Technical University of Munich
TUM School of Computation, Information and Technology

Toward Automated Verification of Dynamical Systems Using Forward and Backward Reachability Analysis

Mark Peter Christoph Wetzlinger

Complete reprint of the dissertation approved by the TUM School of Computation, Information and Technology of the Technical University of Munich for the award of the

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

Chair:

Prof. Dr. Francisco Javier Esparza Estaun

Examiners:

1. Prof. Dr.-Ing. Matthias Althoff
2. Prof. Dr. Goran Frehse
3. Prof. Dr. Sylvie Putot

The dissertation was submitted to the Technical University of Munich on 08.11.2023 and accepted by the TUM School of Computation, Information and Technology on 25.04.2024.

Acknowledgments

First and foremost, I would like to thank my supervisor, Matthias Althoff, for his example, guidance, and support over the past four years. It has been a pleasure to work with you. Furthermore, I am grateful to the Technical University of Munich, our administrative staff Ute, Amy, Katja, and Elisabeth, my mentor Alexander Lenz, my advised students, my colleagues, in particular Tobias, Victor, Felix, Lukas, Hanna, Philipp, Jakob, Eivind, Alex, Hussein, and Paul, my collaborators Alexis, Adrian, Niklas, and renowned researchers Stanley Bak and Murat Arcak. In your individual ways, you have all helped me become a more diligent and prudent researcher, a more concise author and teacher, and a more considerate and caring colleague. On a personal note, I want to thank Stefan, Patrick, Johannes, Leo, Olivier, Sophie, and Theresa; and of course, my ever-supporting family.

I gratefully acknowledge the financial support from my research projects ConVeY (DFG GRK 2428) and justITSELF (ERC grant agreement ID: 817629); in particular, for enabling my participation in many workshops, retreats, and conferences and for facilitating my research stay at the University of California at Berkeley for six months, which has contributed significantly to my professional and personal growth.

Finally, I am honored to have Goran Frehse and Sylvie Putot on my examination committee, as your work has influenced and inspired me since the very beginning.

Munich, November 2023 Mark Wetzlinger

Abstract

Life nowadays has become increasingly interwoven with cyber-physical systems, which consist of physical components that perceive their environment and act upon it based on internal algorithms. Examples of cyber-physical systems include autonomous driving, control of power grids, and robotic surgery. These use cases are safety-critical, as failures can cause ecological and economic damage or harm humans.

In this thesis, we focus on the physical behavior of cyber-physical systems, which can be described by linear and nonlinear dynamical systems. We reason about these models using reachability analysis, a formal method that computes all potential system behaviors under the influence of uncertainties; we call this result the reachable set. Reachability analysis can verify safety by showing that no unsafe behavior is reachable or falsify safety by showing that distinct system behavior is unsafe. However, one cannot compute the exact reachable set in general but only approximations, whose accuracy strongly depends on the parametrization of the reachability algorithm, which ultimately decides whether safety can be determined.

Forward reachability analysis determines all future states that the system can reach. For linear time-invariant systems, we devise a reachability algorithm that computes an arbitrarily accurate enclosure of the reachable set, as well as a verification algorithm that automatically verifies or falsifies a given safety specification. For nonlinear systems, we adaptively tune all algorithm parameters to tightly enclose the reachable set, which can be used for verification in a subsequent step. Backward reachability analysis is used to determine whether a system can avoid an unsafe set or reach a goal set. We propose novel backward reachability algorithms for linear time-invariant systems, which are the first to scale polynomially in the state dimension.

Our numerical evaluation shows that we can verify and falsify safety for linear systems fully automatically up to orders of magnitude faster than comparable state-of-the-art approaches. Furthermore, our reachability algorithm for nonlinear systems solves benchmarks without requiring any manual algorithm parameter tuning as fast as other expert-tuned reachability algorithms. Moreover, our backward reachability algorithms enables the analysis of linear systems with over a thousand states for the first time. In conclusion, these contributions significantly enhance the capabilities of reachability analysis for the purpose of safety verification.

Zusammenfassung

Das Leben ist heutzutage immer enger mit cyber-physischen Systemen verflochten, die mittels physischer Komponenten die Umgebung wahrnehmen und basierend auf internen Algorithmen handeln. Beispiele für cyber-physische Systeme umfassen autonomes Fahren, die Regelung von Stromnetzen und chirurgische Eingriffe durch Roboter. Diese Anwendungsfälle sind allesamt sicherheitskritisch, da Fehlverhalten ökologische und ökonomische Schäden verursachen sowie Menschen verletzen kann.

In dieser Dissertation wird das physikalische Verhalten cyber-physischer Systeme untersucht, das durch lineare und nichtlineare dynamische Systeme beschrieben werden kann. Hierzu werden Methoden aus dem Gebiet der Erreichbarkeitsanalyse verwendet, einer formalen Methode zur Berechnung der erreichbaren Menge, die alle möglichen Systemverhalten unter dem Einfluss von Unsicherheiten darstellt. Mittels Erreichbarkeitsanalyse kann Sicherheit nachgewiesen werden, indem gezeigt wird, dass kein unsicheres Verhalten erreichbar ist. Zudem kann Sicherheit widerlegt werden, indem ein einzelnes Systemverhalten als unsicher identifiziert wird. Im Allgemeinen kann die erreichbare Menge aber nicht exakt, sondern nur näherungsweise berechnet werden. Die Genauigkeit hängt dabei stark von der Parametrisierung des Erreichbarkeitsalgorithmus ab, die schlussendlich darüber entscheidet, ob Sicherheit festgestellt werden kann.

Vorwärtserreichbarkeitsanalyse bestimmt alle erreichbaren zukünftigen Zustände. Für lineare zeit-invariante Systeme wird in dieser Arbeit ein Erreichbarkeitsalgorithmus entworfen, der einen beliebig genauen Einschluss der erreichbaren Menge berechnet, sowie ein Verifikationsalgorithmus, der automatisch Sicherheit nachweist bzw. widerlegt. Für nichtlineare Systeme werden alle algorithmischen Parameter automatisch eingestellt, was zu einem engen Einschluss der erreichbaren Menge führt, der danach zur Verifikation verwendet werden kann. Durch Rückwärtserreichbarkeitsanalyse kann gezeigt werden, ob ein System unsicheres Verhalten vermeiden oder einen Zielzustand erreichen kann. In dieser Arbeit werden die ersten Rückwärtserreichbarkeitsalgorithmen für lineare zeit-invariante Systeme vorgestellt, die polynomiell mit der Zustandsdimension skalieren.

Die Ergebnisse zeigen, dass Sicherheit für lineare Systeme nun vollautomatisch und um Größenordnungen schneller als mittels vergleichbarer Ansätze vom Stand der Technik nachgewiesen bzw. widerlegt werden kann. Zudem löst der vorgestellte Erreichbarkeitsalgorithmus für nichtlineare Systeme Testbeispiele ohne manuelle Einstellung algorithmischer Parameter in vergleichbarer Zeit wie andere von Experten eingestellte Erreichbarkeitsalgorithmen. Der neue Algorithmus für Rückwärtserreichbarkeit ermöglicht erstmals die Analyse linearer Systeme mit über tausend Zuständen. Zusammenfassend trägt die vorliegende Arbeit deutlich zur Erweiterung der Fähigkeiten von Erreichbarkeitsanalyse zum Zwecke der Sicherheitsverifikation bei.

Résumé

La vie quotidienne repose de plus en plus sur des systèmes cyberphysiques, c'est-à-dire des composants physiques capables de percevoir leur environnement et d'y réagir de manière dictée par des algorithmes internes. On peut citer comme exemples de systèmes cyberphysiques les véhicules autonomes, le contrôle des réseaux électriques et les opérations chirurgicales effectuées par des robots. Tous ces cas d'utilisation sont critiques en termes de sûreté : des dysfonctionnements peuvent causer des dommages écologiques ou économiques, ou bien blesser des personnes.

Dans cette thèse, nous nous concentrons sur le comportement physique des systèmes cyberphysiques, lequel peut être décrit par des systèmes dynamiques linéaires ou non linéaires. Nous raisonnons sur ces modèles en utilisant l'analyse d'atteignabilité, une méthode formelle permettant de calculer tous les comportements potentiels du système en présence d'incertitudes ; ce résultat étant appelé l'ensemble atteignable. L'analyse d'atteignabilité permet de vérifier la sûreté en montrant qu'aucun comportement non sûr n'est atteignable, ou de la falsifier en montrant qu'un comportement particulier du système n'est pas sûr. En général, on ne peut calculer que des approximations de l'ensemble atteignable, dont la précision dépend fortement du paramétrage de l'algorithme d'atteignabilité, laquelle décide donc de la capacité à déterminer la sûreté.

L'analyse vers l'avant détermine les états futurs atteignables par le système. Pour les systèmes linéaires invariants en temps, nous concevons un algorithme d'atteignabilité qui calcule une enveloppe arbitrairement précise de l'ensemble atteignable, ainsi qu'un algorithme de vérification qui vérifie ou falsifie automatiquement une spécification de sûreté donnée. Pour les systèmes non linéaires, nous ajustons de façon adaptative tous les paramètres de l'algorithme pour calculer une enveloppe précise de l'ensemble atteignable ; ces résultats peuvent ensuite être utilisé à des fins de vérification. L'atteignabilité en arrière peut être utilisée pour déterminer si un système peut éviter un ensemble non sûr ou atteindre un ensemble-but. Nous proposons de nouveaux algorithmes d'atteignabilité en arrière pour les systèmes linéaires invariants en temps, qui sont les premiers à avoir une complexité polynomiale en fonction des états.

Nos évaluations numériques montrent qu'ils permettent de vérifier et de falsifier la sûreté pour des systèmes linéaires de façon totalement automatique, jusqu'à plusieurs ordres de grandeur plus rapidement que les approches comparables de l'état de l'art. De plus, notre algorithme d'atteignabilité pour les systèmes non linéaires résout nos problèmes de test sans nécessiter d'ajustement manuel des paramètres, et ce, aussi rapidement que des algorithmes d'atteignabilité préalablement paramétrés par des experts. Enfin, nos algorithmes d'atteignabilité en arrière permettent, de façon inédite, l'analyse de systèmes linéaires comportant plus d'un millier d'états. En conclusion, ces contributions améliorent de façon significative les possibilités d'analyse d'atteignabilité pour la vérification de sûreté.

Contents

1 Introduction

Today's society witnesses pervasive changes in all aspects of our lives through the rise of cyber-physical systems—physical devices enhanced by computational capabilities that act upon their environment. Over the last decades, more potent hardware and versatile software packages have allowed computer programs to reach previously unimaginable levels of complexity. Notable domains that rely on these computational capabilities include computer vision, which interprets the information obtained by camera images or videos, or trajectory planning, which determines a path to reach a particular goal while avoiding obstacles. Integrating such algorithms into physical devices like robots, vehicles, or drones gives rise to systems that can autonomously operate within their environment. They determine their next action by executing an internal algorithm that optimally decides how to fulfill their given task under the current circumstances, known through measurements of their surroundings.

A popular example of a cyber-physical system is autonomous driving: The cyber component is the trajectory planning algorithm that aims to steer the vehicle through traffic to a desired destination while respecting traffic regulations, and the physical component is composed of the sensors and actuators of the vehicle. Expected benefits of autonomous driving are the reduction of traffic accidents and traffic jams as well as increased vehicle sharing, leading to a reduction in harmful emissions. Another example of cyber-physical systems is a power grid, where the principal physical components are the power plants and consumers, and the cyber components aim to optimally adjust energy generation to the demand of the consumers while respecting the physical limits of the power grid. Ideally, this balances the generated and consumed energy and prevents excessive loads by self-regulating mechanisms that can quickly react to faults in the grid. There are many more examples of cyber-physical systems, such as medical robots, uncrewed space exploration, and human-robot collaboration.

The above examples represent safety-critical use cases, as failures may cause economic damages, ecological catastrophes, or the loss of human life. Consequently, cyber-physical systems must operate safely to prevent harmful outcomes of their actions. For instance, faults in autonomous vehicles or power grids may cause traffic accidents or blackouts, respectively.

A complete proof of safety encompasses several different perspectives: First, both the physical perception of the environment and the execution of all algorithms need to run reliably. Second, the implementation of the algorithms needs to be correct. Third, the algorithms must return provably correct strategies for solving the given task. Separate research domains are dedicated to the respective levels of safety. In this thesis, we will only consider the third safety level, which assumes reliable hardware and a correct implementation of the algorithms.

1.1 Safety Analysis

The design of a concrete algorithm for determining safety depends on three main factors: The model describing the behavior of the cyber-physical system, the model parameters representing the circumstances under which the system operates, and the specification defining safe behaviors.

Models of cyber-physical systems are designed in discrete and continuous spaces, whose respective states arise naturally from the composition of the cyber-physical system: Discrete states represent distinct modes of operation, such as gears in a gearbox or high-level actions like "turn left/right" or "go straight" in path planning. In contrast, continuous states model the operating domain of the system, e.g., a two-dimensional plane representing the road on which an autonomous vehicle drives. The behavior of discrete states can be modeled using various formalisms, such as finite state automata, Petri nets, and statecharts, whereas the behavior of continuous states can be described using differential equations. Commonly, the behavior of discrete states is modeled in discrete time, whereas the behavior of continuous states is analyzed in both discrete and continuous time. Combining discrete and continuous states yields hybrid systems, e.g., timed automata, hybrid automata, and hybrid statecharts. Each modeling formalism can be divided into different system classes whose members share a set of properties that are exploited to develop well-performing algorithms. System classes can be ordered hierarchically to show which ones generalize or specialize others. The work in this thesis is restricted to systems with purely continuous dynamics described by linear and nonlinear ordinary differential equations. These system classes can describe many real-world phenomena, including the dynamics of vehicles, vessels, spacecraft, thermal conduction, chemical reactions, analog circuits, power grids, population growth, and others.

In addition to the model, we must also define the *model parameters* describing the operating conditions for the system. Examples include the state of the system at the start of the analysis or the time horizon over which the analysis is conducted. Real-world systems are fundamentally uncertain so that each model parameter is usually bounded by a set of possible values. These uncertainties originate from two main sources: First, mathematical models cannot fully accurately capture the actual behavior of a real-world system. Second, measurements are only accurate up to a finite precision, which induces further uncertainty when the system interacts with the environment. In this thesis, we consider uncertainties in the initial state of the system as well as external disturbances that uncontrollably influence the state. Furthermore, the control inputs to the system are chosen from a bounded input set.

Lastly, we formalize the expected system behavior in a *specification*. A common type of specification is liveness, where the system must reach a certain behavior at some future point in time. Fulfilling a fairness specification entails that a certain behavior is reachable repeatedly. The predominant type of specification is safety, which requires certain behaviors to be avoided at all times. Specifications are often encoded with temporal logics, which are modeling languages to formalize the system behavior over time. This thesis focuses on safety specifications, defined using unsafe sets of states that occupy a subspace of the operating domain.

Due to the presence of uncertainties, infinitely many different system behaviors can potentially occur, each of which emanates from a set of deterministic model parameters. Formal methods rigorously reason about all possible system behaviors, allowing us to correctly assess whether a safety specification for a given model and model parameters is fulfilled. Two complementary notions are used to answer this question:

1. Verification: Show that no system behavior is unsafe.

2. Falsification: Show that a specific system behavior is unsafe.

Crucially, formal verification methods must ensure no potential system behavior is missed. In contrast, falsification methods search for a specific system behavior that is unsafe. Since formal falsification requires rigorous proof that there are no deterministic model parameters within their respective uncertain sets that could lead to a safety violation, semi-formal falsification is much more common: Here, one usually employs search techniques to find suitable deterministic model parameters yielding a safety violation. If this search is successful, safety is provably violated; otherwise, the result is inconclusive. There also exist non-rigorous methods that only approximate the system behavior using less computationally demanding algorithms. They are mainly used to gain information and intuition about the system if formal methods fail to return results within a reasonable time.

Verification of dynamical systems is a difficult problem because we generally cannot exactly compute all possible system behaviors [1], also known as the exact reachable set of states. Therefore, one has to overestimate or underestimate the potential behaviors to maintain guarantees regarding safety: If a superset of the reachable states does not violate safety, this also holds for the exact reachable set. Conversely, any subset of the reachable states can immediately falsify safety since we only need a single instance of safety-violating behavior. Crucially, too much overestimation or underestimation of the potential behaviors may prevent safety verification or falsification, respectively, although a conclusive result on safety could be returned if the exact reachable set were available.

