dimensional functions. Specific results for "separable" functions allow for representing common binary functions, e.g., xy, $\frac{x}{y}$ and x^y, by affine combinations of unary functions. The remainder of this section provides methods to construct SOS approximations of continuous unary nonlinear functions.

A trivial and often inefficient placement of breakpoints is to uniformly sample the state space given a fixed number of breakpoints. Many methods exist to generate PWA approximation for different classes of functions. The methods in [75–79] all solve nonlinear optimization programs to determine coefficients to minimize various forms of error, e.g., least-squares or maximum bounds. Alternatively, the methods presented here generate SOS approximations with a tolerance and interval domain set by the user, and the algorithms use as many breakpoints as are required.

When constructing SOS approximations, the resulting memory complexity, defined by the number of breakpoints, and the computational cost of constructing the SOS approximation must be considered. Two approaches are considered including 1) a technique that produces minimal resulting complexity but is computationally expensive to generate, and 2) a technique that is computationally efficient at the expense of added memory complexity. The proposed methods allow for the construction of hybrid zonotopes that contain a nonlinear graph of a unary function and limit the memory complexity of the resulting sets, though these techniques have uses outside the scope of hybrid zonotopes.

3.2.3.1 Method 1 (Bisection Advance):

The first technique is given by Algorithm 1 and summarized as follows. **Lines 1-2:** Initialize the index k, first breakpoint x_1, and assign ε to the maximum error associated with a domain spanning $[x_1, \bar{x}]$. **Line 3:** Check if $\bar{x}$ is the last breakpoint. **Line 4:** Initialize upper, lower, and cut values for a bisection method. **Line 5:** Check if the bisection method has converged. **Lines 6-15:** Evaluate the maximum error with an upper bound associated with the region spanning $[x_k, m]$ and reassign bisection upper, lower, and cut values appropriately. **Lines 16-18:** Assign next breakpoint x_{k+1} equal to the lower bound, ensuring tolerance satisfaction, update the error for the domain spanning $[x_{k+1}, \bar{x}]$, and increment k. **Line 20:** Assign x_{k+1} as the final breakpoint. A visual depiction of the algorithm is shown in Figure 3.6.

The evaluation of the maximum error using the function $\texttt{evalErr}(f(\cdot), [x_1, m])$ is defined in general by

$$\texttt{evalErr}(f(\cdot), [x_k, m]) = \max_{x \in [x_k, m]} |\bar{f}(x) - f(x)| \,, \tag{3.18}$$

where $\bar{f}(x)$ is given by the SOS approximation on the domain $[x_k, m]$ as

$$\bar{f}(x) = f(x_k) + \frac{f(m) - f(x_k)}{m - x_k}(x - x_k)\,. \tag{3.19}$$

In general, the evaluation of (3.18) requires solving a nonlinear program. Additional properties such as whether $f(x)$ is continuous, differentiable, convex, or concave can significantly reduce the complexity of solving (3.18).

Algorithm 1: Generate SOS approximation ($\mathcal{C}^0$). Given: continuous unary scalar-valued function $f(\cdot)$, output error tolerance τ, breakpoint tolerance δ_x, input domain $x \in [\underline{x},\ \bar{x}]$.

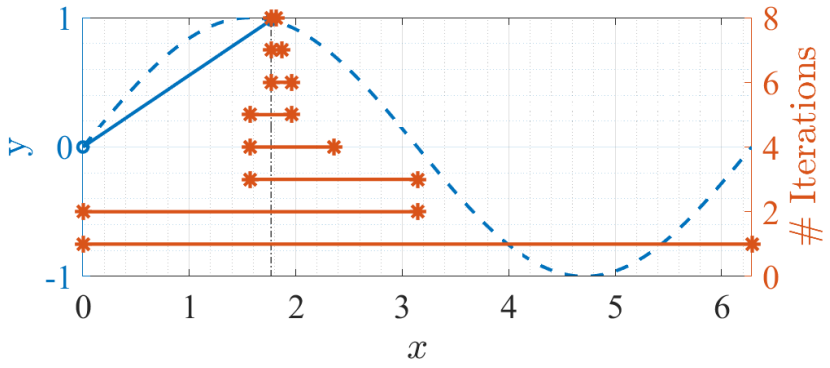
Result: Vector of breakpoints x

$k \leftarrow 1$, $x_1 \leftarrow \underline{x}$
$e \leftarrow \texttt{evalErr}(f(\cdot), [x_1, \bar{x}])$
while $e > \tau$ **do**
  $\ell \leftarrow x_k$, $u \leftarrow \bar{x}$, $m \leftarrow (\ell + u)/2$
  while $u - \ell > \delta_x$ *OR* $e > \tau$ **do**
    $e \leftarrow \texttt{evalErr}(f(\cdot), [x_k, m])$
    if $e < \tau$ **then**
      $\ell \leftarrow m$
    else if $e > \tau$ **then**
      $u \leftarrow m$
    else
      $\ell \leftarrow m$, $u \leftarrow m$
    end
    $m \leftarrow (\ell + u)/2$
  end
  $x_{k+1} \leftarrow \ell$
  $e \leftarrow \texttt{evalErr}(f(\cdot), [x_{k+1}, x_u])$
  $k \leftarrow k + 1$
end
$x_{k+1} \leftarrow \bar{x}$

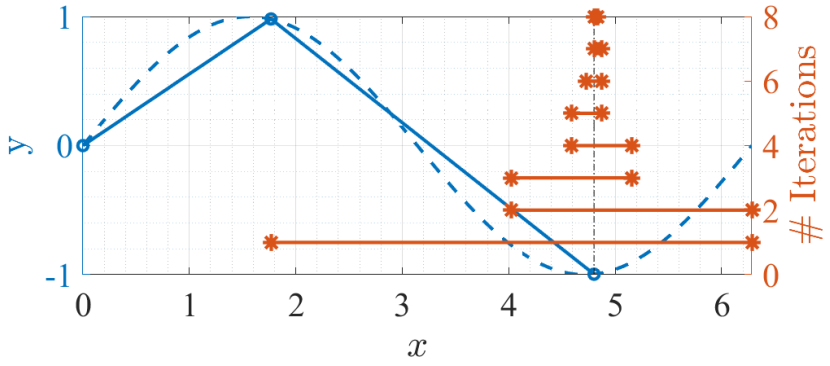
3.2.3.2 Method 2 (Thrice Differentiable Functions):

Algorithm 2 assumes $f(\cdot)$ is a thrice differentiable unary scalar-valued function, and a bound on the magnitude of the third derivative over the domain is given by

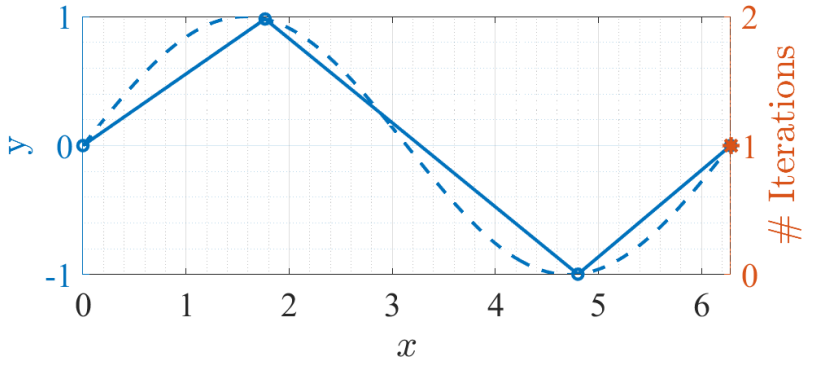
$$d_3 \geq \max_{x \in [\underline{x}, \bar{x}]} |f'''(x)|\,. \tag{3.20}$$



(a)



(b)



(c)

Figure 3.6: Bisection-based construction of SOS approximation.
Method 1 (Algorithm 1). (a) Bisection to determine x_2. (b) Bisection to determine x_3. (c) The upper bound $\bar{x} = 2\pi$ satisfies the error tolerance therefore the bisection method to find the next breakpoint is not required and Algorithm 1 terminates.

Algorithm 2 exploits these assumptions and leverages a *closed-form bound* on the error presented in Theorem 3.3.

Theorem 3.3 *Given a thrice differentiable unary scalar-valued function $f(\cdot)$ over a*

domain $x \in [\underline{x}, \bar{x}]$*, and a bound on the third derivative* d_3 *over the domain* (3.20)*, then*

$$\max_{x \in [\underline{x}, \bar{x}]} |\bar{f}(x) - f(x)| \leq \frac{d_3}{4}(\bar{x} - \underline{x})^3 + \frac{d_2}{4}(\bar{x} - \underline{x})^2 \,, \tag{3.21}$$

where $d_2 = |f''(\underline{x})|$*.*

Proof *The proof begins with the result given in [80, Theorem 4.1], modified to match notation herein and specified to* $f(\cdot) : \mathbb{R} \to \mathbb{R}$*,*

$$\max_{x \in [\underline{x}, \bar{x}]} |\bar{f}(x) - f(x)| \leq \frac{\max_{x \in [\underline{x}, \bar{x}]} |f''(x)|}{4}(\bar{x} - \underline{x})^2 \,. \tag{3.22}$$

By the fundamental theorem of calculus and triangle inequality

$$\begin{aligned} f''(\bar{x}) &= f''(\underline{x}) + \int_{\underline{x}}^{\bar{x}} f'''(t)\ dt \,, \\ |f''(\bar{x})| &= |f''(\underline{x}) + \int_{\underline{x}}^{\bar{x}} f'''(t)\ dt| \,, \\ &\leq |f''(\underline{x})| + \int_{\underline{x}}^{\bar{x}} |f'''(t)|\ dt \,, \\ &\leq |f''(\underline{x})| + \max_{t \in [\underline{x}, \bar{x}]} |f'''(t)|(\bar{x} - \underline{x}) \,. \end{aligned} \tag{3.23}$$

Replacing the maximum magnitude of the second derivative in (3.22) *with an upper bound given by* (3.23) *yields* (3.21)*.* □

Algorithm 2 is summarized as follows. **Line 1:** Initialize index k, first breakpoint x_1, and magnitude of the second derivative at the current breakpoint. **Line 2:** Assign $\bar{\varepsilon}$ to the error bound associated with a domain spanning $[x_1, \bar{x}]$. **Line 3:** Check if $\bar{x}$ is the last breakpoint. **Lines 4-5:** If $\bar{x}$ is not the last breakpoint, assign the next breakpoint x_{k+1} to the largest real solution of

$$\frac{d_3}{4}(x_{k+1} - x_k)^3 + \frac{d_2}{4}(x_{k+1} - x_k)^2 = \tau \,, \tag{3.24}$$

which falls in the domain $[\underline{x}, \bar{x}]$. Note that the function `roots`$(c(x))$ simply returns the three roots s_1, s_2, and s_3 of a cubic polynomial $c(x)$. **Lines 6-9:** Update the error bound $\bar{\varepsilon}$ and d_2 associated with the last breakpoint x_{k+1} and increment k. **Line 10:** Assign $\bar{x}$ as the final breakpoint.

Remark 3.1 *The variation in breakpoint separation is obtained by updating* d_2*, e.g., when* d_2 *is small, the next* x_{k+1} *can be further away.*

Algorithm 2: Generate SOS approximation (Thrice Differentiable). Given: thrice differentiable unary scalar-valued function and its second derivative $f(\cdot)$, $f''(\cdot)$, output error tolerance τ, input domain $x \in [\underline{x},\ \bar{x}]$, and a bound on the magnitude of the third derivative $d_3 \geq \max_{x\in[\underline{x},\bar{x})] } |f'''(x)|$

Result: Vector of breakpoints x

$k \leftarrow 1$, $x_1 \leftarrow \underline{x}$, $d_2 = |f''(x_1)|$

$e \leftarrow \frac{d_3}{4}(\bar{x} - x_1)^3 + \frac{d_2}{4}(\bar{x} - x_1)^2$

while $e > \tau$ **do**

  $\{s_1, s_2, s_3\} \leftarrow \texttt{roots}(\frac{d_3}{4}(x - x_k)^3 + \frac{d_2}{4}(x - x_k)^2 - \tau)$

  $x_{k+1} = \max(s)$ s.t. $s \in [\underline{x}, \bar{x}] \cap \{s_1, s_2, s_3\} \cap \mathbb{R}$

  $d_2 \leftarrow |f''(x_{k+1})|$

  $e \leftarrow \frac{d_3}{4}(\bar{x} - x_{k+1})^3 + \frac{d_2}{4}(\bar{x} - x_{k+1})^2$

  $k \leftarrow k + 1$

end

$x_{k+1} \leftarrow \bar{x}$

3.2.3.3 Comparison of SOS Construction Methods

Figure 3.7 compares construction of SOS approximations using Method 1 and Method 2 in computation time and the number of breakpoints for the functions and domains given in Table 3.1. Method 1 is slower to compute but produces SOS approximations with fewer breakpoints for all the functions tested. Because the techniques in this thesis will be directed at problems solved offline, lower resulting complexities using Method 1 are often worth the longer computation times.

Table 3.1: Functions and domains used to compare Method 1 and Method 2 for SOS construction.

Function	$\sin(x)$	x^2	x^3	$1/x$
Domain	$x \in [0, 2\pi]$	$x \in [-5, 5]$	$x \in [-5, 5]$	$x \in [1, 10]$

Because the number of breakpoints is discrete, as the tolerance is varied the number of breakpoints required to achieve the tolerance will "jump" at critical tolerance values. This is demonstrated in Figure 3.8 for SOS approximations of $\sin(x)$, $x \in [0, 2\pi]$ using Method 1 (Algorithm 1). Dotted lines are projected onto the tolerance axis to indicate critical tolerance values. For a given interval of tolerances, the number of breakpoints does not vary. Red points indicate the tightest tolerance achievable given a specified number of breakpoints. The results in Figure 3.8 were obtained by uniformly sampling the logarithm of the tolerance 500 times. Critical values of tolerances in red correspond to

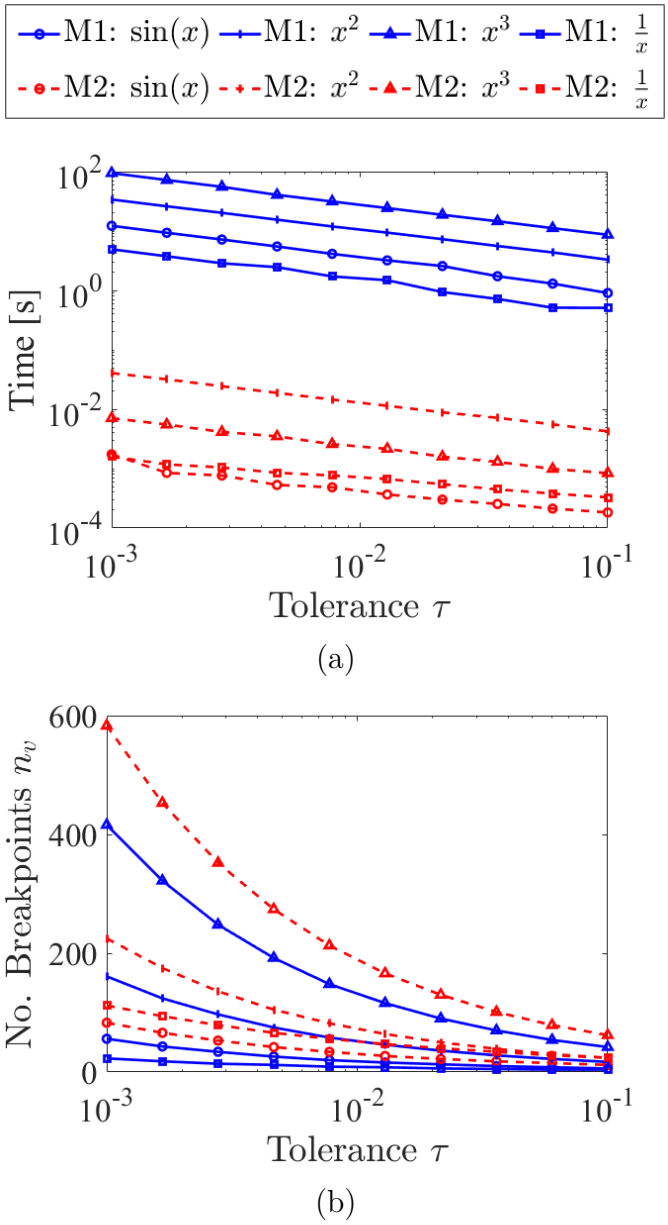


Figure 3.7: Comparison of SOS Construction Methods.
Method 1 (Algorithm 1) and Method 2 (Algorithm 2) for constructing SOS approximations for various functions and domains given in Table 3.1). (a) and (b) show the number of breakpoints and computation time, respectively, as a function of tolerance. Across all functions tested, Method 1 is significantly slower to compute than Method 2, but constructs an SOS approximation with a specified error tolerance using fewer breakpoints.

solutions of optimal placements of breakpoints given a specified number of breakpoints.

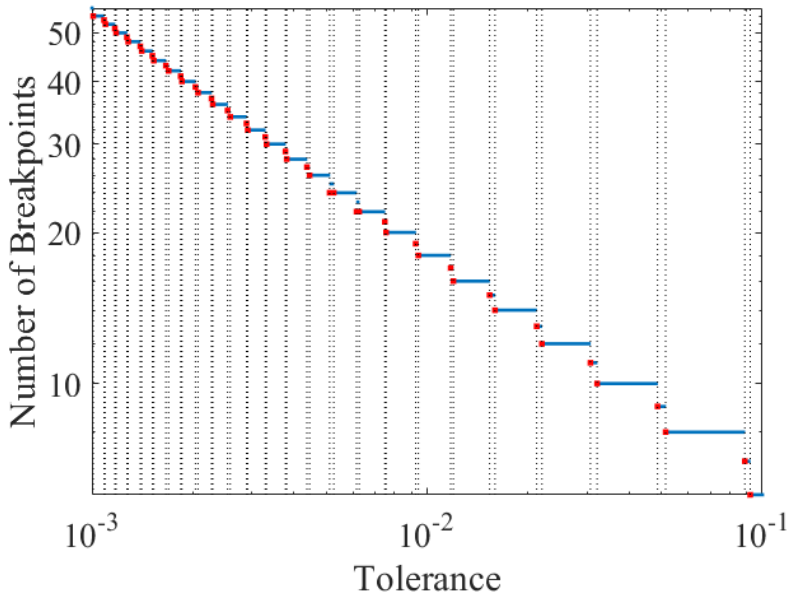


Figure 3.8: Number of Breakpoints required to meet a specified tolerance for the function $\sin(x)$ $x \in [0, 2\pi]$ using Method 1 (Algorithm 1).
Dashed lines indicate values of tolerances at which the number of breakpoints "jumps".

3.3 Functional Decomposition

3.3.1 Introduction and Definition

The Kolmogorov-Arnold Representation Theorem [81, 82] proves that every multivariate continuous function can be written as the composition of continuous functions of a single variable and addition. In [83], the authors represent function decompositions via a parsing tree and describe an algorithm to generate parsing trees. Additionally, [76, 83] define *separable* functions as those which can be expressed as the sum of functions of a single variable, and show how such functions are easily decomposed into unary functions. Work to approximate nonlinear optimization programs uses functional decompositions to avoid exponential scaling of SOS approximations [73, 74] .

The following form of a functional decomposition is modified from [73]. A general nonlinear function $h(x) : \mathbb{R}^n \to \mathbb{R}^m$ is decomposed by introducing intermediate

observables

$$w_j = \begin{cases} x_j \,, & \text{if } j \in \{1, ..., n\} \,, \\ h_j(w_{j1}\{, w_{j2}\}) \,, & \text{if } j \in \{n+1, ..., n+K\} \,, \end{cases} \tag{3.25}$$

where $j1, j2 < j$, giving

$$h(x) = \begin{bmatrix} w_{n+K-m+1} \\ \vdots \\ w_{n+K} \end{bmatrix} .$$

The first n assignments directly correspond to the n elements of the argument vector x, assignments $n+1, ..., n+K$ are defined by the unary function or binary function h_j, and the final m assignments are associated with $h(x)$. In the case that h_j is unary, the second argument is omitted.

Remark 3.2 *Because hybrid zonotopes are closed under linear mapping, functional compositions are also allowed to admit functions $h_j(\cdot)$ that have more than two inputs, provided the function $h_j(\cdot)$ is an affine function of lower-indexed variables, i.e., $w_k \,, \; \forall \; k \; < \; j$.*

Example 3.3 *Consider inverted pendulum dynamics given by*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{g}{l}\sin(x_1) + \frac{u}{I} \end{bmatrix} , \tag{3.26}$$

with gravity $g = 10$, length $l = 1$, mass $m = 1$, and moment of inertia $I = ml^2 = 1$. The continuous-time nonlinear dynamics are discretized with time step $h = 0.1$ using a 2^{nd}-order Taylor polynomial $\mathcal{T}_2(x_k)$ given by

$$\mathcal{T}_2(x_k) = \left\{ \begin{bmatrix} x_{1,k} + \frac{x_{2,k}}{10} + \frac{\sin(x_{1,k})}{20} + \frac{u_k}{200} \\ x_{2,k} + \sin(x_{1,k}) + \frac{x_{2,k}\cos(x_{1,k})}{20} + \frac{u_k}{10} \end{bmatrix} \right\} . \tag{3.27}$$

A functional decomposition of $\mathcal{T}_2(x_k)$ is shown in Table 3.2. For a chosen $D_{\mathcal{H}} = D_1 \times D_2 \times D_3$, bounds on the intermediate and output variables can be found by domain propagation via interval arithmetic.

Table 3.2: Functional decomposition of $\mathcal{T}_2$ (3.27)

w_j	h_ℓ	D_ℓ
$w_1 = x_{1,k}$		$[-4, 4]$
$w_2 = x_{2,k}$		$[-8, 8]$
$w_3 = u_k$		$[-20, 20]$
w_4	$\sin(w_1)$	$[-1, 1]$
w_5	$\cos(w_1)$	$[-1, 1]$
w_6	$w_5 w_2$	$[-8, 8]$
w_7	$w_1 + \frac{w_2}{10} + \frac{w_3}{200} + \frac{w_4}{20}$	$[-4.95, 4.95]$
w_8	$w_2 + w_4 + \frac{w_3}{10} + \frac{w_6}{20}$	$[-11.4, 11.4]$

3.3.2 Construction of High-Dimensional Graphs of Functions

The Kolmogorov-Arnold Representation Theorem proves the existence of a functional decomposition for continuous multivariate functions. This allows for a theoretical scaling of the approaches herein in spite of the fact that functional decompositions are system specific and are not unique. Unfortunately, decompositions based on the Kolmogorov-Arnold Representation Theorem are not always immediately obvious. This section first provides identities to construct high-dimensional graphs of functions assuming a functional decomposition is given in addition to graphs of functions for each nonlinear unary or binary function within the decomposition. Then the computational and memory complexity of the identities is analyzed assuming decomposition based on the Kolmogorov-Arnold Representation Theorem.

While Theorem 3.4 addresses nonlinear functions with a single vector argument x, it is easily applied to functions with multiple arguments by concatenating arguments into a single vector.

Theorem 3.4 (Construct Graph of a Functional Decomposition) [62, Theorem 4] *Consider a general nonlinear function $h(x) : \mathbb{R}^n \to \mathbb{R}^m$ and its decomposition* (3.25)*. Define the set $\mathcal{H}$ as*

$$\mathcal{H} \equiv \left\{ \begin{bmatrix} w_1 \\ \vdots \\ w_{n+K} \end{bmatrix} \;\middle|\; \begin{array}{c} (w_1, w_2, \ldots, w_n) \in \mathrm{D}_{\mathcal{H}}, \\ w_j = h_j(w_{j1}\{, w_{j2}\}) \,, \\ \forall j \in \{n+1, ..., n+K\} \end{array} \right\} . \tag{3.28}$$

Given $\mathrm{D}_j \supseteq [e_j]\mathcal{H}\ ,\ \forall j = n+1, ..., n+K$,[3] *and*

$$\mathcal{H}_\ell = \left\{ \begin{bmatrix} w_{\ell 1} \\ \{w_{\ell 2}\} \\ w_\ell \end{bmatrix} \;\middle|\; \begin{array}{c} (w_{\ell 1}\{, w_{\ell 2}\}) \in \mathrm{D}_{\ell 1}\{\times \mathrm{D}_{\ell 2}\}, \\ w_\ell = h_\ell(w_{\ell 1}\{, w_{\ell 2}\}) \end{array} \right\}, \tag{3.29}$$

for $\ell \in \{n+1, ..., n+K\}$, *then* $\mathcal{H}$ *is given by initializing* $\mathcal{H}_{1:n} = \mathrm{D}_\mathcal{H}$, *iterating* K *times*

$$\mathcal{H}_{1:\ell} = (\mathcal{H}_{1:\ell-1} \times \mathrm{D}_\ell) \cap \begin{bmatrix} e_{\ell 1} \\ \{e_{\ell 2}\} \\ e_\ell \end{bmatrix} \mathcal{H}_\ell\ . \tag{3.30}$$

After K *iterations, the result is given by* $\mathcal{H} = \mathcal{H}_{1:n+K}$.

Proof *From* $\mathcal{H}_{1:n} = \mathrm{D}_\mathcal{H}$ *and* (3.30),

$$\mathcal{H}_{1:n+K} = \left\{ \begin{bmatrix} w_1 \\ \vdots \\ w_{n+K} \end{bmatrix} \;\middle|\; \begin{array}{c} (w_1, w_2, ..., w_n) \in \mathrm{D}_\mathcal{H}, \\ w_j \in \mathrm{D}_j\ ,\ \forall j \in \{n+1, ..., n+K\}, \\ w_j = h_j(w_{j1}\{, w_{j2}\})\ ,\ \forall j \in \{n+1, ..., n+K\} \end{array} \right\}. \tag{3.31}$$

The constraint $w_{n+1} \in \mathrm{D}_{n+1}$ *is redundant given that* $(w_1, w_2, ..., w_n) \in \mathrm{D}_\mathcal{H}$ *and* $w_j = h_j(w_{j1}\{, w_{j2}\})$ *for* $j = n+1$, *and therefore can be removed. The same is then true* $\forall j \in \{n+2, ..., n+K\}$, *yielding* $\mathcal{H}$ *as defined by* (3.28). □

Corollary 3.5 addresses the special case when the decomposition includes affine functions.

Corollary 3.5 (Efficient Construction of Affine Observables) [62, Corollary 1] *Consider a general nonlinear function* $h(x) : \mathbb{R}^n \to \mathbb{R}^m$, *its decomposition* (3.25), *the set* $\mathcal{H}$ *defined by* (3.28) *and the recursion* (3.30). *For all* ℓ *where* $h_\ell(\cdot)$ *is affine, i.e.,* $h_\ell(w_{\ell 1}\{, w_{\ell 2}\}) = m_{\ell 1} w_{\ell 1}\{+m_{\ell 2}\ w_{\ell 2}\} + b_\ell$ *where* $m_{\ell 1}$, $\{m_{\ell 2}\}$, b_ℓ *are scalars, the set* $\mathcal{H}$ *is equivalently given when* (3.30) *is replaced by*

$$\mathcal{H}_{1:\ell} = \begin{bmatrix} \boldsymbol{I}_{l-1} \\ m_{\ell 1} e_{\ell 1}\{+m_{\ell 2} e_{\ell 2}\} \end{bmatrix} \mathcal{H}_{1:\ell-1} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} b_\ell\ . \tag{3.32}$$

[3] Domains $\mathrm{D}_j \supseteq [e_j]\mathcal{H}\ ,\ \forall j = n+1, ..., n+K$ can be found as intervals using domain propagation.

Proof *The proof only requires recognizing that* $m_{\ell 1} w_{\ell 1}\{+m_{\ell 2}\ w_{\ell 2}\} + b_\ell = h_\ell(w_{\ell 1}\{, w_{\ell 2}\})$ *when* $h_\ell(\cdot)$ *is affine to arrive at* (3.31) *and then follows the same procedure as the proof of Theorem 3.4.* □

Although Corollary 3.5 is written for unary or binary affine functions, it is easily extended to functions with an arbitrary number of arguments, i.e., $h_\ell(w_1, ...) = b_\ell + \sum_i m_{\ell i} w_{\ell i}$. Applying Corollary 3.5 can reduce the memory complexity of $\mathcal{H}$ (3.28) when implemented with hybrid zonotopes. This is because the affine transformation in (3.32) does not increase the memory complexity of the hybrid zonotope, whereas the generalized intersection in (3.30) does. This is the reasoning for allowing affine functions in functional decompositions to have more than two inputs (see Remark 3.2).

Often, the intermediate observables created in the process of functional decomposition do not need to be retained as dimensions of the final set. Therefore, Corollary 3.6 provides an identity to remove intermediate observables, only retaining dimensions corresponding to the n arguments and m outputs of $h(x)$.

Corollary 3.6 (Remove Intermediate Observables in Graph of a Functional Decomposition) [62, Corollary 2] *Consider a general nonlinear function* $h(x) : \mathbb{R}^n \to \mathbb{R}^m$ *and its decomposition* (3.25)*. Given the set* $\mathcal{H}$ *as defined by* (3.28)*, the graph of the function* $h(x)$ *defined over input domain* $\mathrm{D}_\mathcal{H}$ *is given by*

$$\Phi = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \;\middle|\; \begin{array}{c} x \in \mathrm{D}_\mathcal{H} \\ y = h(x) \end{array} \right\} = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_m \end{bmatrix} \mathcal{H} \,. \tag{3.33}$$

Proof *By definition of the linear transformation, the right side of* (3.33) *yields*

$$\left\{ \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+K-m+1} \\ \vdots \\ w_{n+K} \end{bmatrix} \;\middle|\; \begin{array}{c} (w_1, w_2, \ldots, w_n) \in \mathrm{D}_\mathcal{H}, \\ w_j = h_j(w_{j1}\{, w_{j2}\}) \,, \\ \forall j \in \{n+1, ..., n+K\} \end{array} \right\} .$$

Substitution of $h_j(\cdot)$ *,* $\forall\ j \in \{n+1, ..., n+K\}$ *yields the desired result.* □

Corollary 3.7 (Constructions using Over-Approximated Sets) *For the identities* (3.30)*,* (3.32)*, and* (3.33) *provided by Theorem 3.4, Corollary 3.5, and Corollary 3.6*

respectively, if argument sets are replaced with an over-approximations, then the identities will instead yield over-approximations of the left sides.