1.2 Literature Review

In this section, we review several approaches for the verification and falsification of dynamical systems. To this end, we introduce their main ideas and critically assess their advantages and drawbacks in three main categories: First, the generality of the system class to which the approach can be applied. Second, the scalability of the approach measured by its runtime complexity in the state dimension. Third, the quality of the results, that is, how accurately the approach approximates the exact reachable set.

Many of the methods reviewed below have been implemented in standalone tools[1], and we will explicitly mention them in the discussion of the corresponding approach. Some participate in the annual ARCH ($\underline{\text{A}}$pplied Ve$\underline{\text{r}}$ification for $\underline{\text{C}}$ontinuous and $\underline{\text{H}}$ybrid Systems) competition[2]. Here, verification tools compete with one another to correctly and efficiently solve challenging benchmarks, allowing for a fair comparison of the respective approaches.

[1]See `https://ieeecss.org/tc/hybrid-systems/tools` for a comprehensive list of tools for hybrid systems doing modeling/identification, verification, and controller synthesis.

[2]See `https://cps-vo.org/group/ARCH`.

Let us preface the following review by briefly listing groups of approaches related to our topic but outside of the scope of this thesis: A single trajectory can be enclosed using *validated integration* [2, 3]. Approaches for *invariant generation* [4–8] can also be used for verification since an invariant set represents the reachable set over an infinite time horizon for all initial states contained in that invariant set. Furthermore, we exclude data-driven approaches [9, 10], learning approaches [11], methods based on constraint propagation [12, 13], methods for stochastic systems [14–16] as implemented in `SReachTools` [17], and theorem-proving approaches [18, 19] as implemented in `Isabelle/HOL` [20], `dReach` [21], and `Keymaera/Keymaera X` [22, 23].

We first review methods using set-based integration as they are closest to the work in this thesis. Then, we discuss simulation-based approaches followed by methods based on abstraction refinement. Finally, we also review range-bounding methods and various approaches based on reformulations of reachability or verification as optimization problems.

Set-Based Integration Numeric integration methods for ordinary differential equations date back several centuries. To incorporate uncertainties, one needs to lift these pointwise methods [24, 25] to set-based methods [26] following the same idea: Integration of the right-hand side of the differential equation—but now over sets instead of single points—to obtain an explicit representation for the reachable set. Figure 1.1 illustrates the iterative propagation of time-point reachable sets and time-interval reachable sets that cover the time between two successive time points. By checking for an intersection between the reachable set and an unsafe set, safety can be verified or falsified.

In principle, there are two important design choices for the reachability algorithm: First, the approximation model, that is, how to simplify the dynamics to enable an integration over sets. Second, a set representation that is expressive enough to accurately represent the reachable set while efficiently evaluating set operations: Examples include polytopes [27], intervals [28], ellipsoids [29], support functions [30], zonotopes [31], constrained zonotopes [32], polynomial zonotopes [33], and Taylor models [34]. Tools for set-based computing include `CORA` [35], written in MATLAB, and `HyPro` [36], written in C++. A comprehensive discussion of set-based arithmetic for various set representations is provided in [37]. We divide the following overview into approaches for linear, nonlinear, and hybrid systems.

A) Linear Systems

Let us first consider the computation of forward reachable sets: An early approach for autonomous systems grids the state space and pushes an approximation computed using polytopes in vertex representation outward or inward to obtain outer or inner approximations, respectively [38, 39]. Exponential runtime complexity in the state dimension follows from gridding the state space. An alternative idea uses families of ellipsoids to approximate the reachable set of controllable systems with varying coefficients [40]: For an outer approximation, an intersection of ellipsoids, each of which touches the reachable set at its boundary in a certain direction, encloses the reachable set. In contrast, a union of ellipsoids touching the reachable set from the inside yields an inner approximation. The work in [41] uses polytopes in halfspace representation and restricts the input to a single trajectory—a necessary simplification

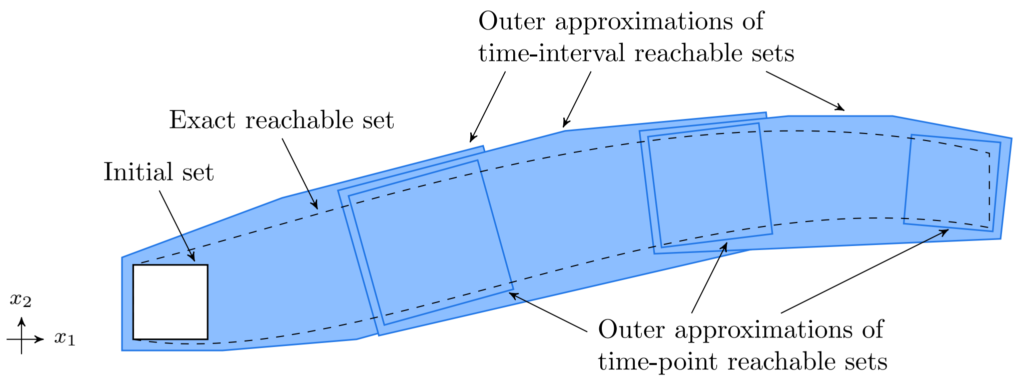

Outer approximations of time-interval reachable sets
Exact reachable set
Initial set
x_2
x_1
Outer approximations of time-point reachable sets

Figure 1.1: Reachability analysis using set propagation: Starting from the initial set, iterative propagation yields a sequence of time-point and time-interval reachable sets, whose union represents an outer (here) or inner approximation of the exact reachable set.

for the Minkowski sum of the homogeneous and particular solutions, which is known to scale exponentially in the set dimension for two polytopes in halfspace representation [42].

A major breakthrough was the introduction of zonotopes to reachability analysis [31, 43], as they allow for an efficient evaluation of the three main set operations—linear map, Minkowski sum, and convex hull—resulting in a reachability algorithm for linear systems with uncertain inputs that has only polynomial runtime complexity in the state dimension. One disadvantage of using zonotopes is the growth of the set representation size over subsequent time steps, which is mitigated by order reduction methods [44, 45]. The zonotope-based approach has also been extended to compute reachable sets for linear systems with parametric uncertainties [46, 47] and linear time-varying systems [47, 48]. A similarly efficient algorithm uses support function evaluations [30, 49], which implicitly describe reachable sets by bounds in pre-defined directions and can thus be used to construct polytopic outer approximations [50]. An extension is flowpipe sampling, where a set of interval-valued functions obtained from support functions capture the non-convexity of reachable sets covering time intervals and thereby drastically reduce the total number of sets [51]. Moreover, combinations of support functions and zonotopes have been explored in [52], where an analysis of the computational complexity suggests the use of support functions for the homogeneous solution and zonotopes for the particular solution.

The work in [49] proposes a wrapping-free approximation model, which avoids the amplification of approximation errors between time steps. A survey of different approximation models discussing the trade-off between computational efficiency and tightness is presented in [53]. Recently, polynomial zonotopes have been used to represent reachable sets of different subclasses of linear systems [54], allowing for a tight enclosure of large time intervals with just a single non-convex set. Zonotope-based methods are implemented in the MATLAB tool `CORA`, while approaches using support functions are implemented in the tools `SpaceEx` [55], written in C++, `JuliaReach` [56], written in Julia, and `XSpeed` [57, 58], written in C++, which exploits parallelization over the directions and partitioning of the time horizon.

As the state dimension grows, the computation of the matrix exponential for the propagation matrix becomes a limiting factor. One method to reduce the complexity is to approximate the matrix exponential in the Krylov subspace [59], allowing the computation of reachable sets for linear systems with up to a billion states [60]. Alternatively, one can also decompose the computation of the matrix exponential into smaller blocks [61], thereby exploiting the sparsity property exhibited by many high-dimensional systems. The decomposition method has also been extended to hybrid systems [62].

In comparison to forward reachability, the literature on backward reachability is so far limited to discrete-time systems: Inner and outer approximations of the minimal and maximal backward reachable sets are represented using ellipsoids in [63], along with controller synthesis techniques. Recent work proposes a more scalable approach to computing inner approximations of the maximal backward reachable set by using zonotopes through novel methods for inner approximating the Minkowski difference between two zonotopes [64].

B) Nonlinear Systems

For nonlinear systems, the right-hand side of the differential equation has to be simplified to a linear or polynomial difference equation to enable an integration over sets. *Hybridization* approaches partition the state space into different regions with simpler dynamics that are sound abstractions of the local nonlinear behavior in that region [65]. Early approaches abstracted the dynamics to constant differential inclusions, as implemented in the tools `HyTech` [66], written in Mathematica/C++, and `PHAVer` [67], written in C++. Using linear differential inclusions instead significantly reduces the induced approximation errors [68]. The originally static partitioning of the state space has been replaced by a dynamic abstraction [69, 70], and improvements to error bounds are presented in [71]. Recent work proposes a method to scale the dynamics in the vicinity of a transition from one region into another to mitigate the induced approximation error by the transition [72].

In contrast, *on-the-fly* abstraction methods approximate the local dynamics of the current time step by a low-order polynomial and overestimate the influence of the higher-order dynamics with a Lagrange remainder [73]. This concept has also been applied to nonlinear differential-algebraic systems [74, 75], which extend standard nonlinear dynamics by a constraint equation with additional algebraic variables. Furthermore, one can decompose the Lagrange remainder into smaller, weakly coupled blocks to accelerate the analysis without compromising the tightness of the resulting reachable sets [76]. The influence of higher-order polynomial abstractions can be accurately captured using polynomial zonotopes as they can represent non-convex sets [77]. By exploiting the dependency preservation of polynomial zonotopes, one can swiftly extract subsets of reachable sets without re-computing the reachable set [78]. These approaches are implemented in the MATLAB tool `CORA`.

While the above approaches compute outer approximations, methods for obtaining inner approximations are more scarce: The work in [79] represents the reachable set as a union of boxes utilizing a sufficient condition to decide whether a box is contained in the image of a nonlinear map [80, 81]. This approach requires an outer approximation of the reachable set, which is computed using Taylor models [34], and its Jacobian, computed using a generalization of affine arithmetic [82]. A similar method, also based on enclosing the state and its derivative, yields inner approximations of projections, which was introduced for autonomous systems [83]

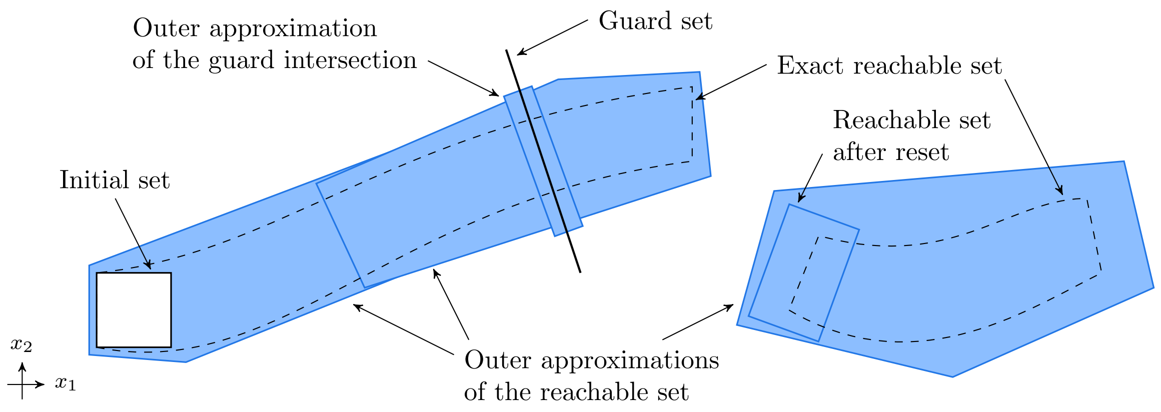


Figure 1.2: Reachability analysis for hybrid systems using set propagation: Since the reachable set within discrete states can be computed using approaches for linear and nonlinear systems, approaches for hybrid systems mainly focus on computing tight approximations of the intersection between the computed reachable set and the guard set.

and later extended to systems with competing inputs and disturbances [84]. For the case of piecewise constant inputs, the limitation to projections has meanwhile been overcome [85].

Inner approximations of the reachable set for autonomous systems can also be obtained by computing outer approximations for the boundary of the initial set; the contained set then represents the inner approximation [86]. After partitioning the boundary into intervals and computing an enclosure of their propagation, contraction via linear programming yields a polytopic inner approximation [87]. An improved variant of the same idea uses sum-of-squares programming to obtain semi-algebraic sets as inner approximations [88]. Both versions suffer from poor scalability due to partitioning the boundary of the initial set. Another approach [89] uses polynomial zonotopes to compute two outer approximations of the reachable set, one using the initial set and another one using only the boundary. The first outer approximation is then shrunk—exploiting that polynomial zonotopes preserve dependencies [78]—until it does not intersect the second outer approximation. This approach has later been extended to deal with inputs [37]. A zonotope-based method for backward reachability of discrete-time systems has been proposed in [90], which focuses on tight inner approximations of the Minkowski difference between a constrained zonotope and a zonotope, as this operation dominates the approximation error of the resulting backward reachable set.

C) Hybrid Systems

Many approaches for linear and nonlinear systems can be used for reachability analysis within the discrete states of hybrid systems. Thus, the main focus lies in evaluating the transition between two discrete states, where the reachable set intersects a guard set and a reset function maps the resulting intersection. As indicated in Figure 1.2, the intersection of the computed reachable set with the guard set generally induces large approximation errors. Consequently, much effort has been dedicated to obtaining tight approximations of the guard intersection.

Predominantly, intersections with guard sets modeled as hyperplanes have been considered: This includes approaches that use template polytopes to enclose the intersection with a zonotopic reachable set [91], directly map the reachable set onto the guard [92], apply domain contraction and range bounding [93], or use convex optimization to compute the intersection, followed by computing the union of reachable sets after the transition to maintain computational efficiency [94]. Another approach scales the dynamics of the flow equation so that only one reachable set intersects the guard set [95], omitting the need for unification later on. Using a combination of contractors and polynomial zonotopes allows for an evaluation of transitions with nonlinear guard sets that has polynomial runtime complexity in the state dimension [96]. Another way to refine the reachable set is to remove spurious transitions resulting from the intersection of states that are part of the outer approximation but not the exact reachable set [97]. To our best knowledge, there exists no guard intersection method that computes an inner approximation of the exact reachable set after the transition, as required for falsification. Many of the tools for linear and nonlinear systems also implement extensions of their respective approaches to hybrid systems using some of the presented guard intersection techniques.

Simulations Single trajectories of the dynamical system can be used as a basis for both verification and falsification: Since the uncertainties in the model parameters vary continuously, verifying safety would require an infinite number of simulations. Thus, many verification approaches expand a finite number of simulated trajectories to sets that cover all possible behaviors. In contrast, a single simulation suffices to falsify safety if the trajectory reaches the unsafe set.

A) Approaches for Verification

Coverage metrics aim to bridge the gap between single simulations and verification by smartly covering the initial set [98, 99]: The notion of star discrepancy describes the degree to which the sampled initial states are equally distributed over the initial set. This metric guides simulations from rapidly exploring random trees toward the unsafe set. While this results in a balanced state exploration, verification requires a quantitative bound on the missing coverage.

Different metrics have been used to obtain sound outer approximations using a finite number of simulations. For linear discrete-time systems with outputs, the work in [100] solves linear matrix inequalities to obtain a bisimulation metric, which is further used in a complete verification algorithm with an upper bound on the number of required simulations. Another approach uses expansion functions that measure how small changes in the initial state affect the resulting trajectory [101]. For linear autonomous systems, sensitivity analysis can be used to expand the trajectories, resulting in a sound outer approximation of the reachable set, as illustrated on the left-hand side of Figure 1.3. Increasing the number of simulations then yields a smaller expansion for each trajectory, which gives rise to an automated verification algorithm based on iterative refinement. This sensitivity-based approach was later extended to inputs [102]. The notion of sensitivity is generalized in [103] to nonlinear dynamics by applying discrepancy functions, such as Lipschitz measures or contraction metrics. These need to be provided by the user for each system to obtain sound outer approximations. The C++/Python tool `C2E2` [104] implements this approach.

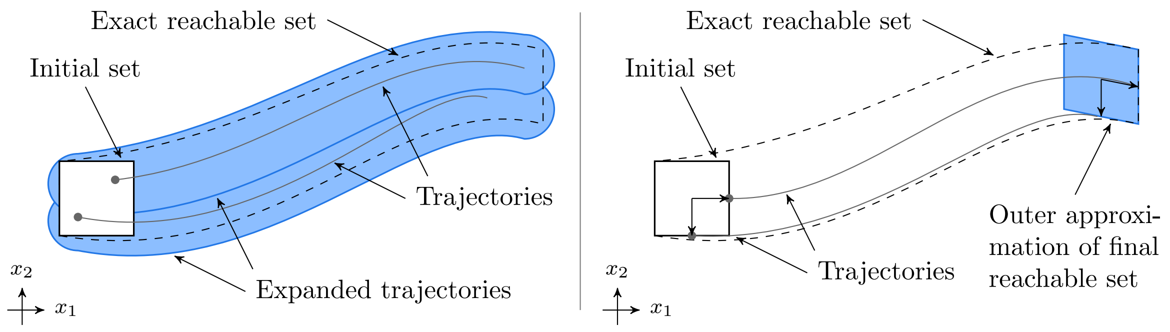


Figure 1.3: Reachability analysis using simulations (inspired by [48, Fig. 1.2] and [105, Fig. 2]): Enlarging a finite number of trajectories by an expansion function yields an outer approximation of the exact reachable set (left); for linear systems, one can exploit the superposition principle to drastically reduce the number of required simulations (right).

The above approaches based on sensitivity analysis suffer from the curse of dimensionality, as an exponential number of initial states is required for an adequate coverage of the initial set. However, one can obtain a polynomial runtime complexity in the state dimension for linear systems by exploiting the superposition principle [105, 106]: For instance, an n-dimensional system with a box as an initial set now only requires $n + 1$ simulations. These are then used to construct inner and outer approximations represented by star sets, as illustrated on the right-hand side of Figure 1.3. This flexible set representation seamlessly deals with non-convex and unbounded initial sets [105]. The approach is implemented in the Python tool `HyLAA` [107].

Individual simulation runs are particularly powerful in the reachability analysis of monotone systems [108]. Here, the derivative of the dynamic function is monotone with a specific ordering. Consequently, the reachable set can be constructed using an interval enclosure of the simulated trajectories emanating from the two initial states, which are minimal and maximal in that ordering [109]. This idea has been extended to mixed-monotone systems, where the system dynamics are decomposed into increasing and decreasing components [110, 111]. It has been shown in [112] that many functions are mixed-monotone, although it is non-trivial to find a decomposition function that rewrites a system as a mixed-monotone system. In general, the enclosure of only two trajetories by an interval yields a coarse outer approximation, although the sets are tight for special cases [111]. The MATLAB tool `TIRA` [113] implements interval-based reachability analysis for various subclasses of monotone systems.

B) Approaches for Falsification

The main idea in safety falsification using simulations is to sample deterministic model parameters to eventually obtain a trajectory that falsifies a given specification. For linear systems, one can exploit the superposition principle to maximize the trajectory in a given direction using support function evaluations [114]. For more general system classes, the notion of *robustness* of trajectories is used to recast the sampling problem as an optimization problem [115, 116]. Figure 1.4 illustrates the main idea: The robustness is a scalar value associated with a trajectory, which describes how close it is to violating a given specification. Consequently, minimizing

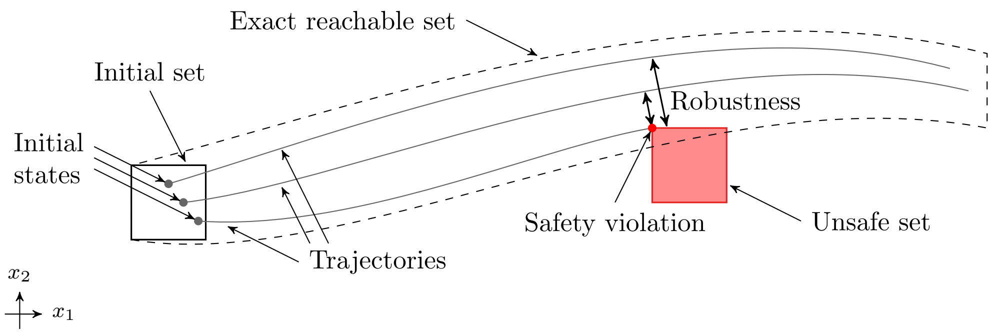


Figure 1.4: Simulation-based falsification based on robustness minimization: The robustness decreases from the uppermost to the lowest trajectory, leading to a safety violation.

the robustness value corresponds to sampling model parameters that yield increasingly critical trajectories.

The main workflow for robustness-based falsification consists of three steps: Sampling of deterministic model parameters, simulation of the trajectory over the entire time horizon, and robustness computation. Since the robustness minimization problem is non-convex [117], stochastic global optimization techniques are applied: The biologically inspired ant colony optimization was extended by Hermite functions and cubic splines for the parametrization of the input space [117]. Monte-Carlo sampling has also been applied [118, 119], as well as simulated annealing, which has been proven to converge to a global minimum [120]. Furthermore, the cross-entropy method samples the initial states and inputs according to a robustness distribution that is iteratively updated using the robustness value of the next simulation [121]. An improvement to the robustness computation for hybrid systems is proposed in [122], addressing the issue that a continuous state has different robustness values depending on the discrete state of the system. Multiple objective functions can help to avoid converging to local minima and reduce the number of simulations until finding a falsifying trajectory [123]. For falsifying several specifications at once, the structure of their composition can be exploited using Bayesian optimization [124]. Many of these approaches have contributed to the growth of the MATLAB tool `S-TaLiRo` [125]. Recently, robustness-guided falsification has been recast as a reinforcement learning problem, exploiting that neural networks can approximate any nonlinear function, in this case, the robustness [126].

Abstraction Refinement Complex systems are often time-consuming to analyze, even though a conservative abstraction might suffice to quickly verify safety. Abstraction refinement methods gradually increase the computational effort by iteratively refining coarse models or representations. The well-known concept of *counterexample-guided abstraction refinement* (CEGAR) uses counterexamples generated from an abstracted system to propose directions in which to refine the abstraction: Safety is verified by proving no unsafe behavior for the abstraction and falsified by concretizing an abstract counterexample to a safety-violating system trajectory.

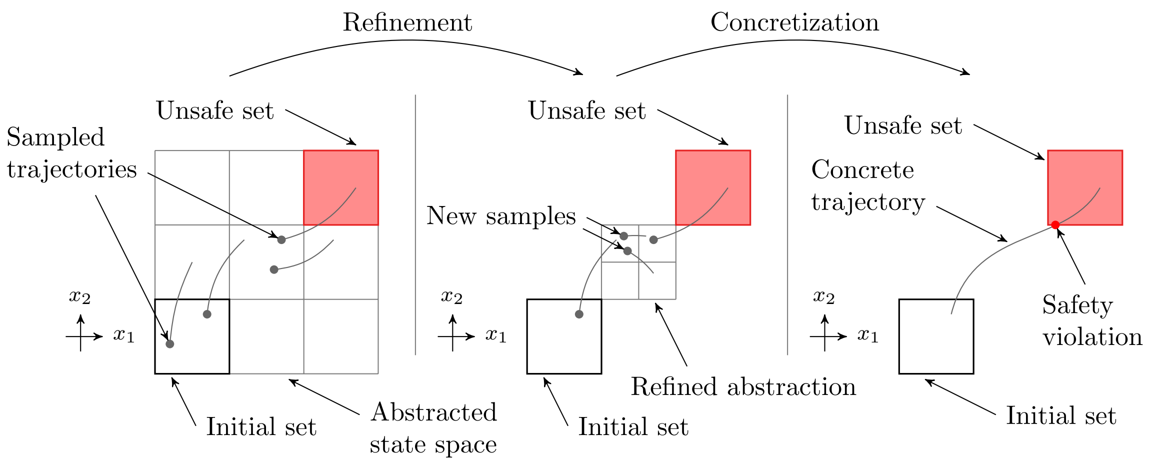


Figure 1.5: Falsification using abstraction refinement (inspired by [130, Fig. 1]): Individual partial trajectories link cells to one another until an abstract path from the initial set to the unsafe set is found. Refinement of the partial trajectories eventually leads to a concrete safety-violating trajectory.

An early application of this concept abstracted hybrid automata to finite state automata and proposed various validation methods for the generated counterexample [127]. Another abstraction method grids the state space and uses transitions between cells for forward and backward reachability to verify cyclic invariants for oscillating dynamics [128]. An abstraction-based approach for falsification is illustrated in Figure 1.5: To this end, the state space is abstracted to cells and an abstract counterexample—a trajectory that reaches the unsafe set by transitioning from cell to cell—is iteratively refined until it can be concretized to a safety-violating trajectory [129, 130]. Commonly, methods that abstract the state space to cells suffer from poor scalability due to the curse of dimensionality.

Instead of abstracting and refining the system itself, one can also refine the algorithm parameters that strongly influence the outcome of the reachability analysis. The work in [131] refines the directions of the support function evaluation used in the reachability algorithm based on a cost function similar to the robustness mentioned above. This idea has been extended by automatically generating template directions from spurious counterexamples [132]. Another method incorporates real-time constraints by allowing a switch to a less conservative controller if it can be verified online by a tight enough reachable set computed within the available computation time [133]. The tool `Hypy` [134] presents a high-level framework, which proposes refined algorithm parameters given the reachable sets obtained from different toolboxes. A similar general framework for algorithm parameter refinement is proposed in [135] to reduce the approximation errors.

Range Bounding Range-bounding methods compute outer or inner approximations of the image of a function over a domain of values. Common techniques include interval arithmetic [28], affine arithmetic [82], or Taylor models [34], which represent non-convex sets by a vector-

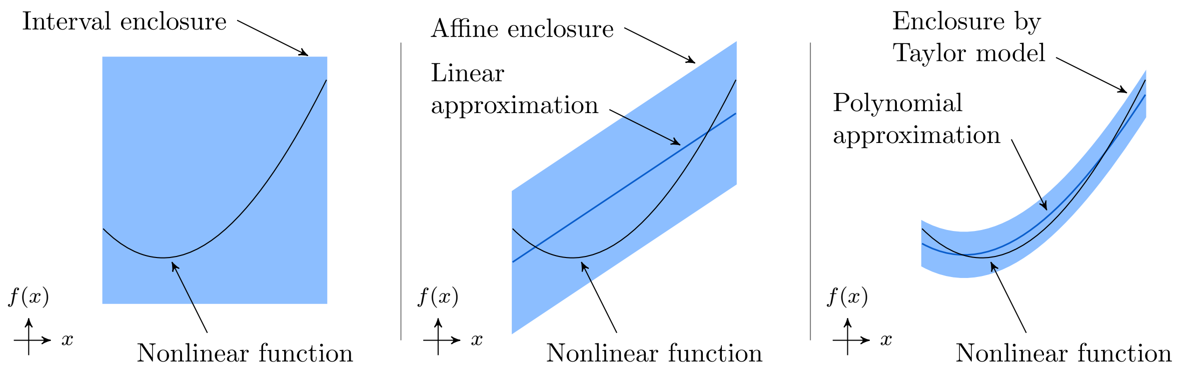


Figure 1.6: Range bounding of a scalar nonlinear function using interval arithmetic (left), affine arithmetic (middle), and Taylor models (right).

valued polynomial function enlarged by an interval [34]. Figure 1.6 illustrates the enclosure of a nonlinear function with several range-bounding methods.

For discrete-time systems, one bound the nonlinear function over the range of the current reachable set to compute the successor reachable set. A popular method to obtain tight enclosures is the Bernstein expansion of nonlinear functions since the approximation error with respect to the exact solution is known. For a multivariate polynomial function, the coefficients of the corresponding Bernstein expanded polynomial has exponential runtime complexity in the state dimension [136].

The image of polynomial maps was first bounded using Bézier simplices and polynomial optimization [137]. Using the Bernstein expansion of polynomials, one can enclose the successor reachable set using linear programming instead of polynomial programming; additionally, template polytopes replaced the previous triangulation method [138]. This method restricts the preimage of the nonlinear map to boxes, which was later extended to arbitrary polytopes [139, 140]. Improvements to the tightness of the enclosure include using piecewise affine functions for bounding the image and dynamic selection of template directions for the polytopes representing the reachable set [139]. Template polytopes were later replaced by parallelotope bundles [141, 142], which are the intersection of a set of parallelotopes. The tool `Sapo` [143], written in C++, implements this version of the approach. The automated selection of the normal vectors that compose the individual parallelotopes is addressed in [144]. This approach was implemented in the Python tool `Kaa` [145].

Reachable set computations using Bernstein polynomials have also been applied to autonomous linear systems in continuous time [146]: In this case, the dynamics can be integrated analytically, and the resulting function is enclosed over a time interval using Bernstein polynomials. The reachable set is obtained via the convex hull of all propagated vertices, which is subsequently enlarged by the approximation error of the Bernstein polynomial with respect to the exact integration. Notably, this approach adaptively tunes the time step size by enforcing a bound on the approximation error. However, the runtime complexity is exponential in the state dimension due to using the vertices of the initial set, which suffers from the curse of dimensionality.

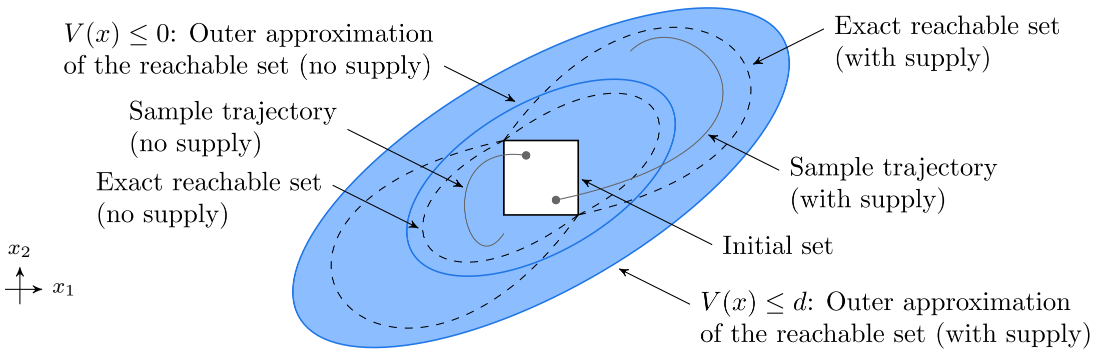
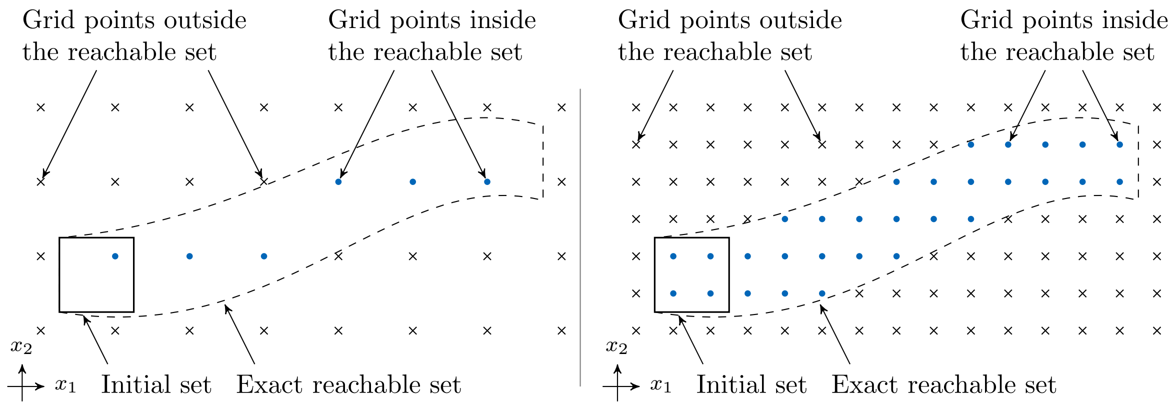

$V(x) \leq 0$: Outer approximation of the reachable set (no supply)
Sample trajectory (no supply)
Exact reachable set (no supply)
x_2
x_1
Exact reachable set (with supply)
Sample trajectory (with supply)
Initial set
$V(x) \leq d$: Outer approximation of the reachable set (with supply)

Figure 1.7: Reachability analysis of dissipative systems (inspired by [150, Fig. 1.3]): Outer approximations of the exact reachable set are represented using sublevel sets of the storage function $V(x)$, whose elevation depends on the norm of the supply rate d to the system.