Proof *Set containment is preserved under linear transformation and generalized intersection.* □

Example 3.4 *A functional decomposition of* $x_{k+1} = \cos(\pi \sin(x_k))$ *over a domain* $\mathrm{D}_{\mathcal{H}} = [-\pi, \pi]$ *and its visual representation are given by Table 3.3 and Figure 3.9, respectively. Over-approximations* $\bar{\mathcal{H}}_2$ *and* $\bar{\mathcal{H}}_3$ *are constructed using SOS approximations and closed-form solution error bounds from [74], which are combined using Corollary 3.4. The effect of over-approximation per Corollary 3.7 is demonstrated in Figure 3.9(e)-(h). In Figure 3.9(d) and (h), Corollary 3.6 is used as the last step in construction of the state-update set* Φ *and its over-approximation* $\bar{\Phi}$*, which are shown in magenta.*

Table 3.3: Functional decomposition and domain propagation of $x_{k+1} = \cos(\pi \sin(x_k))$ over a domain $\mathrm{D}_{\mathcal{H}} = [-\pi,\ \pi]$.

w_j	$\mathcal{H}_j$	D_j
$w_1 = x_k$		$[-\pi,\ \pi]$
w_2	$\pi \sin(w_1)$	$[-\pi,\ \pi]$
$w_3 = x_{k+1}$	$\cos(w_2)$	$[-1,\ 1]$

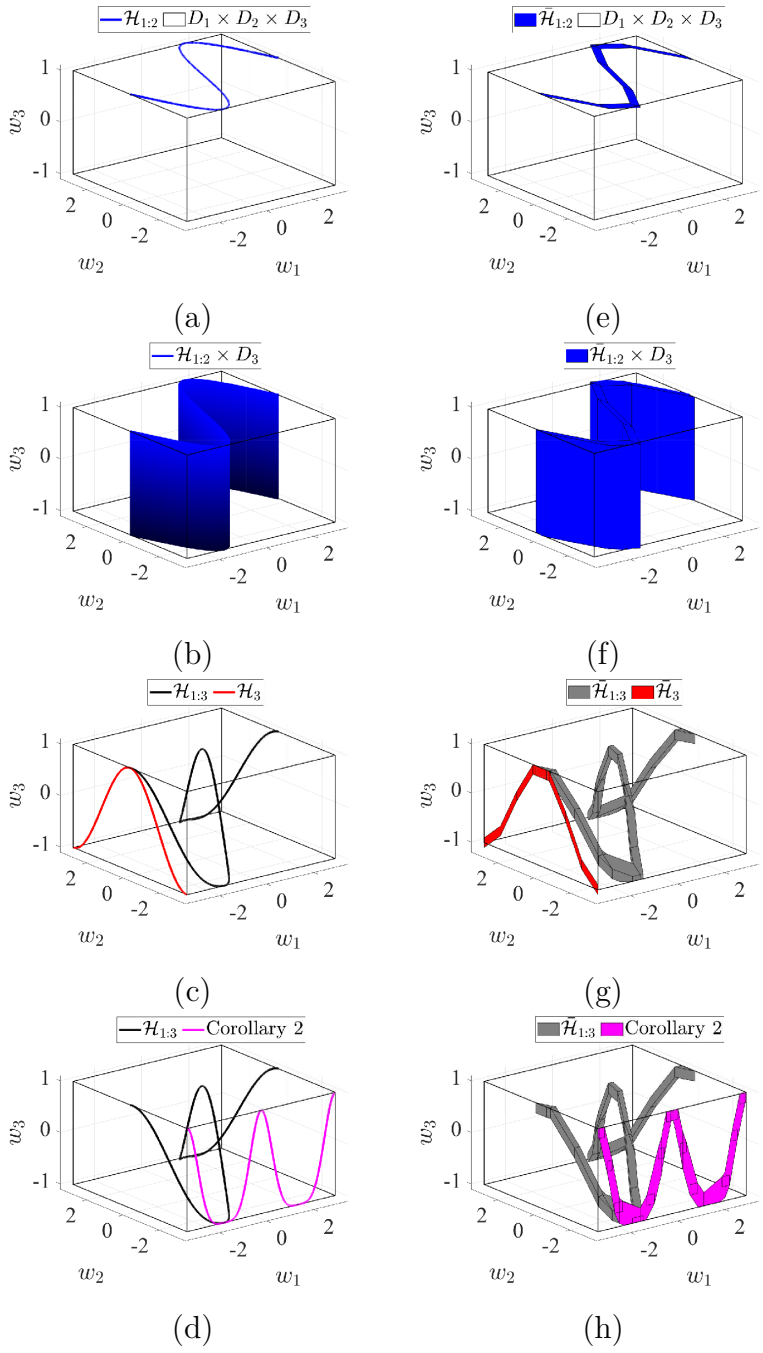


Figure 3.9: Visual depiction of functional decomposition and Theorem 3.4 applied to $x_{k+1} = \cos(\pi \sin(x_k))$ with the decomposition shown in Table 3.3.
Subfigures (a-d) show the exact case while subfigures (e-h) show the effect of using over-approximations $\mathcal{H}_2 \subset \bar{\mathcal{H}}_2$, and $\mathcal{H}_3 \subset \bar{\mathcal{H}}_3$.

3.3.2.1 Avoiding the Curse of Dimensionality

This section provides a theoretical bound on the memory complexity of the graph of the function given by Theorem 3.4 and Corollary 3.5 to quantify the scalability of the proposed methods with respect to the state dimension and number of vertices used to approximate nonlinear functions.

Consider a general nonlinear function $h(x) : \mathbb{R}^n \to \mathbb{R}^m$. It is assumed that the functional decomposition consists of n intermediate observables corresponding to the dimensions of x, K_{aff} intermediate observables corresponding to multivariate affine functions, and K_{NL} intermediate observables corresponding to nonlinear functions that are unary or binary. The set of indices such that $h_\ell(\cdot)$ is nonlinear is denoted $\mathcal{N}$. Assuming $D_{\mathcal{H}}$ and D_ℓ are intervals, the memory complexity of $\mathcal{H}$ as constructed by Theorem 3.4 and Corollary 3.5 in terms of complexities of H_ℓ , $\forall \ell$ such that $h_\ell(\cdot)$ is nonlinear, is given by

$$n_{g,\mathcal{H}} = n + K_{NL} + \sum_{\ell \in \mathcal{N}} n_{g,\mathcal{H}_\ell} \,, \tag{3.34}$$

$$n_{b,\mathcal{H}} = \sum_{\ell \in \mathcal{N}} n_{b,\mathcal{H}_\ell} \,, \tag{3.35}$$

$$n_{c,\mathcal{H}} = \sum_{\ell \in \mathcal{N}} \left(n_{c,\mathcal{H}_\ell} + n_\ell \right), \quad n_\ell = \begin{cases} 2 & \text{if } h_\ell(\cdot) \text{ is unary} \,, \\ 3 & \text{if } h_\ell(\cdot) \text{ is binary} \,. \end{cases} \tag{3.36}$$

The computational complexity of constructing $\mathcal{H}$ using Theorem 3.4 and Corollary 3.5 is dominated by the K_{NL} generalized intersections in the recursions of (3.30). Each generalized intersection has computational complexity $\mathcal{O}(\ell(n_g + n_b))$, where n_g and n_b grow with the complexity of $\mathcal{H}_\ell$.

For a given system, the functional decomposition plays a critical role in both the computational and memory complexity of building a graph of the function. This presents a challenge to quantifying the scalability of the proposed approaches with the state dimension as it can be expected that K_{NL} will grow with n. This challenge is exacerbated when considering that functional decompositions are dependent on the system and are not unique, i.e., multiple functional decompositions exist for a single system.

In order to provide a theoretical bound for the proposed method, let us assume that a decomposition based on the Kolmogorov-Arnold Representation Theorem [81, 82] is performed for each $h_i(x)$, $\forall\, i \in \{1, ..., m\}$. Then the number of unary nonlinear decom-

position functions is given by $K_{NL} = 2mn^2 + mn$ and no binary nonlinear decomposition functions are required. Assuming that each unary nonlinear decomposition function is approximated using an SOS approximation with n_v vertices, the resulting complexity of $\mathcal{H}$ is given by

$$\begin{aligned} n_{g,\mathcal{H}} &= n + (2mn^2 + mn)(2n_v + 1) \,, \\ n_{b,\mathcal{H}} &= (2mn^2 + mn)(n_v - 1) \,, \\ n_{c,\mathcal{H}} &= 8mn^2 + 4mn \,. \end{aligned} \tag{3.37}$$

It is clear from (3.37) that the memory complexity of $\mathcal{H}$ scales as a polynomial with n, avoiding the exponential growth associated with the curse of dimensionality. In comparison, any method that sampled the n-dimensional space to generate an approximation would scale as n_v^n, not including any additional complexity incurred to generate $\mathcal{H}$ from those samples.

3.3.3 Automated Functional Decomposition

This section presents methods to automate production of a functional decomposition in the form of (3.25) from an expression in infix notation, which represents mathematical operators placed between operands. The process uses an existing algorithm to convert from infix notation to another standard notation, and then presents a novel algorithm to convert the result to a functional decomposition. The proposed approach addresses two challenges when constructing functional decompositions, namely removing redundant observables[4] and avoiding excessive decomposition of unary functions.

3.3.3.1 Reverse Polish Notation

Infix notation is commonly used to represent mathematical expressions with operators placed between operands, e.g., $x + y * z$. Reverse Polish Notation (RPN), also referred to as reverse Likasiewicz notation and postfix notation, represents a mathematical expression with operands preceding their operators. The Shunting Yard algorithm (SYA) [84] uses precedence and associativity to parse infix notation and outputs an equivalent expression

[4]Redudant observables are referred to as *multiple presence* in [74].

in RPN, e.g.,

$$\begin{array}{ccc} \underline{\text{Infix Notation}} & \underline{\text{SYA}} & \underline{\text{RPN}} \\ x + y \times z & \longrightarrow & x\ y\ z\ \times \end{array} \tag{3.38}$$

The SYA can similarly handle functions, e.g.,

$$\begin{array}{ccc} \underline{\text{Infix Notation}} & \underline{\text{SYA}} & \underline{\text{RPN}} \\ 3 \times y \times \cos(x)^2 & \longrightarrow & 3\ y\ x\ \cos\ 2\ \wedge\ \times\ \times \end{array} \tag{3.39}$$

Each number, variable, operator, and function is referred to as a *token*. The RPN expressions in (3.38) and (3.39) have 5 and 8 tokens, respectively. To read RPN, start at the left of the expression and *push* tokens onto a stack until a operator or function token is reached. Then *pop* the appropriate number of tokens off of the stack, perform the operation to produce a single token, and *push* the resulting token to the stack. The process is repeated until there are no remaining tokens in the expression and the result is on the stack.

3.3.3.2 RPN to Functional Decomposition (Scalar-valued Function)

A string of tokens is denoted by $\mathcal{E}$ and $\mathcal{E}_1$ denotes the first token of $\mathcal{E}$. As tokens are pushed or popped to a stack, the value of $\mathcal{E}_1$ changes. For ease of notation and without loss of generality, it is assumed that the input variables of the expression of interest are $w_1, ...w_{n_x}$ where n_x is the number of input variables.

Algorithm 3: RPN to Functional Decomposition. Given: a string of n tokens $\mathcal{E}$ in RPN.

Result: Function Decomposition

```
k ← n_x + 1
while E is not empty do
    while E_1 is a number or a variable do
        push E_1 to stack.
    end
    pop tokens from stack associated with the operator or function E_1, form
    observable expression and save as w_k.
    k ← k + 1
end
```

Algorithm 3 provides and automated method to convert a string of tokens in RPN into a functional decomposition of the form (3.25) by leveraging the order of operations

and evaluations of sub-expressions to generate observables. Algorithm 3 has two apparent flaws when used with piecewise affine and piecewise polytope approximations, which are described as a) redundant observables, and b) excessive decomposition of unary functions.

3.3.3.2.1 Redundant Observables

First, consider the expression

$$\sin(w_1) + \sin(w_1)^2 \,. \tag{3.40}$$

The SYA yields the RPN expression

$$w_1 \;\; \sin \;\; w_1 \;\; \sin \;\; 2 \;\; \wedge \;\; + \tag{3.41}$$

and Algorithm 3 gives the functional decomposition

$$\begin{aligned} &\text{Input: } w_1 \\ &\quad w_3 = \sin(w_1) \,, \\ &\quad w_4 = \sin(w_1) \,, \\ &\quad w_5 = w_4^2 \,, \\ &\quad w_6 = w_4 + w_5 \,. \end{aligned} \tag{3.42}$$

Note that w_3 and w_4 are redundant as they represent the same quantity. When considering use of the functional decomposition with piecewise affine or piecewise polytopic approximations of nonlinear functions, this redundancy should be avoided because 1) unnecessary complexity will be accrued in approximating the quantity $\sin(w_1)$ twice instead of once, and 2) for piecewise polytopic approximations, w_3 and w_4 will be uncoupled, i.e., values within over-approximations of w_3 and w_4 are not required to be equal to each other given the same w_1. This motivates a modification of Algorithm 3 to avoid constructing a functional decomposition containing redundant observables.

Algorithm 4 builds on Algorithm 3 and addresses redundant observables by introducing a check to see if an equivalent observable already exists. New observables are only created if an equivalent observable does not already exist. Applying Algorithm 4 to the RPN expression given in (3.41) results in the more compact functional decomposition

$$\begin{aligned} &\text{Input: } w_1 \\ &\quad w_3 = \sin(w_1) \,, \end{aligned} \tag{3.43}$$

$$w_4 = w_3^2 \,,$$
$$w_5 = w_3 + w_4 \,.$$

Figure 3.10 and Table 3.4 demonstrate the accuracy and memory complexity benefits of accounting for redundant observables (Algorithm 3 vs. Algorithm 4). In this case, hybrid zonotope over-approximations of the nonlinear sinusoidal and quadratic observables are generated using the following steps.

1. Algorithm 1 generates breakpoints for an SOS approximations of $\sin(x)$ and x^2 with tolerances of $\tau_{\sin(x)} = 0.1$ and $\tau_{x^2} = 0.01$.

2. Using Theorem 2.1, the SOS approximations of $\sin(x)$ and x^2 are represented as hybrid zonotopes.

3. Using Theorem 3.4, Corollary 3.5, and Corollary 3.6, over-approximated graphs of $\sin(w_1) + \sin(w_1)^2$ are computed for resulting functional decompositions given by both Algorithm 3 and Algorithm 4.

Because Algorithm 3 creates a redundant observable, the resulting hybrid zonotope is both less accurate and more complex than achievable with Algorithm 4. Table 3.4 provides complexities of the resulting graphs of functions without and with using hybrid zonotope order reduction methods [22, 58]. Checking for redundant observables comes with computational cost, as Algorithm 3 scales with the number of tokens n as $\mathcal{O}(n)$ whereas Algorithm 4 scales as $\mathcal{O}(n^2)$, however this is often negligible compared to the benefits of a more accurate and compact functional decomposition.

Table 3.4: Memory complexity of hybrid zonotope over-approximations of graphs of $\sin(w_1) + \sin(w_1)^2$ with (3.43) and without (3.42) redundancy in the functional decomposition.

Decomposition	Memory Complexity (n_g, n_b, n_c)
Algorithm 3	$(58, 24, 36)$ $(53, 19, 32)_{\text{red}}$
Algorithm 4	$(43, 18, 26)$ $(39, 15, 23)_{\text{red}}$

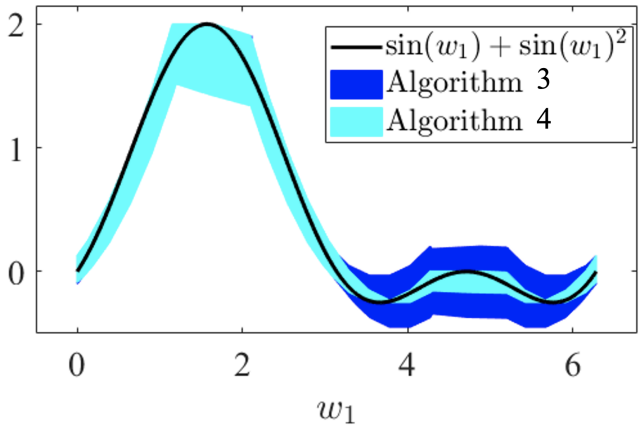


Figure 3.10: Hybrid zonotopes generated using functional decompositions with and without redundant observables.
Over-approximations of the graph of a function of (3.41) obtained using functional decompositions generated by Algorithm 3 and Algorithm 4 are shown in cyan and dark blue, respectively. Both over-approximate nonlinear functions with identical error, however Algorithm 4 is obtained without redundant observables, leading to less error and lower memory complexity.

3.3.3.2.2 Excessive Decomposition of Unary Functions Consider the expression

$$\cos(\sin(w_1 \times w_2)) + \sin(\cos(\sin(w_1 \times w_2))) + \sin(w_1 \times w_2)\,.$$

The SYA yields the RPN expression

$$w_1\ w_2\ \times\ \sin\ \cos\ w_1\ w_2\ \times\ \sin\ \cos\ \sin\ +\ w_1\ w_2\ \times\ \sin\ +\,, \tag{3.44}$$

and Algorithm 3 gives the functional decomposition

$$\begin{aligned}
\text{Inputs}: \ & w_1,\ w_2\,,\\
& w_3 = w_1 \times w_2\,,\\
& w_4 = \sin(w_3)\,,\\
& w_5 = \cos(w_4)\,,\\
& w_6 = \sin(w_5)\,,\\
& w_7 = w_5 + w_6\,,\\
& w_8 = w_4 + w_7.
\end{aligned} \tag{3.45}$$

Algorithm 4: Modified RPN to Functional Decomposition. Given a string of n tokens $\mathcal{E}$ in RPN

Result: Function Decomposition: $w_1, ..., w_{n_x+K}$

$k = n_x + 1$
while $\mathcal{E}$ *is not empty* **do**
  while $\mathcal{E}_1$ *is a number or a variable* **do**
    push $\mathcal{E}_1$ to stack.
  end
  pop tokens from stack associated with the operator or function $\mathcal{E}_1$ and form *candidate* observable expression, w_c.
  if $w_c \neq w_i\ ,\ \forall i \in \{1, ..., k-1\}$ **then**
    Save $w_k = w_c$
    $k \leftarrow k + 1$
  else
    Substitute previously defined and equivalent observable for w_c and push to stack.
  end
end

A more concise decomposition may be realized by combining w_4, w_5, w_6, w_7, e.g.,

$$\begin{aligned} &\text{Inputs}: \ w_1,\ w_2\ , \\ &w_3 = w_1 \times w_2\ , \\ &w_8 = \sin(\cos(\sin(w_3)))\ldots \\ &\qquad + \cos(\sin(w_3)) + \sin(w_3)\ . \end{aligned} \tag{3.46}$$

Graphs provide a convenient and powerful representation to detect and reduce excessive unary decompositions. The example is briefly set aside to introduce and review graphs as they are used within this thesis.

A directed graph $\mathcal{G}$, not to be confused with a graph of a function, consists of a set of n_V vertices $V = \{V_1, V_2, \ldots, V_{n_V}\}$ and a set of n_E directed edges E which connect pairs of vertices. The structure of $\mathcal{G}$ can be represented by an adjacency matrix $A \in \mathbb{R}^{n_V \times n_V}$ where

$$A_{ij} = \begin{cases} 1\ , & \text{if there exists an edge from } V_i \text{ to } V_j\ , \\ 0\ , & \text{otherwise}\ . \end{cases}$$

If there exists an edge with vertex V_i as its tail and vertex V_j as its head (i.e., $A_{ij} = 1$), then V_j is a successor of V_i and V_i is a predecessor of V_j. Herein, the adjacency matrix is not symmetric, yielding a "directed" graph. The transpose of a graph, $\mathcal{G}'$, is a directed

graph with the same vertices as $\mathcal{G}$, but with edges of opposite orientation. For a vertex V_i, the number of edges entering it, referred to as its indegree, is denoted by $d^-(V_i)$, and the number of edges leaving the vertex, referred to as its outdegree, is denoted by $d^+(V_i)$. A walk is an ordered list of vertices $V_1, V_2, ..., V_k$ where $A_{i,i+1} = 1\ ,\ \forall i \in \{1, ..., k-1\}$. A reverse walk, also called a moonwalk [85], is the walk of the transposed graph, $\mathcal{G}'$. A directed graph is said to be acyclic if $\forall V_i \in V$, there exists no possible walk from V_i to itself.

The more compact functional decomposition of (3.46) as compared to (3.45) allows for the approximation of fewer nonlinear functions, reducing the number of nonlinear approximations that need to be made. These excessive unary decompositions can be detected and eliminated by representing the functional decomposition structure as a directed acyclic graph. For example, the composition of (3.45) can be represented as a graph with adjacency matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.47}$$

where $A_{i,j} \in \{0, 1\}$ and $A_{i,j} = 1 \iff w_i$ is an argument of $h_j(\cdot)$. The associated graph of (3.45) represented by adjacency matrix (3.47) is plotted in Figure 3.11.

Detection of an excessive unary decomposition can be achieved by finding a pair of vertices, V_i and V_j, $i \neq j$, such that observable w_j is only a function of w_i. Then, the subgraph between V_i and V_j can be replaced by a single edge which captures the unary function composition. The proposed detection process, given as Algorithm 5, starts by forming two sets for each vertex. The first set, $\mathcal{W}_i$, is the intersection of travelled vertices on all forward walks from V_i and the second set, $\mathcal{M}_i$, is the intersection of travelled vertices on all reverse walks from V_i, i.e., $\mathcal{W}_i$ and $\mathcal{M}_i$ contain vertices that *must* be visited on every forward and reverse walk, respectively. Then, the graph is searched for pairs of vertices V_i and V_j which have mutual dependency in the original and transposed graphs, i.e., $V_j \in \mathcal{W}_i$ and $V_i \in \mathcal{W}_j$.

To perform the simplification step of this method, a recursive composition function

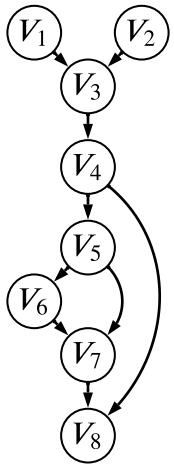


Figure 3.11: Visualization of graph defined by adjacency matrix (3.47).

`comp(·)` is defined as

$$\texttt{comp}(V_i, V_j) = \begin{Bmatrix} w_i\,, & \text{if } i = j \\ f_j(\texttt{comp}(V_i, P_1)\{, \texttt{comp}(V_i, P_2)\})\,, & \text{if } i \neq j \end{Bmatrix}, \tag{3.48}$$

where P is the set of one or two predecessors of V_j. The result of $\texttt{comp}(V_i, V_j)$ is the composition of unary and binary functions with the single input w_i. A graphical representation of reducing (3.45) to (3.46) is demonstrated in Figure 3.12. Note that Algorithm 5 does not simplify the graph in Figures 3.12(a)-(c) as $V_1 \notin \mathcal{M}_3$, $V_2 \notin \mathcal{M}_3$, and $d^+(V_3) = 1$, respectively. Repeating Algorithm 5 until the number of observables remains the same will result in the desired and concise functional decomposition.

Remark 3.3 *When generating a functional decomposition, it may be desired to protect certain intermediate observables from simplification, e.g., in the case with multiple outputs. Therefore, forward and reverse walks in this method are terminated when they reach either a vertex with zero outdegree or a vertex contained in a predefined set of protected vertices.*

3.3.3.3 RPN to Functional Decomposition (Vector-valued Function)

The proposed method in Section 3.3.3.2 completes and simplifies the functional decomposition of a scalar-valued function. This result is now extended to vector functions of finite output dimension, $h(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$. Creating a functional decomposition for

each element if $h(\cdot)$ may result in redundant observables across decompositions, e.g.,

$$h(w_1) = \begin{bmatrix} \sin(w_1) \\ \cos(\sin(w_1)) \end{bmatrix}, \tag{3.49}$$

has similar terms in h_1 and h_2. The functional decomposition of each element would be

$$\begin{array}{rlrl} \text{Inputs:} & w_1 & \text{Inputs:} & w_1 \\ w_2 = & \sin(w_1) & w_3 = & \sin(w_1) \\ & & w_4 = & \cos(w_3) \end{array} \tag{3.50}$$

The simplest way to address this is to concatenate the functional decompositions of each element of $h(\cdot)$ into one single decomposition, ensuring the observables maintain unique indices, e.g.,

$$\begin{array}{rl} \text{Inputs:} & w_1 \\ w_2 = & \sin(w_1) \\ w_3 = & \cos(w_2) \end{array} \tag{3.51}$$

One way to accomplish this while protecting output observables from simplification is to introduce a new function $\bullet(\cdot)$, which evaluates its argument, i.e., $\bullet(x) = x$, and run Algorithm 5 on the function $\sum_{i=1}^{n_y} \bullet(h_i)$. This prevents redundant observables across elements of h as these are detected using Algorithm 4 and allows for easy identification of output observables of h.

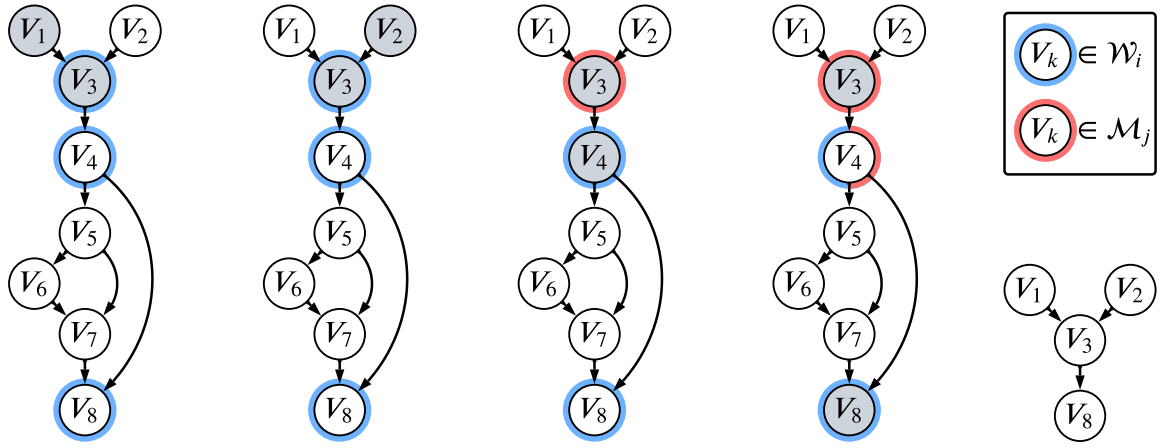
While performing the conversion from RPN to a functional decomposition in Algorithm 4, output observables are identified as the tokens which the identity operators act on. When $\mathcal{E}_1 = \bullet(\cdot)$, the first token is popped from the stack, the observable it corresponds with is noted and pushed back onto the stack, and $\mathcal{E}_1$ is discarded. The vertices corresponding to these noted observables will be added to the set of protected vertices, as described in Remark 3.3.

Again, consider the functional decomposition given in (3.45), but with two ouputs w_4 and w_8, which are added to a set of protected variables. Algorithm 5 results in the functional decomposition

$$\text{Inputs}: \ w_1, \ w_2 \, ,$$

$$
\begin{aligned}
w_3 &= w_1 \times w_2 \,, \\
w_4 &= \sin(w_3) \,, \\
w_8 &= \sin(\cos(w_4)) + \cos(w_4) + w_4,
\end{aligned}
\tag{3.52}
$$

and the process is depicted in Figure 3.13.



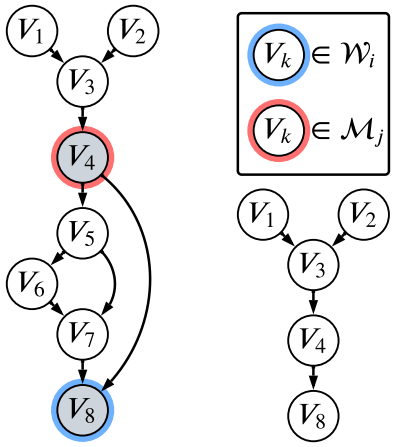
(a) $(i, j) = (1, 3)$ (b) $(i, j) = (2, 3)$ (c) $(i, j) = (3, 4)$ (d) $(i, j) = (3, 8)$ (e) Result

Figure 3.12: Graph representation of (3.45) being simplified to (3.46) according to Algorithm 5.
Gray-filled vertices indicate V_i and V_j. Blue and red-outlined vertices indicate elements of the intersection of forward walks $\mathcal{W}_i$ from V_i and reverse walks $\mathcal{M}_j$ from V_j, respectively. (a) $V_1 \notin \mathcal{M}_3$. (b) $V_2 \notin \mathcal{M}_3$. (c): $A_{i,j} = 0$. (d) This iteration satisfies all conditions in lines 6, 7, and 8 of Algorithm 5. (e) The resulting graph from lines 9 and 10 of Algorithm 5.

3.3.4 Separable Bilinear Functions

It has been assumed in Section 3.3.1 that functional decompositions can include unary or binary nonlinear functions and identities to construct graphs of functional decompositions from graphs of unary and binary functions were given in Section 3.3.2. Section 3.3.3 then presented methods to automate the process of decomposing functions into unary and binary functions. To this point, only methods for constructing hybrid zonotope approximations of unary functions have been addressed (Section 3.2.1).

For some binary functions, existing over-approximation techniques can be leveraged, as will be shown for the bilinear function xy. This section presents a class of functions called "separable" functions which are compatible with methods for constructing hybrid



(a) $(i, j) = (4, 8)$ (b) Result

Figure 3.13: Graph representation of (3.45) being simplified to (3.46) according to Algorithm 5 with V_4 and V_8 listed as protected vertices.
Note the difference between (b) and the result in Figure 3.12(e), as V_4 is no longer removed. This is because forward walks terminate at protected nodes, thus V_8 is no longer in $\mathcal{W}_3$.

zonotope approximations of unary functions. Several key binary functions, xy, $\frac{x}{y}$, and x^y, can each be represented using functional decompositions composed exclusively of separable functions.

Definition 3.3 *A function is separable if it can be equivalently expressed as a summation of unary functions i.e., $f(x_1, ..., x_n)$ is separable if there exist $f_i(x_i)$, $\forall\ i \in \{1, ..., n\}$ such that*

$$f(x_1, ..., x_n) = \sum_{i=1}^{n} f_i(x_i) \ . \tag{3.53}$$

Note that an affine term, while not explicitly written, can be included in any combination of the unary functions $f_i(\cdot)$. The interested reader is referred to [86, Chapter 7.3] and [76, Section 4.2] for a comprehensive review of separable functions.

Proposition 3.1 (Separating Common Binary Functions) *Using an affine change of variables, $w_1 w_2$, $\frac{w_1}{w_2}$, and $w_1^{w_2}$, can each be represented using a functional decomposition composed exclusively of separable functions.*

Proof *A constructive proof is given for each binary function. Recursive substitution of the last observable in each decomposition, until the result is only a function of w_1 and w_2, verifies equality with each expression.*

Algorithm 5: Given a functional decomposition $\mathbb{H}$, generate a functional decomposition $\mathbb{H}_s$ which has fewer unary redundancies.

Result: $\mathbb{H}_s$

Generate directed graph $\mathcal{G}$ from $\mathbb{H}$, with vertices V, edges E, and adjacency matrix A
for $V_i \in V$ **do**
    Generate $\mathcal{W}_i$ and $\mathcal{M}_i$
end
for $i \in \{1, ..., n_V\}$ **do**
    for $V_j \in \mathcal{W}_i$ **do**
        if $V_i \in \mathcal{M}_j$ **then**
            if $A_{ij} = 0$ *OR* $d^+(V_i) > 1$ **then**
                $w_j \leftarrow \texttt{comp}(V_i, V_j)$
                Remove all observables w_k associated with vertices visited on all walks between V_i and V_j, $k \neq i, j$
            end
        end
    end
end
$\mathbb{H}_s \leftarrow$ functional decomposition from $\mathcal{G}$ with re-indexed observables

$w_1 w_2$:

$$\begin{aligned}
\text{Inputs: } & w_1, w_2 \ , \\
& w_3 = w_1 + w_2 \ , \\
& w_4 = w_1 - w_2 \ , \\
& w_5 = w_3^2 \ , \\
& w_6 = w_4^2 \ , \\
& w_7 = \frac{1}{4}(w_5 - w_6) = w_1 w_2 \ ,
\end{aligned}$$

$\frac{w_1}{w_2}$:

$$\begin{aligned}
\text{Inputs: } & w_1, w_2 \ , \\
& w_3 = \frac{1}{w_2} \ , \\
& w_4 = w_1 + w_3 \\
& w_5 = w_1 - w_3 \ ,
\end{aligned}$$

$$
\begin{aligned}
w_6 &= w_4^2 \ , \\
w_7 &= w_5^2 \ , \\
w_8 &= \frac{1}{4}(w_6 - w_7) = \frac{w_1}{w_2} \ ,
\end{aligned}
$$

$w_1^{w_2}$:

$$
\begin{aligned}
&\text{Inputs: } w_1, w_2 \ , \\
&\quad w_3 = \ln w_1 \ , \\
&\quad w_4 = w_2 + w_3 \ , \\
&\quad w_5 = w_2 - w_3 \ , \\
&\quad w_6 = w_4^2 \ , \\
&\quad w_7 = w_5^2 \ , \\
&\quad w_8 = \frac{1}{4}(w_6 - w_7) \ , \\
&\quad w_9 = e^{w_8} = w_1^{w_2} \ .
\end{aligned} \tag{3.54}
$$

□

In the following example, three methods to compute hybrid zonotope over-approximations of the graph of a bilinear function are compared.