For nonlinear systems in continuous time, one can combine the *Picard iteration* with range bounding using Taylor models: A polynomial solution trajectory of fixed degree can be approximated using the Picard iteration or a truncated series of Lie derivatives [93]. This approximative polynomial solution needs to be enlarged by an interval remainder term, which has to be large enough to contract if inserted into the Picard iteration, to obtain a sound outer approximation. Automated tuning methods for the time step size and the degree of the approximating polynomial have been proposed [147]. Furthermore, decomposition methods have been leveraged to improve the scalability of the approach [148]. An inner approximation can be obtained by computing outer approximations using the boundary as the initial set and then identifying the contained set [86]: We first compute a Taylor model enclosing the reachable set for each constraint of a semi-algebraic initial set. Then, the interval remainder terms are adapted to push the reachable sets inward until their intersection is contained in the exact reachable set, which is validated via interval constraint propagation. The tools `Flow*` [149], written in C++, and `JuliaReach`, written in Julia, implement approaches using the Picard iteration and Taylor models.

Dissipativity Another group of approaches is based on the notion of dissipativity [150]. This property holds for systems for which there exists a storage function $V : \mathbb{R}^n \mapsto \mathbb{R}$ fulfilling the following conditions: The value must be 0 at the origin and nonnegative everywhere else; additionally, the difference between two subsequent states along a trajectory is bounded via a supply rate, which models, e.g., disturbances. These conditions allow for reformulating reachability to an optimization problem with dissipation inequalities [151] whose solution is the storage function that represents the reachable set. Ellipsoidal storage functions are illustrated in Figure 1.7 for a system with and without supply. As an ansatz for the storage function, one often uses a parametrized semi-algebraic set, which is restricted to sum-of-squares polynomials. Consequently, we obtain an optimization problem that can be relaxed to a sum-of-squares decomposition, which can be modeled as a semi-definite program yielding the coefficients for the polynomial storage function that encodes the reachable set. The resulting framework can



Grid points outside the reachable set
Grid points inside the reachable set
Grid points outside the reachable set
Grid points inside the reachable set
x_2
x_1
Initial set
Exact reachable set
x_2
x_1
Initial set
Exact reachable set

Figure 1.8: Reachability analysis by evaluating a reformulation to the Hamilton-Jacobi-Isaacs partial differential equation over a gridded state space. A coarse grid (left) is computationally less demanding, while a refined grid (right) better captures the shape of the reachable set.

be applied to linear and nonlinear dynamics and incorporate multiple sources of uncertainty, e.g., time-varying parameters. However, the number of variables of the semi-definite program scales polynomially in the state dimension but exponentially in the degree of the polynomial representing the storage function [152]. Another disadvantage is the dependence on a manually defined ansatz function.

This dissipativity-based approach has been applied to compute outer approximations of forward reachable sets [151] and inner approximations of backward reachable sets [153–155], which are predominantly used for controller synthesis. The integration of integral-quadratic constraints into the framework [151, 154] widened the applicability to more general systems, as these constraints can also account for system behavior induced by unmodeled dynamics.

Hamilton-Jacobi-Isaacs Equation The reachable set of a dynamical system has proven to be equivalent to the zero sublevel set of the solution to a particular Hamilton-Jacobi-Isaacs partial differential equation [156]. Thus, one can reformulate the reachable set computation as an optimization problem, commonly known as *Hamilton-Jacobi* reachability analysis [157–159]. The optimization problem is then solved over a gridded state space, see Figure 1.8: Grid points with a negative value are part of the reachable set, while grid points with a positive value are outside of the reachable set. Due to the curse of dimensionality induced by gridding the state space, the analysis scales exponentially in the state dimension [158]. Nevertheless, the framework is very flexible as it allows for computing forward and backward reachable sets for linear and nonlinear dynamics while also respecting state constraints [160].

Significant focus has been dedicated to alleviating the computational burden of Hamilton-Jacobi reachability: To this end, techniques to decompose unperturbed dynamics and subsequently reconstruct the solution were explored in [161, 162], with an extension to mutually dependent inputs between subsystems in [163]. Furthermore, decoupling approaches for perturbed dynamics were investigated [159, 164]. Another way to reduce the computational effort

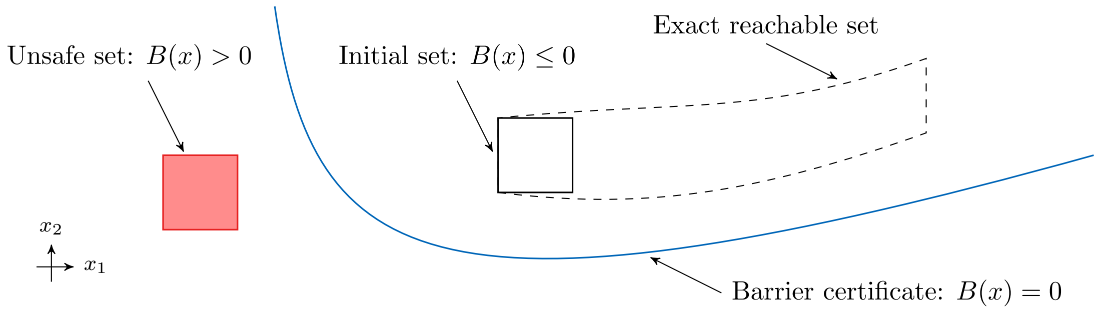


Figure 1.9: Verification using barrier certificates (inspired by [172, Fig. 1]): The zero level set of the barrier certificate $B(x)$ separates the reachable set from the unsafe set.

is by computing outer and inner approximations instead of using the exact reformulation [165]. Approximations of polynomial solutions to the reformulated Hamilton-Jacobi equation for autonomous systems can be computed by sum-of-squares programming, resulting in semi-algebraic reachable sets [166]. This approach has later been extended to deal with inputs as well [167]. The reachable set computation was originally implemented in the MATLAB tool `helperOC`[3] using level set arithmetic from the MATLAB tool `toolboxls` [168] or the C++ implementation in `beacls`[4] and recently refurbished as the Python package `hj-reachability`[5]. Recent work also integrates machine learning techniques to estimate the value function, which improves the scalability but relinquishes all formal guarantees [169, 170]. The Python tool `DeepReach` [171] implements these learning-based approaches.

Barrier Certificates The concept of barrier certificates is to prove safety by finding a level set separating the reachable set and the unsafe set, as illustrated in Figure 1.9. A barrier certificate $B : \mathbb{R}^n \mapsto \mathbb{R}$ must fulfill three conditions [172]: The value at every initial state must be nonpositive, the value at every unsafe state must be positive, and the Lie derivative of the dynamic function is nonpositive on the level set $B(x) = 0$. These conditions accommodate many different system classes—including nonlinear and hybrid dynamics—and can also incorporate constraints. For polynomial dynamics and semi-algebraic initial and unsafe sets, the conditions can be encoded in an optimization problem. We obtain a semi-definite program via sum-of-squares decomposition, which suffers from the same scalability issues mentioned for dissipativity-based approaches above.

Main issues include the conservativeness of the approach as well as the complexity of the optimization problem: To improve the scalability, the structure of interconnected subsystems can be exploited by composing the conditions for each system individually and accounting for their interdependency via additional constraints [173]. More expressive barrier certificates can be synthesized by altering the condition on the Lie derivative [174]. One can also exploit

[3]Available at `https://github.com/HJReachability/helperOC`.
[4]Available at `https://github.com/HJReachability/beacls`.
[5]Available at `https://pypi.org/project/hj-reachability`.

the structure of the semi-definite program and combine two candidate functions into a barrier certificate [175]. Recently, neural networks have been used to synthesize barrier certificates for autonomous systems [176, 177]: Given training data sampled from the initial set, the unsafe set, and the considered domain, a neural network with rectified linear units as activation functions is trained using the three conditions from above in the loss function. The trained network is, however, only a candidate barrier certificate, as one has to validate that the three conditions hold for all states within the respective sets. The validation can be cast as an optimization problem and solved using interval constraint propagation [176] or mixed-integer linearly/quadratically constrained programming [177]. Finally, different types of specifications other than safety have been addressed in [178]. A popular extension called *control barrier functions* combines barrier certificates with control Lyapunov functions to synthesize safety-preserving controllers [179, 180].

Current Challenges Despite their shared goal of proving safety, the verification and falsification approaches reviewed in this section differ strongly from one another. Let us now analyze the challenges faced by current approaches using forward and backward reachability analysis.

A) Verification using Forward Reachability

Safety verification of linear systems is mainly addressed by simulation-based approaches or set-based integration techniques due to their low polynomial runtime complexity. Still, these methods often require manual tuning of algorithm parameters—mainly the time step size—until the reachable set is tight enough to verify or falsify safety. Furthermore, the overwhelming majority of reachability algorithms do not provide quantitative information on the tightness of the reachable set. This complicates the manual tuning as one can never be sure whether different values for the algorithm parameters could return a conclusive result on safety. Also, inner and outer approximations are usually not computed jointly, so one would have to run to different algorithms in parallel.

Nonlinear systems are notoriously difficult to analyze, which aggravates the issue of manual parameter tuning that all approaches require, as detailed in the annual ARCH reports [W9, W11, W13]. Simulation-based reachability algorithms require additional information about the system, such as contraction metrics, to compute sound outer approximations. They generally scale unfavorably to higher-dimensional systems due to an exponential number of sampled initial states. The sum-of-squares framework used for dissipativity-based methods requires a user-defined degree for the polynomial as well as a manually parametrized ansatz for the storage function describing the reachable set. Approaches evaluating the Picard iteration using Taylor models have been extended by automated parameter tuning based on manually defined threshold values on the size of the interval remainder as well as upper and lower bounds for the time step size[147]. Nevertheless, the evaluation of ARCH benchmarks reveals that many algorithm parameters, including the degree of the approximating polynomial, are still fixed for each system [W11]. The computation of barrier certificates also depends on the suitable parametrization for the sum-of-squares program synthesizing the certificate.

In summary, the main drawback of current approaches using forward reachability for safety verification of dynamical systems is their dependence on the algorithm parameters. Consequently, practitioners require expert knowledge for the tuning of the respective algorithms in order to obtain tight outer or inner approximations of the exact reachable set that suffice to verify or falsify safety. Ideally, an algorithm could automatically adapt the algorithm parameters to the dynamics and the model parameters, such that the reachable set approximations are tight enough for the given safety specification.

B) Verification using Backward Reachability

For systems with competing control inputs and disturbances, mainly dissipativity-based approaches and Hamilton-Jacobi reachability are used to compute backward reachable sets. While both frameworks can handle very general cases, encompassing linear and nonlinear dynamics and time-varying parameters, their scalability is limited: The optimization problem constructed by dissipation inequalities is solved via sum-of-squares programming, whose runtime complexity is exponential in the degree of the polynomial [152]. The reformulated Hamilton-Jacobi-Issacs equation is evaluated over a gridded state space, leading to an exponential runtime complexity in the state dimension [158]. Furthermore, the approximation error of the reachable sets computed using Hamilton-Jacobi reachability depends on the grid resolution and the discretization error in the solution of the partial differential equation. Both approaches, however, often utilize the computation of backward reachable sets as an intermediate step toward synthesizing safety-preserving controllers [155, 181]. An alternative approach computes backward reachable sets using set propagation for discrete-time systems [64, 90], which cannot be used for safety verification of a continuous bounded time horizon.

Compared to forward reachability, safety verification using backward reachability has yet to be thoroughly studied. In particular, there are no backward reachability algorithms using set propagation for continuous-time systems to date, and current approaches suffer from poor scalability in the state dimension.

1.3 Contributions

In this thesis, we focus on verify and falsifying safety specifications for dynamical systems modeled by linear and nonlinear ordinary differential equations. We use set-based integration methods, which represent a set-based analog to classical solvers for ordinary differential equations, to compute tight reachable sets sufficiently fast and embed the computation in a subsequent verification algorithm. In detail, we contribute the following:

Appendices A.1 and A.2 For a state-of-the-art reachability algorithm for linear systems using zonotopes, we rigorously derive bounds for all induced approximation errors. Furthermore, we determine how different algorithm parameters, such as the time step size, affect these errors and analyze the convergence behavior in the limiting values of said algorithm parameters. Consequently, the tightness of the outer approximation with respect to the exact reachable set is now known, enabling us to extract an inner approximation with the same bound on its approximation error. By combining these error bounds with automated tuning methods for the individual algorithm parameters, we can compute outer and inner approximations that respect any given admissible error bound larger than zero. Next, we iteratively refine this error bound using a set-based distance between the unsafe sets and the current reachable set approximation to devise an automated verification algorithm. This allows us to verify or falsify safety for any safety specification that does not require the exact reachable set since we can fulfill all error bounds apart from zero.

Appendix A.3 Some safety specifications do not require explicit sets but can be verified or falsified using support function evaluations only. Thus, we extend a reachability algorithm using support functions by automated parameter tuning methods, which exploit the convergence behavior to the exact reachable set for a decreasing time step size, resulting in automated verification algorithms for unsafe sets represented by halfspaces or arbitrary convex sets. Since we only require support function evaluations of the homogeneous and particular solutions composing the reachable set, these algorithms work for many different set representations. Additionally, we show a significant acceleration of the propagation if the homogeneous and particular solutions can be pre-computed using an explicit set representation, such as zonotopes. Furthermore, we can extract a disturbance trajectory that yields a safety-violating counterexample. The developed verification algorithms are capable of verifying and falsifying safety for linear systems whose state dimensions are too large for state-of-the-art approaches.

Appendix A.4 We propose the first backward reachability algorithms for perturbed continuous-time linear systems using set propagation by combining polytopes, zonotopes, and constrained zonotopes. The soundness of the outer and inner approximations is proven so that they are amenable to solving verification tasks. Moreover, we analyze the approximation errors and discuss refinement methods to obtain tighter reachable sets. The low polynomial runtime complexity in the state dimension is a significant improvement over state-of-the-art methods, allowing for the analysis of linear systems with over a thousand states.

Appendices A.5 and A.6 For nonlinear systems, we thoroughly analyze the abstraction error for a set-based reachability algorithm based on on-the-fly abstraction. The critical dependence of the abstraction error on the time step size is formalized by our novel concept called *gain order*, which measures the ratio of the abstraction error for two different time step sizes. Analyzing the limit behavior reveals a linear gain for many nonlinear dynamics, which entails a lower bound on the time step size, as other approximation errors incurred by the set-based computation outweigh the additional gain achieved by reducing the time step size. In other words, we show that continuously reducing the time step size does not yield a monotonic improvement in the tightness of the reachable set approximation. Using our insights about the behavior of the abstraction error and the effects of other algorithm parameters, we formulate a local optimization problem to determine the time step size. Other algorithm parameters are tuned using threshold criteria and comparing various abstraction orders for the nonlinear dynamics to swiftly adapt to stronger or milder nonlinearities in the state space. We utilize these tuning methods to develop the first reachability algorithm based on set propagation that runs fully automatically and verifies safety specifications for nonlinear systems without requiring expert knowledge about the reachability algorithm.

Below is the list of publications reprinted in Appendices A.1 to A.6:

[W1] **M. Wetzlinger**, N. Kochdumper, and M. Althoff. "Adaptive parameter tuning for reachability analysis of linear systems". In: *Proc. of the 59th Conference on Decision and Control.* IEEE, 2020, pp. 5145–5152. DOI: `10.1109/CDC42340.2020.9304431`.

[W2] **M. Wetzlinger**, N. Kochdumper, S. Bak, and M. Althoff. "Fully automated verification of linear systems using inner and outer approximations of reachable sets". In: *IEEE Transactions on Automatic Control* 68.12 (2023), pp. 7771–7786. DOI: `10.1109/TAC.2023.3292008`.

[W3] **M. Wetzlinger**, N. Kochdumper, S. Bak, and M. Althoff. "Fully-automated verification of linear systems using reachability analysis with support functions". In: *Proc. of the 26th International Conference on Hybrid Systems: Computation and Control.* ACM, 2023. DOI: `10.1145/3575870.3587121`.