Example 3.5 *Three methods for building hybrid zonotope over-approximated graphs of functions for the bilinear function xy are considered. These methods are:*

- *Method 1 (M1)*

  1. *uniformly samples the two dimensional input space, i.e., $n_x = n_y$ where n_x and n_y are the number of breakpoints in the x and y dimensions,*
  2. *constructs V-rep polytopes over-approximations using [73, Theorem 3.4] for each partition, and*
  3. *represents their union as a hybrid zonotope using Theorem 2.1.*

- *Method 2 (M2)*

  1. *adopts the functional decomposition composed of separable functions given in Proposition 3.1,*

2. *constructs uniformly spaced SOS approximations* [5] *for both quadratic functions within the decomposition using identical sampling, i.e.,* $n_u = n_v$ *where* n_u *and* n_v *are the number of breakpoints to approximate each quadratic function,*

3. *converts SOS approximations of quadratic function to HCG-rep using Theorem 2.1,*

4. *generates hybrid zonotope over-approximations of quadratic functions using error bounds from [73, Proposition 3.1] and Corollary 3.4, and*

5. *constructs HCG-rep over-approximation of the graph of the bilinear function using Theorem 3.4, Corollary 3.5, Corollary 3.6, and Corollary 3.7.*

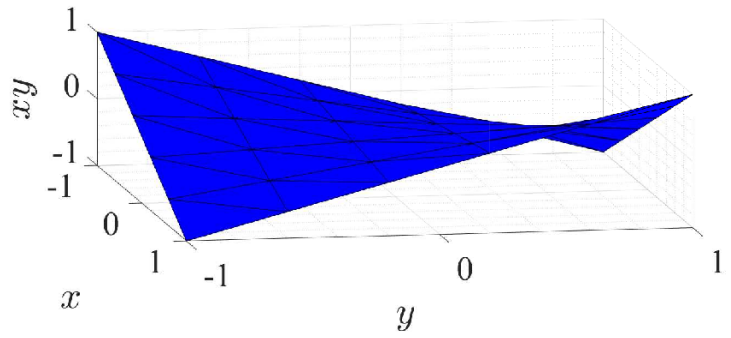
- *Method 3 (M3) constructs approximations in a manner similar to Method 1, but only partitions one dimension using* n_x *breakpoints. The intuition of M3 is that for a fixed value of* x*,* xy *is linear with respect to the* y *dimension; thus it is possible to to obtain tighter over-approximations of* xy *just by sampling the* x *dimension (or* y *dimension).*

The domain of interest is specified as $(x, y) \in \left[-1\,,1\right]^2$*. The number of samples for M2,* n_u*, is chosen as* $n_u = \lceil \sqrt{2} n_x \rceil$ *to ensure that the spacing of the approximation for M2 partitions the domain into squares with a side length at least as small as those using M1. To compare scalability with M2, the sampling for M3 is chosen to result in the same number of binary factors as M2 by setting* $n_{x,M3} = 2n_{u,M2}$

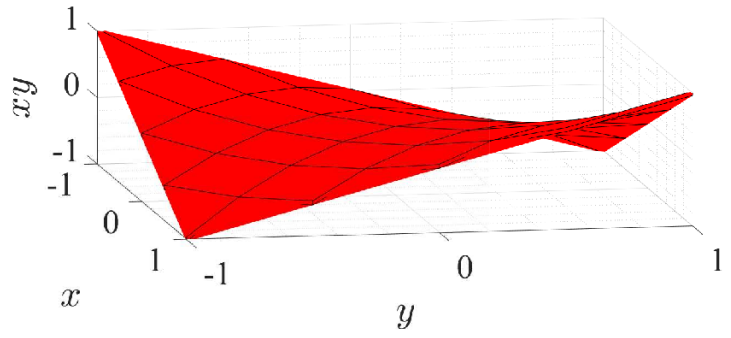
Table 3.5 compares the complexity of the three methods, demonstrating how the separable decomposition in Proposition 3.1 enables more scalable over-approximations of the bilinear function as sampling is increased and over-approximations become more accurate. Figure 3.14 plots over-approximations of the bilinear function using all three methods corresponding to Case $(M1 : n_x = 6, M2 : n_u = 17, M3 : n_x = 34)$ *in Table 3.5. The partitioning in Figure 3.14(a) and Figure 3.14(c) corresponds to sampling in the* (x, y) *space, while the partitioning in Figure 3.14(b) corresponds to sampling quadratic functions in a rotated space.*

Remark 3.4 *For the bilinear function* xy*, M3 has been found to outperform M2 when the domains of* $x \in \left[\underline{x}\,,\ \ \bar{x}\right]$ *and* $y \in \left[\underline{y}\,,\ \ \bar{y}\right]$ *are such that* $\bar{x} - \underline{x} << \bar{y} - \underline{y}$ *or* $\bar{x} - \underline{x} >> \bar{y} - \underline{y}$*.*

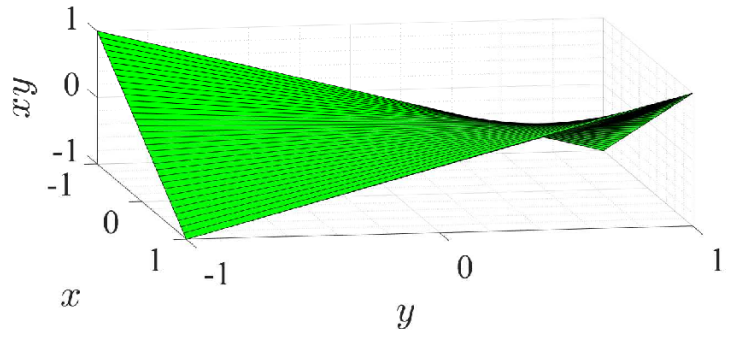
[5] Uniform sampling is optimal for quadratic functions as the error is dependent only of the difference between breakpoints [73, Proposition 3.1].



(a): M1 $n_x = n_y = 6$



(b): M2 $n_u = n_v = 17$



(c): M3 $n_x = 34,\ n_y = 2$

Figure 3.14: Comparison of 2-D and 1-D sampling methods for over-approximating $\{xy \mid x, y \in [-1\ 1]\}$.

3.4 Hybrid Zonotope Graphs of Common Functions

This section provides explicit examples of commonly used and foundational functions for which hybrid zonotopes are well-suited. For each function, the following are provided,

- an assumed domain,
- a corresponding vertex and incidence matrix for which Theorem 2.1 would generate a hybrid zonotope graph of the function (or its over-approximation), and

Table 3.5: Complexity comparison of three methods for approximating the bilinear function $f(x, y) = xy$.

The leftmost column denotes (n_x for M1, n_u for M2, n_x for M3) and these values are selected for parity in the comparison.

Case	M1: $n_x = n_y$	M2: $n_u = n_v$	M3: n_x ($n_y = 2$)
(3,9,18)	$n_g = 19$ $n_b = 5$ $n_c = 11$	$n_g = 24$ $n_b = 10$ $n_c = 14$	$n_g = 41$ $n_b = 10$ $n_c = 22$
(6,17,34)	$n_g = 73$ $n_b = 26$ $n_c = 38$	$n_g = 40$ $n_b = 18$ $n_c = 22$	$n_g = 73$ $n_b = 18$ $n_c = 38$
(9,26,52)	$n_g = 163$ $n_b = 65$ $n_c = 83$	$n_g = 56$ $n_b = 26$ $n_c = 30$	$n_g = 105$ $n_b = 26$ $n_c = 54$
(12,34,68)	$n_g = 289$ $n_b = 122$ $n_c = 146$	$n_g = 72$ $n_b = 34$ $n_c = 38$	$n_g = 137$ $n_b = 34$ $n_c = 70$

- resulting memory complexity with and without order reduction methods.

SOS approximations are not included again in this section, as they are previously discussed in detail in Section 3.2

3.4.1 Absolute Value (Exact)

Consider the function

$$y = |x| \,. \tag{3.55}$$

The domain is given as $x \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix}$ and it is assumed that $\underline{x} < 0$ and $\bar{x} > 0$, which corresponds to the most general case. The *exact* graph of the function (3.55) over the domain can be generated using the vertex and incidence matrices given by

$$V_{|x|} = \begin{bmatrix} \underline{x} & 0 & \bar{x} \\ -\underline{x} & 0 & \bar{x} \end{bmatrix}, \quad M_{|x|} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 6$	$n_{g,red} = 4$
$n_b = 2$	$n_{b,red} = 1$
$n_c = 5$	$n_{c,red} = 2$

3.4.2 Satisfaction of Inequality (Approximate)

Consider the function

$$y = \begin{cases} 1\,, & \text{if } x \leq c\,, \\ 0\,, & \text{if } x > c\,, \end{cases} \tag{3.56}$$

where c is a finite constant. The domain is given as $x \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix}$ and it is assumed that $\underline{x} < c$ and $\bar{x} > c$, which corresponds to the most general case. The *approximate* graph of the function (3.56) can be generated using the vertex and incidence matrices given by

$$V_{x \leq c} = \begin{bmatrix} \underline{x} & c & c+\delta & \bar{x} \\ 1 & 1 & 0 & 0 \end{bmatrix}, \quad M_{x \leq c} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 8$	$n_{g,red} = 8$
$n_b = 2$	$n_{b,red} = 1$
$n_c = 6$	$n_{c,red} = 5$

Over-approximations are given when $\delta = 0$ and inner-approximations when $\delta > 0$. For the best inner-approximation, δ can be set to the machine precision. Then the graph of the function (3.56) is open, and therefore cannot be represented exactly using a collection of V-rep polytopes.

3.4.3 Sign (Approximate)

Consider the function

$$y = \begin{cases} -1\,, & \text{if } x < 0\,, \\ 0\,, & \text{if } x = 0\,, \\ 1\,, & \text{if } x > 0\,. \end{cases} \tag{3.57}$$

The domain is given as $x \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix}$ and it is assumed that $\underline{x} < 0$ and $\bar{x} > 0$, which corresponds to the most general case. The *approximate* graph of the function (3.57) can be generated using the vertex and incidence matrices given by

$$V_{\text{sign}(x)} = \begin{bmatrix} \underline{x} & -\delta & 0 & \delta & \bar{x} \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix}, \quad M_{\text{sign}(x)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 10$	$n_{g,red} = 9$
$n_b = 3$	$n_{b,red} = 3$
$n_c = 7$	$n_{c,red} = 7$

Over-approximations are given when $\delta = 0$ and inner-approximations when $\delta > 0$. For the best inner-approximation, δ can be set to the machine precision. Then the graph of the function (3.57) is open, and therefore cannot be represented exactly using a collection of V-rep polytopes.

3.4.4 Rectified Linear Unit (Exact)

Consider the function

$$y = \begin{cases} x\,, & \text{if } x > 0\,, \\ 0\,, & \text{if } x \leq 0\,. \end{cases} \tag{3.58}$$

The domain is given as $x \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix}$ and it is assumed that $\underline{x} < 0$ and $\bar{x} > 0$, which corresponds to the most general case. The *exact* graph of the function (3.58) over the domain can be generated using the vertex and incidence matrices given by

$$V_{\mathrm{ReLU}(x)} = \begin{bmatrix} \underline{x} & 0 & \bar{x} \\ 0 & 0 & \bar{x} \end{bmatrix}, \quad M_{\mathrm{ReLU}(x)} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 6$	$n_{g,red} = 4$
$n_b = 2$	$n_{b,red} = 1$
$n_c = 5$	$n_{c,red} = 2$

The graph of the function (3.58) is represented exactly in spite of the inequality constraint, $x > 0$, because the graph of the function can be represented exactly as the union of two polytopes.

3.4.5 Minimum (Exact)

Consider the function

$$y = \begin{cases} x_1 \,, & \text{if } x_1 \leq x_2 \,, \\ x_2 \,, & \text{if } x_1 > x_2 \,. \end{cases} \tag{3.59}$$

The domain is given as $(x_1, x_2) \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix} \times \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix}$, though cases of independent domains for x_1 and x_2 can also be handled via minor extension. The *exact* graph of the function (3.59) over the domain can be generated using the vertex and incidence matrices given by

$$V_{\min(x_1,x_2)} = \begin{bmatrix} \underline{x} & \underline{x} & \bar{x} & \bar{x} \\ \underline{x} & \bar{x} & \underline{x} & \bar{x} \\ \underline{x} & \underline{x} & \underline{x} & \bar{x} \end{bmatrix}, \quad M_{\min(x_1,x_2)} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 8$	$n_{g,red} = 6$
$n_b = 2$	$n_{b,red} = 1$
$n_c = 6$	$n_{c,red} = 3$

The graph of the function (3.59) is represented exactly in spite of the inequality constraint, $x_1 > x_2$, because the graph of the function can be represented exactly as the union of two polytopes.

Remark 3.5 *The maximum function can be achieved over the same domain using the matrices*

$$V_{\max(x_1,x_2)} = \begin{bmatrix} \underline{x} & \underline{x} & \bar{x} & \bar{x} \\ \underline{x} & \bar{x} & \underline{x} & \bar{x} \\ \underline{x} & \bar{x} & \bar{x} & \bar{x} \end{bmatrix}, \quad M_{\max(x_1,x_2)} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix},$$

and has the same memory complexity reported for the minimum function.

3.4.6 Boolean (Exact)

Consider any boolean function $y = f_{\text{bool}}(x_1, x_2)$, where

$$y = \begin{cases} f_{\text{bool}}(0,0)\,, & \text{if } (x_1,x_2) = (0,0)\,, \\ f_{\text{bool}}(0,1)\,, & \text{if } (x_1,x_2) = (0,1)\,, \\ f_{\text{bool}}(1,0)\,, & \text{if } (x_1,x_2) = (1,0)\,, \\ f_{\text{bool}}(1,1)\,, & \text{if } (x_1,x_2) = (1,1)\,. \end{cases} \tag{3.60}$$

The domain is given as $(x_1, x_2) \in \{0,1\}^2$. The *exact* graph of the function (3.60) over the domain can be generated using the vertex and incidence matrices given by

$$V_{\text{bool}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ f_{\text{bool}}(0,0) & f_{\text{bool}}(0,1) & f_{\text{bool}}(1,0) & f_{\text{bool}}(1,1) \end{bmatrix}, \quad M_{\text{bool}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 8$	$n_{g,red} = 0$
$n_b = 4$	$n_{b,red} = 4$
$n_c = 6$	$n_{c,red} = 1$

Note that the graph of a function is the union of four V-rep polytopes, each with a single vertex.

3.4.7 Identity with Boolean On/Off Switch (Exact)

Consider the function $y = f(x, b)$, where

$$y = \begin{cases} x\,, & \text{if } b = 1\,, \\ 0\,, & \text{if } b = 0\,. \end{cases} \tag{3.61}$$

The domain is given as $(x, b) \in \begin{bmatrix} \underline{x} & \bar{x} \end{bmatrix} \times \{0, 1\}$. The *exact* graph of the function (3.61) over the domain can be generated using the vertex and incidence matrices given by

$$V_{\text{bool}} = \begin{bmatrix} \underline{x} & \bar{x} & \underline{x} & \bar{x} \\ 1 & 1 & 0 & 0 \\ \underline{x} & \bar{x} & 0 & 0 \end{bmatrix}, \quad M_{\text{bool}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

which when converted to HCG-rep results in the memory complexity below.

Theorem 2.1	Order Reduction via [22, 58]
$n_g = 8$	$n_{g,red} = 8$
$n_b = 2$	$n_{b,red} = 1$
$n_c = 6$	$n_{c,red} = 5$

Chapter 4
Reachability Analysis

4.1 Introduction

Reachability analysis—the process of calculating reachable sets—is used to evaluate system performance and ensure constraint satisfaction in safety-critical applications, while accounting for the effects of input, disturbance, and parameter uncertainties. However, the scalability of existing approaches for hybrid, logical, neural networks, and nonlinear systems is limited by their nonconvexity and the computational complexity that this induces.

For discrete-time systems, reachable and invariant sets are calculated by recursion of precursor and successor sets, also referred to as one-step backward and one-step forward reachable sets, respectively [31]. Set propagation techniques for continuous-time systems often resemble their discrete-time counterparts, as the former often still propagate reachable sets over discrete time intervals [87]. For this reason, there are many shared challenges in calculating reachable sets for continuous-time and discrete-time systems. Notionally, the successor set and precursor set are defined as follows.

Definition 4.1 *(Successor Set) The successor set from a given set is the set of states that are achievable from a current set of states in exactly one time step.*

Definition 4.2 *(Precursor Set) The precursor set from a given set is the set of states from which the given set can be achieved in exactly one time step.*

Successor and precursor sets depend on the given set at the current time step, the system dynamics, and the presence of inputs, disturbances, and parameter uncertainties. Definition 4.1 and Definition 4.2 are left intentionally vague as assumptions vary between the classes of systems presented herein. Within each section, formal definitions of successor and precursor sets are given.

Verification and Falsification via Forward Reachability: Forward reachable sets, calculated by recursively finding successor sets, are used to assess the safety of a dynamic system by determining whether any state starting in an initial set can reach an unsafe state in a finite number of time steps. The system safety is falsified, i.e., the system is proven to be unsafe, if the exact forward reachable sets have a non-empty intersection with an unsafe state or set. Rather than exact forward reachable sets, if over-approximations of forward reachable sets are computed instead, then the system is again verified to be safe if the intersection of these sets with the unsafe set is empty. Otherwise, the safety of the system cannot be decided, e.g., the portion of the over-approximated reachable set that intersects the unsafe region may, or may not, be in the exact forward reachable set.

As uncertainties in inputs, disturbances, and parameters increase, forward reachable sets become larger. In the case of an additive uncertainty, this corresponds to a Minkowski sum with a larger set. Disturbance and parameter uncertainties are unknown, and for this reason, if a forward reachable set intersects a region of interest, it *cannot* be concluded that "the system can be driven into the region of interest". When disturbance and parameter uncertainties are present in a system with a closed-loop controller, it can concluded that it is possible to drive a system into the region of interest in a manner that is robust to disturbance and parameter uncertainty if and only if the entire forward reachable set is contained within a region of interest.

Verification and Falsification via Backward Reachability: Backward reachable sets, calculated by recursively finding precursor sets, are often calculated in a manner that is *robust* to disturbance and parameter uncertainties. Robust backward reachable sets are often used for verification of systems with a closed-loop controller as they represent the set of states that will be driven to a target set for any disturbances and/or parameter uncertainties within some assumed bounds. Robust backwards reachable sets can also be calculated for open-loop systems with a set of feasible inputs and represent the set of states that can be driven to a target set given a suitable control sequence.

As uncertainties in disturbances and parameters become larger, robust backward reachable sets become smaller, and potentially even empty. In the case of an additive uncertainty, this corresponds to a Minkowski difference with a larger set. One interpretation is that disturbance and parameter uncertainties can be viewed as adversarial/non-cooperative agents working to prevent the evolution of a trajectory to the target set, whereas the input can be viewed as a cooperative agent to drive the state trajectory to the target set.

4.2 Hybrid Systems

4.2.1 Literature Review

Forward reachable sets of linear hybrid systems may be calculated using a collection of convex sets by partitioning the state space into locations separated by guards and applying techniques developed for linear systems within each location [25, 28]. Successive intersections with guards at each time step result in worst-case exponential growth in complexity, leading to computational-intractability for long time horizons. Over-approximation techniques like clustering-based methods reduce complexity of sets at the cost of conservatism [29]. The extent of this conservatism is application-dependent and difficult to quantify [30].

Hamilton-Jacobi reachability is applicable to nonlinear dynamics including hybrid systems; however, its computational complexity scales exponentially with the system state dimension. Techniques to reduce computational complexity decompose the system into subsystems based on its structure [6–9].

Hybrid zonotopes have been shown to enable scalable closed-form solutions of successor sets for broad classes of discrete-time linear hybrid systems. This includes Mixed Logical Dynamical (MLD) systems [22] and linear systems in closed-loop with Model Predictive Control (MPC) [23], of which the explicit solution can yield piecewise affine (PWA) control laws [31]. Previous work does not address the use of hybrid zonotopes to calculate precursor sets for backward reachability.

In general, precursor sets of hybrid systems are challenging to calculate. Even for *autonomous* discrete-time hybrid systems, there may be many states that converge to a single successor state. This complexity is often compounded when considering disturbances, as analysis relies on computing or approximating Minkowski differences [35, 36]. Similar to finding successor sets of hybrid systems, precursor sets may be found by considering collections of convex sets within each location, with worst-case exponential growth in set complexity with time [37, 38]. Many MPC formulations leverage invariant sets to guarantee properties such as recursive feasibility. Lacking scalable methods to calculate precursor sets under the PWA control laws of MPC, artificial constraints are introduced to ensure recursive feasibility, resulting in conservatism [31].

This section proposes exact, closed-form identities for calculating robust precursor and successor sets of discrete-time linear hybrid systems using hybrid zonotopes. Using graphs of functions for the dynamics, called state-update sets, successor sets of MLD systems,

closed-loop MPC, and DHA can be computed with lower computational complexity and set representation complexity than in previous work. Robust precursor sets can also be calculated with state-update sets. This in turn enables calculation of invariant sets for linear systems under MPC with reduced conservatism as compared to conventional methods.

4.2.2 Mixed Logical Dynamical Systems

4.2.2.1 Introduction

Introduced in [88], the MLD system modeling framework combines continuous and binary variables with logical relations in mixed-integer inequalities to express complex system dynamics. MLD systems can be used to model mixed continuous and discrete states and inputs, PWA and bilinear dynamics, finite state machines, and those with any combination of the former [33, 88]. An MLD system with an additive disturbance may be expressed as

$$x_+ = Ax + B_u u + B_v v + B_{aff} + w \,, \tag{4.1a}$$

$$\text{s.t.} \;\; E_x x + E_u u + E_v v \leq E_{aff} \,, \tag{4.1b}$$

where $x \in \mathbb{R}^{n_{xc}} \times \{0,1\}^{n_{xl}}$ are the system states, $u \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ul}}$ are the control inputs, $v \in \mathbb{R}^{n_{rc}} \times \{0,1\}^{n_{rl}}$ are auxiliary variables, and $w \in \mathcal{W} \subset \mathbb{R}^{xc} \times \{0,1\}^{n_{xl}}$ is a disturbance. The number of inequality constraints is denoted by n_e such that $E_{aff} \in \mathbb{R}^{n_e}$. The total number of system states is denoted by $n = n_{xc} + n_{xl}$.

When formulating an MLD system model (4.1), the so-called "big-M" constants used in the mixed-integer inequalities to relate continuous values to logical statements [89] are chosen for a user defined subset of the state space, $\mathcal{X} \subset \mathbb{R}^{n_{xc}} \times \{0,1\}^{n_{xl}}$, and set of admissible control inputs, $\mathcal{U} \subset \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ul}}$ [88]. It follows that for the bounded state-input domain over which the MLD model is defined, the auxiliary variables will belong to a bounded set $\mathcal{V} \subset \mathbb{R}^{n_{rc}} \times \{0,1\}^{n_{rl}}$ and may be found through domain propagation.

Assumption 4.1 (Well-Posed) *The MLD system given by* (4.1) *is well-posed, meaning that x_+ is uniquely determined by x, u, and w.*

Definition 4.3 (Robust Successor Set) [60, Definition 2] *The* robust successor

set *from* $\mathcal{R}_k \subseteq \mathcal{X}$ *by the MLD system* (4.1) *is given by*

$$\mathrm{Suc}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) = \left\{ x_+ \;\middle|\; \begin{array}{c} x_+ = Ax + B_u u + B_v v + B_{aff} + w \\ x \in \mathcal{R}_k,\ u \in \mathcal{U},\ v \in \mathcal{V}, w \in \mathcal{W}, \\ E_x x + E_u u + E_v v \leq E_{aff} \end{array} \right\}. \tag{4.2}$$

Definition 4.4 (Robust Precursor Set) [60, Definition 3] *The* robust precursor set *for* $\mathcal{R}_k \subseteq \mathcal{X}$ *by the MLD system* (4.1) *is given by*

$$\mathrm{Pre}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) = \left\{ \begin{array}{c} x \in \mathbb{R}^{n_{xc}} \times \{0,1\}^{n_{xl}} \mid \exists u \in \mathcal{U},\ s.t. \\ Ax + B_u u + B_v v + B_{aff} + w \in \mathcal{R}_k, \\ \forall v \in \mathcal{V}\ s.t.\ E_x x + E_u u + E_v v \leq E_{aff}, \\ \forall w \in \mathcal{W} \end{array} \right\}. \tag{4.3}$$

4.2.2.2 Proposed Method

The state-update set encodes all possible state transitions of (4.1) for a specified input set $\mathcal{U}$ and no disturbance, i.e., $\mathcal{W} = \{\mathbf{0}\}$, thus the state-update set is defined as the graph of the function $\phi(x) = \mathrm{Suc}(x, \mathcal{U}, \{\mathbf{0}\})$ over a domain $x \in \mathrm{D}_\Phi$. Identities are given for precursor and successor sets using the state-update set, and closely mirror the input-output identities (3.3) and (3.5).

Definition 4.5 (State-Update Set) *The state-update set* $\Phi \subseteq \mathbb{R}^{2n}$ *is defined as*

$$\Phi = \left\{ \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{c} x_{k+1} \in \mathrm{Suc}(\{x_k\}, \mathcal{U}, \{\mathbf{0}\}), \\ x_k \in D_\Phi \end{array} \right\}. \tag{4.4}$$

We refer to $\mathrm{D}_\Phi \subseteq \mathcal{X}$ as the *domain set* of Φ, typically chosen as the region of interest for analysis. Similarly, the *range set* of Φ is defined as $\mathrm{R}_\Phi = \{x_{k+1} | (x_k, x_{k+1}) \in \Phi\}$. Given a state-update set Φ, D_Φ and R_Φ can be calculated as $\mathrm{D}_\Phi = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \Phi$ and $\mathrm{R}_\Phi = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \Phi$.

Theorem 4.1 (Robust Successor Set Identity) [60, Theorem 1] *Given a set of states* $\mathcal{R}_k \subseteq \mathbb{R}^n$ *and state-update set* Φ*, if* $\mathcal{R}_k \subseteq D_\Phi$*, then*

$$\mathrm{Suc}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \end{bmatrix} \left(\Phi \cap_{[\boldsymbol{I}\ \mathbf{0}]} \mathcal{R}_k \right) \oplus \mathcal{W}. \tag{4.5}$$

Proof *By definition of the generalized intersection and Definition 4.5,*

$$\Phi \cap_{[\boldsymbol{I}\ \mathbf{0}]} \mathcal{R}_k = \left\{ \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} \middle| \begin{array}{c} x_{k+1} \in \mathrm{Suc}(\{x_k\}, \mathcal{U}, \{\mathbf{0}\}), \\ x_k \in \mathcal{R}_k \cap D_\Phi \end{array} \right\}.$$

If $\mathcal{R}_k \subseteq D_\Phi$*, then* $\mathcal{R}_k \cap D_\Phi = \mathcal{R}_k$*. Thus* (4.5) *yields* $\{x_{k+1} | x_{k+1} \in \mathrm{Suc}(\{x_k\}, \mathcal{U}, \mathcal{W}),\ x_k \in \mathcal{R}_k\}$. □

The containment condition in Theorem 4.1, $\mathcal{R}_k \subseteq \mathrm{D}_\Phi$, is not restrictive as modeled dynamics are often only valid over some region of interest, which the user may specify as D_Φ. If $\mathcal{R}_k \not\subseteq \mathrm{D}_\Phi$, it can be shown that the right side of (4.5) gives $\mathrm{Suc}(\mathcal{R}_k \cap \mathrm{D}_\Phi, \mathcal{U}, \mathcal{W})$. We now consider the robust precursor set.

Lemma 4.1 (Equivalent State-Update Set Definition) [60, Lemma 1] *The state-update set* (4.4) *is equivalently defined in terms of the precursor set as*

$$\Phi = \left\{ \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} \middle| \begin{array}{c} x_k \in \mathrm{Pre}(\{x_{k+1}\}, \mathcal{U}, \{\mathbf{0}\}), \\ x_k \in D_\Phi \end{array} \right\}. \tag{4.6}$$

Proof *This may be shown for well-posed MLD systems (Assumption 4.1) by expanding* (4.4) *and* (4.6) *using the successor* (4.2) *and precursor* (4.3) *set definitions.* □

Theorem 4.2 (Robust Precursor Set Identity) [60, Theorem 2] *Given a set of states* $\mathcal{R}_k \subseteq \mathbb{R}^n$ *and state-update set* Φ*, if* $\mathcal{R}_k \ominus \mathcal{W} \subseteq R_\Phi$*, then*

$$\mathrm{Pre}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) \cap D_\Phi = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} \end{bmatrix} \left(\Phi \cap_{[\mathbf{0}\ \boldsymbol{I}]} (\mathcal{R}_k \ominus \mathcal{W}) \right). \tag{4.7}$$

Proof *By definition of the Minkowski difference, generalized intersection, and Lemma 4.1,*

$$\Phi \cap_{[\mathbf{0}\ \boldsymbol{I}]} (\mathcal{R}_k \ominus \mathcal{W}) = \left\{ \begin{bmatrix} x_{k-1} \\ \hat{x}_k \end{bmatrix} \middle| \begin{array}{c} x_{k-1} \in \mathrm{Pre}(\{\hat{x}_k\}, \mathcal{U}, \{\mathbf{0}\}), \\ x_{k-1} \in D_\Phi, \\ \hat{x}_k + w \subseteq \mathcal{R}_k\ ,\ \forall w \in \mathcal{W} \end{array} \right\}.$$

The linear transformation in (4.7) *yields the desired result.* □

When Φ is generated for a domain D_Φ corresponding to a region of interest for analysis, the condition of Theorem 4.2 is not restrictive. If $\mathcal{R}_k \ominus \mathcal{W} \not\subseteq \mathrm{R}_\Phi$, then $\exists x_k \in \mathcal{R}_k$ that can be reached, but for which no combination of $x_{k-1} \in \mathrm{D}_\Phi$, $u \in \mathcal{U}$ and

$w \in \mathcal{W}$ can account. Regarding the left side of (4.7), D_Φ can be chosen such that $\mathrm{Pre}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) \subseteq \mathrm{D}_\Phi \implies \mathrm{Pre}(\mathcal{R}_k, \mathcal{U}, \mathcal{W}) \cap \mathrm{D}_\Phi = \mathrm{Pre}(\mathcal{R}_k, \mathcal{U}, \mathcal{W})$.