[W4] **M. Wetzlinger** and M. Althoff. "Backward reachability analysis of perturbed continuous-time linear systems using set propagation". In: *arXiv preprint arXiv:2310.19083* (2023). DOI: `10.48550/arXiv.2310.19083`.

[W5] **M. Wetzlinger**, A. Kulmburg, and M. Althoff. "Adaptive parameter tuning for reachability analysis of nonlinear systems". In: *Proc. of the 24th International Conference on Hybrid Systems: Computation and Control.* ACM, 2021. DOI: `10.1145/3447928.3456643`.

[W6] **M. Wetzlinger**, A. Kulmburg, A. Le Penven, and M. Althoff. "Adaptive reachability algorithms for nonlinear systems using abstraction error analysis". In: *Nonlinear Analysis: Hybrid Systems* 46 (2022), p. 101252. ISSN: 1751-570X. DOI: `10.1016/j.nahs.2022.101252`.

The following publications are reports of the ARCH competition from 2020 to 2023, in which the author has participated using the tool `CORA`. Some of the novel algorithms introduced in the publications reprinted in Appendix A have been successfully applied to the state-of-the-art benchmarks of the ARCH competition, as discussed in the corresponding numerical evaluation sections.

[W7] M. Althoff, S. Bak, Z. Bao, M. Forets, G. Frehse, D. Freire, N. Kochdumper, Y. Li, S. Mitra, R. Ray, C. Schilling, S. Schupp, and **M. Wetzlinger**. "ARCH-COMP20 category report: Continuous and hybrid systems with linear continuous dynamics". In: *Proc. of the 7th International Workshop on Applied Verification of Continuous and Hybrid Systems.* EasyChair. 2020, pp. 16–48. DOI: `10.29007/7dt2`.

[W8] M. Althoff, E. Ábrahám, M. Forets, G. Frehse, D. Freire, C. Schilling, S. Schupp, and **M. Wetzlinger**. "ARCH-COMP21 category report: Continuous and hybrid systems with linear continuous dynamics". In: *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems.* EasyChair. 2021, pp. 1–31. DOI: `10.29007/lhbw`.

[W9] L. Geretti, J. Alexandre Dit Sandretto, M. Althoff, L. Benet, A. Chapoutot, P. Collins, P. S. Duggirala, M. Forets, E. Kim, U. Linares, D. P. Sanders, C. Schilling, and **M. Wetzlinger**. "ARCH-COMP21 category report: Continuous and hybrid systems with nonlinear dynamics". In: *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair, 2021, pp. 32–54. DOI: 10.29007/2jw8.

[W10] M. Althoff, M. Forets, C. Schilling, and **M. Wetzlinger**. "ARCH-COMP22 category report: Continuous and hybrid systems with linear continuous dynamics". In: *Proc. of the 9th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair. 2022, pp. 58–85. DOI: 10.29007/mmzc.

[W11] L. Geretti, J. Alexandre Dit Sandretto, M. Althoff, L. Benet, A. Chapoutot, P. Collins, P. S. Duggirala, M. Forets, E. Kim, S. Mitsch, C. Schilling, and **M. Wetzlinger**. "ARCH-COMP22 category report: Continuous and hybrid systems with nonlinear dynamics". In: *Proc. of the 9th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair, 2022, pp. 86–112. DOI: 10.29007/fnzc.

[W12] M. Althoff, M. Forets, Y. Li, C. Schilling, **M. Wetzlinger**, and D. Zhuang. "ARCH-COMP23 category report: Continuous and hybrid systems with linear continuous dynamics". In: *Proc. of the 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair. 2023, pp. 34–60. DOI: 10.29007/nl86.

[W13] L. Geretti, J. Alexandre Dit Sandretto, M. Althoff, L. Benet, P. Collins, M. Forets, E. Ivanova, Y. Li, S. Mitra, S. Mitsch, C. Schilling, **M. Wetzlinger**, and D. Zhuang. "ARCH-COMP23 category report: Continuous and hybrid systems with nonlinear dynamics". In: *Proc. of the 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair, 2023, pp. 61–88. DOI: 10.29007/93f2.

The remainder of this work is structured as follows: In Chapter 2, we will first formally introduce the problem statement for the verification tasks. Then, we will outline our solution concept for the automated safety verification of dynamical systems using reachability analysis. Appendix A contains the reproduction of the six included publications [W1–W6], which represent the main body of this thesis. Finally, we will summarize and critically discuss the work in Chapter 3 and briefly describe potential directions for future work.

2 Automated Verification Using Reachability Analysis

In this chapter, we formally define the considered system classes, the computed reachable sets, and the verification tasks. Finally, we concisely outline our solution concept for safety verification, which underlies all published work reprinted in Appendix A.

Let us briefly introduce some basic notation: Scalars, vectors, and functions are denoted by lowercase letters, matrices by Latin uppercase letters, discrete sets by Greek uppercase letters, and continuous sets by calligraphic uppercase letters. The Minkowski sum of two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ is defined as $\mathcal{S}_1 \oplus \mathcal{S}_2 := \{s_1 + s_2 \,|\, s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$. Furthermore, we use $[a, b]$ to represent real-valued scalar intervals with $a \leq b$. All other notation is specified at the first usage.

2.1 Problem Statement

Verification algorithms are often finely tailored to specific system classes in order to exploit their properties. We first define linear continuous-time time-invariant systems.

Definition 1 (Linear Time-Invariant System)**.** *We consider perturbed linear time-invariant (LTI) systems of the form*

$$\dot{x}(t) = Ax(t) + Bu(t) + Ew(t), \tag{2.1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $A \in \mathbb{R}^{n \times n}$ is the state matrix, $B \in \mathbb{R}^{n \times m}$ is the input matrix, $u(t) \in \mathbb{R}^m$ is the input vector, $E \in \mathbb{R}^{n \times r}$ is the disturbance matrix, and $w(t) \in \mathbb{R}^r$ is the disturbance vector. We also work with the simplified dynamics

$$\dot{x}(t) = Ax(t) + Ew(t) \tag{2.2}$$

that we obtain in two ways: either by insertion of a control law, such as $u(t) = Kx(t)$ with $K \in \mathbb{R}^{m \times n}$, or by integrating the term $Bu(t)$ into the term $Ew(t)$, e.g., if we have a piecewise constant input trajectory $u(t)$. □

Next, we introduce nonlinear continuous-time systems.

Definition 2 (Nonlinear System)**.** *We consider perturbed nonlinear systems of the form*

$$\dot{x}(t) = f(x(t), w(t)), \tag{2.3}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $w(t) \in \mathbb{R}^r$ is the disturbance vector, and the function $f : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is sufficiently smooth. □

In this case, we assume a synthesized controller has already been inserted into the differential equation. Let us now symbolically introduce the solution to the ODEs in Definitions 1 and 2, which we later use to define reachable sets.

Definition 3 (State Trajectory)**.** *Given an initial state $x_0 \in \mathbb{R}^n$ at time $t_0 \in \mathbb{R}$, an input trajectory $u : \mathbb{R} \to \mathbb{R}^m$, and a disturbance trajectory $w : \mathbb{R} \to \mathbb{R}^r$, we denote the solution at time $t \geq t_0$ of an ODE of the form* (2.1) *by $\xi(t; x_0, u(\cdot), w(\cdot))$. For systems of the forms in* (2.2)-(2.3)*, we omit the dependence on the input trajectory $u(\cdot)$ accordingly.* □

Reachability analysis extends well-known analytical and numerical methods for solving ODEs by considering uncertainties in the initial state, the control input, and the disturbance. These are part of the model parameters representing the circumstances under which we analyze safety.

Definition 4 (Model Parameters)**.** *We consider bounded time horizons $t \in \tau = [t_0, t_{\text{end}}]$ with $t_{\text{end}} > t_0$. The initial state $x_0 \in \mathbb{R}^n$ at time $t_0 \in \mathbb{R}$ is uncertain within the bounded initial set $\mathcal{X}_0 \subset \mathbb{R}^n$. Similarly, we define the bounded target set $\mathcal{X}_{\text{end}} \subset \mathbb{R}^n$ for states at the end of the time horizon $t_{\text{end}} \in \mathbb{R}$, which either represents an unsafe set or a goal set. Let us denote a single input trajectory by $u(\cdot)$, for which $\forall t \in \tau : u(t) \in \mathcal{U} \subset \mathbb{R}^m$ holds, where $\mathcal{U} \subset \mathbb{R}^m$ is the bounded input set. We use $\mathbb{U} : \mathbb{R} \to \mathbb{R}^m$ to represent the set of all input trajectories. Analogously, we denote a single disturbance trajectory by $w(\cdot)$, which fulfills $\forall t \in \tau : w(t) \in \mathcal{W} \subset \mathbb{R}^r$ with $\mathcal{W} \subset \mathbb{R}^r$ being the bounded set of disturbances. The set of all disturbance trajectories is written as $\mathbb{W} : \mathbb{R} \to \mathbb{R}^r$. We use Ω to denote the problem-specific set of model parameters, defined individually for each verification problem.* □

In general, reachable sets contain all states a dynamical system can reach under a particular interplay of states, inputs, and disturbances. Let us now define different notions of reachable sets, starting with the maximal forward reachable set: For a given initial set, it contains all successor states that are reachable through the dynamics of the differential equation.

Definition 5 (Maximal Forward Reachable Set)**.** *For a system* (2.1) *and the model parameters $\Omega = \{\tau, \mathcal{X}_0, \mathbb{U}, \mathbb{W}\}$, the maximal forward reachable set is defined as*

$$\mathcal{R}_{\exists\exists}(\tau; \mathcal{X}_0, \mathbb{U}, \mathbb{W}) := \big\{\xi(t; x_0, u(\cdot), w(\cdot)) \,\big|\, \exists t \in \tau\ \exists x_0 \in \mathcal{X}_0\ \exists u(\cdot) \in \mathbb{U}\ \exists w(\cdot) \in \mathbb{W}\big\}, \tag{2.4}$$

which contains all reachable states for any combination of initial state $x_0 \in \mathcal{X}_0$, input trajectory $u(\cdot) \in \mathbb{U}$, and disturbance trajectory $w(\cdot) \in \mathbb{W}$ over the time interval $\tau \in [t_0, t_{\text{end}}]$. For systems of the form (2.2)-(2.3)*, the dependence on the set of input trajectories $\mathbb{U}$ is omitted accordingly.* □

The most common verification task is to show that no unsafe behavior can be reached for any initial state despite worst-case disturbances. The maximal forward reachable set plays an integral part in solving this so-called reach-avoid problem.

Problem 1 (Forward Reach-Avoid Problem)**.** *Given a system* (2.2)-(2.3)*, the model parameters* $\Omega = \{\tau, \mathcal{X}_0, \mathbb{W}\}$*, and a number* $\ell \in \mathbb{N}$ *of unsafe sets* $\forall j \in \{1, ..., \ell\} : \mathcal{F}_j \subset \mathbb{R}^n$*, decide whether*

$$\Bigg(\mathcal{R}_{\exists\exists}(\tau; \mathcal{X}_0, \mathbb{W}) \cap \bigcup_{j\in\{1,\dots,\ell\}} \mathcal{F}_j\Bigg) = \emptyset, \tag{2.5}$$

that is, whether there exists any disturbance trajectory $w(\cdot) \in \mathbb{W}$ *causing the state to reach some unsafe set.* □

While forward reachability reasons about the future behavior, backward reachability computes past states: Backward reachable sets contain all initial states emanating a trajectory that can reach a given target set $\mathcal{X}_{\text{end}} \subset \mathbb{R}^n$ under certain conditions.

Definition 6 (Minimal and Maximal Backward Reachable Set)**.** *For a system* (2.1) *with the model parameters* $\Omega = \{\tau, \mathcal{X}_{\text{end}}, \mathbb{U}, \mathbb{W}\}$*, the minimal backward reachable set over a time interval* $\tau \in [t_0, t_{\text{end}}]$ *with* $t_0 \in \mathbb{R}_{\geq 0}$ *is defined as*

$$\mathcal{R}_{\forall\exists}(-\tau; \mathcal{X}_{\text{end}}, \mathbb{U}, \mathbb{W}) := \big\{x_0 \in \mathbb{R}^n \,\big|\, \forall u(\cdot) \in \mathbb{U}\; \exists w(\cdot) \in \mathbb{W}\; \exists t \in \tau : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}\big\} \tag{2.6}$$

and contains all initial states $x_0 \in \mathbb{R}^n$*, where for every input trajectory* $u(\cdot) \in \mathbb{U}$*, there exists a corresponding disturbance trajectory* $w(\cdot) \in \mathbb{W}$ *such that the resulting state trajectory* $\xi(t; x_0, u(\cdot), w(\cdot))$ *enters the target set at time* $t \in \tau$*. Moreover, the maximal backward reachable set over a time interval* $\tau \in [t_0, t_{\text{end}}]$ *with* $t_0 \in \mathbb{R}_{\geq 0}$ *is defined as*

$$\mathcal{R}_{\exists\forall}(-\tau; \mathcal{X}_{\text{end}}, \mathbb{U}, \mathbb{W}) := \big\{x_0 \in \mathbb{R}^n \,\big|\, \exists u(\cdot) \in \mathbb{U}\; \forall w(\cdot) \in \mathbb{W}\; \exists t \in \tau : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}\big\} \tag{2.7}$$

and contains all initial states $x_0 \in \mathbb{R}^n$*, where there exists an input trajectory* $u(\cdot) \in \mathbb{U}$ *such that the resulting state trajectory* $\xi(t; x_0, u(\cdot), w(\cdot))$ *enters the target set for all potential disturbance trajectories* $w(\cdot) \in \mathbb{W}$ *at some time* $t \in \tau$*.* □

A notable difference from forward reachability, see Definition 5, is the competing influence of inputs and disturbances, also known as a two-player game [156]: In the definition (2.6) for the minimal backward reachable set, the disturbance wants to steer the state *toward* the target set, while the input wants to steer the state *away* from the target set. These roles are inverted in the definition (2.4) for the maximal backward reachable set.

Each backward reachable set has its respective use in verification: If the target set represents an unsafe set, we must also avoid entering the minimal backward reachable set, as it contains all states that cannot avoid entering the unsafe set under worst-case disturbances. Instead, if the target set is a goal set, the maximal backward reachable set contains all initial states for which a controller exists to reach that goal set despite worst-case disturbances. Problems 2 and 3 formulate these perspectives as verification tasks.

Problem 2 (Robust Collision Avoidance)**.** *Given a system* (2.1) *with the model parameters* $\Omega = \{\tau, \mathcal{X}_{\text{end}}, \mathbb{U}, \mathbb{W}\}$*, decide for an initial state* $x_0 \in \mathbb{R}^n$ *whether*

$$\exists u(\cdot) \in \mathbb{U} : \mathcal{R}_{\exists\exists}(\tau; x_0, u(\cdot), \mathbb{W}) \cap \mathcal{X}_{\text{end}} = \emptyset, \tag{2.8}$$

that is, whether there exists an input trajectory $u(\cdot) \in \mathbb{U}$ such that all trajectories emanating from the initial state x_0 avoid the unsafe target set $\mathcal{X}_{\text{end}}$ regardless of the disturbance trajectory $w(\cdot) \in \mathbb{W}$. □

Problem 3 (Reaching a Target Set)**.** *Given a system* (2.1) *with the model parameters $\Omega = \{\tau, \mathcal{X}_{\text{end}}, \mathbb{U}, \mathbb{W}\}$, decide for an initial state $x_0 \in \mathbb{R}^n$ whether*

$$\exists u(\cdot) \in \mathbb{U}\ \exists t \in \tau : \mathcal{R}_{\exists\exists}(t; x_0, u(\cdot), \mathbb{W}) \in \mathcal{X}_{\text{end}}, \tag{2.9}$$

that is, whether there exists an input trajectory $u(\cdot) \in \mathbb{U}$ such that all trajectories emanating from the initial state x_0 can be steered into the target set $\mathcal{X}_{\text{end}}$ at some time $t \in \tau$ regardless of the disturbance trajectory $w(\cdot) \in \mathbb{W}$. □

In the next section, we outline our approach to solving the above-defined verification tasks.

2.2 Solution Concept

This section outlines our solution concept for solving Problem 1 using the forward reachable set defined in Definition 5. Please note that Problems 2 and 3 can be addressed analogously with the help of the backward reachable sets from Definition 6.