Time complexity of the successor set given by (4.5) is $\mathcal{O}(n)$, as the linear mapping $[\mathbf{I}\ \mathbf{0}]$ under the generalized intersection amounts to matrix concatenation. This is a reduction from $\mathcal{O}(n^3)$ reported for successor sets of MLD systems in [22]. Time complexity of the precursor set given by (4.7) is $\mathcal{O}(nn_{g,w})$ due to the Minkowski difference. Set complexity growth of the successor and precursor sets is given by

$$
\begin{aligned}
n_{g,\mathrm{Suc}} &= n_{g,r} + n_{g,\phi} + n_{g,w}\,, & n_{g,\mathrm{Pre}} &= 2^{n_{g,w}} n_{g,r} + n_{g,\phi}\,,\\
n_{b,\mathrm{Suc}} &= n_{b,r} + n_{b,\phi}\,, & n_{b,\mathrm{Pre}} &= 2^{n_{g,w}} n_{b,r} + n_{b,\phi}\,,\\
n_{c,\mathrm{Suc}} &= n_{c,r} + n_{c,\phi} + n\,, & n_{c,\mathrm{Pre}} &= 2^{n_{g,w}} n_{c,r} + 2^{n_{g,w}} n + n_{c,\phi}\,.
\end{aligned}
$$

Iterative calculation of successor sets (4.5) results in linear complexity growth dependent on the complexity of Φ and $\mathcal{W}$. If $n_{g,w} = 0$, time complexity of precursor sets (4.7) reduces to $\mathcal{O}(n)$ and iterative applications give linear complexity growth. In order to obtain the reported computational and memory complexity for MLD systems given by (4.5) and (4.7), the state-update set for an MLD system must be represented as a hybrid zonotope. Theorem 4.3 provides a closed-form hybrid zonotope solution for state-update sets of MLD systems.

Theorem 4.3 (Construct MLD State-Update Set) [60, Theorem 3] *Given a well-posed MLD system (Assumption 4.1), defined over $\mathcal{X}$ and described by* (4.1)*, let*

$$
\mathcal{Q} = \begin{bmatrix}\mathbf{0} & B_u^\mathsf{T} & E_u^\mathsf{T}\end{bmatrix}^\mathsf{T} \mathcal{U} \oplus \begin{bmatrix}\mathbf{0} & B_v^\mathsf{T} & E_v^\mathsf{T}\end{bmatrix}^\mathsf{T} \mathcal{V} \oplus \begin{bmatrix}\mathbf{0} & B_{aff}^\mathsf{T} & \mathbf{0}\end{bmatrix}^\mathsf{T}\,,
$$

and define $\mathcal{H} = \{x \in \mathbb{R}^{n_e} : x \leq E_{aff}\}$. Then a state-update set with $D_\Phi = \mathcal{X}$ for the MLD system is given by

$$
\Phi = [\boldsymbol{I}_{2n}\ \mathbf{0}] \left[\Big([\boldsymbol{I}\ A^\mathsf{T}\ E_x^\mathsf{T}]^\mathsf{T} \mathcal{X} \oplus \mathcal{Q} \Big) \cap_{[\mathbf{0}\ \boldsymbol{I}_{n_e}]} \mathcal{H} \right]\,. \tag{4.8}
$$

Proof *Let $\hat{\Phi}$ be the hybrid zonotope given by the right side of* (4.8)*. For any $\hat{\phi} \in \hat{\Phi}$, there exists $p \in [\boldsymbol{I}\ A^\mathsf{T}\ E_x^\mathsf{T}]^\mathsf{T} \mathcal{X} \oplus \mathcal{Q}$, $x \in \mathcal{X}, u \in \mathcal{U}$, and $v \in \mathcal{V}$ such that*

$$
p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} x \\ Ax + B_u u + B_v v + B_{aff} \\ E_x x + E_u u + E_v v \end{bmatrix}\,,
$$

$\hat{\phi} = [\boldsymbol{I}_{2n}\ \mathbf{0}]p$, and $[\mathbf{0}\ \boldsymbol{I}_{n_e}]p \in \mathcal{H}$. By [22, Theorem 8], $p_2 \in \mathrm{Suc}(p_1, \mathcal{U}, \{\mathbf{0}\})$, thus $\hat{\phi} \in \Phi$.

The MLD is defined $\forall x \in \mathcal{X}$*, therefore* $D(\hat{\Phi}) = \mathcal{X}$ *and* $\hat{\Phi} = \Phi$. □

4.2.2.3 Numerical Examples

4.2.2.3.1 Two-Equilibrium System [60]

Consider the PWA system with two equilibria,

$$x[k+1] = \begin{cases} A_1 x[k] + B_1 + u[k] + w[k]\,, & \text{if } x_1[k] \leq 0 \\ A_2 x[k] + B_2 + u[k] + w[k]\,, & \text{if } x_1[k] > \varepsilon \end{cases},$$

$$A_1 = \begin{bmatrix} 0.75 & 0.25 \\ -0.25 & 0.75 \end{bmatrix}\,,\; B_1 = \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix}\,,\; A_2 = A_1^{\mathsf{T}}\,,\; B_2 = \begin{bmatrix} 0.25 \\ -0.25 \end{bmatrix}.$$

where ε is the machine precision. The inputs and disturbances are constrained to belong to scaled unit hypercubes such that $u[k] \in \mathcal{U} = s_u \mathcal{B}_\infty^2$ and $w[k] \in \mathcal{W} = s_w \mathcal{B}_\infty^2$. Forward $(\mathcal{R}_4, \mathcal{R}_5, \mathcal{R}_6)$ and backward $(\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2)$ reachable sets are calculated from

$$\mathcal{R}_3 = \left\langle \begin{bmatrix} 0.15 & 0.05 \\ -0.05 & 0.15 \end{bmatrix}, \begin{bmatrix} -0.0520 \\ 0.8465 \end{bmatrix} \right\rangle,$$

and shown in Figure 4.1 for three cases of input and disturbance sets defined in Table 4.1. This example system and choice of $\mathcal{R}_3$ are intended to facilitate comparison with forward reachability results from [22]. The state-update set for Case 1 and Case 2 (no input) is calculated and found to have complexity $n_{g,\phi} = 16,\; n_{b,\phi} = 1,\; n_{c,\phi} = 10$. The state-update set for Case 3 has 2 additional continuous generators associated with the input. Computation times and reachable set complexities are reported in Table 4.1.

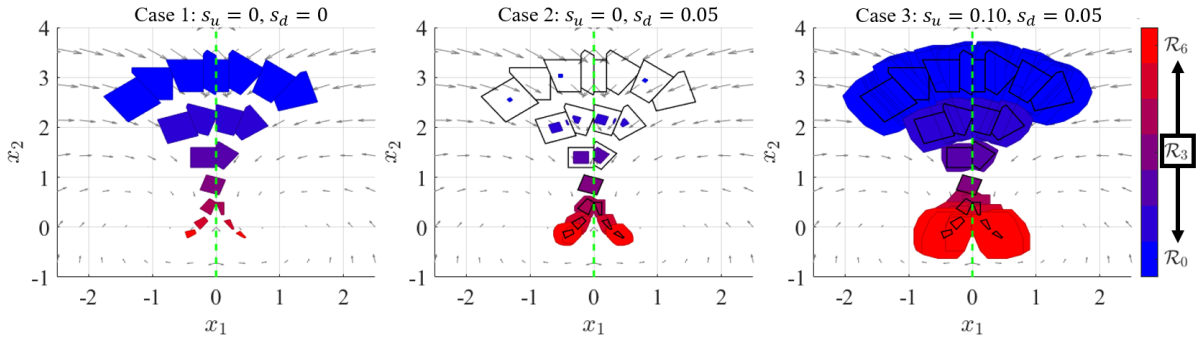


Figure 4.1: Forward $(\mathcal{R}_4, \mathcal{R}_5, \mathcal{R}_6)$ and backward $(\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2)$ reachable sets of a two-equilibrium system from $\mathcal{R}_3$ for three cases of input and disturbance sets. Sets from the Case 1 subplot are also shown in wire frame in the Case 2 and 3 subplots for comparison.

Table 4.1: Forward and backward reachable set complexities and computation times for the two-equilibrium system.

Case	Direction/Set	$n_{g,r}$	$n_{b,r}$	$n_{c,r}$	Time [ms]
1) $s_u = 0\,, s_w = 0$	Forward: $\mathcal{R}_6$	50	3	36	0.13
	Backward: $\mathcal{R}_0$	50	3	36	0.12
2) $s_u = 0\,, s_w = 0.05$	Forward: $\mathcal{R}_6$	56	3	36	0.23
	Backward: $\mathcal{R}_0$	464	21	378	2.92
3) $s_u = 0.10\,, s_w = 0.05$	Forward: $\mathcal{R}_6$	62	3	36	0.24
	Backward: $\mathcal{R}_0$	506	21	378	3.12

In Figure 4.1, forward reachable sets are shown to split along the guard, while backward reachable sets branch when their precursor sets span both sides of the guard. Comparison of Case 1 and Case 2 demonstrates that backward reachable sets are smaller when required to be robust to a disturbance, while forward reachable sets become larger. Comparison of Case 2 and Case 3 demonstrates that adding control authority causes forward and backward reachable sets to become larger.

In Case 1, backward and forward reachable sets are calculated with similar computation times and have identical complexity growth when there is no disturbance. The addition of a disturbance in Case 2, and both a disturbance and an input in Case 3, has a small effect on computation times and complexity growth of forward reachable sets. However, because the robust precursor set relies on a Minkowski difference, when a disturbance is present, the computation time and complexity growth of backward reachable sets increase significantly.

Using the methods from [22] for *exact* order reduction, 5 continuous generators and 3 constraints are removed from the state-update set. The 6-step backward reachable sets from $\mathcal{R}_3$ for Case 2 using the nominal ($\mathcal{R}_{-3}$) and reduced ($\mathcal{R}_{-3}^{red}$) state-update sets are shown in Table 4.2. Computation time of the reduced set includes 0.5 seconds to reduce the state-update set prior to iteration over precursor sets. This enables a 62% reduction in total computation time, 22% reduction in number of continuous generators, and 6% reduction in number of constraints.

Table 4.2: Backward reachability with nominal (row 1) vs. reduced (row 2) state-update set for the two-equilibrium system.

Case 2	Direction/Set	$n_{g,r}$	$n_{b,r}$	$n_{c,r}$	Time [s]
$s_u = 0\,, s_w = 0.05$	Backward: $\mathcal{R}_{-3}$	30032	1365	24570	50
	Backward: $\mathcal{R}_{-3}^{red}$	23207	1365	23207	19

4.2.2.3.2 Thermostat-Controlled Heated Room [60]

To demonstrate scalability, backward reachability is performed for a benchmark room heating example with 6 rooms and 2 heaters, previously studied for forward reachability as "Case (6,2)" of [22, Section 6.2]. This example extends the heated room scenario given in [90], where the thermostatic control and heat exchange among adjacent rooms is modeled as a hybrid system. The continuous temperature dynamic of the i^{th} room is modeled as

$$\dot{x}_i = c \cdot h_i + b_i(u - x_i) + \sum_{i \neq j} a_{ij}(x_j - x_i) \,, \tag{4.9}$$

where the heat transfer coefficient a_{ij} is 1 between adjacent rooms and 0 otherwise, the heat transfer coefficient between the room and the outside environment is $b_i = 0.08q$ where q is the number of exposed walls, the heating power is $c = 15$ with $h_i \in \{0, 1\}$ for rooms with heaters and $h_i = 0$ otherwise, and the outside temperature may take on any value within the interval $u \in [0, 0.1]$ [90]. Heaters located in select rooms are controlled by discrete-time thermostats that turn on when the sampled temperature in the room decreases below $22 - \varepsilon$ °C and turn off when it increases above $24 + \varepsilon$ °C where ε is machine precision. The closed-loop temperature dynamics of the building may be discretized for a time-step of 0.01 assuming a zero-order hold on u, and converted to a MLD system using HYSDEL [34]. This introduces one binary state, three auxiliary binary variables, and nine inequality constraints for each heater.

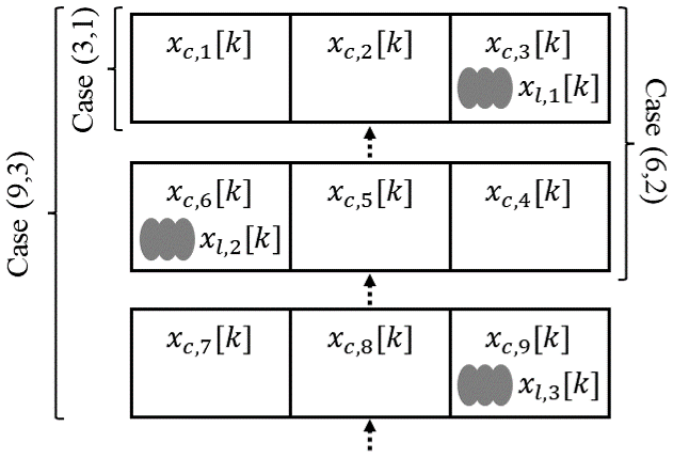


Figure 4.2: Room layout and heater locations for a varying number of rooms. [58]

The state-update set has complexity $n_{g,\phi} = 24,\ n_{b,\phi} = 8,\ n_{c,\phi} = 18$. Backward reachable sets, shown in Figure 4.3, are found from $\mathcal{R}_{50}$ (chosen to facilitate comparison with [22]), which has complexity $n_{g,50} = 68,\ n_{b,50} = 13,\ n_{c,50} = 62$. The 50-step backward

reachable set $\mathcal{R}_0$ is calculated in 0.34 seconds with complexity $n_{g,0} = 1268$, $n_{b,0} = 413$, $n_{c,r0} = 1362$.

An alternative to the proposed approach is to calculate the reachable set as a collection of convex sets, as done for forward reachability in [28] and backward reachability in [37]. This exhibits worst-case exponential growth in set complexity over time, while the proposed approach has linear complexity growth. At every time step, *each* convex set in the collection is propagated backward under the dynamics of *each* mode, and sets corresponding to inactive modes are eliminated. For comparison to the proposed approach, $\mathcal{R}_0$ is calculated using [37, Section IV] and implemented using constrained zonotopes [21]. This results in a collection of 1125 constrained zonotopes with 147,637 total generators and 28,387 total constraints and takes 202 seconds to compute. This is nearly 600 times longer than the proposed approach due to the large number of convex sets to be propagated and eliminated.

This example is revisited in Section 4.2.4.2.2, without the need to convert the dynamics to a MLD system.

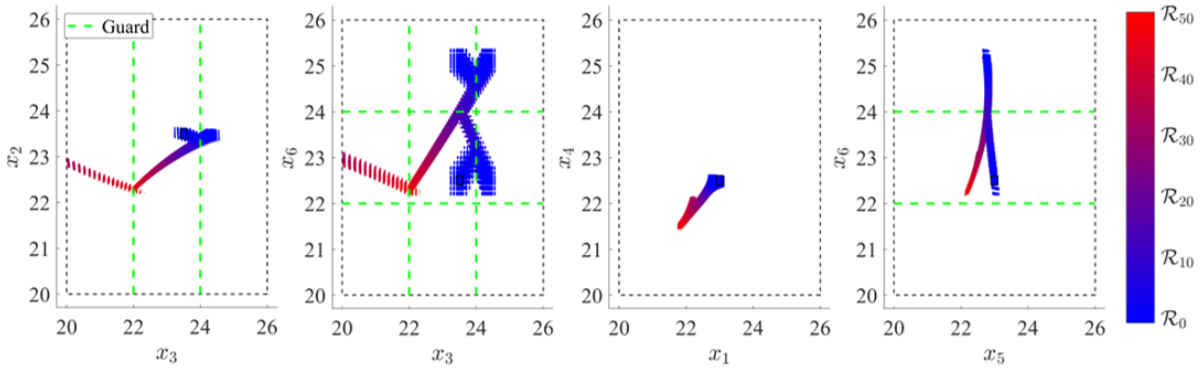


Figure 4.3: Projections of backward reachable sets calculated from $\mathcal{R}_{50}$ for 50 steps. Guards determining heater logic (green dashed lines) and the region over which the MLD is defined (black dashed lines) are also plotted.

4.2.3 Linear Systems with Model Predictive Controllers

4.2.3.1 Introduction

Consider the linear time invariant discrete-time system,

$$x_{k+1} = Ax_k + Bu_k \,, \tag{4.10}$$

where $x_k \in \mathbb{R}^n$ is the vector of states and $u_k \in \mathbb{R}^{n_u}$ is the vector of inputs at time k, and the linear dynamics are defined by $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n_u}$. Under MPC, the control input is determined by solving the optimization program

$$\begin{aligned} &\min_{u,x} x_N^\mathsf{T} Q_N x_N + \sum_{k=0}^{N-1} x_k^\mathsf{T} Q x_k + u_k^\mathsf{T} R u_k \\ &\text{s.t. } x_{k+1} = Ax_k + Bu_k \,,\; u_k \in \mathcal{U} \,,\; \forall\, k \in \{1, ..., N-1\} \,, \\ &\qquad x_k \in \mathcal{X} \,,\; \forall\, k \in \{1, ..., N-1\} \,,\; x_N \in \mathcal{X}_N \,, \end{aligned} \tag{4.11}$$

where $\mathcal{U}$, $\mathcal{X}$ and $\mathcal{X}_N$ are convex bounded polytopes and the initial state is given as x_0. States and inputs have weights $Q, Q_N \succeq 0$ and $R \succ 0$, respectively. For ease of readability the MPC formulation is provided as a regulator problem. However, it can be modified for other cases [91]. Explicit solution of (4.11) results in PWA control laws [31].

4.2.3.2 Proposed Method

It is possible to construct an MLD system equivalent to the closed-loop dynamics and then calculate a state-update set using the results of Section 4.2.2.2, though the number of critical regions can grow exponentially with the number of constraints and limits the process of generating explicit MPC control laws needed to construct the MLD system. Alternatively, we now describe a more direct method using results from [23], which defines a hybrid zonotope (implicit) representation of the parametric solution using the Karush-Kuhn-Tucker conditions of optimality. Given the system (4.10) and MPC formulation (4.11), consider the augmented system

$$\begin{aligned} &\hat{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & A \end{bmatrix} ,\; \hat{B} = \begin{bmatrix} \mathbf{0} \\ B \end{bmatrix} ,\; \hat{Q} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q \end{bmatrix} , \hat{Q}_N = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_N \end{bmatrix} , \\ &\hat{R} = R \,,\; \hat{X} = \mathcal{X} \times \mathcal{X} \,,\; \hat{X}_n = \mathcal{X} \times \mathcal{X}_n \,,\; \hat{\mathcal{U}} = \mathcal{U} \,. \end{aligned} \tag{4.12}$$

The augmented system (4.12) stacks a static system with the linear system of interest. Static states are not penalized and have no effect on the optimal inputs. Leveraging [23, Theorem 2] to find the one-step forward reachable set of (4.10) under closed-loop MPC from the initial set $\hat{\mathcal{X}}_0 = [\mathbf{0}\ \mathbf{I}]^T \mathcal{X}$ yields the state-update set with $\mathrm{D}_\Phi = \mathcal{X}$. This allows for the computation of forward and backward reachable sets of linear systems with MPC to be computed *without* solving optimization programs.

4.2.3.3 Numerical Example

4.2.3.3.1 Maximal Positive Invariant Set for a Double Integrator Under MPC [60]

The maximal positive invariant set $\mathcal{O}_\infty^{MPC}$ of a closed-loop system under MPC consists of all initial conditions that generate recursively feasible trajectories. We next exemplify how the proposed methods enable computation of a less conservative $\mathcal{O}_\infty^{MPC}$ as compared to conventional methods.

Consider the double integrator from [31, Example 12.1],

$$x[k+1] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k] \,,$$

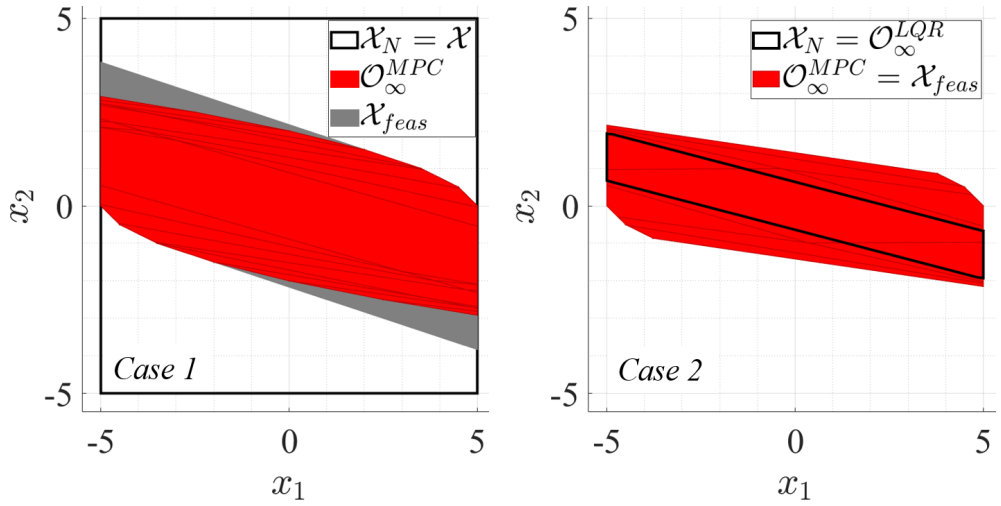
under MPC (4.11) with $P = Q = \mathbf{I}$, $R = 10$, and $N = 3$. Input and state trajectories are constrained by $u[k] \in \mathcal{U} = \frac{1}{2}\mathcal{B}_\infty^1$, $x[k] \in \mathcal{X} = 5\mathcal{B}_\infty^2$, $\forall k \in \{1, ..., N-1\}$. Two cases of terminal state constraints are considered.

Case 1: For this case, we set $\mathcal{X}_N = \mathcal{X}$. Algorithm 10.1 of [31] provides a general method for calculating $\mathcal{O}_\infty$ for autonomous dynamics using precursor and intersection calculations, but its application to hybrid systems was previously limited, in part, due to the lack of a scalable precursor set identity. However, this algorithm can be applied using hybrid zonotopes and the precursor set identity in Theorem 4.2 to calculate $\mathcal{O}_\infty^{MPC}$, as plotted in Figure 4.4(a). Also plotted is $\mathcal{X}_{feas}$, the set of states for which the optimization program has a feasible solution but not necessarily recursively feasible trajectories.

Case 2: Absent efficient methods to compute $\mathcal{O}_\infty^{MPC}$ under the PWA control laws, conventional approaches ensure recursive feasibility by artificially constraining $\mathcal{X}_N$ to a positively invariant set associated with a simpler control law. This results in $\mathcal{O}_\infty^{MPC} = \mathcal{X}_{feas}$ [31, Chapter 12.3.1]. A common choice is $\mathcal{X}_N = \mathcal{O}_\infty^{LQR}$, where $\mathcal{O}_\infty^{LQR}$ is the maximal linear quadratic regulator (LQR) invariant set [31, Definition 11.1]. This set and the resulting $\mathcal{X}_{feas}$ are shown in Figure 4.4(b).

Comparison of cases 1 and 2 in Figure 4.4 demonstrates that introducing the terminal constraint $\mathcal{X}_N = \mathcal{O}_\infty^{LQR}$ results in a smaller maximal region of recursive feasibility than when $\mathcal{X}_N = N$. Thus the proposed methods enable reduced conservatism in the control design by allowing $\mathcal{O}_\infty^{MPC}$ to be computed for a less restrictive terminal constraint. The terminal constraint in case 2 may also negatively affect performance [31, Remark 12.2].

Remark 4.1 (Convergence and Complexity Growth) *Using [31, Algorithm 10.1], directly will result in exponential complexity growth due to an intersection with the*



(a) Contribution (b) Conventional

Figure 4.4: Maximal positive invariant sets under MPC for a double integrator. (a) The maximal positive invariant set under MPC $\mathcal{O}_\infty^{MPC}$ is calculated using the scalable precursor set identity presented in this paper. (b) Conventional methods use an artificial terminal constraint to obtain $\mathcal{O}_\infty^{MPC}$, however this makes the set smaller.

previous iteration. This results in more than a doubling of memory complexity at each time step, and can quickly overwhelm memory limitations if the algorithm does not terminate in a relatively small number of time-steps. Complicating matters, the check to detect convergence is computationally expensive. It requires detecting whether two hybrid zonotopes are equal, which can be achieved by check set containment in both directions. Set containment of hybrid zonotopes requires solving a bi-level MILP. Methods aimed at overcoming the complexity growth and convergence detection challenges are now presented, though in general, complexity growth and convergence detection remain open challenges.

Method 1) By design, each iteration after initialization results in a positively invariant set, thus the algorithm could be iterated until computational resources are spent or convergence is detected, whichever comes first.

Method 2) To avoid exponential memory complexity growth with iterations, the intersection in [31, Algorithm 10.1] could be ignored, which results in linear memory complexity growth This eliminates the guarantee that each iteration is positively invariant. Nonetheless, if the algorithm converges and terminates, the maximal positive invariant set produced.

4.2.4 Discrete Hybrid Automata

4.2.4.1 Introduction

DHA are a class of discrete-time hybrid systems that combine four components, namely 1) an event generator (EG), 2) a finite state machine (FSM), 3) a mode selector (MS), and 4) a switched affine system (SAS). The interested reader is referred to [31, Chapter 16] for a detailed review of DHA.

Previous work by the author provided reachability methods for an equivalent class of systems called mixed-logical dynamical (MLD) systems [22, 60], thus a DHA could be converted to an MLD system using techniques and tools such as HYSDEL [34], and reachability analysis could be performed on an equivalent MLD system. This section presents an alternative approach that avoids an unnecessary conversion to an equivalent MLD system by exploiting the structure of DHA subsystems for functional decomposition.

The approach is demonstrated first by exposing the exploitable structure of DHA for functional decomposition using an example DHA adopted from [31, Example 16.6]. Then the method is used to generate an alternative but equivalent state-update set for the heated room example from Section 4.2.2.3.2.

4.2.4.2 Numerical Examples

4.2.4.2.1 Example DHA Functional Decomposition

Consider the DHA system [31, Modified from Example 16.6] consisting of a continuous state $x_k \in \mathbb{R}$, a continuous input $u_k \in \mathbb{R}$, a mode indicator signal $i_k \in \{1, 2, 3\}$, and event signals $\delta_1, \delta_2 \in \{0, 1\}$ given by

$$\text{SAS: } x_{k+1} = \begin{cases} x_k + u_k - 1\,, & \text{if } i_k = 1\,, \\ 2x_k\,, & \text{if } i_k = 2\,, \\ 2\,, & \text{if } i_k = 3\,, \end{cases}$$

$$\text{EG: } \begin{cases} \delta_1 = \begin{cases} 0\,, & \text{if } x_k \leq -\varepsilon\,, \\ 1\,, & \text{if } x_k \geq 0\,, \end{cases} \\ \delta_2 = \begin{cases} 0\,, & \text{if } x_k + u_k - 1 \leq -\varepsilon\,, \\ 1\,, & \text{if } x_k + u_k - 1 \geq 0\,, \end{cases} \end{cases}$$

$$\text{MS: } i_k = \begin{cases} 1\,, & \text{if } (\delta_1, \delta_2) = (0,0)\,, \\ 2\,, & \text{if } \delta_1 = 1\,, \\ 3\,, & \text{if } (\delta_1, \delta_2) = (0,1)\,, \end{cases}$$

where ε is machine precision. A functional decomposition is obtained as

$$\begin{aligned}
\text{Inputs: } & \begin{cases} w_1 \leftarrow x_k, \\ w_2 \leftarrow u_k \end{cases} \\
\text{EG: } & \begin{cases} w_3 = \begin{cases} 0\,, & \text{if } w_1 \leq -\varepsilon\,, \\ 1\,, & \text{if } w_1 \geq 0\,, \end{cases} \\ w_4 = w_1 + w_2 - 1 \\ w_5 = \begin{cases} 0\,, & \text{if } w_4 \leq -\varepsilon\,, \\ 1\,, & \text{if } w_4 \geq 0\,, \end{cases} \end{cases} \\
\text{MS: } & \begin{cases} w_6 = \begin{cases} 1\,, & \text{if } (w_3, w_5) = (0,0)\,, \\ 2\,, & \text{if } w_3 = 1\,, \\ 3\,, & \text{if } (w_3, w_5) = (0,1)\,, \end{cases} \end{cases} \\
\text{SAS: } & \begin{cases} w_7 = \begin{cases} 1\,, & \text{if } w_6 = 1\,, \\ 0\,, & \text{if } w_6 = 2 \vee w_6 = 3\,, \end{cases} \\ w_8 = \begin{cases} 1\,, & \text{if } w_6 = 2\,, \\ 0\,, & \text{if } w_6 = 1 \vee w_6 = 3\,, \end{cases} \\ w_9 = \begin{cases} 1\,, & \text{if } w_6 = 3\,, \\ 0\,, & \text{if } w_6 = 1 \vee w_6 = 2\,, \end{cases} \\ w_{10} = w_1 + w_2 - 1\,, \\ w_{11} = 2w_1\,, \\ w_{12} = 2\,, \\ w_{13} = w_7 w_{10}\,, \\ w_{14} = w_8 w_{11}\,, \\ w_{15} = w_9 w_{12}\,, \\ w_{16} = w_{13} + w_{14} + w_{15}\,. \leftarrow \text{Output} \end{cases}
\end{aligned}$$

The DHA is represented by a functional decomposition with only unary and binary functions with the exception of w_{16} which is affine. Using results from Chapter 3, a graph of the function for the dynamics, also called a state-update set, can be generated.

4.2.4.2.2 Thermostat-Controlled Heated Room Revisited

Consider again the heated room example from Section 4.2.2.3.2. The goal of this example is to demonstrate how a state-update set can be generated without the need for conversion to an MLD system, and to compare the resulting state-update set complexity. Additionally, this example explicitly demonstrates each step used to generate the state-update set using methods from Chapter 3.

The room temperature dynamics for Case (6,2) are equivalently written as

$$x_{k+1} = Ax_k + B_h h_k + B_u u_k \tag{4.13}$$

where $x \in \mathbb{R}^6$ are the room temperatures, $h \in \{0,1\}^2$ are the on/off modes for heaters in rooms 3 and 6, and $u \in \mathbb{R}^1$ is the outside temperature. The matrices A, B_h, and B_u can be found by discretizing the linear dynamics (4.9) with a time step of $\delta = 0.01$, assuming a zero-order hold on h and u, and are given by

$$A = \begin{bmatrix} 0.9787 & 0.0097 & 0.0000 & 0.0000 & 0.0001 & 0.0098 \\ 0.0097 & 0.9698 & 0.0097 & 0.0001 & 0.0097 & 0.0001 \\ 0.0000 & 0.0097 & 0.9787 & 0.0098 & 0.0001 & 0.0000 \\ 0.0000 & 0.0001 & 0.0098 & 0.9787 & 0.0097 & 0.0000 \\ 0.0001 & 0.0097 & 0.0001 & 0.0097 & 0.9698 & 0.0097 \\ 0.0098 & 0.0001 & 0.0000 & 0.0000 & 0.0097 & 0.9787 \end{bmatrix},$$

$$B_h = \begin{bmatrix} 0.0000 & 0.0007 \\ 0.0007 & 0.0000 \\ 0.1484 & 0.0000 \\ 0.0007 & 0.0000 \\ 0.0000 & 0.0007 \\ 0.0000 & 0.1484 \end{bmatrix}, \quad B_u = \begin{bmatrix} 0.1600 \\ 0.0800 \\ 0.1600 \\ 0.1600 \\ 0.0800 \\ 0.1600 \end{bmatrix}.$$

The heater state for each room with a heater depends on the room temperature and the state of the heater at the previous time step. The room heater logic for room 3 (heater 1)

and room 6 (heater 2) are given by

$$h_{1,k+1} = \begin{cases} 1\,, & \text{if } x_{3,k+1} \leq 22 - \varepsilon\,, \\ 0\,, & \text{if } x_{3,k+1} \geq 24 + \varepsilon\,, \\ h_{1,k}\,, & \text{if } 22 \leq x_{3,k+1} \leq 24\,, \end{cases} \tag{4.14}$$

$$h_{2,k+1} = \begin{cases} 1\,, & \text{if } x_{6,k+1} \leq 22 - \varepsilon\,, \\ 0\,, & \text{if } x_{6,k+1} \geq 24 + \varepsilon\,, \\ h_{2,k}\,, & \text{if } 22 \leq x_{6,k+1} \leq 24\,, \end{cases} \tag{4.15}$$

where ε is a small number. The graph of the function is visualized in Figure 4.5, and two methods to generate a hybrid zonotope representation are discussed.