We verify safety by checking for an intersection between the unsafe set and the reachable set, which we compute using difference inclusions and set propagation. Hence, let us briefly summarize the computation of outer approximations of reachable sets for linear systems of the form (2.2) and nonlinear systems of the form (2.3): The time horizon τ is partitioned into ω time steps, i.e.,

$$\tau = \bigcup_{k=0}^{\omega-1} \tau_k, \quad \forall k \in \{0, ..., \omega-1\} : \tau_k = [t_k, t_{k+1}], \Delta t_k = t_{k+1} - t_k. \tag{2.10}$$

For linear systems, we can exploit the superposition principle to separately propagate outer approximations of the homogeneous solution $\widehat{\mathcal{H}}(t)$ and the particular solution $\widehat{\mathcal{P}}(t)$ due to the disturbance set $\mathcal{W}$. We compute an enclosure of the time-interval reachable set $\widehat{\mathcal{R}}(\tau_k)$ using the Minkowski sum of the homogeneous and particular solutions over the time interval τ_k:

$$\forall k \in \{0, ..., \omega-1\} : \ \widehat{\mathcal{R}}_{\text{lin}}(\tau_k) = \widehat{\mathcal{H}}(\tau_k) \oplus \widehat{\mathcal{P}}(\tau_k). \tag{2.11}$$

Analogously, one can enclose the time-point reachable set $\widehat{\mathcal{R}}_{\text{lin}}(t_{k+1})$ by using time-point homogeneous and particular solutions.

For nonlinear systems, we Taylor expand the nonlinear dynamics (2.3) around the linearization points for the state x^* and the disturbance w^*, yielding the difference inclusion

$$\dot{x}(t) \in \left.\frac{\partial f(x,w)}{\partial x}\right|_{x^*} (x(t) - x^*) + \left.\frac{\partial f(x,w)}{\partial w}\right|_{w^*} (w(t) - w^*) \oplus \mathcal{L}(t). \tag{2.12}$$

The Lagrange remainder $\mathcal{L}(t)$ overestimates the influence of the higher-order dynamics and is defined by

$$\mathcal{L}(t) := \left\{ \frac{1}{2}(z - z^*)^\top \frac{\partial^2 f_i(\zeta)}{\partial z^2}(z - z^*) \,\middle|\, \zeta = z^* + \lambda(z(t) - z^*), \lambda \in [0, 1] \right\}, \tag{2.13}$$

using the stacked vector $z(t) = [x(t)^\top \, u(t)^\top]^\top$ and the stacked linearization point $z^* = [x^{*\top} \, w^{*\top}]^\top$. This set is evaluated using range-bounding methods, such as interval arithmetic [28]. By interpreting the Lagrange remainder as an additive disturbance, we can apply the superposition principle for linear systems to compute the reachable set due to the linearized dynamics and the Lagrange remainder separately. For the first time interval $\tau_0 = [0, \Delta t_0]$, we enclose the reachable set by

$$\widehat{\mathcal{R}}_{\text{nonlin}}(\tau_0) = \widehat{\mathcal{R}}_{\text{lin}}(\tau_0) \oplus \widehat{\mathcal{R}}_{\text{abs}}(\tau_0), \tag{2.14}$$

where $\widehat{\mathcal{R}}_{\text{lin}}(\tau_0)$ is computed as in (2.11) using the linearized dynamics around the first linearization point, and $\widehat{\mathcal{R}}_{\text{abs}}(\tau_0)$ represents the integrated Lagrange remainder $\mathcal{L}(\tau_0)$. However, the set $\mathcal{L}(\tau_0)$ depends on the reachable set $\widehat{\mathcal{R}}_{\text{nonlin}}(\tau_0)$, which determines the range of $z(t)$ in (2.13) over the time interval τ_0. This mutual dependency between the Lagrange remainder and the reachable set can be resolved using the iterative scheme proposed in [73, Section III], allowing us to treat the Lagrange remainder $\mathcal{L}(t)$ as an additive disturbance. An outer approximation of the time-point reachable set $\widehat{\mathcal{R}}_{\text{nonlin}}(t_1)$ can be obtained by replacing the time-interval solution $\widehat{\mathcal{R}}_{\text{lin}}(\tau_0)$ in (2.14) with its time-point analog $\widehat{\mathcal{R}}_{\text{lin}}(t_1)$. For all subsequent time steps, the time-point solution of the previous time step is used as the initial set to evaluate (2.14) using the current abstracted dynamics. This process is illustrated in Figure 1.1 in Section 1.2.

For the remainder of this overview, we abbreviate the reachable set computation using the operator $\texttt{Reach}(\Omega, \Phi)$, which takes two input arguments: the model parameters Ω, e.g., the time horizon, and the algorithm parameters Φ, e.g., the time step size. Since we use set propagation methods, its output is a sequence of ω time-interval reachable sets

$$\left\{ \widehat{\mathcal{R}}(\tau_0), \widehat{\mathcal{R}}(\tau_1), ..., \widehat{\mathcal{R}}(\tau_{\omega-1}) \right\} \leftarrow \texttt{Reach}(\Omega, \Phi), \tag{2.15}$$

which compose the reachable set over the entire time horizon τ:

$$\widehat{\mathcal{R}}(\tau) = \bigcup_{k=0}^{\omega-1} \widehat{\mathcal{R}}(\tau_k). \tag{2.16}$$

While the reachability algorithm $\texttt{Reach}(\Omega, \Phi)$ is guaranteed to return a sound outer approximation, the tightness of the computed sets with respect to the exact reachable set strongly depends on the tuning of the algorithm parameters Φ. Among these, the time step size and the complexity of the set representation have the most significant effect on the result. Let us briefly showcase how these algorithm parameters affect the reachable set computations introduced above: Since the exact homogeneous time-interval solution is, in general, a non-convex set, smaller time step sizes yield smaller approximation errors when the set representation is

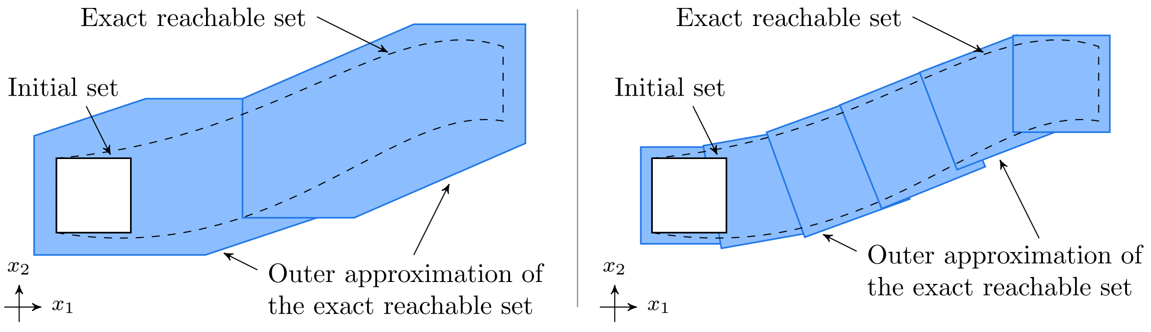

Exact reachable set
Initial set
Outer approximation of the exact reachable set
x_2
x_1
Exact reachable set
Initial set
Outer approximation of the exact reachable set
x_2
x_1

Figure 2.1: Influence of the algorithm parameters on the approximation error: Different time step sizes partitioning the time horizon (left: $\omega = 2$ time steps, right: $\omega = 6$ time steps) and different complexities for representing the computed outer approximation of the exact reachable set (left: six vertices per set, right: four vertices per set).

restricted to convex sets. In contrast, the computation of the particular solution benefits from larger time step sizes since its propagation over subsequent time steps corresponds to a sequence of Minkowski sums, each of which increases the complexity of the resulting shape and, thus, the required representation size for an accurate representation. Figure 2.1 illustrates how a suitable choice of algorithm parameters can influence the resulting reachable set size: On the left-hand side, the time horizon is partitioned using larger time step sizes and the reachable sets are represented using more vertices as opposed to smaller time step sizes and fewer vertices on the left-hand side. We prefer to obtain the result on the right-hand side from the reachability algorithm $\texttt{Reach}(\Omega, \Phi)$.

Let us now show how the tightness of the computed outer (or inner) approximation affects the verification task. To this end, consider Figure 2.2, where we are given an initial set $\mathcal{X}_0$ and compute an outer approximation $\widehat{\mathcal{R}}(\tau) \supseteq \mathcal{R}(\tau)$ and an inner approximation $\check{\mathcal{R}}(\tau) \subseteq \mathcal{R}(\tau)$ of the exact reachable set $\mathcal{R}(\tau)$ over a bounded time horizon τ. We now examine whether we can verify safety regarding a set of safety specifications, which are defined via the unsafe sets $\mathcal{F}_1, \dots, \mathcal{F}_4$: There is no intersection between the outer approximation $\widehat{\mathcal{R}}(\tau)$ and the leftmost unsafe set $\mathcal{F}_1$ so that the corresponding specification is verified. In contrast, the rightmost unsafe set $\mathcal{F}_4$ intersects the inner approximation $\check{\mathcal{R}}(\tau)$, resulting in a safety violation since there is definitely some system behavior that reaches this unsafe set. The other two unsafe sets intersect the outer approximation but not the inner approximation, from which safety can neither be verified nor falsified. With the help of the exact reachable set $\mathcal{R}(\tau)$—which we cannot compute—safety could be determined: The second-to-left unsafe set $\mathcal{F}_2$ does not intersect the exact reachable set so that the corresponding specification is verified, whereas there is indeed an intersection between the second-to-right unsafe set $\mathcal{F}_3$ and the exact reachable set, thereby falsifying safety with regard to that specification.

This abstract illustration underlines the criticality of the tightness of the reachable set approximations. We can only verify or falsify safety for a given specification if the computed outer and inner approximations are tight enough. To quantitatively measure the tightness of

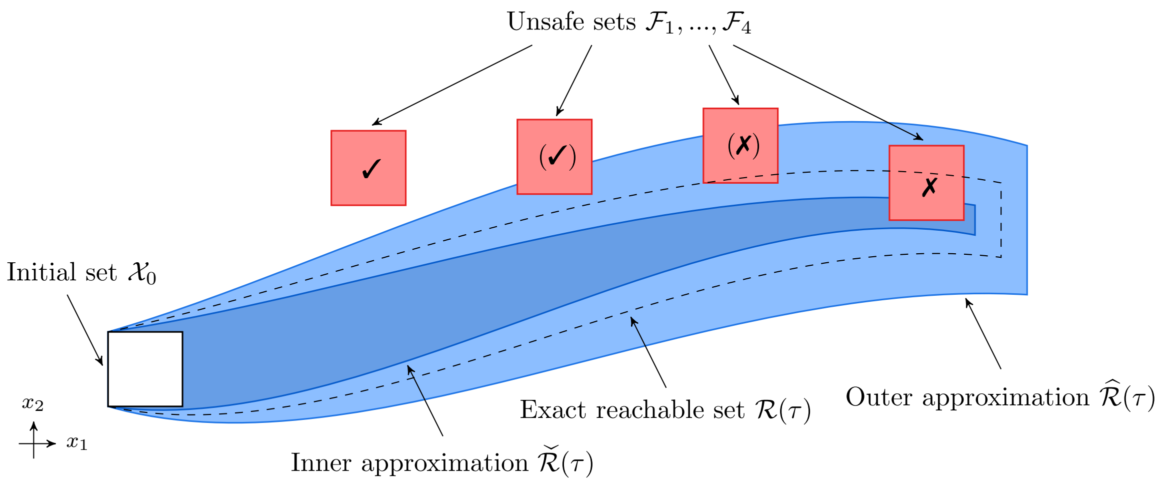

Unsafe sets $\mathcal{F}_1, ..., \mathcal{F}_4$
✓
(✓)
(✗)
✗
Initial set $\mathcal{X}_0$
x_2
x_1
Exact reachable set $\mathcal{R}(\tau)$
Outer approximation $\widehat{\mathcal{R}}(\tau)$
Inner approximation $\breve{\mathcal{R}}(\tau)$

Figure 2.2: Safety verification using inner and outer approximations of the exact reachable set: Safety can be verified if an unsafe set and the outer approximation do not intersect (case ✓). In contrast, if an unsafe set intersects the inner approximation, safety is provably violated (case ✗). If the inner and outer approximation are not tight enough to determine safety, we do not obtain any conclusive result (cases (✓) and (✗)).

an outer or inner approximation of the reachable set with respect to its exact counterpart, we use the Hausdorff distance between sets.

Definition 7 (Hausdorff Distance)**.** *The Hausdorff distance between two compact sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ is defined as*

$$d_H(\mathcal{S}_1, \mathcal{S}_2) := \max\left\{ \max_{s_1 \in \mathcal{S}_1} \Big(\min_{s_2 \in \mathcal{S}_2} \|s_1 - s_2\|_2 \Big), \max_{s_2 \in \mathcal{S}_2} \Big(\min_{s_1 \in \mathcal{S}_1} \|s_1 - s_2\|_2 \Big) \right\} \tag{2.17}$$

with respect to the Euclidean norm. □

Informally, the Hausdorff distance is the longest shortest distance between any two points within their respective sets, as shown on the left-hand side of Figure 2.3. In this thesis, we use the Hausdorff distance, as illustrated on the right-hand side of Figure 2.3, to measure the approximation error

$$\varepsilon(\Phi) = d_H\Big(\mathcal{R}(\tau; \Omega), \widehat{\mathcal{R}}(\tau; \Omega, \Phi)\Big) \tag{2.18}$$

between the exact reachable set $\mathcal{R}(\tau; \Omega)$ and an outer approximation $\widehat{\mathcal{R}}(\tau; \Omega, \Phi)$ returned by a reachability algorithm `Reach`(Ω, Φ). To highlight the crucial dependence of the approximation error on the algorithm parameters Φ, we include Φ as an argument in $\widehat{\mathcal{R}}(\tau; \Omega, \Phi)$ in addition to the user-provided model parameters Ω. Please note that we can define the error (2.18) analogously with respect to an inner approximation $\breve{\mathcal{R}}(\tau; \Omega, \Phi)$.

Let us now outline our solution concept for the verification tasks introduced in the previous section. For all three tasks, we need a reachability algorithm computing the required reachable sets. To this end, we either re-use well-known state-of-the-art algorithms or introduce novel ones. The automated verification consists of three main parts, namely,

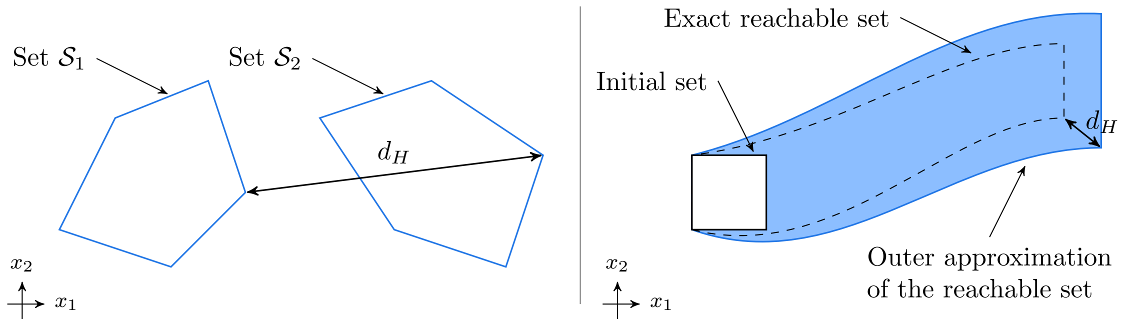


Figure 2.3: The Hausdorff distance d_H measures the distance between sets (left). We interpret the Hausdorff distance between the outer approximation and the exact reachable set as the approximation error (right).

1. the derivation or estimation of bounds on the approximation error (2.18),
2. devising strategies for the automated tuning of algorithm parameters Φ, and
3. iterative refinement of the reachable set approximations $\widehat{\mathcal{R}}(\tau)$ and $\check{\mathcal{R}}(\tau)$ based on the tuning methods from step 2.

Knowing how each algorithm parameter affects the approximation error facilitates the development of tuning strategies for the reachability algorithm. We then compute a tightening sequence of approximations by iteratively refining the algorithm parameters for the reachability algorithm to eventually verify or falsify safety. This concept automates the verification process so that the user only has to provide the dynamics (2.1)-(2.3), the model parameters Ω, and the safety specification to obtain a provably correct result on the safety of the system.