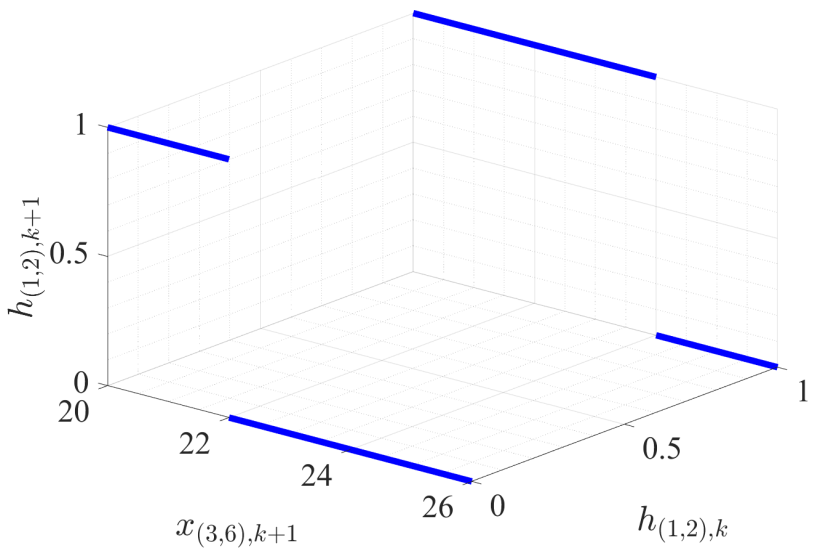


Figure 4.5: Graph of the heater logic function for (4.14) and (4.15)

Construct $\mathcal{H}_{heater}$ via Method 1: The graph of the function for the heater logic, denoted $\mathcal{H}_{heater}$, is generated using the vertex and incidence matrices

$$V_{heater} = \begin{bmatrix} 20 & 22-\varepsilon & 20 & 22-\varepsilon & 22 & 24 & 22 & 24 & 24+\varepsilon & 26 & 24+\varepsilon & 26 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.16}$$

$$M_{heater} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.17}$$

for temperatures between 20°C to 26°C. This method is visualized in Figure 4.6.

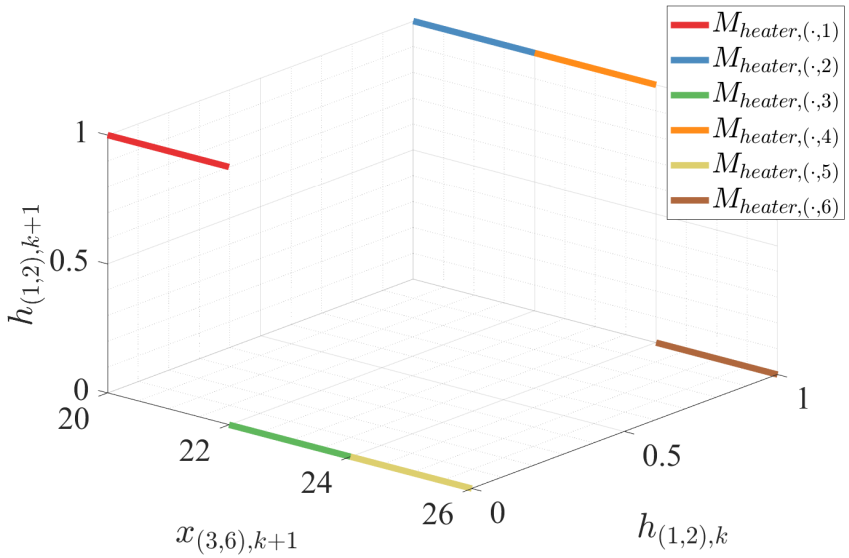


Figure 4.6: HCG-rep graph of the heater logic function for Method 1. The graph of the function is represented as collection of V-rep polytopes, converted to HCG using vertex matrix (4.16) incidence matrix (4.17), and Theorem 2.1. The segments are color-coded for each column of the incidence matrix (4.17). After application of order reduction methods, the resulting complexity is $n_g = 24$, $n_b = 6$, $n_c = 14$.

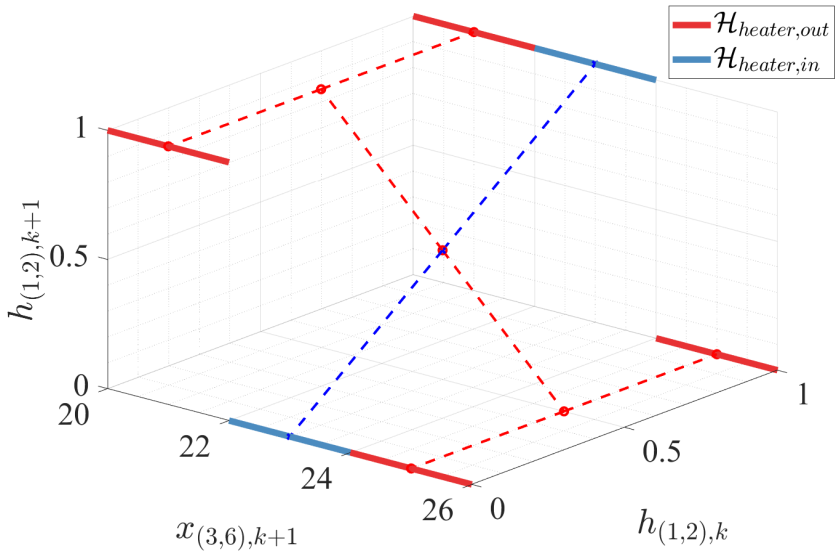
Construct $\mathcal{H}_{heater}$ via Method 2: Alternatively, the graph of the heater logic

function can be represented as the union of the two hybrid zonotopes

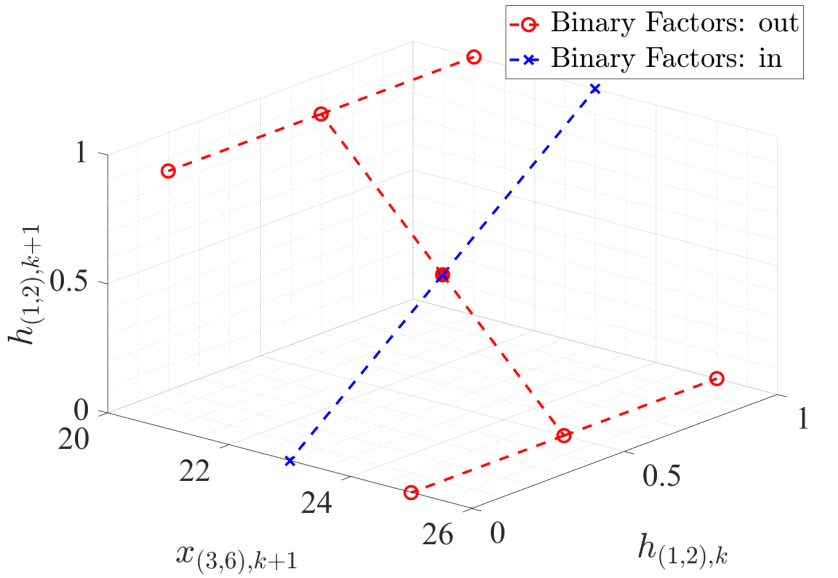
$$\mathcal{H}_{in} = \left\langle \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} 23 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \emptyset, \emptyset, \emptyset \right\rangle, \quad \text{and} \tag{4.18}$$

$$\mathcal{H}_{out} = \left\langle \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 2+\varepsilon \\ \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} 23 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \emptyset, \emptyset, \emptyset \right\rangle, \tag{4.19}$$

which are plotted in Figure 4.7. Both sets are equivalent, though constructing $\mathcal{H}$ using Method 2 results in lower memory complexity, i.e., (n_g, n_b, n_c) is $(24, 6, 14)_{M1}$ for Method 1 vs. $(7, 4, 5)_{M2}$ for Method 2.



(a)



(b)

Figure 4.7: HCG-rep graph of the heater logic function for Method 2. (a) The graph of the function is represented as the union of two hybrid zonotopes given by (4.18) and (4.19), converted to a single hybrid zonotope using Proposition 2.4. Combinations of the centers shifted by binary factors are plotted as dashed lines, and (b) shows these combinations by themselves, i.e., without plotting $\mathcal{H}_{heater}$. After application of order reduction methods, the resulting complexity is $n_g = 7$, $n_b = 4$, $n_c = 5$.

Functional Decomposition: A functional decomposition is given by

$$\begin{array}{c} \text{time step} \\ k \end{array} \begin{cases} w_1 &= x_{1,k} \,, \\ w_2 &= x_{2,k} \,, \\ w_3 &= x_{3,k} \,, \\ w_4 &= x_{4,k} \,, \\ w_5 &= x_{5,k} \,, \\ w_6 &= x_{6,k} \,, \\ w_7 &= h_{1,k} \,, \\ w_8 &= h_{2,k} \,, \\ w_9 &= u_k \,, \end{cases} = w_{1:9}$$

$$\begin{array}{c} \text{room temp.} \\ k+1 \end{array} \begin{cases} w_{10} = x_{1,k+1} \,, \\ w_{11} = x_{2,k+1} \,, \\ w_{12} = x_{3,k+1} \,, \\ w_{13} = x_{4,k+1} \,, \\ w_{14} = x_{5,k+1} \,, \\ w_{15} = x_{6,k+1} \,, \end{cases} = \begin{bmatrix} A & B_h & B_u \end{bmatrix} w_{1:9}$$

$$\begin{array}{c} \text{heater state} \\ k+1 \end{array} \begin{cases} w_{16}(w_{12}, w_7) = h_{1,k+1}(x_{3,k+1}, h_{1,k}) \,, \\ w_{17}(w_{15}, w_8) = h_{2,k+1}(x_{6,k+1}, h_{2,k}) \,. \end{cases}$$

Constructing the State-Update Set: Construction of graph of the functional decomposition of the heated room dynamics and reduction to a state-update set is as follows. The state-update set is built to include the room temperature states and the heater states, and encodes the dynamics from $(x, h)_k$ to $(x, h)_{k+1}$.

1. Initialize $\mathrm{D}_{\mathcal{H}} \leftarrow \begin{bmatrix} 20 & 26 \end{bmatrix}^6 \times \{0, 1\}^2 \times \begin{bmatrix} 0 & 0.1 \end{bmatrix}$ and set $\mathcal{H}_{1:9} = \mathrm{D}_{\mathcal{H}}$.

2. Encode the linear room temperature dynamics using Corollary 3.5,

$$\mathcal{H}_{1:15} = \begin{bmatrix} \mathbf{I}_6 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \\ A & B_h & B_u \end{bmatrix} \mathcal{H}_{1:9} \,.$$

3. Encode the dynamics of heater 1 using (3.30) from Theorem 3.4,

$$\mathcal{H}_{1:16} = (\mathcal{H}_{1:15} \times \mathrm{D}_{16}) \cap \begin{bmatrix} e_{12} \\ e_7 \\ e_{16} \end{bmatrix} \mathcal{H}_{heater} \,,$$

where $\mathrm{D}_{16} = \{0, 1\}$ is the possible states for $h_{1,k+1}$.

4. Encode the dynamics of heater 2 using (3.30) from Theorem 3.4,

$$\mathcal{H}_{1:17} = (\mathcal{H}_{1:16} \times \mathrm{D}_{17}) \cap \begin{bmatrix} e_{15} \\ e_8 \\ e_{17} \end{bmatrix} \mathcal{H}_{heater} \,,$$

where $\mathrm{D}_{17} = \{0, 1\}$ is the possible states for $h_{2,k+1}$.

5. By Corollary 3.6,

$$\Phi = \Big[(e_1, e_2, ...e_8, e_{10}, e_{11}, ..., e_{17})\Big] \mathcal{H}_{1:17} \,. \tag{4.20}$$

Generation of Φ in this section did not require the conversion to an MLD system and results in an equivalent state-update set to the state-update set generated using (4.8), and can be used to generate equivalent reachable sets. This can be seen by comparing reachable sets in Figure 4.8 with [22, Figure 4]. The complexity of the state-update set built with $\mathcal{H}$ constructed using Method 1 is $(n_g, n_b, n_c)_{\mathrm{M1}} = (47, 11, 25)$ and using Method 2 is $(n_g, n_b, n_c)_{\mathrm{M2}} = (21, 9, 13)$. For comparison, the complexity of the state-update set constructed by converting to an equivalent MLD system in Section 4.2.2.3.2 was $(n_g, n_b, n_c)_{\mathrm{MLD}} = (24, 8, 18)$; thus, the state-update set can be constructed with comparable memory complexity using functional decomposition.

4.3 Logical Systems

4.3.1 Introduction

Logical system dynamics with inputs are written as

$$x_{k+1} = f(x_k, u_k) \,,$$

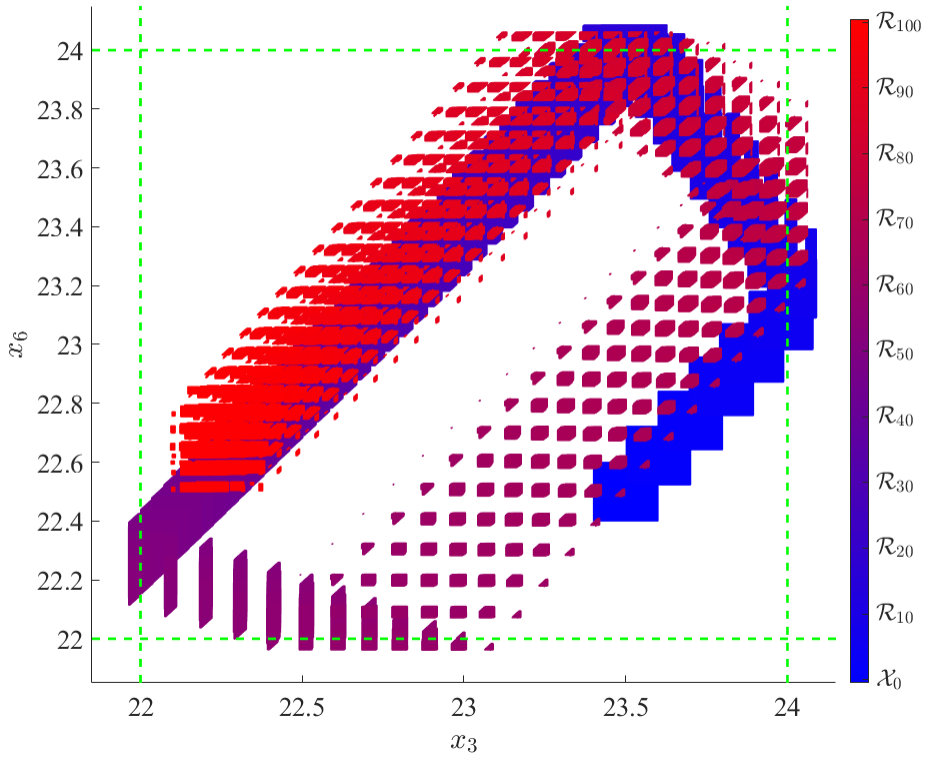


Figure 4.8: Forward reachable sets of heated room example case (6,2) using state-update set calculated using functional decomposition. See [22, Figure 4] for comparison.

where $f : \mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \to \mathbb{B}^{nx}$, $x_k \in \mathbb{B}^{n_x} = \{0,1\}^{n_x}$, $u_k \in \mathbb{B}^{n_u} = \{0,1\}^{n_u}$, and $f(\cdot)$ is exclusively composed of logical functions, e.g., OR $\vee$, XNOR $\odot$, AND $\wedge$, and NAND $\barwedge$. Logical systems are a subset of hybrid systems as they do not include continuous dynamics. Logical zonotopes [19] and polynomial logical zonotopes [20] are novel set representations for which identities have been developed for reachability analysis of logical systems. In this section, it is demonstrated that hybrid zonotopes are capable of exact and efficient reachability analysis of logical system, using the same methods developed for hybrid systems.

4.3.2 Proposed Method

This section demonstrates the ability of hybrid systems to represent Boolean sets (sets with elements that exclusively exist in $\{0,1\}$) and graphs of logical functions. Consider

the AND $\wedge$ function

$$b_1 \wedge b_2 = \begin{cases} 0\,, & \text{if } (b_1, b_2) = (0,0)\,, \\ 0\,, & \text{if } (b_1, b_2) = (1,0)\,, \\ 0\,, & \text{if } (b_1, b_2) = (0,1)\,, \\ 1\,, & \text{if } (b_1, b_2) = (1,1)\,. \end{cases}$$

The graph of the AND $\wedge$ function consists of four points, which can be represented as the union of sets consisting of each point, i.e.,

$$\Phi_\wedge = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\} \tag{4.21}$$

$$= \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} \cup \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\} \cup \left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\} \cup \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}. \tag{4.22}$$

It is apparent that hybrid zonotopes can represent (4.22). This is because each singleton can be represented as a hybrid zonotope with no continuous factors, binary factors, or constraints by assigning the center of each hybrid zonotope to the value of the corresponding singleton. Iterative unions using Proposition 2.4 result in the desired set represented as a single hybrid zonotope. Alternatively, one can observe that the set is the union of four V-rep polytopes, each with a single vertex, thus a hybrid zonotope representation can be generated using Theorem 2.1 with vertex and incidence matrices given by

$$V = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

resulting in the hybrid zonotope $\Phi_\wedge = \langle G_{c,\wedge}, G_{b,\wedge}, c_\wedge, A_{c,\wedge}, A_{b,\wedge}, b_\wedge \rangle$ where

$$G_{c,\wedge} = \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_{b,\wedge} = \mathbf{0}_{3\times 5}\,, \quad c_\wedge = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \end{bmatrix}, \quad b_\wedge = -\mathbf{1}_{6\times 1}\,,$$

$$A_{c,\wedge} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad A_{b,\wedge} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} \end{bmatrix}.$$

Due to the large number of generators and constraints, it is challenging to see how these matrices result in the desired set, though using order reduction methods from [22, 58], the state-update set can be represented equivalently by

$$\Phi_{\wedge,red} = \langle G_{c,\wedge,red}, G_{b,\wedge,red}, c_{\wedge,red}, A_{c,\wedge,red}, A_{b,\wedge,red}, b_{\wedge,red} \rangle ,$$
$$G_{c,\wedge,red} = \emptyset , \quad G_{b,\wedge,red} = \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}, \quad c_{\wedge,red} = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \end{bmatrix},$$
$$A_{c,\wedge,red} = \emptyset , \quad A_{b,\wedge,red} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \quad b_{\wedge,red} = -1 .$$

The single linear equality constraint

$$\frac{1}{2}\xi^{b,1} + \frac{1}{2}\xi^{b,2} + \frac{1}{2}\xi^{b,3} + \frac{1}{2}\xi^{b,4} = -1 ,$$

coupled with $\xi^{b,i} \in \{-1, 1\}$, $\forall\ i \in \{1, 4\}$ enforces that exactly one of the four binary factors equals one, i.e., $\xi^{b,i} = 1$ and $\forall j \neq i,\ \xi^{b,j} = -1$. There are four combinations of binary factors that satisfy the constraints, given by the columns of

$$B = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}.$$

By considering each column of B, the four points in (4.22) are achieved. For exampled, consider $B_{(\cdot,1)}$, thus

$$x = \begin{bmatrix} 1 \\ 1 \\ \frac{1}{2} \end{bmatrix} + \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} .$$

In fact, all 16 binary Boolean functions can be represented exactly using the same process (see Section 3.4.6) and all have reduced complexities of 0 continuous factors, 4 binary factors, and 1 constraint. Because logical functions are defined as the composition of unary and binary logical functions, there will always exist a functional decomposition consisting of unary and binary logical functions. Therefor hybrid zonotopes can represent the graph of logical functions exactly.

4.3.3 Numerical Example: High Dimensional Boolean Function

We adopt the following example from [20]. In this example, notation is abused to accommodate the higher dimensional state and input space. It is assumed that operators act element-wise on the vectors and elements of the functional decomposition are allowed to be vectors. Consider the Boolean function with $x_i, u_i \in \{0,1\}^{10}$, $i \in \{1,2,3\}$

$$\begin{aligned} x_{1,k+1} &= u_{1,k} \vee (x_{2,k} \odot x_{1,k}) \,, \\ x_{2,k+1} &= x_{2,k} \odot (x_{1,k} \wedge u_{2,k}) \,, \\ x_{3,k+1} &= x_{3,k} \barwedge (u_{2,k} \odot u_{3,k}) \,. \end{aligned}$$

A functional decomposition is given by

$$\begin{aligned} \text{Inputs:} \quad & \begin{cases} w_1 = x_{1,k} \,, \\ w_2 = x_{2,k} \,, \\ w_3 = x_{3,k} \,, \\ w_4 = u_{1,k} \,, \\ w_5 = u_{2,k} \,, \\ w_6 = u_{3,k} \,, \end{cases} \\ \text{Intermediate:} \quad & \begin{cases} w_7 = w_2 \odot w_1 = x_{2,k} \odot x_{1,k} \,, \\ w_8 = w_1 \wedge w_5 = x_{1,k} \wedge u_{2,k} \,, \\ w_9 = w_5 \odot w_6 = u_{2,k} \odot u_{3,k} \,, \end{cases} \\ \text{Outputs:} \quad & \begin{cases} w_{10} = w_4 \vee w_7 = u_{1,k} \vee (x_{2,k} \odot x_{1,k}) \,, \\ w_{11} = w_2 \odot w_8 = x_{2,k} \odot (x_{1,k} \wedge u_{2,k}) \,, \\ w_{12} = w_3 \barwedge w_9 = x_{3,k} \barwedge (u_{2,k} \odot u_{3,k}) \,. \end{cases} \end{aligned}$$

A hybrid zonotope graph of the functional decomposition is constructed using methods

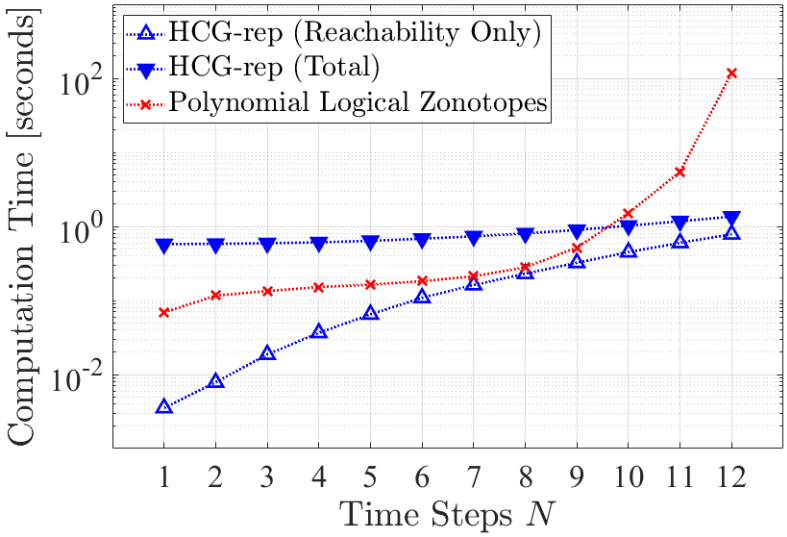


Figure 4.9: Computation time of reachable sets of a logical function. Two lines are plotted for HCG-rep, one including the time to generate the state-update set (Total) and one that only included the computation time associated with the generation of reachable sets using (3.3). The latter better demonstrates the scaling of the computation time with N, as it does not include the one-time computation time cost of generating Φ. Computation times for polynomial logical zonotopes are also shown.

from Chapter 3, from which the state-update set is generated using Corollary 3.6. Generating Φ in HCG-rep, including order reduction, took slightly less than 0.6 seconds.

An initial set for ($x_{1,k}$,$x_{2,k}$,$x_{3,k}$) consists of 8 possible values. Input sets for ($u_{1,k}$, $u_{2,k}$, $u_{3,k}$) also consist of 8 possible values. Reachable sets are calculated for 1-12 time steps. Computation times are plotted in Figure 4.9 comparing the methods presented here using HCG-rep to those developed for polynomial logical zonotopes. Computation time using the methods developed for polynomial logical zonotopes grow rapidly for time steps $N > 8$. Computation time using hybrid zonotopes appears to scale polynomially, as expected, and is significantly faster to compute than polynomial logical zonotopes for $N = \{10, 11, 12\}$.

4.4 Neural Network Control Systems

4.4.1 Introduction

This section addresses systems with neural network controllers using Rectified Linear Unit (ReLU) activation function and focuses on comparisons to the four state-of-the-art

reachability tools included in the Artificial Intelligence and Neural Network Control Systems category of the Applied veRification for Continuous and Hybrid systems (ARCH) competitions in 2022 [43] and 2023 [44]. Benchmark problems from [43, 44] are used to demonstrate the proposed methods.

COntinuous Reachability Analyzer (CORA) uses polynomial zonotopes to approximate the input-output relationship of activation functions and nonlinear dynamics [45,46]. Over-approximated reachable sets are calculated by efficient mappings via the over-approximated activation functions of the neural network and nonlinear functions associated with the plant dynamics. JuliaReach utilizes methods to efficiently convert structured zonotopes to and from Taylor models, and then leverages existing reachability tools using structured zonotopes for the neural network and Taylor Models for the plant, to efficiently construct reachable sets of the closed-loop system [47]. The Neural Network Verification (NNV) tool uses star sets for efficient computation of exact or approximated reachable sets through neural networks, and utilizes CORA for nonlinear plant dynamics. The POLynomial ARithmetic-base (POLAR) framework computes functional over-approximations of the flowmap. POLAR uses univariate Bernstein polynomial abstractions for activation functions to address non-differentiable ReLU function.

In most cases, each of these state-of-the-art tools performs well for the benchmark examples in [43, 44], though it is noted that the initial sets for the benchmark problems are relatively small. Large initial sets pose computational challenges as stated in a review of set propagation techniques in [12], which includes techniques used by these tools: *"Several challenging research problems remain to be addressed in the field, such as handling large initial sets for nonlinear systems and many guard intersections in hybrid systems. Both aspects are especially relevant when verifying systems involving neural networks."* To avoid these challenges, large initial sets may be partitioned, though for systems beyond a few dimensions the partitioning itself introduces worst-case exponential complexity.

While hybrid zonotopes have previously been used for reachability analysis of neural networks [92, 93], the methods proposed here achieve improved scalability and go beyond the scope of these previous works by analyzing the coupling of a neural network controller to a nonlinear plant. Parallel work presents methods to over-approximate nonlinear graphs of functions using hybrid zonotope representations of graphs of nonlinear functions [94], which can be coupled with exact representations of the graph of the neural network. Those methods are very similar to those proposed in Chapter 3, many of which were previously presented in [62].

4.4.2 Proposed Method

In this section we present methods based on the combination of graphs of functions and hybrid zonotopes for reachability analysis of nonlinear systems in closed-loop with neural network controllers. It is shown how ReLU neural networks have an exploitable structure for functional decomposition, and how graphs of the plant dynamics and controller can be generated separately and then combined to produce a closed-loop state-update set.

4.4.2.1 Dynamics

Consider a class of discrete-time nonlinear dynamics given by

$$x_{k+1} = f(x_k, u_k)\,, \tag{4.23}$$

where $f : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$, with state and input constraint sets given by $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$. The i^{th} row of $f(x_k, u_k)$ is a scalar-valued function and denoted by $f_i(x_k, u_k)$. Disturbances are omitted for simplicity of exposition, although the results here are easily extendable to systems with disturbances. Because hybrid zonotopes are inherently bounded, the following assumption regarding the dynamics is made.[1]

Assumption 4.2 (Boundedness) *For all* $(x, u) \in \mathcal{X} \times \mathcal{U}$, $||f(x, u)|| < \infty$.

The successor set is defined as follows.

Definition 4.6 (Successor Set) **[95, Definition 2]** *The* successor set *from* $\mathcal{R}_k \subseteq \mathcal{X}$ *with inputs bounded by* $\mathcal{U}_k \subseteq \mathcal{U}$ *is given by*

$$\text{Suc}(\mathcal{R}_k, \mathcal{U}_k) \equiv \left\{ f(x, u) \mid x \in \mathcal{R}_k,\ u \in \mathcal{U}_k \right\}\,. \tag{4.24}$$

The k^{th} forward reachable set, $\mathcal{R}_k$, from an initial set $\mathcal{R}_0$ can be found by k recursions of successor sets (4.24), i.e., $\mathcal{R}_{k+1} = \text{Suc}(\mathcal{R}_k, \mathcal{U}_k)$.

4.4.2.2 Open-Loop and Closed-Loop State-Update Sets

This section first introduces the open-loop state-update set (Definition 4.7), which encodes all possible state transitions of (4.23) over a user-specified domain of states and inputs, and is used to calculate successor sets over discrete time steps (Theorem 4.4). Then,

[1] See Corollary 3.3.

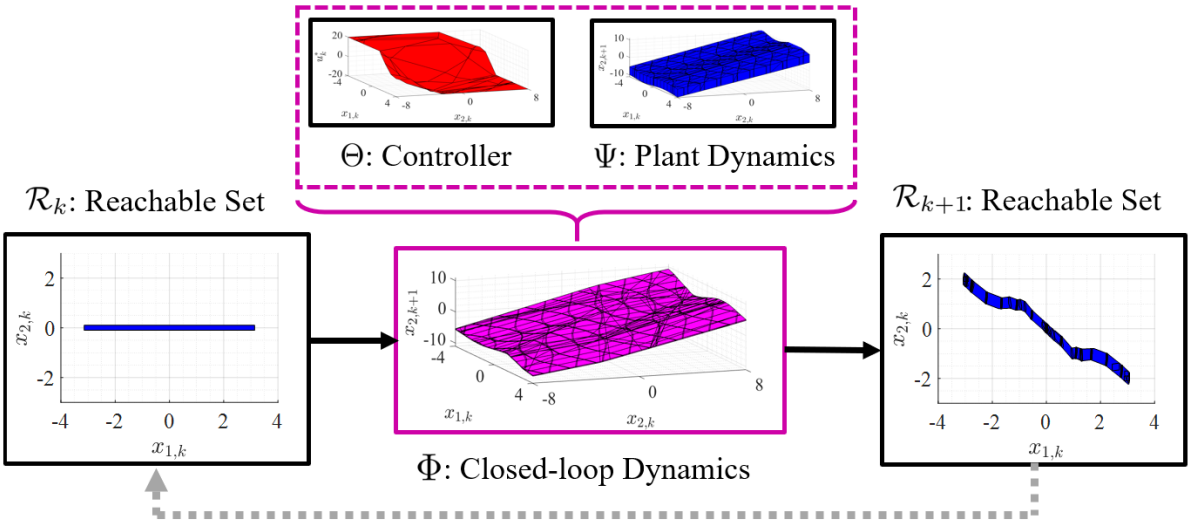


Figure 4.10: The closed-loop successor set identity uses a set-based representation of the closed-loop dynamics, called the closed-loop state-update set Φ, to generate the one-step forward reachable set $\mathcal{R}_{k+1}$ from $\mathcal{R}_k$. The closed-loop state-update set is created by combining sets representing the open-loop dynamics and a state-feedback controller, called the open-loop state-update set Ψ and the state-input map Θ, respectively.

after defining a state-input map as all possible inputs of a given control law over a user-specified domain of states (Definition 4.8), the set of possible state transitions of the closed-loop system is constructed by combining the state-input map and the open-loop state-update set (Theorem 4.5). It is shown how this closed-loop state-update set can be used to calculate successor sets of the closed-loop system (Theorem 4.6). This process is depicted in Figure 4.10.

Remark 4.2 *Note that the open-loop state-update set, state-input map, and closed-loop state-update set are graphs of the functions describing the open-loop dynamics, state-feedback control law, and closed-loop dynamics, respectively. The identities to generate successor sets correspond to that given in Theorem 3.1. The combination of an open-loop state-update set and a state-input maps to produce a closed-loop state-update set is a special case of how graphs of functions are constructed via* (3.30) *and* (3.33).

Definition 4.7 (Open-Loop State-Update Set) **[95, Definition 4]** *The open-loop state-update set* $\Psi \subseteq \mathbb{R}^{2n+n_u}$ *is defined as*

$$\Psi = \left\{ \begin{bmatrix} x_k \\ u \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{c} x_{k+1} \in \mathrm{Suc}(\{x_k\}, \{u\}), \\ (x_k, u) \in \mathrm{D}_\Psi \end{array} \right\}. \tag{4.25}$$

We refer to $\mathrm{D}_\Psi \subset \mathbb{R}^{n+n_u}$ as the *domain set* of Ψ, typically chosen as the region of interest for analysis, and $\mathrm{R}_\Psi = [\mathbf{0}\ \mathbf{I}_n]\Psi \subset \mathbb{R}^n$ as the *range set* of Ψ.

Theorem 4.4 (Open-Loop Successor Set Identity) [95, Theorem 1] *Given a set of states $\mathcal{R}_k \subseteq \mathbb{R}^n$, a set of inputs $\mathcal{U}_k \subseteq \mathbb{R}^{n_u}$, and an open-loop state-update set Ψ, if $\mathcal{R}_k \times \mathcal{U}_k \subseteq \mathrm{D}_\Psi$, then the open-loop successor set is given by*

$$\mathrm{Suc}(\mathcal{R}_k, \mathcal{U}_k) = \begin{bmatrix} 0 & \boldsymbol{I}_n \end{bmatrix} \left(\Psi \cap_{[\boldsymbol{I}_{n+n_u}\ \mathbf{0}]} (\mathcal{R}_k \times \mathcal{U}_k) \right). \tag{4.26}$$

Proof *By definition of the generalized intersection,*

$$\Psi \cap_{[\boldsymbol{I}_{n+n_u}\ \mathbf{0}]} (\mathcal{R}_k \times \mathcal{U}_k) = \left\{ \begin{bmatrix} x_k \\ u \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{l} x_{k+1} \in \mathrm{Suc}(\{x_k\}, \{u\}), \\ \begin{bmatrix} x_k \\ u \end{bmatrix} \in \mathrm{D}_\Psi \cap (\mathcal{R}_k \times \mathcal{U}_k) \end{array} \right\}.$$

If $\mathcal{R}_k \times \mathcal{U}_k \subseteq \mathrm{D}_\Psi$, then $\mathrm{D}_\Psi \cap (\mathcal{R}_k \times \mathcal{U}_k) = \mathcal{R}_k \times \mathcal{U}_k$, and (4.26) gives $\{x_{k+1} | x_{k+1} \in \mathrm{Suc}(\{x_k\}, \{u\}),\ x_k \in \mathcal{R}_k,\ u \in \mathcal{U}_k\}$. □

The containment condition in Theorem 4.4, $\mathcal{R}_k \times \mathcal{U}_k \subseteq \mathrm{D}_\Psi$, is not restrictive as modeled dynamics are often only valid over some region of states and inputs, which the user may specify as $\mathrm{D}_\Psi = \mathcal{X} \times \mathcal{U}$.

Consider a set-valued function $\mathcal{C}(x_k)$ corresponding to a state-feedback controller, such that $\mathcal{C}(x_k)$ is the set of all possible inputs that the controller may provide given the current state, x_k. For example, for a linear feedback control law given by $u(x_k) = Kx_k$ with no actuator uncertainty, $\mathcal{C}(x_k) = \{Kx_k\}$ would be a single vector. In the case of a linear feedback control law with actuator uncertainty given by $u = Kx_k + \delta_u$ where $\delta_u \in \Delta_u$, we would have $\mathcal{C}(x_k) = \{Kx_k + \delta_u \mid \delta_u \in \Delta_u\}$. The *state-input map* encodes the feedback control law given by $\mathcal{C}(x_k)$ as a set over a domain of states.

Definition 4.8 (State-Input Map) [95, Definition 5] *The state-input map is defined as $\Theta = \{(x_k, u) \mid u \in \mathcal{C}(x_k),\ x_k \in \mathrm{D}_\Theta\}$, where D_Θ is the* domain set *of Θ.*

Next, the closed-loop state-update set under a controller given by $\mathcal{C}(x_k)$ is defined. Then it will be shown how to construct a closed-loop state-update set given an open-loop state-update set and a state-input map.

Definition 4.9 (Closed-Loop State-Update Set) [95, Definition 6] *The closed-loop state-update set $\Phi \subseteq \mathbb{R}^{2n}$ for a controller given by $\mathcal{C}(x_k)$ is defined as*

$$\Phi = \left\{ \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{c} x_{k+1} \in \mathrm{Suc}\left(\{x_k\}, \mathcal{C}(x_k)\right), \\ x_k \in \mathrm{D}_\Phi \end{array} \right\}, \tag{4.27}$$

where $\mathrm{D}_\Phi \subset \mathbb{R}^n$ is the domain set of Φ.

Theorem 4.5 (Closed-Loop State-Update Set Construction) [95, Theorem 2] *Given an open-loop state-update set Ψ and state-input map Θ, the closed-loop state-update set Φ with $D_\Phi = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0} \end{bmatrix}(\mathrm{D}_\Psi \cap \Theta)$ is given by*

$$\Phi = \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_n \end{bmatrix} \left(\Psi \cap_{\begin{bmatrix} \boldsymbol{I}_{n+n_u} & \mathbf{0} \end{bmatrix}} \Theta \right). \tag{4.28}$$

Proof *By definition of the generalized intersection,*

$$\Psi \cap_{\begin{bmatrix} \boldsymbol{I}_{n+n_u} & \mathbf{0} \end{bmatrix}} \Theta = \left\{ \begin{bmatrix} x_k \\ u \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{c} x_{k+1} \in \mathrm{Suc}(\{x_k\}, \{u\}), \\ (x_k, u) \in \mathrm{D}_\Psi \cap \Theta, \\ u \in \mathcal{C}(x_k) \end{array} \right\}.$$

Thus the right side of (4.28) *equals*

$$\left\{ \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} \;\middle|\; \begin{array}{c} x_{k+1} \in \mathrm{Suc}\left(\{x_k\}, \mathcal{C}(x_k)\right), \\ x_k \in [\boldsymbol{I}_n \;\; \mathbf{0}]\big(\mathrm{D}_\Psi \cap \Theta\big) \end{array} \right\}.$$

Comparison to Def. 4.9 completes the proof. □

Theorem 4.6 provides an identity for the successor set of a closed-loop system with the feedback control law described by the set-valued function $\mathcal{C}(x_k)$. For closed-loop successor sets, the input set argument $\mathcal{U}_k$ is omitted and the successor set is instead denoted by $\mathrm{Suc}(\mathcal{R}_k, \mathcal{C})$.

Theorem 4.6 (Closed-Loop Successor Set Identity) [95, Theorem 3] *Given a set of states $\mathcal{R}_k \subseteq \mathbb{R}^n$ and closed-loop state-update set Φ, if $\mathcal{R}_k \subseteq \mathrm{D}_\Phi$ then the closed-loop successor set is given by*

$$\mathrm{Suc}(\mathcal{R}_k, \mathcal{C}) = \begin{bmatrix} 0 & \boldsymbol{I}_n \end{bmatrix} \left(\Phi \cap_{[\boldsymbol{I}_n \;\; \mathbf{0}]} \mathcal{R}_k \right). \tag{4.29}$$

The identities in (4.26), (4.28), and (4.29) utilize the open-loop state-update set and state-input map. Over-approximated open-loop state-update sets for nonlinear plant dynamics can be generated using methods from Chapter 3. The following section demonstrates how neural network controllers have an exploitable structure for functional decompositions, which can then be used in conjunction with methods from Chapter 3 to generate the state-input map.

4.4.2.3 Functional Decomposition of Neural Networks

Feed-Forward Neural Networks (FFNN) with unary activation functions have a structure that is conducive to a functional decomposition. Consider a fully connected neural network with n_x inputs and n_y outputs. Without loss of generality and for simplicity of exposition, it is assumed that there are two hidden layers, each with k_N nodes that utilize an activation function $\phi(\cdot)$ and weights and biases for the i^{th} layer are given by W_i, b_i. A functional decomposition is readily given by

$$\begin{aligned} \text{Inputs: } & w_1 \,, \\ & w_2 = W_1 w_1 + b_1 \,, \\ & w_3 = \phi(w_2) \,, \\ & w_4 = W_2 w_3 + b_2 \,, \\ & w_5 = \phi(w_4) \,, \\ & w_6 = W_3 w_5 + b_3 \,, \end{aligned} \tag{4.30}$$

where w_6 corresponds to the n_y outputs of the FFNN. For ease of reading, (4.30) abuses notation as it does not assign an independent index to each scalar observable, i.e., $w_1 \in \mathbb{R}^{n_x}$, $w_2, w_3, w_4, w_5 \in \mathbb{R}^{k_N}$, and $w_6 \in \mathbb{R}^{n_y}$. Additionally, it is assumed that $\phi(\cdot)$ acts element-wise when given an argument of a vector, e.g., $\phi(w_3) = \begin{bmatrix} \phi(w_{(3,1)}) & \cdots & \phi(w_{(3,k_N)}) \end{bmatrix}^T$ where $w_{3,j}$ represent the j^{th} element of w_3.

The resulting memory complexity for constructing the graph of a FFNN as a hybrid zonotope is only dependent on the total number of nodes with nonlinear activation functions, k_{tot} and the complexity of the graph of the unary activation function ($n_{g,\phi}$,$n_{b,\phi}$,$n_{c,\phi}$), and is given by

$$\begin{aligned} n_{g,NN} &= n + k_{tot}(n_{g,\phi} + 1) \,, \\ n_{b,NN} &= k_{tot} n_{b,\phi} \,, \end{aligned}$$

$$n_{c,NN} = k_{tot}(n_{c,\phi} + 2) \, .$$

Assuming a ReLU activation function is used and its graph is produced using the method from Section 3.4.4, an exact graph of the neural network is given by

$$\begin{aligned} n_{g,NN} &= n + 5k_{tot} \, , \\ n_{b,NN} &= k_{tot} \, , \\ n_{c,NN} &= 2k_{tot} + 4 \, . \end{aligned}$$

4.4.3 Numerical Examples

This section presents two numerical examples,

1. Single Pendulum with Neural Network Controller (Adapted from ARCH-COMP), and

2. Vertical Collision Avoidance System (Adapted from ARCH-COMP).

4.4.3.1 Single Pendulum with Neural Network Controller

Two cases of parameters and controllers are presented. The latter correspond to those used in ARCH-COMP, and are discussed in Section 4.4.3.1.2 with comparison to state-of-the-art reachability tools. The former, presented in Section 4.4.3.1.1, is uses a less complex neural network than that used in ARCH-COMP, so that state-update sets and state-input maps can be visualized.

4.4.3.1.1 Case 1:

Consider the dynamics of an inverted pendulum given by (3.26) with gravity $g = 10$, length $l = 1$, mass $m = 1$, and moment of inertia $I = ml^2 = 1$. The continuous-time nonlinear dynamics are discretized with time step $h = 0.1$ using a 2^{nd}-order Taylor polynomial $\mathcal{T}_2(x_k)$, given by (3.27) and over-approximated as

$$x_{k+1} \in \mathcal{T}_2(x_k) \oplus \mathcal{L} \, , \tag{4.31}$$

where $\mathcal{L}$ is constructed using the Taylor inequality to bound the error due to truncating higher-order terms. The input torque is controlled in discrete time with a zero-order hold and bounded as $u_k \in [-20, 20] \, , \; \forall k$.

Table 4.3: Memory complexity of the open-loop state-update set, state-input map, and closed-loop state-update set.

Set	n_g	n_b	n_c
$\bar{\Psi}$	113	45	63
Θ	76	20	56
$\bar{\Phi}$	184	63	117

A functional decomposition of $\mathcal{T}_2(x_k)$ is shown in Table 3.2. For a chosen domain $(\mathrm{D}_1, \mathrm{D}_2, \mathrm{D}_3)$, bounds on the intermediate and output variables can be found by domain propagation via interval arithmetic. This choice of the domain also yields $\mathcal{L}$ in (4.31) as $\mathcal{L} = \left[-0.02, 0.02\right] \times \left[-0.26, 0.26\right]$. The sets $\bar{\mathcal{H}}_\ell\,,\ \forall \ell \in \{4,5,6\}$ are constructed using methods from Section 3.2. The set $\bar{\mathcal{H}}$ constructed using methods from Section 3.2 with a Minkowski summation to account for the Taylor remainder $\mathcal{L}$ yields an over-approximation of the open-loop state-update set,

$$\Psi \subset \bar{\Psi} = \bar{\mathcal{H}} \oplus \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathcal{L}\,. \tag{4.32}$$

A projection of $\bar{\Psi}$ is shown in Figure 4.11(a). The thickness of the set in the $x_{2,k}$ dimension is primarily due to the open-loop state-update set capturing variability in the input $u_k \in [-20, 20]$, though some of this is also a result of the Taylor remainder $\mathcal{L}$ and over-approximation of nonlinear functions, with $\bar{\mathcal{H}}_\ell \supset \mathcal{H}_\ell\,,\ \forall \ell \in \{4,5,6\}$.

Computing $\bar{\mathcal{H}}_\ell\,,\ \forall \ell \in \{4,5,6\}$ required 0.02 seconds, and application of exact complexity reduction techniques from [22, 58] required an additional 2 seconds. From this, $\bar{\Psi}$ was computed in 8 milliseconds using Theorem 3.4 and Corollary 3.5.

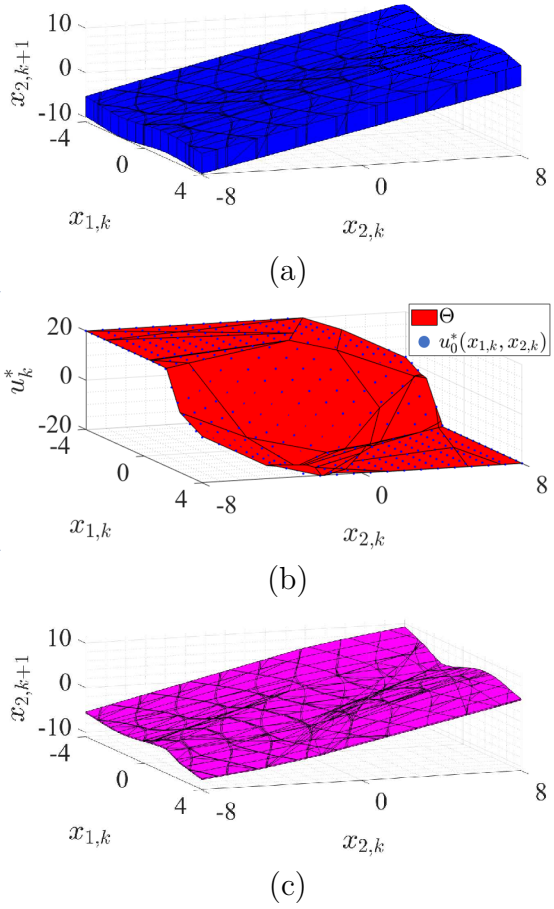


Figure 4.11: (a) Projection of over-approximated open-loop state-update set $\bar{\Psi}$ bounding dynamics of a pendulum at discrete time steps. (b) State-input map Θ of a neural network trained to mimic NMPC. (c) Projection of over-approximated closed-loop state-update set found using Theorem 4.5 by coupling $\bar{\Psi}$ and Θ.

To train the neural network controller, a nonlinear MPC is formulated as

$$\begin{aligned} \min_{u(\cdot)} \quad & \sum_{k=1}^{10} x_k^T \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} x_k + u_{k-1}^T u_{k-1} \\ \text{s.t.} \quad & \text{trapezoidal discretization of (3.26) holds}\,, \\ & u_k \in [-20, 20]\,,\ \forall k \in \{0, ..., 9\}\,. \end{aligned} \tag{4.33}$$

The solution of (4.33) is found using the MATLAB Model Predictive Control Toolbox [96] for 400 uniformly sampled initial conditions. The sampled initial conditions and the first optimal input of the solution trajectory u_0^* are then used as input-output pairs to train a neural network with 2 hidden layers, each with 10 nodes and ReLU activation functions, using the MATLAB Deep Learning Toolbox. A functional decomposition of the neural network with saturated output to obey torque constraints is performed and Theorem 3.4 is used to generate the state-input map Θ. The state-input map is shown in Figure 4.11(b). Using Theorem 3.4 and Corollary 3.5, this took 4 seconds to compute and an additional 45 seconds to apply the exact complexity reduction techniques from [22, 58].

Given an over-approximation of the open-loop state-update set $\bar{\Psi}$ and the exact state-input map Θ, an over-approximation of the closed-loop state-update set $\bar{\Phi}$ was constructed using the identity in Theorem 4.5 in less than 1 millisecond and exact reduction methods were completed in an additional 35 seconds. A projection of $\bar{\Phi}$ is shown in Figure 4.11(c). This projection is a subset of the projection of $\bar{\Psi}$ in Figure 4.11(a), as variability in the input is eliminated when creating the closed-loop state-update set using the state-input map Θ. Although difficult to perceive in the figure, some thickness in the $x_{k,2}$ dimension remains as a result of the Taylor remainder $\mathcal{L}$ and over-approximating nonlinear functions of the plant model. Table 4.3 reports the memory complexity of $\bar{\Psi}$, Θ, and $\bar{\Phi}$ for this example.

Using (4.29), over-approximations of forward reachable sets $\mathcal{R}_i$, $i \in \{1, ..., 15\}$ are calculated from an initial set given by

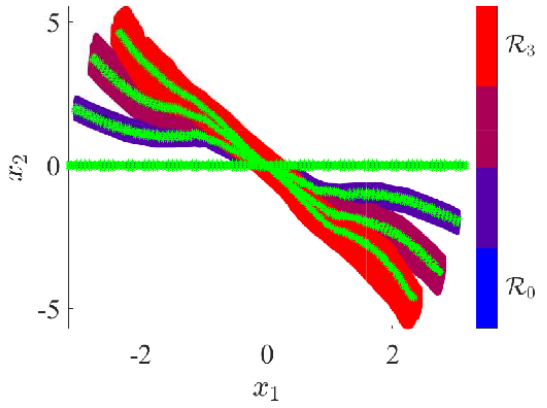
$$\mathcal{R}_0 = \left\langle \begin{bmatrix} \pi & 0 \\ 0 & 0.1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\rangle . \tag{4.34}$$

The over-approximated reachable sets up to $\mathcal{R}_3$ are plotted in Figure 4.12(a), overlaid by exact closed-loop trajectories found by randomly sampling points in $\mathcal{X}_0$ and propagating using (3.26). Examination of the exact trajectories suggests that a successful nonconvex over-approximation of the reachable sets is achieved. Computation time to execute the

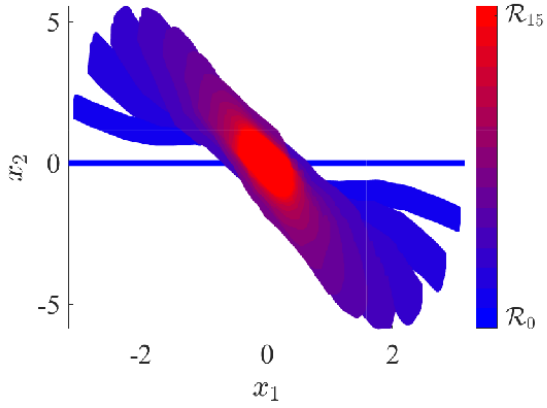
successor set identity was 5 milliseconds per time step on average.

To handle growth in set memory complexity over time steps, set propagation methods often utilize over-approximations to reduce complexity. Using techniques from [58], over-approximations of the reachable set are taken periodically every three time steps beginning at $k = 3$, resulting in 4 total approximations that took an average of 49 seconds each to compute, in a manner similar to [97]. At the time steps corresponding to over-approximations, the set is first saved and analyzed before being over-approximated. The over-approximation is used to calculate the reachable set of the subsequent time step. Figure 4.12(b) plots the over-approximated reachable sets and Figure 4.12(c) plots the corresponding memory complexity. The periodic memory complexity reduction is apparent in Figure 4.12(c). It is clear that the containment condition of Theorem 4.6, $\mathcal{R}_i \subseteq \mathrm{D}_\Phi$, is met at each time step.

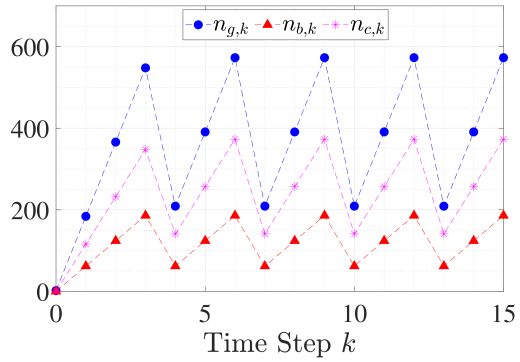
While plotting the reachable sets can be used to visually confirm performance and the containment condition of Theorem 4.6, this is a computationally expensive process, taking 7378 seconds to produce Figure 4.12(b). Much of the same information can be obtained with lower computational burden by sampling the support function in the axis-aligned directions, which took a total of 34 seconds for all 15 steps, less than 0.5% of the time to plot.



(a)



(b)



(c)

Figure 4.12: (a) Over-approximation of reachable sets $\mathcal{R}_0 \to \mathcal{R}_3$ of the inverted pendulum in closed-loop with a saturated neural network controller, overlaid by samples of exact trajectories in green. (b) Over-approximated reachable sets $\mathcal{R}_0 \to \mathcal{R}_{15}$ with over-approximations taken every three time steps. (c) Memory complexity of the over-approximated reachable sets.

4.4.3.1.2 Case 2 (ARCH-COMP Single Pendulum):

This case uses the single pendulum example found in [44, Section 3.5], with two alterations. Firstly, the goal of falsification/verification is removed so that each method will generate reachable sets over the time period $t \in \begin{bmatrix} 0 & 1 \end{bmatrix}$ seconds with a time step of 0.05 seconds. In [44, Section 3.5], the example is constructed such that it is unsafe and may be falsified. Many of the existing methods leverage specialized algorithms to generate trajectories to prove that the system is unsafe. In order to compare *reachability* tools, the consideration of an unsafe region is removed. Secondly, two initial sets are considered, one smaller initial set matches that given in [44, Section 3.5], and the case with a larger initial set is also considered. The sets are given by

$$\begin{aligned} &\text{Small Initial Set: } \mathcal{R}_0 = \begin{bmatrix} 1 & 1.2 \end{bmatrix} \times \begin{bmatrix} 0 & 0.2 \end{bmatrix} \text{, and} \\ &\text{Large Initial Set: } \mathcal{R}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix} \times \begin{bmatrix} -0.1 & 0.1 \end{bmatrix} . \end{aligned}$$

The proposed method follows the same procedure to construct the open-loop state-update set, state-input map, closed-loop state-update set, and reachable sets as done in the previous example. The changes amount to adjusting the parameters, including such that the controller has 2 hidden layers with 25 nodes each (50 total ReLU nodes). The complexity of the closed-loop state-update set is $n_{g,\phi} = 321$, $n_{b,\phi} = 75$, $n_{c,\phi} = 240$. As was done for the previous example, reachable sets are over-approximated every 3 time steps to handle growth in memory complexity.

The proposed method is compared to four state-of-the-art tools from the literature. Computation times and an indication of whether reachable sets undergo diverging over-approximation error for the small initial set and large initial set are given in Table 4.4 and Table 4.5, respectively. For the small initial set, all methods successfully generate reachable sets without diverging approximation error and the proposed method is substantially slower than the other tools. For the larg initial set, the proposed method was the only method to compute the reachable set without diverging approximation error, though it required almost 40 minutes to do so.

Tuning Parameters and Partitioning the Initial Set: CORA, JuliaReach, NNV, and POLAR each have tuning parameters that allow the user to adjust a trade-off between computation time and error. The results reported for the larger initial set were generated without altering these the tuning parameters as chosen for the small initial set, though the author did make several attempts to adjust tuning parameters for each to handle the larger initial set. The author was unable to find tuning parameters that

Table 4.4: Comparison of state-of-the-art tools for reachability analysis of an inverted pendulum with a neural network controller from a small initial set.

Tool	Small Initial Set	
	Computation Time [s]	Approximation Error
Proposed Method	312	Does not diverge
CORA	0.5	Does not diverge
JuliaReach	0.5	Does not diverge
NNV	2086	Does not diverge
POLAR	0.2	Does not diverge

yield better results for the four state-of-the-art tools, but a solution via adjusting the tuning parameters cannot be ruled out. Additionally, it is possible for CORA, JuliaReach, NNV, and POLAR to analyze the larger initial set by partitioning, e.g., when the initial set is split into 160 partitions, NNV can compute the reachable sets without diverging approximation error. In general, this may require a significant number of partitions, especially for systems with many states, that must be tuned to the problem at hand; thus addressing large initial sets remains an open challenge.

Breakdown of Proposed Method Computation Time: Over 96% of the proposed method computation time is spent calculating periodic convex over-approximations. The time to reduce the order of the closed-loop state-update set is 82 seconds ($> 3\%$). Thus the time spent to construct the open-loop state-update set, state-input map, closed-loop state-update set, and successor sets for all time steps is only 8 seconds. The periodic over-approximations are taken so that the analysis of the resulting sets can be achieved, i.e., if no periodic approximations are taken, GUROBI was unable to solve for the support function (2.14) beyond the first 6 times steps. This motivates work to efficiently generate hybrid zonotope over-approximations with acceptable approximation error, though approximations of hybrid zonotopes are challenging due to their implicit nature [58].

Table 4.5: Comparison of state-of-the-art tools for reachability analysis of an inverted pendulum with a neural network controller from a large initial set.

Tool	Large Initial Set	
	Computation Time [s]	Approximation Error
Proposed Method	2377	Does not diverge
CORA	0.6	Diverges
JuliaReach	0.7	Diverges
NNV	>7200	Diverges
POLAR	0.15* Terminates after 7 steps.	Diverges

4.4.3.2 Vertical Collision Avoidance System

The Vertical Collision Avoid System (VCAS) example (see [43] and [44] for details) is adopted for comparison between the proposed methods and state-of-the-art reachability tools. The plant is a linear discrete-time model with 3 states

- h_k: the relative height of the ownship from an intruder flying at a constant altitude,
- $\dot{h}_{0,k}$: the derivative of the height of the ownship,[2]
- $\tau_k \in \{25, 24, ..., 15\}$: the time until the ownship and intruder are no longer horizontally separated,[3] and

$$\begin{aligned} h_{k+1} &= h_k - \dot{h}_{0,k} - \frac{1}{2}\ddot{h}_k \,, \\ \dot{h}_{0,k+1} &= \dot{h}_{0,k} + \ddot{h}_k \,, \\ \tau_{k+1} &= \tau_k - 1 \,. \end{aligned}$$

An additional state $\text{adv}_k \in \{1, 2, ..., 9\}$, denotes the flight advisory, for which there are corresponding choices of $\ddot{h}$ to be selected by the pilot. At each time step, one of 9 neural networks, each with 5 fully connected hidden layers and 20 ReLU nodes per layer, is selected based on the previous time step advisory. Each neural network has 3 inputs, corresponding to h_k, $\dot{h}_{0,k}$, and τ_k, and has 9 outputs. The index of the largest output (of

[2] h and $\dot{h}_0$ have opposite sign convention.

[3] τ decrements by one each time step, i.e., $\tau_{k+1} = \tau_k - 1$. There appears to be a sign error in [44].

the neural network corresponding to the previous time step advisory) determines the advisory for the current time step. If the previous advisory and current advisory coincide, and $\dot{h}_{0,k}$ complies with the advisory, then $\ddot{h} = 0$. Otherwise, $\ddot{h}_k$ is selected according to the current advisory. Advisories and associated ranges of $\dot{h}$ that are compliant, and choices of $\ddot{h}$, are listed in Table 4.6.

CORA, JuliaReach, and POLAR are not capable of modeling the VCAS dynamics. In [43], POLAR does not participate in the VCAS benchmark problem. In both [43, 44], CORA and JuliaReach provide custom simulation algorithms to falsify the VCAS, and do not employ reachability algorithms. NNV is the only tool to employ reachability algorithms for this benchmark problem and is able to verify/falsify various initial conditions by partitioning the set. Additionally, NNV considers simplified versions of the problem where uncertainty in $\ddot{h}$ given an advisory is eliminated by assuming a "middle" (middle $\ddot{h}$ is chosen) or "worst-case" ($\ddot{h}$ that results in driving h closest to 0) strategy.

Table 4.6: VCAS Advisories

adv	Advisory	Compliant $\dot{h}_{0,k}$	Choice of $\ddot{h}_k$
1	COC	$\emptyset$	$\{-\frac{g}{8},\ 0,\ \frac{g}{8}\}$
2	DNC	$\dot{h}_{0,k} \leq 0$	$\{-\frac{g}{3},\ -\frac{7g}{24},\ -\frac{g}{4}\}$
3	DND	$0 \leq \dot{h}_{0,k}$	$\{\frac{g}{4},\ \frac{7g}{24},\ \frac{g}{3}\}$
4	DES1500	$\dot{h}_{0,k} \leq -1500$	$\{-\frac{g}{3},\ -\frac{7g}{24},\ -\frac{g}{4}\}$
5	CL1500	$1500 \leq \dot{h}_{0,k}$	$\{\frac{g}{4},\ \frac{7g}{24},\ \frac{g}{3}\}$
6	SDES1500	$\dot{h}_{0,k} \leq -1500$	$\{-\frac{g}{3}\}$
7	SCL1500	$1500 \leq \dot{h}_{0,k}$	$\{\ \ \frac{g}{3}\}$
8	SDES2500	$\dot{h}_{0,k} \leq -2500$	$\{-\frac{g}{3}\}$
9	SCL2500	$2500 \leq \dot{h}_{0,k}$	$\{\ \ \frac{g}{3}\}$

Proposed Method: The proposed method first generates HCG-representation graphs of the 9 neural networks associated with each advisory. To reduce the complexity of the problem, it is first shown that for a large region of the state space, only 4 advisories, $\text{adv}_k \in \{1, 5, 7, 9\}$, are achievable for trajectories of the aircraft starting from an initial set where the previous advisory is COC, $\text{adv}_{k-1} = 1$, and that do not exit the constraints $h \in \begin{bmatrix}-400 & -100\end{bmatrix}$, and $\dot{h}_{0,k} \in \begin{bmatrix}-100 & 100\end{bmatrix}$. The following functional decomposition relates the inputs of h_k, $\dot{h}_{0,k}$, τ_k, and adv_k to the next advisory. It is clear that the compliant regions of advisories $4 - 9$ would never be met for the domain chosen. It will be shown that under these assumptions, advisories 2 and 3 are never achieved, and thus

the logic associated with compliant regions can be neglected.