3 Discussion and Conclusion

In this chapter, we summarize and critically assess the work presented in this thesis, in addition to giving potential directions for work. We address our contributions to the automated verification of linear systems using forward reachability in Section 3.1 and using backward reachability in Section 3.2. In Section 3.3, we recap our automated reachability algorithm of nonlinear systems and identify the missing components toward automated verification.

3.1 Verification of Linear Systems Using Forward Reachability

Our works [W1–W3] reprinted in Appendices A.1 to A.3 are dedicated to solving the verification task in Problem 1 for linear systems of the form (2.2). We have devised two automated verification algorithms that tune the algorithm parameters based on knowledge about the induced approximation errors. Hence, we can verify any safety specification that does not require the exact reachable set.

Summary The first algorithm (Appendices A.1 and A.2) computes explicit outer and inner approximations of the exact reachable set represented by zonotopes and constrained zonotopes, respectively. A subsequent intersection check with the unsafe sets either verifies safety or falsifies safety or demands refined algorithm parameters for a tighter approximation. The runtime complexity for computing an outer approximation respecting a bound on the admissible error is $\mathcal{O}(n^3)$, equal to the base algorithm [31, 48]. Halving the admissible error bound at most doubles the number of time steps required for the reachable set computation. The inner approximation is computed by the Minkowski difference between a zonotope and an error ball representing the approximation error. Our evaluation of this operation yields a constrained zonotope with a significantly larger representation size. The subsequent intersection check between the inner approximation and the unsafe set has complexity $\mathcal{O}(n^7)$, which thus becomes the bottleneck of the verification algorithm. However, one can also circumvent the computation of the Minkowski difference altogether, as shown in Figure 3.1: On the left-hand side, we falsify safety by determining that the unsafe set intersects the inner approximation, which is computed by the Minkowski difference between the outer approximation and the error ball. Instead, one can tighten the unsafe set by Minkowski difference with that same error ball, as shown on the right-hand side. Safety is then falsified by showing that the outer approximation intersects the tightened unsafe set. At the cost of computing an explicit inner approximation, this would improve the runtime complexity of the verification algorithm to about $\mathcal{O}(n^4)$, depending on the linear programs used for the intersection checks.

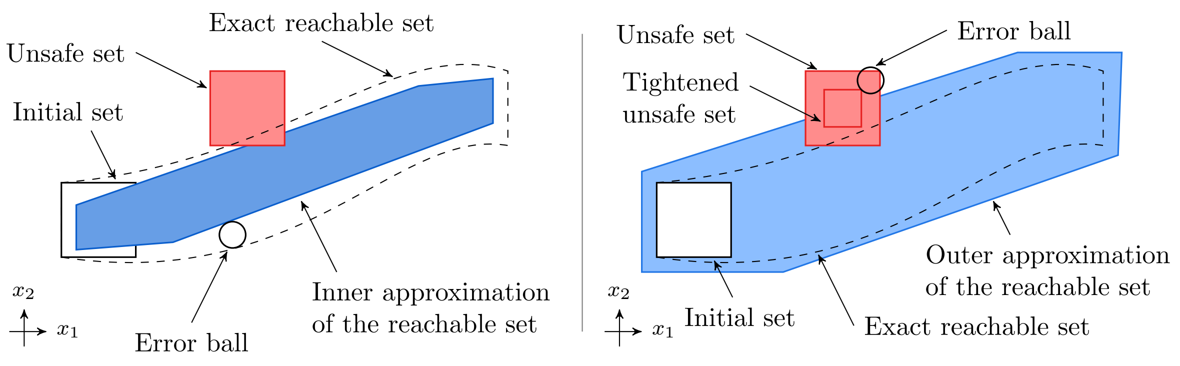


Figure 3.1: Alternative approach for the falsification of linear systems: Instead of computing an explicit inner approximation and checking for an intersection with the unsafe set (left), one can also check for an intersection between the computed outer approximation and a tightened unsafe set (right).

The second algorithm (Appendix A.3) uses support function evaluations to compute outer and inner approximations of the exact reachable set. For specific set representations describing the initial set and the disturbance set, it is possible to pre-compute the homogeneous and particular solutions, which significantly decreases the computation time since the runtime complexity of the propagation reduces to $\mathcal{O}(n^2)$. Although the algorithm is primarily designed to solve the verification task, one can also extract explicit reachable sets represented as polytopes—except for the inner approximation of the time-interval reachable set—but without knowledge about their approximation error with respect to the exact reachable set. For safety specifications formulated using halfspaces as unsafe sets, this algorithm provides a faster verification result than our first algorithm. In the case of more complex representations of the unsafe sets, it is unclear whether the intersection check is faster using an a priori unknown number of support function evaluations as in this algorithm or a single linear program for explicitly computed reachable sets as in the first algorithm.

In summary, both our verification algorithms iteratively and automatically refine the algorithm parameters until safety can be verified or falsified. Consequently, only the model, the model parameters, and the safety specifications have to be defined to obtain a conclusive result on safety. Our numerical evaluation has shown comparative or faster performance on benchmarks of the ARCH competition, which assesses the current capabilities of academic verification tools. Hence, we claim that the verification task formulated in Problem 1 has been successfully solved by the work in this thesis.

Future Work One can extend our solution concept to reachability algorithms using dimensionality reduction to improve the analysis of high-dimensional linear systems [59–62]. Similar to our rigorous approximation error analysis, the errors induced by the dimensionality reduction have to be bounded, and their dependence on the algorithm parameters, most notably the time step size, has to be determined. For a Krylov subspace algorithm [59, 60], the dimension of the Krylov subspace becomes another tunable algorithm parameter; for a decomposition algorithm [61, 62], the block partitioning of the state matrix has to be tuned. These new algorithm parameters are responsible for the degree of abstraction that is imposed on the dynamics before starting the reachable set computation. This introduces a level of abstraction refinement to the resulting verification algorithm, where the outer layer refines the abstraction and the inner layer refines the approximation errors of the reachable set computed for that abstraction. Other than for decoupled systems, such abstractions are not loss-free and the refinement will converge to the original full-state dynamics as the admissible error bound decreases.

Another direction for future work is the verification of more complex specifications, such as temporal logic specifications [182, 183], which cannot be solved by simple intersection checks due to the time dependency of the specifications. One method introduces reachset temporal logic, which rewrites signal temporal logic formulae into conjunctions and disjunctions of reach-avoid problems [183]. Our approach might thus be extensible to this more general class of specifications by tailoring the refinement process to many concurrent individual reach-avoid problems.

3.2 Verification of Linear Systems Using Backward Reachability

Our work [W4] reprinted in Appendix A.4 addresses the verification tasks formulated in Problems 2 and 3 for linear systems of the form (2.1). We have developed novel backward reachability algorithms for time-point and time-interval minimal and maximal backward reachable sets whose runtime complexity is polynomial in the state dimension.

Summary Under mild assumption for the model parameters, our backward reachability algorithms exploit the benefits of three different set representations—polytopes, zonotopes, and constrained zonotopes—and conversions between them, such that all set operations have polynomial runtime complexity in the state dimension. The combination of different set representations becomes necessary due to the presence of competing control inputs and disturbances, which results in a set-based algorithm with a Minkowski sum and a Minkowski difference: The Minkowski difference is essentially a containment problem, which can be solved efficiently using support function evaluations if the outer body is a polytope in halfspace representation. If, instead, the outer body is a zonotope in generator representation, the runtime complexity is exponential in the state dimension [184]. The Minkowski sum can be very efficiently evaluated for zonotopes and constrained zonotopes, but only in exponential runtime with respect to the state dimension for two polytopes [42]. Our algorithms successfully avoid evaluating both set operations in exponential runtime. Regarding the approximation error, the computed time-point solutions converge to the respective exact backward reachable sets in the limit for a time step size going to zero and by evaluating an infinite number of support functions for the Minkowski sum of a polytope and a zonotope. Still, an exact bound for the approximation error is yet to be derived. Moreover, our computation of time-interval solutions is based on approximations whose induced errors are unknown. A comparison to the state of the art, namely Hamilton-Jacobi reachability and dissipativity-based approaches, greatly favors our proposed algorithms due to their polynomial runtime complexity in the state dimension. While those approaches

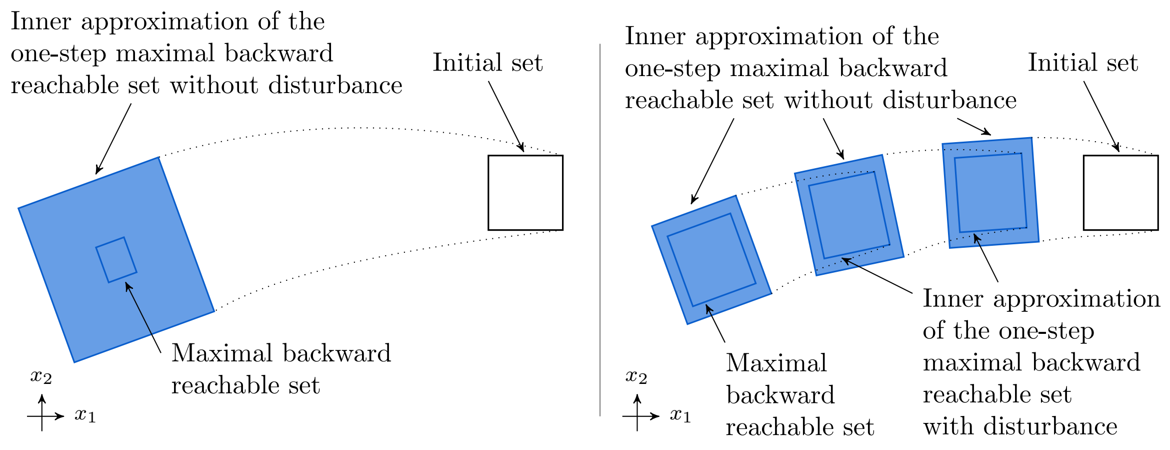

Inner approximation of the one-step maximal backward reachable set without disturbance
Initial set
Maximal backward reachable set
x_2
x_1
Inner approximation of the one-step maximal backward reachable set without disturbance
Initial set
Inner approximation of the one-step maximal backward reachable set with disturbance
Maximal backward reachable set
x_2
x_1

Figure 3.2: Maximal backward reachability analysis using feedforward (left) or feedback (right) control. The final maximal backward reachable set is larger for feedback control because the control input can iteratively react to the disturbance.

can deal with more general dynamics, including time-varying parameters and state constraints, we still claim that our novel backward reachability algorithms significantly outperform them, restricted to the computation of backward reachable sets for linear systems.

Future Work Since our work on backward reachability is the first of its kind, there are multiple directions for future investigations: A rigorous derivation of the approximation errors would provide a quantitative assessment of the tightness of the computed inner and outer approximations. In contrast to our approximation error analysis for the forward reachable set computation in [W2], this is a more challenging endeavor for backward reachability: Some errors originate from approximations with unknown exact solutions, which impedes any attempt to bound the approximation error. Another idea is to compute both inner and outer approximations and bound the approximation error using the Hausdorff distance between the two. However, it is non-trivial to obtain the complementary inner or outer approximation for the computation of the approximation error, and even then, the Hausdorff distance is generally hard to compute unless similarities in the structure of the two sets can be exploited.

A second extension is to incorporate automated tuning of algorithm parameters to refine the tightness of the computed backward reachable sets. By applying our solution concept, one could then fully automatically solve the verification tasks in Problems 2 and 3. Note that an explicit bound on the approximation error is not strictly necessary as long as the parameter tuning can be proven to tighten the reachable sets, e.g., by analyzing the behavior in the limit of a time step size going to zero. However, this requires all approximation errors to be known, which is currently not the case, as mentioned above.

Another idea is to consider feedback control instead of feedforward control. Figure 3.2 shows the differences between the resulting maximal backward reachable sets: Our set-based computation corresponds to feedforward control due to the superposition of the two particular solutions for the influences of control inputs and disturbances. Since the superposition is maintained for the entire time under consideration, the control input cannot react to the disturbance (left-hand side of Figure 3.2). Feedback control corresponds to an iterative computation of the reachable set (right-hand side of Figure 3.2), where both particular solutions affect the computation of the reachable set within each time step. The main challenge for the resulting backward reachability algorithm is the evaluation of the Minkowski sum and Minkowski difference in each time step. A source of inspiration might be the reachable set computation using set propagation for linear discrete-time systems [64], where iterative evaluation of both set operations is unavoidable.

Finally, to date, there exist no backward reachability algorithms using set propagation for nonlinear continuous-time systems. A promising starting point is on-the-fly linearization [73], which abstracts the nonlinear dynamics to linear difference inclusions while rigorously accounting for the abstraction error due to the truncated higher-order dynamics. This results in a sequence of one-step backward reachable set computations for iteratively changing linear dynamics, which is reminiscent of the feedback control illustrated in Figure 3.2.

3.3 Verification of Nonlinear Systems

Our works [W5, W6] reprinted in Appendices A.5 and A.6 are dedicated to solving the verification task in Problem 1 for nonlinear systems of the form (2.3). We have extended a reachability algorithm based on set propagation and on-the-fly linearization by automated parameter tuning methods to obtain tight reachable sets, which can be used for verification in a subsequent step.

Summary The main challenge for automated parameter tuning is the unavoidable wrapping effect of the set propagation. In contrast to our proposed parameter tuning methods for linear systems, we cannot fulfill an admissible error bound for the reachable set since we do not have bounds on the errors induced in future time steps—these depend on the future size of the reachable set and the future abstracted dynamics. Therefore, our parameter tuning methods for nonlinear systems focus on analyzing the local behavior of the abstraction error: After simplifying the set propagation over a sequence of time steps until reaching a finite time horizon, we obtain a scalar function estimating the radius of the reachable set at the end of that finite time horizon. Crucially, the influence of the time step size on the abstraction error is measured using the novel concept of a gain order, which describes the relative change in the abstraction error for different time step sizes. While previous approaches were restricted to fixed values, automatically tuned time step sizes allow the reachability analysis to adapt to the current dynamics: A smaller time step size is used in regions with stronger nonlinearities to limit the induced abstraction errors, while a larger time step size reduces the number of time steps in regions with milder nonlinearities.

Future Work Finally, let us outline how to build upon our work [W5, W6] to devise an automated verification algorithm. For the type of reachability algorithm that we used, it has been shown that spatial refinement—often called *splitting*—is required for convergence to the exact reachable set [185]. This convergence to the exact solution is a critical condition for

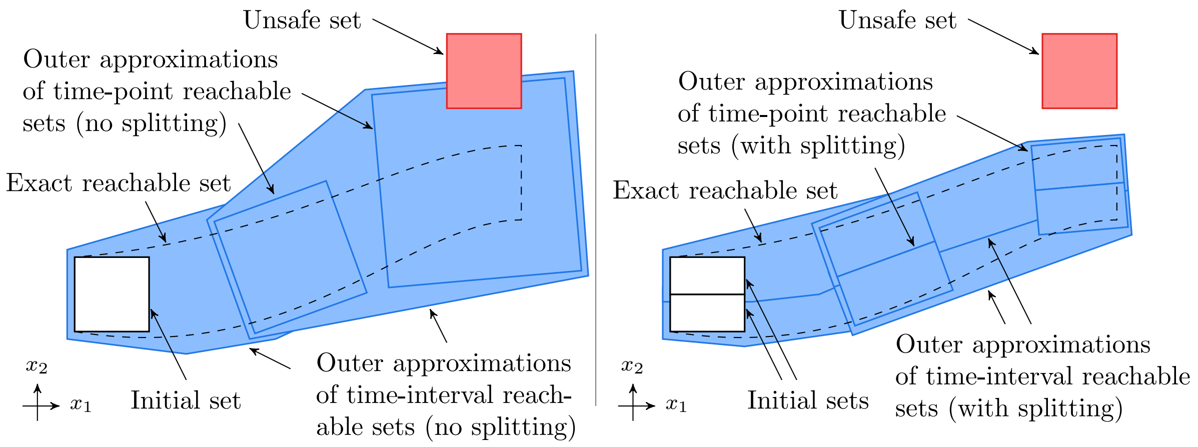

Unsafe set
Outer approximations of time-point reachable sets (no splitting)
Exact reachable set
x_2
x_1
Initial set
Outer approximations of time-interval reachable sets (no splitting)
Unsafe set
Outer approximations of time-point reachable sets (with splitting)
Exact reachable set
x_2
x_1
Initial sets
Outer approximations of time-interval reachable sets (with splitting)

Figure 3.3: Reachability analysis of nonlinear systems: Large initial sets result in large approximation errors, impeding verification (left). However, splitting the initial set into smaller sets reduces the approximation error, leading to a successful verification (right).

a complete verification algorithm, especially for nonlinear systems, where reachability analysis often aborts due to an uncontrollable growth of the abstraction error, as illustrated in Figure 3.3: On the left-hand side, the reachable set is computed without splitting, resulting in an outer approximation with a large approximation error that impedes safety verification. In contrast, the reachable set on the right-hand side is computed for a split initial set, and since neither of the reachable sets intersects the unsafe set, safety is verified. Therefore, the effects of splitting have to be analyzed with similar rigor as demonstrated for the dependence of the abstraction error on the time step size in [W6]. One idea is to define a spatial gain order describing how the abstraction error changes when the set is split in twain, analogously to the temporal gain order we have introduced. This quantitative information can then be used to decide when to split, ultimately yielding an algorithm that refines time and space and, thus, converges to the exact solution [185]. Nevertheless, the approximation error of the computed outer approximation would still be unknown, so that we cannot falsify safety unless we incorporate falsification techniques.