Consider the functional decomposition in Table 4.7. Note that the inequalities associated with $\overrightarrow{w}_{16,1:8}$ are non-strict in both directions. This is consistent with uncertainty when two outputs of the active neural network produce exactly the same value. Using the proposed methods and the functional decomposition, the graph of the function $\text{adv}_k = f(h_k, \dot{h}_{0,k}, \tau_k, \text{adv}_{k-1})$ can be generated. Using the identity (3.3), the set of advisories that can be active, given the set of states and previous advisory, is calculated. This is done iteratively in Table 4.8. Iteration 4 results in the same potential advisories $\{1, 5, 7, 9\}$ in the output set as the input set. The assumption that advisories 2 and 3 are not visited is confirmed, and the only advisories ever achieved starting from the input set in iteration 4, and remaining within the bounds for $(h_k, \dot{h}_k, \tau_k)$, is $\{1, 5, 7, 9\}$. This is a powerful result, as for analysis within a large domain can neglect 5 of the neural networks, significantly reducing the complexity of the problem.

Table 4.7: Functional decomposition of VCAS: States →Advisory

$$
\begin{aligned}
w_1 &= h_k \\
w_2 &= \dot{h}_{0,k} \\
w_3 &= \tau_k \\
w_4 &= \mathrm{adv}_{k-1} \\
\overrightarrow{w}_{i+4,1:9} &= \begin{cases} f_{NN,i}(w_1, w_2, w_3) & \text{if } \mathrm{adv}_{k-1} = i \\ \mathbf{0} & \text{otherwise} \end{cases}, \quad \forall\, i \in \{1,2,3,4,5,6,7,8,9\} \\
\overrightarrow{w}_{14,1:9} &= \textstyle\sum_{i=5}^{13} \overrightarrow{w}_i \\
\overrightarrow{w}_{15,1:7} &= \begin{bmatrix} \max(w_{14,1}, w_{14,2}) \\ \max(w_{15,1}, w_{14,3}) \\ \max(w_{15,2}, w_{14,4}) \\ \max(w_{15,3}, w_{14,5}) \\ \max(w_{15,4}, w_{14,6}) \\ \max(w_{15,5}, w_{14,7}) \\ \max(w_{15,6}, w_{14,8}) \end{bmatrix} \\
\overrightarrow{w}_{16,1:8} &= \begin{bmatrix} w_{16,1} = \begin{cases} 0 & \text{if } w_{14,1} \geq w_{14,2} \\ 1 & \text{if } w_{14,1} \leq w_{14,2} \end{cases} \\ w_{16,2} = \begin{cases} 0 & \text{if } w_{15,1} \geq w_{14,3} \\ 1 & \text{if } w_{15,1} \leq w_{14,3} \end{cases} \\ \vdots \\ w_{16,8} = \begin{cases} 0 & \text{if } w_{15,7} \geq w_{14,9} \\ 1 & \text{if } w_{15,7} \leq w_{14,9} \end{cases} \end{bmatrix} \\
w_{17} &= \begin{cases} 1 & \text{if } (w_{16,1:8}) = \mathbf{0} \\ 2 & \text{if } (w_{16,2:8}) = \mathbf{0} \wedge w_{16,1} = 1 \\ 3 & \text{if } (w_{16,3:8}) = \mathbf{0} \wedge w_{16,2} = 1 \\ 4 & \text{if } (w_{16,4:8}) = \mathbf{0} \wedge w_{16,3} = 1 \\ 5 & \text{if } (w_{16,5:8}) = \mathbf{0} \wedge w_{16,4} = 1 \\ 6 & \text{if } (w_{16,6:8}) = \mathbf{0} \wedge w_{16,5} = 1 \\ 7 & \text{if } (w_{16,7:8}) = \mathbf{0} \wedge w_{16,6} = 1 \\ 8 & \text{if } (w_{16,8:8}) = \mathbf{0} \wedge w_{16,7} = 1 \\ 9 & \text{if } w_{16,8} = 1 \end{cases}
\end{aligned}
$$

Table 4.8: Iterative domain propagation of advisories for an assumed domain of interest.

Iteration	Input Set $(h_k, \dot{h}_k, \tau_k, \text{adv}_{k-1})$	Potential Advisories adv_k
1	$[-400,\ -100]...$ $\times[-100,\ 100]...$ $\times\{25, 24, ...15\}...$ $\times\{1\}$	$\{1, 5\}$
2	$[-400,\ -100]...$ $\times[-100,\ 100]...$ $\times\{25, 24, ...15\}...$ $\times\{1, 5\}$	$\{1, 5, 7\}$
3	$[-400,\ -100]...$ $\times[-100,\ 100]...$ $\times\{25, 24, ...15\}...$ $\times\{1, 5, 7\}$	$\{1, 5, 7, 9\}$
4	$[-400,\ -100]...$ $\times[-100,\ 100]...$ $\times\{25, 24, ...15\}...$ $\times\{1, 5, 7, 9\}$	$\{1, 5, 7, 9\}$

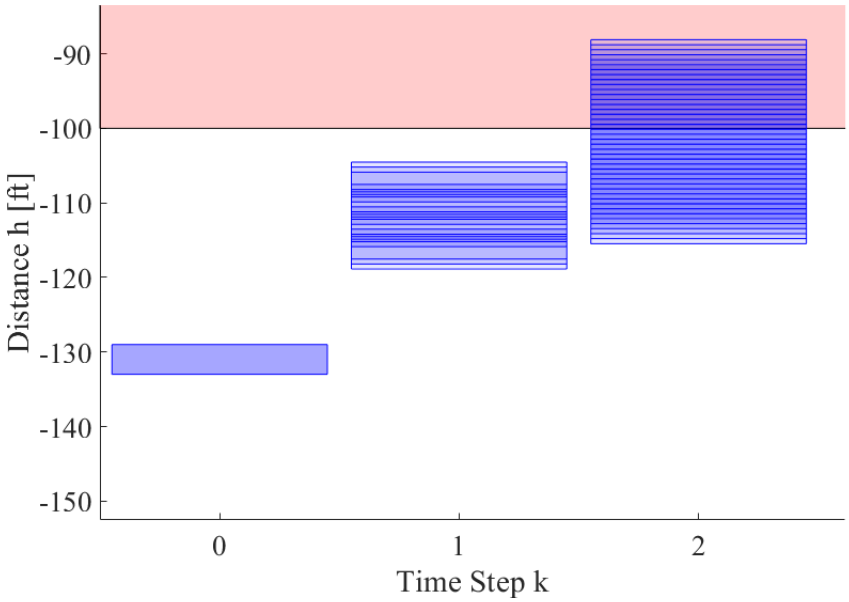


Figure 4.13: Reachable sets for VCAS, calculated and falsified in 0.8 seconds, and plotted in 4.3 seconds.

Now consider the domain

$$(h_k, \dot{h}_{0,k}, \tau_k, \text{adv}_k) \in \text{D}_1 \times \text{D}_2 \times \text{D}_3 \times \text{D}_4 \\ = \begin{bmatrix} -400 & -100 \end{bmatrix} \times \begin{bmatrix} -100 & 100 \end{bmatrix} \times \{25, 24, ..., 15\} \times \{1, 5, 7, 9\} .$$

Using the domain and the functional decomposition given by Table 4.9, a closed-loop state-update set is generated in 2.8 seconds with complexity $(n_{g,\phi}, n_{b,\phi}, n_{c,\phi}) = (2339, 842, 2049)$, which encodes the transition from $(w_1, w_2, w_3, w_4) \to (w_{14}, w_{15}, w_{16}, w_{12})$. Reachable sets are generated by recursion of (4.29) and checked for falsification each step, taking a total of 0.8 seconds to falsify the VCAS *without* restricting the inputs sets via "middle" or "worst-case" assumptions or partitioning the initial set. The reachable sets are plotted in Figure 4.13.

Although the proposed methods are able to falsify the VCAS system without restricting the potential inputs or partitioning the initial set, a closed-loop state-update set and reachable sets are generated for the the so-called "worst-case" scenario,[4] and are plotted in Figure 4.14. Worst-case reachable sets were calculated in 16 seconds. Portions of the reachable set that go beyond the domain of the state-update set are not propagated

[4]See [44] for details regarding the worst-case assumption.

Table 4.9: Functional decomposition of VCAS: States →Updated States

$$
\begin{array}{lcl}
w_1 & = & h_k \\
w_2 & = & \dot{h}_{0,k} \\
w_3 & = & \tau_k \\
w_4 & = & \mathrm{adv}_{k-1} \\
\overrightarrow{w}_{i+4,1:4} & = & \begin{cases} [e_1^T, e_5^T, e_7^T, e_9^T]^T f_{NN,i}(w_1, w_2, w_3) & \text{if } \mathrm{adv}_{k-1} = i \\ \mathbf{0} & \text{otherwise} \end{cases}, \quad \forall\ i \in \{1,5,7,9\} \\
\overrightarrow{w}_{9,1:4} & = & \sum_{i=5}^{8} \overrightarrow{w}_i \\
\overrightarrow{w}_{10,1:2} & = & \begin{bmatrix} \max(w_{9,1}, w_{9,2}) \\ \max(w_{10,1}, w_{9,3}) \end{bmatrix} \\
\overrightarrow{w}_{11,1:3} & = & \begin{bmatrix} w_{11,1} = \begin{cases} 0 & \text{if } w_{9,1} \geq w_{9,2} \\ 1 & \text{if } w_{9,1} \leq w_{9,2} \end{cases} \\ w_{11,2} = \begin{cases} 0 & \text{if } w_{10,1} \geq w_{9,3} \\ 1 & \text{if } w_{10,1} \leq w_{9,3} \end{cases} \\ w_{11,3} = \begin{cases} 0 & \text{if } w_{10,2} \geq w_{9,4} \\ 1 & \text{if } w_{10,2} \leq w_{9,4} \end{cases} \end{bmatrix} \\
w_{12} & = & \begin{cases} 1 & \text{if } (w_{11,1:3}) = \mathbf{0} \\ 5 & \text{if } (w_{11,2:3}) = \mathbf{0} \wedge w_{11,1} = 1 \\ 7 & \text{if } (w_{11,3:3}) = \mathbf{0} \wedge w_{11,2} = 1 \\ 9 & \text{if } w_{11,3} = 1 \end{cases} \\
w_{13} & \in & \begin{cases} \{-\frac{g}{8}, 0, \frac{g}{8}\} & \text{if } w_{12} = 1 \\ \{\frac{g}{4}, \frac{7g}{24}, \frac{g}{3}\} & \text{if } w_{12} = 5 \\ \{\frac{g}{3}\} & \text{if } w_{12} = 7 \\ \{\frac{g}{3}\} & \text{if } w_{12} = 9 \end{cases} \\
w_{14} & = & w_1 - w_2 - w_{13} \\
w_{15} & = & w_2 + w_{13} \\
w_{16} & = & w_3 - 1
\end{array}
$$

through subsequent time steps.[5] The reachable sets can be compared to those for NNV in [44, Figure 30] and correspond to overlaying the four subplots therein, minus the portions that are lost due to portions of the reachable sets leaving the domain of the state-update set.

[5]See assumption in 4.6.

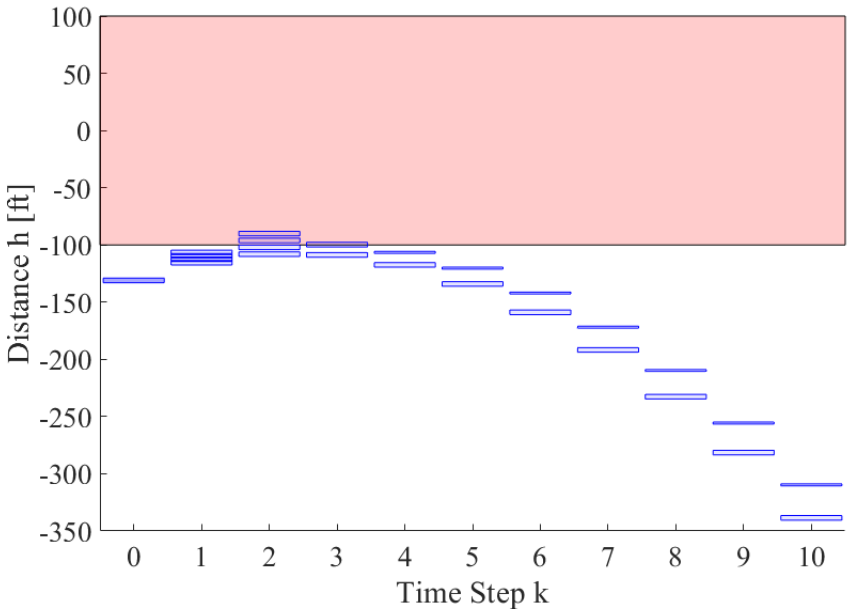


Figure 4.14: Reachable sets for VCAS with a simplifying "worst-case" assumption are calculated in 16 seconds. The reachable set loses pieces associated with the portion of a reachable set at the previous time step not included in the domain of the state-update set $h \notin \begin{bmatrix} -400 & -100 \end{bmatrix}$.

Chapter 5
Set-Valued State Estimation and Parameter Identification

5.1 Introduction

Estimation methods are often needed to implement state-feedback controllers [98] or detect faults and failures within a system [99]. Approaches for state estimation include stochastic methods such as Kalman filtering [98] and set-based methods [51, 100, 101]. Stochastic approaches have been successfully applied to a wide range of systems due to their efficient implementation and ability to provide a statistical characterization of the estimate uncertainty. This is contingent upon accurate representation of random processes, e.g., measurement noise. Statistical methods do not seek to provide guaranteed bounds on estimate uncertainties.

This chapter addresses set-valued state estimation (SVSE) methods, also referred to as set-membership state estimation, which compute sets containing *all* possible state values consistent with a dynamic plant model, output measurements, and uncertainty bounds. Additionally, this chapter addresses set-valued parameter identification (SVPI) methods, which compute sets containing *all* possible parameters consistent with a dynamic plant model, output measurements and uncertainty bounds.

SVSE and SVPI methods commonly use convex set representations, such as zonotopes and constrained zonotopes [12,21,51,102–105], as these are closed under key set operations for propagating linear dynamics [12]. However, convex sets are *not* closed under nonlinear mappings. To address a larger class of dynamics and measurement models, a common approach is to generate linear approximations, e.g., via mean value and first-order Taylor extensions [51, 105].

Convex approaches are successful when the dynamics are represented well using affine

approximations and when the exact set of states/parameters consistent with a given measurement can be tightly approximated with a convex set. When the true set of states/parameters consistent with the dynamic model, measurements, and uncertainty is significantly nonconvex, over-approximations using affine and convex approaches incur significant error. Methods considering collections of convex sets coupled with piecewise-affine approximations have been developed, though computational challenges arise, for example successive intersection with guards can result in exponential growth in the number of convex sets with time [12]. It is especially challenging to compute successor sets used in the dynamic update of nonconvex SVSE when the set-valued state estimate is large [12]. Similar difficulties arise in the measurement update.

Contribution: This chapter proposes a new approaches for SVSE and SVPI of nonlinear systems. Graphs of functions are represented with a nonconvex set representation called the hybrid zonotope, providing set estimates tighter to the true state than possible using convex methods while admitting efficient and scalable calculations.

5.2 Set-Valued State Estimation

Consider a class of discrete-time nonlinear systems given by dynamic plant model $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ and measurement model $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_y}$, where

$$x_{k+1} = f(x_k, u_k, w_k)\,, \tag{5.1}$$

$$y_{k+1} = g(x_{k+1}, v_{k+1})\,, \tag{5.2}$$

with bounded state $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$, input $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$, process uncertainty $w \in \mathcal{W} \subset \mathbb{R}^{n_w}$, and measurement noise $v \in \mathcal{V} \subset \mathbb{R}^{n_v}$.

Assumption 5.1 *The functions $f(\cdot)$ and $g(\cdot)$ are defined and bounded over their respective domains corresponding to $\mathcal{X}$, $\mathcal{U}$, $\mathcal{W}$, and $\mathcal{V}$, which themselves are also bounded.*[1]

The set $\hat{\mathcal{X}}_{k+1|k}$ denotes the set of states that can be achieved by the dynamics at time step $k+1$ that are consistent with all measurements through time step k and may be found as the *successor set* of $\hat{\mathcal{X}}_{k|k}$, $\text{Suc}(\hat{\mathcal{X}}_{k|k}, \mathcal{U}_k, \mathcal{W}_k)$.

Definition 5.1 *The successor set from $\mathcal{R}_k \subseteq \mathcal{X}$ with inputs and process uncertainty*

[1] See Corollary 3.3.

bounded by $\mathcal{U}_k \subseteq \mathcal{U}$ *and* $\mathcal{W}_k \subseteq \mathcal{W}$*, respectively, is given by*

$$\mathrm{Suc}(\mathcal{R}_k, \mathcal{U}_k, \mathcal{W}_k) \equiv \left\{ f(x,u,w) \mid \begin{array}{c} x \in \mathcal{R}_k,\ u \in \mathcal{U}_k\ , \\ w \in \mathcal{W}_k \end{array} \right\} . \tag{5.3}$$

Definition 5.2 *The set* $\hat{\mathcal{X}}_{y_{k+1}}$ *is the set of states consistent with measurement* y_{k+1} *and measurement noise* $\mathcal{V}_{k+1} \subseteq \mathcal{V}$ *within the feasible region* $\mathcal{X}$ *and is given by*

$$\hat{\mathcal{X}}_{y_{k+1}} \equiv \{x \mid y_{k+1} = g(x,v),\ x \in \mathcal{X},\ v \in \mathcal{V}_{k+1}\} . \tag{5.4}$$

Assuming the initialized set-valued state estimate contains the true initial state, i.e., $x_0 \in \hat{\mathcal{X}}_{0|0}$, SVSE combines dynamic and measurement updates using the recursion

$$\hat{\mathcal{X}}_{k+1|k} = \mathrm{Suc}(\hat{\mathcal{X}}_{k|k}, \mathcal{U}_k, \mathcal{W}_k) , \tag{5.5}$$

$$\hat{\mathcal{X}}_{k+1|k+1} = \hat{\mathcal{X}}_{k+1|k} \cap \hat{\mathcal{X}}_{y_{k+1}} , \tag{5.6}$$

to generate set-valued state estimates guaranteed to contain the true state. The dynamic update (5.5) can be addressed with methods from Chapter 4, which demonstrates the hybrid zonotopes effective use in calculating successor sets for many classes of hybrid and nonlinear systems. The measurement update (5.6) consists of generating $\hat{\mathcal{X}}_{y_{k+1}}$ and a generalized intersection, for which hybrid zonotopes are well-suited (2.25), with the result of the dynamic update. The generation of $\hat{\mathcal{X}}_{y_{k+1}}$ is accomplished using methods in Chapter 3 by 1) generating an over-approximation of the graph of the measurement function represented as a hybrid zonotope, and using the identity (3.5) to generate a set of states (inputs to the graph of the function) that are consistent with a measurement (outputs of the graph of the function).

5.2.1 Numerical Example: Sum of Signal Strengths

Consider the integrator dynamics and measurement

$$x_{k+1} = x_k + u_k , \tag{5.7}$$

$$y = \sum_{i=1}^{4} \frac{1}{d_i^2 + 1} , \tag{5.8}$$

Table 5.1: Memory complexity for set-based over-approximations of (5.8).

Method	n_g	n_b	n_c
M1 (100 breakpoints)	202	163	102
M3 (2 layers, 20 nodes each)	127	31	93

where $d_i = ||x - s_i||_2$ and

$$s_1 = (1, 3),\ s_2 = (-2, 2),\ s_3 = (3, 0),\ s_4 = (-1, -4)\,. \tag{5.9}$$

with $x_k, u_k \in \mathbb{R}^2$. One interpretation is that y is a measurement of the summation of signal strengths emitted by four sources located at s_i , $\forall i \in \{1, 2, 3, 4\}$, where the strength of signal from each source decreases with the inverse square of the distance from the source. While the proposed methods account for input uncertainty, process uncertainty, and measurement noise, for simplicity of exposition this example omits these. This makes it easier to assess the effect of using an over-approximated graph of the nonlinear measurement function.

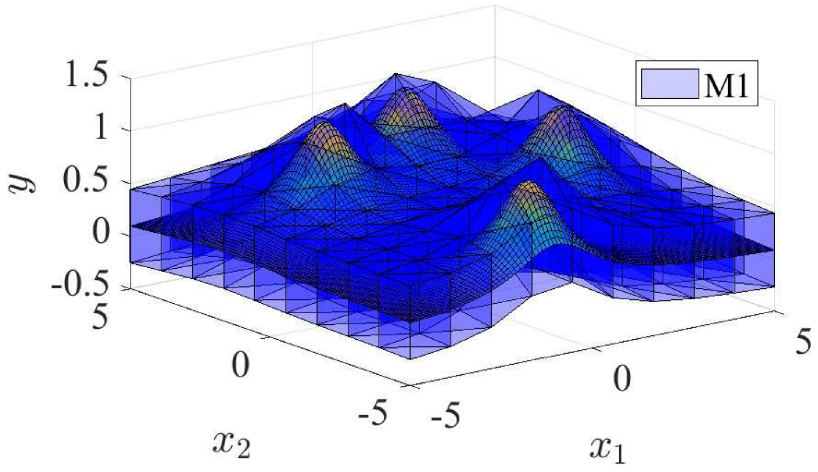
Figure 5.1 shows two methods (M1 and M3)[2] for over-approximating (5.8) over a domain of $(x_1, x_2) \in [-5, 5] \times [-5, 5]$. M1 uses 100 uniformly spaced breakpoints to generate an SOS approximation, which is represented in HCG-rep using Theorem 2.1. Bounds on the worst-case errors are found by solving (3.15) and (3.16), and an over-approximated graph of the function is constructed using Corollary 3.4. M3 trains a ReLU neural network using the MATLAB Deep Learning Toolbox [96]. The ReLU network has 2 layers with 20 nodes each. The neural network is represented exactly as a hybrid zonotope using methods from Chapter 3 and a functional decomposition of the form presented in Section 4.4.2.3. Bounds on the worst-case error are found by solving nonlinear programs similar to (3.15) and (3.16) over each PWA region of the graph of the neural network function. Table 5.1 and Figure 5.1 show that M3 can achieve a much tighter approximation with less than a quarter of the maximum error of M1 and considerably lower memory complexity. For these reasons, Method M3 is chosen as the over-approximation of a graph of a function $\bar{\Phi}_{meas}$ used for the remainder of the example.

For this example, the dynamic update (5.5) specifies to

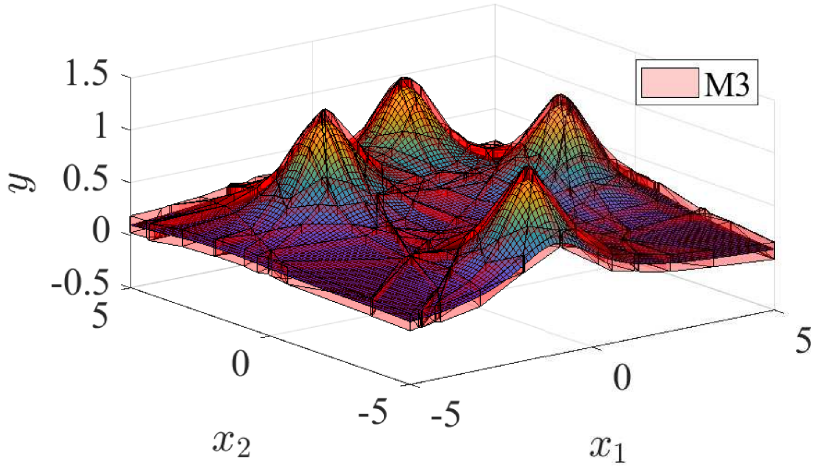
$$\hat{\mathcal{X}}_{k+1|k} = \hat{\mathcal{X}}_{k|k} \oplus \{u_k\}\,, \tag{5.10}$$

where u_k is known to the estimator. The set $\hat{\mathcal{X}}_{y_{k+1}}$ in (5.4) is calculated using Theorem

[2]M2 from [63] is not presented here but the names are unchanged for consistency with that publication.



(a)



(b)

Figure 5.1: Comparison of methods for hybrid zonotope over-approximation of (5.8). M1: uniformly spaced breakpoints and M3: trained neural network. A surface plot of (5.8) is also shown in both (a) and (b).

3.2, Corollary 3.1, and $\bar{\Phi}_{meas}$ as

$$\hat{\mathcal{X}}_{y_{k+1}} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \end{bmatrix} \left(\bar{\Phi}_{meas} \cap \begin{bmatrix} \mathbf{0} & 1 \end{bmatrix} \{y_{k+1}\} \right) ,$$

for use in (5.6). The set-valued estimate is initialized as the interval set corresponding to state constraints

$$\hat{\mathcal{X}}_{0|-1} = \mathcal{X} = [-5, 5] \times [-5, 5] , \tag{5.11}$$

and the true initial state and input trajectory are given by

$$x_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T , \tag{5.12}$$

$$[u_0 \ u_1 \ u_2 \ u_3] = \begin{bmatrix} -1 & -2 & -1 & 2 \\ 1 & -1 & -1 & -1 \end{bmatrix} . \tag{5.13}$$

Figure 5.2 shows the set-valued *a priori* state estimate $\bar{\hat{\mathcal{X}}}_{k|k-1}$, *a posteriori* state estimate $\bar{\hat{\mathcal{X}}}_{k|k}$, exact set of measurement-consistent states $\hat{\mathcal{X}}_{y_k}$ and its over-approximation $\bar{\hat{\mathcal{X}}}_{y_k}$, and true state x_k for several time steps. This shows how the initial state estimate is gradually reduced to a small region by combining knowledge of the dynamics and measurement, and highlights how the nonlinearity of the measurement model is tightly captured by the nonconvexity of the hybrid zonotope set representation. When $k = 2$, the state estimate consists of several disjoint regions, and after the third measurement, the state estimate consists of only two disjoint regions. After the fourth dynamic update, one of these regions escapes the set of feasible states, so only one region remains.

For comparison, a convex approach M_{convex} is shown that estimates $\hat{\mathcal{X}}_{k|k}$ using convex methods by taking an inner convex approximation of $\hat{\mathcal{X}}_{y_k}$. This is achieved using the same dynamic update (5.10) and initialization (5.11), but samples the exact level set of (5.8) within $\mathcal{X}$ at each time step to generate a polytope in vertex representation to approximate $\hat{\mathcal{X}}_{y_{k+1}}$. It can be seen from M_{convex} that *any* convex approach would produce a much larger over-approximation than the proposed approach, which leverages nonconvex representations of $\hat{\mathcal{X}}_{y_{k+1}}$.

Set-valued state estimates for $k \in \{0, 1, ..., 4\}$ are computed by the proposed method in less than 0.2 seconds. Plotting requires 740 seconds as this involves solving mixed-integer linear programs to find each nonempty convex set within the hybrid zonotope, associated with specific values of the binary factors. However, upper and lower bounds on each of the states can be calculated much more quickly by sampling the support function in the axis-aligned directions. For this example, these bounds are obtained for all time steps in a total of 6 seconds.

In some applications it may be desirable to obtain a single point within the set-valued set estimate, for example as state feedback for a controller. This becomes nontrivial for nonconvex and/or disjoint set-valued state estimates, for which the "center" of a set may not be within the set, such as a toroid. A natural alternative would be to find a single feasible point within the set, which in this case can be found by solving a mixed-integer linear feasibility problem. This required 0.8 seconds for $\hat{\mathcal{X}}_{4|4}$.

5.3 Set-Valued Parameter Identification

Consider a bounded[3] nonlinear measurement model with additive measurement noise given by $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$, where

$$y_k = g_{x,p}(x_k, p) \ , \tag{5.14}$$

$$\hat{y}_k = g(x_k, v_k, p) = g_{x,p}(x_k, p) + v_k \ , \tag{5.15}$$

with bounded state $x_k \in \mathcal{X} \subset \mathbb{R}^{n_x}$, measurement noise $v_k \in \mathcal{V} \subset \mathbb{R}^{n_v}$, and bounded uncertain and time-invariant parameters $p \in \mathcal{P} \subset \mathbb{R}^{n_p}$. It is assumed that the state x_k is estimated as $\hat{x}_k$ sufficiently well such that error is bounded, i.e.,

$$\hat{x}_k - x_k \in \mathcal{W} \ , \quad \forall \ k \ , \tag{5.16}$$

and $||w|| < \infty \ , \ \forall w \in \mathcal{W}$.

The set $\hat{\mathcal{P}}_{k|k}$ denotes the set of parameters that are consistent with all state estimate and measurement pairs $(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), ..., (\hat{x}_k, \hat{y}_k)$ and may be found as the intersection of sets that are consistent *each* measurement $\hat{\mathcal{P}}_k$, i.e.,

$$\hat{\mathcal{P}}_{k|k} = \bigcap_{1}^{k} \hat{\mathcal{P}}_k \ , \tag{5.17}$$

where

$$\hat{\mathcal{P}}_k = \left\{ p \ \middle| \ \begin{array}{c} y_k = g_{x,p}(x_k, p) \ , \\ (x_k, y_k) \in (\hat{x}_k \oplus (-\mathcal{W})) \times (\hat{y}_k \oplus (-\mathcal{V})) \ , \\ x_k \in \mathcal{X} \ , \ v_k \in \mathcal{V} \ , \ p \in \mathcal{P} \ . \end{array} \right\}$$

The process to generate $\hat{P}_k$ is an extension of the methods from Chapter 3. The identities given in (3.3) and (3.5) map a set of inputs of a function to a set of outputs and vice versa, while the process of generating $\bar{\hat{P}}_k$, maps a set that contains outputs and a subset of inputs $(\hat{x}_k, \hat{y}_k)$ to a subset of the inputs of $g(\cdot)$, associated with the parameters p.

[3]See Corollary 3.3.

Consider the graph of the function (5.15) given by

$$\Phi = \left\{ \begin{bmatrix} x_k \\ v_k \\ p \\ \hat{y}_k \end{bmatrix} \;\middle|\; \begin{array}{c} \hat{y}_k = g(x_k, v_k, p) \ , \\ x_k \in \mathcal{X} \ , \\ v_k \in \mathcal{V} \ , \\ p \in \mathcal{P} \end{array} \right\} , \tag{5.18}$$

and the set $\mathcal{S}$ given by

$$\mathcal{S} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I}_{n_p} & \mathbf{0} \end{bmatrix} \left(\Phi \cap \begin{bmatrix} \mathbf{I}_{n_x} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix} ((\hat{x}_k \oplus (-\mathcal{W})) \times (\hat{y}_k \oplus (-\mathcal{V}))) \right) . \tag{5.19}$$

By direct applications of the generalized intersection and linear mapping identities, $\mathcal{S}$ is equivalent to $\hat{P}_k$. The relationship of (3.5) to (5.19) is as follows. The set $((\hat{x} \oplus (-\mathcal{W})) \times (\hat{y} \oplus (-\mathcal{V})))$ from (5.19) can be cast as the set of outputs $\mathcal{Q}$ from (3.5). Then the only differences are in the matrices of the generalized intersection and linear mapping, which are needed to accommodate a relationship that is *not* a strict mapping from outputs to inputs of the function $g(\cdot)$. If an over-approximation of $\bar{\Phi} \supset \Phi$ is used, then $\mathcal{S} \supset \hat{P}_k$, as set containment is preserved under generalized intersection and linear mapping.

5.3.1 Numerical Example: Sum of Signal Strengths

Consider the measurement

$$y = \sum_{i=1}^{2} \frac{1}{d_i^2 + 1} \, , \tag{5.20}$$

where $d_i = ||x - p_i||_2$ and

$$p_1 = (-3, 2), \ p_2 = (2, -2) \, , \tag{5.21}$$

with $x_k, u_k \in \mathbb{R}^2$. One interpretation is that y is a measurement of the summation of signal strengths emitted by two sources located at p_1 and p_2, where the strength of signal from each source decreases with the inverse square of the distance from the source. While the previous example focused on estimating the location of a vehicle based on measurements from signals at known locations, the current example focuses on estimating the signal locations as parameters.

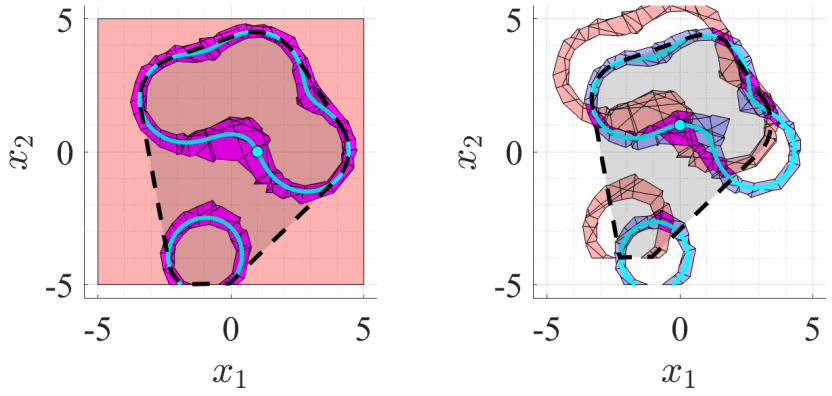
The proposed methods account for estimation uncertainty and measurement noise, for simplicity of exposition this example omits these, i.e., $\mathcal{W} = \{\mathbf{0}\}$ and $\mathcal{V} = \{\mathbf{0}\}$. As in the previous example, this makes it easier to assess the effect of using an over-approximated graph of the nonlinear measurement function.

A functional decomposition is given by

$$
\begin{aligned}
w_1 &= x_1, & w_{11} &= w_7^2 \,, \\
w_2 &= x_2 \,, & w_{12} &= w_8^2 \,, \\
w_3 &= p_{(1,1)} \,, & w_{13} &= w_9^2 \,, \\
w_4 &= p_{(1,2)} \,, & w_{14} &= w_{10}^2 \,, \\
w_5 &= p_{(2,1)} \,, & w_{15} &= w_{11} + w_{12} + 1 \,, \\
w_6 &= p_{(2,2)} \,, & w_{16} &= w_{13} + w_{14} + 1 \,, \\
w_7 &= w_1 - w_3 \,, & w_{17} &= \tfrac{1}{w_{15}} \,, \\
w_8 &= w_1 - w_4 \,, & w_{18} &= \tfrac{1}{w_{16}} \,, \\
w_9 &= w_2 - w_5 \,, & w_{19} &= w_{17} + w_{18} \,, \\
w_{10} &= w_2 - w_6
\end{aligned}
$$

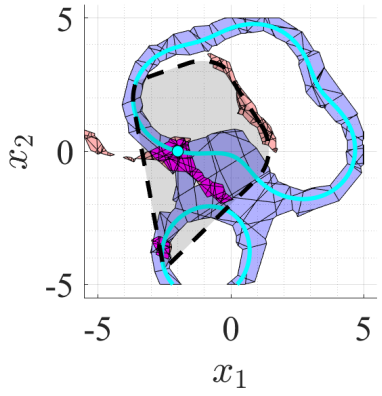
and using the methods in Chapter 3 an over-approximation of graph of the function $\bar{\Phi}$ as defined by (5.18) is constructed in 12 seconds for a domain $x_k, p_1, p_2 \in \begin{bmatrix} -5, & 5 \end{bmatrix}^2$. Using the recursion (5.17), $\bar{\hat{P}}_{k|k}$, $\forall\, i \in \{1, 2, ..., 10\}$ are constructed in 0.2 seconds for ordered pairs of states, shown in Figure 5.3, and outputs given by (5.15). The parameter sets $\bar{\hat{P}}_{k|k}$, $\forall\, i \in \{1, 2, ..., 10\}$ for the locations of each source p_1 and p_2 are shown in Figure 5.4, and support functions taken at each time-step are shown in Figure 5.5. The 80 support functions[4] to generate the plot in Figure 5.5 were evaluated in 132 seconds, and indicate that signal locations were narrowed down to a relatively small set containing the true locations.

[4] 2 directions per dimension, 2 dimensions per per source, 2 sources, 10 steps

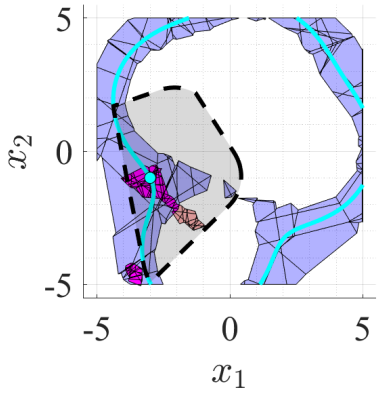


(a) $k = 0$

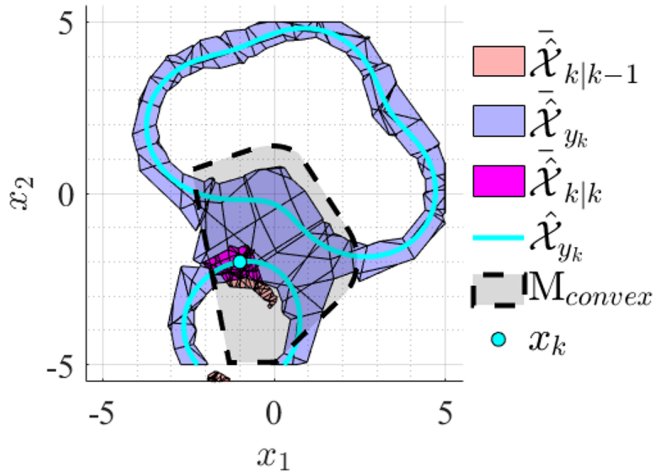
(b) $k = 1$



(c) $k = 2$



(d) $k = 3$



(e) $k = 4$

Figure 5.2: Set valued state estimation (5.5)-(5.6) of (5.7) with measurement model (5.8) and its over-approximation represented as a hybrid zonotope given by Figure 5.1(b).

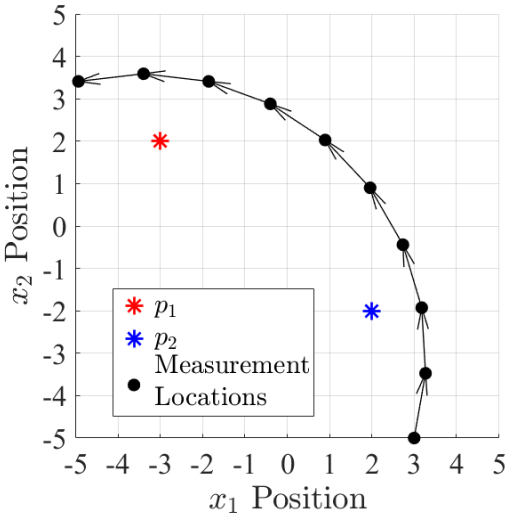
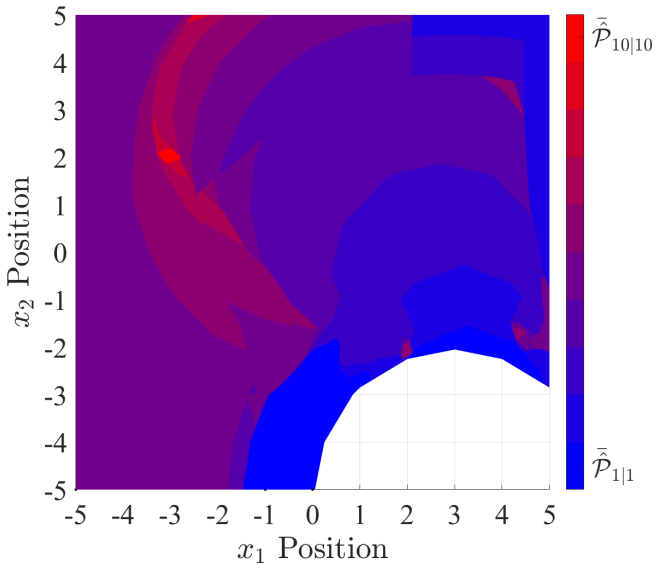
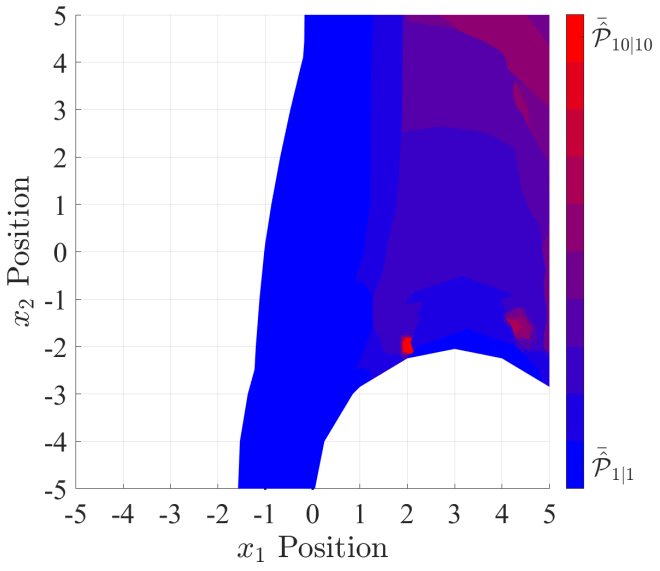


Figure 5.3: Measurement and signal source locations.

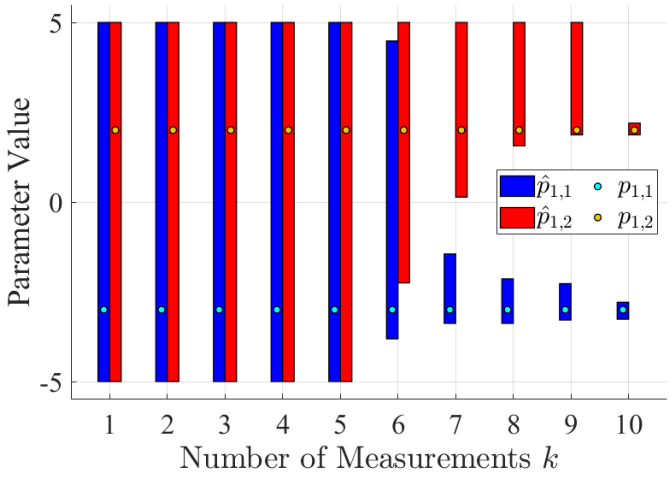


(a)

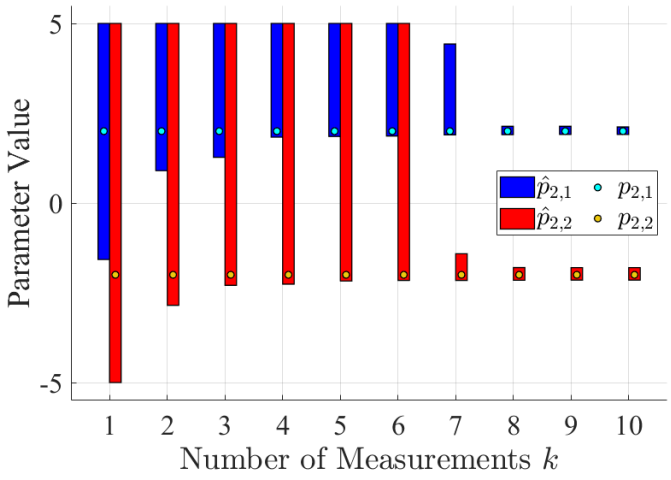


(b)

Figure 5.4: (a) Parameter identification of the location of the first source, $p_1 = (-3, 2)$. (b) Parameter identification of the location of second source $p_2 = (2, -2)$.



(a)



(b)

Figure 5.5: (a) Bounds in axis-aligned directions for $p_1 = (-3, 2)$ found by support functions. (b) Bounds in axis-aligned directions for $p_2 = (2, -2)$ found by support functions. The 80 support functions used to find the axis-aligned bounds were evaluated in 132 seconds.

Chapter 6
Conclusion

This thesis provides fundamental theoretic contributions to set-based methods for reachability analysis and demonstrates their utility for verification, state estimation, and parameter identification of hybrid and nonlinear systems. The proposed methods share a common framework that leverages graphs of functions, hybrid zonotopes, special-ordered set approximations, and functional decomposition.

The reachability analysis techniques address several classes of systems including mixed-logical dynamical systems, linear systems in closed loop with MPC, discrete hybrid automata, logical systems, and nonlinear systems in both open loop and closed loop with neural network controllers. By generating sets that represent dynamics over a domain of interest, the techniques can efficiently produce accurate reachable sets from large initial sets, which often pose challenges to existing techniques, including state-of-the-art software tools. Because analysis of reachable sets represented as hybrid zonotopes requires solving mixed-integer linear programs, the approaches may be too computationally expensive for online verification in many applications. However, because they can efficiently analyze large initial sets and broad classes of systems, the proposed methods are well suited for offline design and verification of complex systems and controllers.

The proposed state estimation and parameter identification techniques address nonlinear systems. Numerical examples demonstrate their ability to handle highly nonconvex estimation problems, where more efficient convex approaches would incur excessive approximation error.

While propagating the implicit reachable sets is fast, a fundamental challenge of the methods in this thesis is that the analysis of resulting sets requires solving computationally-intensive MILPs. Future work will seek to address these challenges through the use of novel order reduction methods. Additionally, future work will seek to apply the proposed methods for verification of real-world systems.

Appendix
Computer Hardware Specifications

All numerical results were generated on a desktop computer using a 9th Generation Intel® Core™ i7 processor with 16GB of random access memory storage.

Bibliography

[1] T. J. Bird, "Hybrid zonotopes: A mixed-integer set representation for the analysis of hybrid systems," Ph.D. dissertation, Purdue University, 2022.

[2] T. J. Bird, H. C. Pangborn, N. Jain, and J. P. Koeln, "Hybrid zonotopes: A new set representation for reachability analysis of mixed logical dynamical systems," *Automatica*, vol. 154, p. 111107, 2023.

[3] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[4] M. Chen, Q. Hu, C. Mackin, J. F. Fisac, and C. J. Tomlin, "Safe platooning of unmanned aerial vehicles via reachability," in *IEEE Conference on Decision and Control*, 2015, pp. 4695–4701.

[5] L. C. Barrett, "Applied reachability analysis for time-optimal spacecraft attitude reorientations," Ph.D. dissertation, Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio, 2021.

[6] A. A. Geraldes, L. Geretti, D. Bresolin, R. Muradore, P. Fiorini, L. S. Mattos, and T. Villa, "Formal verification of medical cps: A laser incision case study," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 4, 2018.

[7] A. Olama, P. R. Mendes, and E. F. Camacho, "Lyapunov-based hybrid model predictive control for energy management of microgrids," *IET Generation, Transmission & Distribution*, vol. 12, no. 21, pp. 5770–5780, 2018.

[8] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control*, 2017, pp. 2242–2253.

[9] M. Bui, M. Lu, R. Hojabr, M. Chen, and A. Shriraman, "Real-time hamilton-jacobi reachability analysis of autonomous system with an fpga," in *IEEE/RSJ Conference on Intelligent Robots and Systems*, 2021, pp. 1666–1673.

[10] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[11] M. Chen, S. Herbert, and C. J. Tomlin, "Fast reachable set approximations via state decoupling disturbances," in *IEEE Conference on Decision and Control*, 2016, pp. 191–196.

[12] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *ACM Conference on Hybrid Systems: Computation and Control*, 2004, pp. 477–492.

[13] Z. Yang, M. Wu, and W. Lin, "An efficient framework for barrier certificate generation of uncertain nonlinear hybrid systems," *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100837, 2020.

[14] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 369–395, 2021.

[15] A. Kurzhanski and P. Varaiya, "On ellipsoidal techniques for reachability analysis. part i: External approximations," *Optimization Methods & Software*, vol. 17, no. 2, pp. 177–206, 2002.

[16] H. Tiwary, "On the hardness of computing intersection, union and minkowski sum of polytopes," *Discrete & Computational Geometry*, vol. 40, p. 469–479, 2008.

[17] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2021.

[18] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *International Journal of Pure and Applied Mathematics*, vol. 6, pp. 239–316, 2003.

[19] N. Kochdumper and M. Althoff, "Constrained polynomial zonotopes," *Acta Informatica*, vol. 60, pp. 279–316, 2023.

[20] P. S. Duggirala and M. Viswanathan, "Parsimonious, simulation based verification of linear systems," in *Computer Aided Verification*, 2016, pp. 477–494.

[21] A. Alanwar, F. J. Jiang, S. Amin, and K. H. Johansson, "Logical zonotopes: A set representation for the formal verification of boolean functions," *arXiv: 2210.08596*, 2022.

[22] A. Alanwar, F. J. Jiang, and K. H. Johansson, "Polynomial logical zonotopes: A set representation for reachability analysis of logical systems," *arXiv: 2306.12508*, 2023.

[23] 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*, vol. 69, pp. 126–136, 2016.

[24] T. J. Bird, N. Jain, H. C. Pangborn, and J. P. Koeln, "Set-based reachability and the explicit solution of linear mpc using hybrid zonotopes," in *American Control Conference*, 2022, pp. 158–165.

[25] T. J. Bird and N. Jain, "Unions and complements of hybrid zonotopes," *IEEE Control Systems Letters*, vol. 6, pp. 1778–1783, 2022.

[26] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995, hybrid Systems.

[27] A. Bemporad, F. D. Torrisi, and M. Morari, "Optimization-based verification and stability characterization of piecewise affine and hybrid systems," in *ACM Hybrid Systems: Computation and Control*, 2000, pp. 45–58.

[28] D. Liberzon, *Switching in Systems & Control*. Springer, 2003.

[29] A. Bemporad, "Modeling, control, and reachability analysis of discrete-time hybrid systems," *University of Sienna*, 2003.

[30] G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," in *ACM Conference on Hybrid Systems: Computation and Control*, 2013, p. 203–212.

[31] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *ACM Conference on Hybrid Systems: Computation and Control*, 2000, pp. 20–31.

[32] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[33] A. Bemporad, W. Heemels, and B. De Schutter, "On hybrid systems and closed-loop mpc systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 863–869, 2002.

[34] W. Heemels, B. De Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, 2001.

[35] F. D. Torrisi and A. Bemporad, "Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems," *IEEE transactions on control systems technology*, vol. 12, no. 2, pp. 235–249, 2004.

[36] M. Althoff, "On computing the minkowski difference of zonotopes," *arXiv: 1512.02794*, 2016.

[37] L. Yang and N. Ozay, "Scalable zonotopic under-approximation of backward reachable sets for uncertain linear systems," *IEEE Control Systems Letters*, vol. 6, pp. 1555–1560, 2022.

[38] S. V. Rakovic, M. Baric, and M. Morari, "Max-min control problems for constrained discrete time systems," in *IEEE Conference on Decision and Control*, 2008, pp. 333–338.

[39] L. G. Giovanni Palmieri, Miroslav Barić and F. Borrelli, "Robust vehicle lateral stabilisation via set-based methods for uncertain piecewise affine systems," *Vehicle System Dynamics*, vol. 50, no. 6, pp. 861–882, 2012.

[40] M. Althoff, "An introduction to cora 2015," in *Applied Verification for Continuous and Hybrid Systems*, vol. 34, 2015, pp. 120–151.

[41] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling, "Juliareach: A toolbox for set-based reachability," in *ACM Conference on Hybrid Systems: Computation and Control*, 2019, p. 39–44.

[42] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *Computer Aided Verification*, 2020, pp. 3–17.

[43] C. Huang, J. Fan, X. Chen, W. Li, and Q. Zhu, "Polar: A polynomial arithmetic framework for verifying neural-network controlled systems," in *Symposium on Automated Technology for Verification and Analysis*, 2022, pp. 414–430.

[44] D. M. Lopez, M. Althoff, L. Benet, X. Chen, J. Fan, M. Forets, C. Huang, T. T. Johnson, T. Ladner, W. Li *et al.*, "ARCH-COMP22 category report: artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," in *Applied Verification of Continuous and Hybrid Systems*, 2022.

[45] D. M. Lopez, M. Althoff, M. Forets, T. T. Johnson, T. Ladner, and C. Schilling, "ARCH-COMP23 category report: Artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," in *Applied Verification of Continuous and Hybrid Systems*, vol. 96, 2023, pp. 89–125.

[46] N. Kochdumper, C. Schilling, M. Althoff, and S. Bak, "Open- and closed-loop neural network verification using polynomial zonotopes," in *NASA Formal Methods*, K. Y. Rozier and S. Chaudhuri, Eds., 2023, pp. 16–36.

[47] T. Ladner and M. Althoff, "Automatic abstraction refinement in neural network verification using sensitivity analysis," in *ACM Conference on Hybrid Systems: Computation and Control*, 2023.

[48] C. Schilling, M. Forets, and S. Guadalupe, "Verification of neural-network control systems by integrating taylor models and zonotopes," in *AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 8169–8177.

[49] F. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs," *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 22–28, 1968.

[50] L. Chisci, A. Garulli, and G. Zappa, "Recursive state bounding by parallelotopes," *Automatica*, vol. 32, no. 7, pp. 1049–1055, 1996.

[51] H. Wang, I. Kolmanovsky, and J. Sun, "Zonotope-based set-membership parameter identification of linear systems with additive and multiplicative uncertainties: A new algorithm," in *American Control Conference*, 2017, pp. 1481–1486.

[52] B. S. Rego, J. K. Scott, D. M. Raimondo, and G. V. Raffo, "Set-valued state estimation of nonlinear discrete-time systems with nonlinear invariants based on constrained zonotopes," *Automatica*, vol. 129, p. 109638, 2021.

[53] B. S. Rego, D. Locatelli, D. M. Raimondo, and G. V. Raffo, "Joint state and parameter estimation based on constrained zonotopes," *Automatica*, vol. 142, p. 110425, 2022.

[54] J. Koeln, T. J. Bird, J. Siefert, J. Ruths, H. Pangborn, and N. Jain, "zonolab: A matlab toolbox for set-based control systems analysis using hybrid zonotopes," *arXiv: 2310.15426*, 2023.

[55] G. Brassard and P. Bratley, *Fundamentals of algorithmics*. Prentice-Hall, Inc., 1996.

[56] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal toolbox (ET)," in *IEEE Conference on Decision and Control*, 2006, pp. 1498–1503.

[57] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling, "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *ACM Conference on Hybrid Systems: Computation and Control*, 2018, pp. 41–50.

[58] M. Althoff and G. Frehse, "Combining zonotopes and support functions for efficient reachability analysis of linear systems," in *IEEE Conference on Decision and Control*, 2016, pp. 7439–7446.

[59] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[60] J. A. Siefert, T. J. Bird, J. P. Koeln, N. Jain, and H. C. Pangborn, "Robust successor and precursor sets of hybrid systems using hybrid zonotopes," in *IEEE Control Systems Letters*, vol. 7, 2023, pp. 355–360.

[61] J. Glunt, J. Siefert, H. Pangborn, and S. Brennan, "Challenges in integrating low-level path following and high-level path planning over polytopic maps," in *Modeling, Estimation, and Controls Conference*, 2023.

[62] J. A. Siefert, T. J. Bird, J. P. Koeln, N. Jain, and H. C. Pangborn, "Reachability analysis of nonlinear systems using hybrid zonotopes and functional decomposition," *arXiv: 2304.06827v1*, 2023.

[63] J. A. Siefert, A. F. Thompson, J. J. Glunt, and H. C. Pangborn, "Set-valued state estimation for nonlinear systems using hybrid zonotopes," in *IEEE Conference on Decision and Control*, 2023.

[64] T. Gan, M. Chen, Y. Li, B. Xia, and N. Zhan, "Reachability analysis for solvable dynamical systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2003–2018, 2017.

[65] A. Bemporad, G. Ferrari-Trecate, and M. Morari, "Observability and controllability of piecewise affine and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1864–1876, 2000.

[66] J. Roll, "Local and piecewise affine approaches to system identification," Ph.D. dissertation, Linköping University, 2003.

[67] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, "A bounded-error approach to piecewise affine system identification," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1567–1580, 2005.

[68] C. Y. Lai, C. Xiang, and T. H. Lee, "Identification and control of nonlinear systems via piecewise affine approximation," in *IEEE Conference on Decision and Control*, 2010, pp. 6395–6402.

[69] ——, "Data-based identification and control of nonlinear systems via piecewise affine approximation," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2189–2200, 2011.

[70] E. Asarin, T. Dang, and A. Girard, "Reachability analysis of nonlinear systems using conservative approximation," in *ACM Conference on Hybrid Systems: Computation and Control*, 2003, pp. 20–35.

[71] R. Baier, C. Büskens, I. A. Chahma, and M. Gerdts, "Approximation of reachable sets by direct solution methods for optimal control problems," *Optimisation Methods and Software*, vol. 22, no. 3, pp. 433–452, 2007.

[72] E. M. L. Beale and J. A. Tomlin, "Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables," *Operational Research*, vol. 69, no. 447-454, p. 99, 1970.

[73] S. Leyffer, A. Sartenaer, and E. Wanufelle, "Branch-and-refine for mixed-integer nonconvex global optimization," *Mathematics and Computer Science Division, Argonne National Laboratory*, vol. 39, pp. 40–78, 2008.

[74] E. Wanufelle, "A global optimization method for mixed integer nonlinear non-convex problems related to power systems analysis," Ph.D. dissertation, Facultés Universitaires Notre-Dame de la Paix, 2007.

[75] S. Kozak and J. Stevek, "Improved piecewise linear approximation of nonlinear functions in hybrid control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 14 982–14 987, 2011.

[76] A. Szűcs, M. Kvasnica, and M. Fikar, "Optimal piecewise affine approximations of nonlinear functions obtained from measurements," *IFAC Proceedings Volumes*, vol. 45, no. 9, pp. 160–165, 2012.

[77] E. H. S. Diop, A. Ngom, and V. S. Prasath, "Signal approximations based on nonlinear and optimal piecewise affine functions," *Circuits, Systems, and Signal Processing*, vol. 42, no. 4, pp. 2366–2384, 2023.

[78] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization and Engineering*, vol. 10, pp. 1–17, 2009.

[79] S. Rebennack and J. Kallrath, "Continuous piecewise linear delta-approximations for univariate functions: computing minimal breakpoint systems," *Journal of Optimization Theory and Applications*, vol. 167, no. 2, pp. 617–643, 2015.

[80] M. Stämpfle, "Optimal estimates for the linear interpolation error on simplices," *Journal of Approximation Theory*, vol. 103, no. 1, pp. 78–90, 2000.

[81] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," in *Doklady Akademii Nauk*, vol. 114, no. 5, 1957, pp. 953–956.

[82] V. Kůrková, "Kolmogorov's theorem is relevant," *Neural Computation*, vol. 3, no. 4, pp. 617–622, 1991.

[83] M. Kvasnica, A. Szücs, and M. Fikar, "Automatic derivation of optimal piecewise affine approximations of nonlinear systems," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 8675–8680, 2011.

[84] E. Dijkstra, "Algol 60 translation : An algol 60 translator for the x1 and making a translator for algol 60," 1961.

[85] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm origin identification using random moonwalks," in *IEEE Symposium on Security and Privacy*, 2005, pp. 242–256.

[86] H. P. Williams, *Model building in mathematical programming.* John Wiley & Sons, 2013.

[87] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Institute of Automatic Control Engineering, Technische Universität München, 2010.

[88] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[89] A. Lodi, *Mixed Integer Programming Computation*. Springer Berlin Heidelberg, 2010, pp. 619–645.

[90] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 233–249, 2010.

[91] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[92] Y. Zhang and X. Xu, "Reachability analysis and safety verification of neural feedback systems via hybrid zonotopes," in *American Control Conference*, 2023, pp. 1915–1921.

[93] J. Ortiz, A. Vellucci, J. Koeln, and J. Ruths, "Hybrid zonotopes exactly represent relu neural networks," in *Machine Learning Research*, 2023.

[94] H. Zhang, Y. Zhang, and X. Xu, "Hybrid zonotope-based backward reachability analysis for neural feedback systems with nonlinear system models," *arXiv: 2310.06921*, 2023.

[95] J. A. Siefert, T. J. Bird, J. P. Koeln, N. Jain, and H. C. Pangborn, "Successor sets of discrete-time nonlinear systems using hybrid zonotopes," in *American Control Conference*, 2023, pp. 1383–1389.

[96] MATLAB, "version 9.10.0 (r2021a)," The MathWorks Inc., Natick, Massachusetts, 2022.

[97] J. A. Siefert, D. D. Leister, J. P. Koeln, and H. C. Pangborn, "Discrete reachability analysis with bounded error sets," *IEEE Control Systems Letters*, vol. 6, pp. 1694–1699, 2021.

[98] D. Simon, *Optimal State Estimation*. John Wiley & Sons, Ltd, 2006.

[99] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott, "Closed-loop input design for guaranteed fault diagnosis using set-valued observers," *Automatica*, vol. 74, pp. 107–117, 2016.

[100] N. Loukkas, J. J. Martinez, and N. Meslem, "Set-membership observer design based on ellipsoidal invariant sets," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6471–6476, 2017.

[101] Y. Wang, V. Puig, and G. Cembrano, "Set-membership approach and kalman observer based on zonotopes for discrete-time descriptor systems," *Automatica*, vol. 93, pp. 435–443, 2018.

[102] V. T. H. Le, C. Stoica, T. Alamo, E. F. Camacho, and D. Dumur, "Zonotopic guaranteed state estimation for uncertain systems," *Automatica*, vol. 49, no. 11, pp. 3418–3424, 2013.

[103] C. Combastel, "A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes," in *IEEE Conference on Decision and Control*, 2005, pp. 7228–7234.

[104] B. S. Rego, D. M. Raimondo, and G. V. Raffo, "Set-based state estimation of nonlinear systems using constrained zonotopes and interval arithmetic," in *European Control Conference*, 2018, pp. 1584–1589.

[105] B. S. Rego, G. V. Raffo, J. K. Scott, and D. M. Raimondo, "Guaranteed methods based on constrained zonotopes for set-valued state estimation of nonlinear discrete-time systems," *Automatica*, vol. 111, p. 108614, 2020.

Vita

Jacob A. Siefert

Jacob Siefert received his B.S. degree in Mechanical Engineering from the University of Maryland in 2016, and went on to complete his M.S. degree in Mechanical Engineering from the University of Minnesota in 2021. In 2020 he began working as a Ph.D. student and Research Assistant with his advisor, Professor Herschel Pangborn, at The Pennsylvania State University. He has accepted a Research Faculty position at The Pennsylvania State University starting in January 2024. His research interests include optimal control, co-design, hybrid systems, and reachability-based verification.