In summary, we have made important steps toward automated verification of uncertain dynamical systems. Nevertheless, there are still many ways to improve reachability analysis, as foreshadowed above. Our overarching goal is to design algorithms for safety verification based on automatically refining the tightness of the computed reachable sets. Further improvements to our work will steadily advance automated safety verification until future approaches can eventually be used in industrial practice in order to guarantee the safe operation of cyber-physical systems.

Finis.

List of Figures

Bibliography

[1] T. Gan, M. Chen, Y. Li, B. Xia, and N. Zhan. "Reachability analysis for solvable dynamical systems". In: *IEEE Transactions on Automatic Control* 63.7 (2018), pp. 2003–2018. DOI: 10.1109/TAC.2017.2763785.

[2] N. S. Nedialkov. "Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation." Dissertation. University of Toronto, 2000.

[3] K. Makino and M. Berz. "Rigorous integration of flows and ODEs using Taylor models". In: *Proc. of Symbolic-Numeric Computation.* ACM. 2009, pp. 79–84. DOI: 10.1145/1577190.1577206.

[4] J. Liu, N. Zhan, and H. Zhao. "Computing semi-algebraic invariants for polynomial dynamical systems". In: *Proc. of the 9th International Conference on Embedded Software.* ACM, 2011, pp. 97–106. DOI: 10.1145/2038642.2038659.

[5] M. A. Ben Sassi and A. Girard. "Computation of polytopic invariants for polynomial dynamical systems using linear programming". In: *Automatica* 48.12 (2012), pp. 3114–3121. DOI: 10.1016/j.automatica.2012.08.014.

[6] M. A. Ben Sassi, A. Girard, and S. Sankaranarayanan. "Iterative computation of polyhedral invariants sets for polynomial dynamical systems". In: *Proc. of the 53rd Conference on Decision and Control.* IEEE. 2014, pp. 6348–6353. DOI: 10.1109/CDC.2014.7040384.

[7] K. Ghorbal and A. Platzer. "Characterizing algebraic invariants by differential radical invariants". In: *Proc. of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems.* Springer, 2014, pp. 279–294. DOI: 10.1007/978-3-642-54862-8_19.

[8] M. Boreale. "Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial ODEs". In: *Science of Computer Programming* 193 (2020). DOI: 10.1016/j.scico.2020.102441.

[9] A. Devonport and M. Arcak. "Estimating reachable sets with scenario optimization". In: *Proc. of the 2nd Conference on Learning for Dynamics and Control.* PMLR. 2020, pp. 75–84.

[10] A. Devonport, F. Yang, L. El Ghaoui, and M. Arcak. "Data-driven reachability analysis with Christoffel functions". In: *Proc. of the 60th Conference on Decision and Control.* IEEE, 2021, pp. 5067–5072. DOI: 10.1109/CDC45484.2021.9682860.

[11] A. J. Thorpe, K. R. Ortiz, and M. M. K. Oishi. "Learning approximate forward reachable sets using separating kernels". In: *Proc. of the 3rd Conference on Learning for Dynamics and Control.* 2021, pp. 201–212.

[12] S. Ratschan and Z. She. "Safety verification of hybrid systems by constraint propagation-based abstraction refinement". In: *Transactions on Embedded Computing Systems* 6.1 (2007), pp. 8–31. DOI: 10.1145/1210268.1210276.

[13] N. Ramdani and N. S. Nedialkov. "Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques". In: *Nonlinear Analysis: Hybrid Systems* 5.2 (2011), pp. 149–162. DOI: 10.1016/j.nahs.2010.05.010.

[14] B. HomChaudhuri, A. P. Vinod, and M. M. K. Oishi. "Computation of forward stochastic reach sets: Application to stochastic, dynamic obstacle avoidance". In: *Proc. of the American Control Conference.* IEEE. 2017, pp. 4404–4411. DOI: 10.23919/ACC.2017.7963633.

[15] A. P. Vinod, B. HomChaudhuri, and M. M. K. Oishi. "Forward stochastic reachability analysis for uncontrolled linear systems using Fourier transforms". In: *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control.* ACM. 2017, pp. 35–44. DOI: 10.1145/3049797.3049818.

[16] A. P. Vinod and M. M. K. Oishi. "Stochastic reachability of a target tube: Theory and computation". In: *Automatica* 125 (2021), p. 109458. DOI: 10.1016/j.automatica.2020.109458.

[17] A. P. Vinod, J. P. Gleason, and M. M. K. Oishi. "SReachTools: A MATLAB stochastic reachability toolbox". In: *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control.* ACM. 2019, pp. 33–38. DOI: 10.1145/3302504.3311809.

[18] A. Taly and A. Tiwari. "Deductive verification of continuous dynamical systems". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science.* Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2009.

[19] A. Platzer. *Logical analysis of hybrid systems: Proving theorems for complex dynamics.* Springer, 2010. ISBN: 978-3-642-14508-7. DOI: 10.1007/978-3-642-14509-4.

[20] F. Immler. "Tool presentation: Isabelle/HOL for reachability analysis of continuous systems". In: *Proc. of the 2nd International Workshop on Applied Verification for Continuous and Hybrid Systems.* 2015, pp. 180–187. DOI: 10.29007/b3wr.

[21] S. Kong, S. Gao, W. Chen, and E. Clarke. "dReach: δ-reachability analysis for hybrid systems". In: *Proc. of the Conference on Tools and Algorithms for the Construction and Analysis of Systems.* Springer, 2015. DOI: 10.1007/978-3-662-46681-0_15.

[22] A. Platzer and J.-D. Quesel. "KeYmaera: A hybrid theorem prover for hybrid systems (system description)". In: *International Joint Conference on Automated Reasoning.* Springer. 2008, pp. 171–178. DOI: 10.1007/978-3-540-71070-7_15.

[23] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. “KeYmaera X: An axiomatic tactical theorem prover for hybrid systems”. In: *Proc. of the 25th International Conference on Automated Deduction*. Springer. 2015, pp. 527–538. DOI: 10.1007/978-3-319-21401-6_36.

[24] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016. ISBN: 9781119121503. DOI: 10.1002/9781119121534.

[25] D. F. Griffiths and D. J. Higham. *Numerical methods for ordinary differential equations: Initial value problems*. Springer, 2010. ISBN: 978-0-85729-148-6. DOI: 10.1016/C2013-0-10643-5.

[26] M. Althoff, G. Frehse, and A. Girard. “Set propagation techniques for reachability analysis”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 369–395. DOI: 10.1146/annurev-control-071420-081941.

[27] G. M. Ziegler. *Lectures on polytopes*. Springer Science & Business Media, 2012.

[28] G. Alefeld and G. Mayer. “Interval analysis: Theory and applications”. In: *Computational and Applied Mathematics* 121.1-2 (2000), pp. 421–464. DOI: 10.1016/S0377-0427(00)00342-3.

[29] A. A. Kurzhanskiy and P. Varaiya. “Ellipsoidal toolbox (ET)”. In: *Proc. of the 45th Conference on Decision and Control*. IEEE. 2006, pp. 1498–1503. DOI: 10.1109/CDC.2006.377036.

[30] A. Girard and C. Le Guernic. “Efficient reachability analysis for linear systems using support functions”. In: *IFAC Proceedings Volumes* 41.2 (2008). DOI: 10.3182/20080706-5-KR-1001.01514.

[31] A. Girard. “Reachability of uncertain linear systems using zonotopes”. In: *Proc. of the 8th International Workshop on Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305. DOI: 10.1007/978-3-540-31954-2_19.

[32] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. “Constrained zonotopes: A new tool for set-based estimation and fault detection”. In: *Automatica* 69 (2016), pp. 126–136. DOI: 10.1016/j.automatica.2016.02.036.

[33] N. Kochdumper and M. Althoff. “Sparse polynomial zonotopes: A novel set representation for reachability analysis”. In: *IEEE Transactions on Automatic Control* 66.2 (2021), pp. 4043–4058. DOI: 10.1109/TAC.2020.3024348.

[34] M. Berz and G. Hoffstätter. “Computation and application of Taylor polynomials with interval remainder bounds”. In: *Reliable Computing* 4 (1998), pp. 83–97. DOI: 10.1023/A:1009958918582.

[35] M. Althoff. “An introduction to CORA 2015”. In: *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*. 2015, pp. 120–151. DOI: 10.29007/zbkv.

[36] S. Schupp, E. Ábrahám, I. Ben Makhlouf, and S. Kowalewski. "HyPro: A C++ library of state set representations for hybrid systems reachability analysis". In: *Proc. of the 9th NASA Formal Methods Symposium.* 2017, pp. 288–294. DOI: `10.1007/978-3-319-57288-8_20`.

[37] N. Kochdumper. "Extensions of polynomial zonotopes and their application to verification of cyber-physical systems". Dissertation. Technische Universität München, 2022.

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

[39] T. Dang. "Verification and synthesis of hybrid systems". Dissertation. Institut National Polytechnique de Grenoble, 2000.

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

[41] A. Chutinan and B. H. Krogh. "Computational techniques for hybrid system verification". In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 64–75. DOI: `10.1109/TAC.2002.806655`.

[42] H. R. Tiwary. "On the hardness of computing intersection, union and Minkowski sum of polytopes". In: *Discrete & Computational Geometry* 40.3 (2008), pp. 469–479. DOI: `10.1007/s00454-008-9097-3`.

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

[44] X. Yang and J. K. Scott. "A comparison of zonotope order reduction techniques". In: *Automatica* 95 (2016), pp. 378–384. DOI: `10.1016/j.automatica.2018.06.006`.

[45] A.-K. Kopetzki, B. Schürmann, and M. Althoff. "Methods for order reduction of zonotopes". In: *Proc. of the 56th Conference on Decision and Control.* IEEE. 2017, pp. 5626–5633. DOI: `10.1109/CDC.2017.8264508`.

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

[47] M. Althoff, C. Le Guernic, and B. H. Krogh. "Reachable set computation for uncertain time-varying linear systems". In: *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control.* ACM, 2011, pp. 93–102. DOI: `10.1145/1967701.1967717`.

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

[49] C. Le Guernic. “Reachability analysis of hybrid systems with linear continuous dynamics”. Dissertation. Université Joseph-Fourier – Grenoble I, 2009.

[50] C. Le Guernic and A. Girard. “Reachability analysis of linear systems using support functions”. In: *Nonlinear Analysis: Hybrid Systems* 4.2 (2010), pp. 250–262. DOI: 10.1016/j.nahs.2009.03.002.

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

[52] M. Althoff and G. Frehse. “Combining zonotopes and support functions for efficient reachability analysis of linear systems”. In: *Proc. of the 55th Conference on Decision and Control.* IEEE. 2016, pp. 7439–7446. DOI: 10.1109/CDC.2016.7799418.

[53] M. Forets and C. Schilling. “Conservative time discretization: A comparative study”. In: *Proc. of the International Conference on Integrated Formal Methods.* Springer. 2022, pp. 149–167. DOI: 10.1007/978-3-031-07727-2_9.

[54] E. Luo, N. Kochdumper, and S. Bak. “Reachability analysis for linear systems with uncertain parameters using polynomial zonotopes”. In: *Proc. of the 26th International Conference on Hybrid Systems: Computation and Control.* ACM, 2023. DOI: 10.1145/3575870.3587130.

[55] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. “SpaceEx: Scalable verification of hybrid systems”. In: *Proc. of the 23rd International Conference on Computer Aided Verification.* LNCS 6806. Springer, 2011, pp. 379–395. DOI: 10.1007/978-3-642-22110-1_30.

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

[57] R. Ray, A. Gurung, B. Das, E. Bartocci, S. Bogomolov, and R. Grosu. “XSpeed: Accelerating reachability analysis on multi-core processors”. In: *Haifa Verification Conference.* Springer. 2015, pp. 3–18. DOI: 10.1007/978-3-319-26287-1_1.

[58] A. Gurung, R. Ray, E. Bartocci, S. Bogomolov, and R. Grosu. “Parallel reachability analysis of hybrid systems in XSpeed”. In: *International Journal on Software Tools for Technology Transfer* 21.4 (2019), pp. 401–423. DOI: 10.1007/s10009-018-0485-6.

[59] M. Althoff. “Reachability analysis of large linear systems with uncertain inputs in the Krylov subspace”. In: *IEEE Transactions on Automatic Control* 65.2 (2020), pp. 477–492. DOI: 10.1109/TAC.2019.2906432.

[60] S. Bak, H.-D. Tran, and T. T. Johnson. “Numerical verification of affine systems with up to a billion dimensions”. In: *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control.* ACM, 2019, pp. 23–32. DOI: 10.1145/3302504.3311792.

[61] 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: *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control.* ACM, 2018, pp. 41–50. DOI: 10.1145/3178126.3178128.

[62] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. “Reachability analysis of linear hybrid systems via block decomposition”. In: *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 39.11 (2020), pp. 4018–4029. DOI: 10.1109/TCAD.2020.3012859.

[63] A. A. Kurzhanskiy and P. Varaiya. “Reach set computation and control synthesis for discrete-time dynamical systems with disturbances”. In: *Automatica* 47.7 (2011), pp. 1414–1426. DOI: 10.1016/j.automatica.2011.02.009.

[64] L. Yang and N. Ozay. “Scalable zonotopic under-approximation of backward reachable sets for uncertain linear systems”. In: *IEEE Control Systems Letters* 6 (2022), pp. 1555–1560. DOI: 10.1109/LCSYS.2021.3123228.

[65] E. Asarin, T. Dang, and A. Girard. “Hybridization methods for the analysis of nonlinear systems”. In: *Acta Informatica* 43 (2007), pp. 451–476. DOI: 10.1007/s00236-006-0035-7.

[66] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. “HyTech: A model checker for hybrid systems”. In: *Proc. of the 9th International Conference on Computer Aided Verification.* Springer. 1997, pp. 460–463. DOI: 10.1007/3-540-63166-6_48.

[67] G. Frehse. “PHAVer: Algorithmic verification of hybrid systems past HyTech”. In: *Proc. of the 8th International workshop on Hybrid Systems: Computation and Control.* Springer. 2005, pp. 258–273. DOI: 10.1007/978-3-540-31954-2_17.

[68] E. Asarin, T. Dang, and A. Girard. “Reachability analysis of nonlinear systems using conservative approximation”. In: *6th International Workshop on Hybrid Systems: Computation and Control.* Springer. 2003, pp. 20–35. DOI: 10.1007/3-540-36580-X_5.

[69] T. Dang, C. Le Guernic, and O. Maler. “Computing reachable states for nonlinear biological models”. In: *Proc. of the International Conference on Computational Methods in Systems Biology.* Springer. 2009, pp. 126–141. DOI: 10.1007/978-3-642-03845-7_9.

[70] T. Dang, C. Le Guernic, and O. Maler. “Computing reachable states for nonlinear biological models”. In: *Theoretical Computer Science* 412.21 (2011), pp. 2095–2107. DOI: 10.1016/j.tcs.2011.01.014.

[71] T. Dang, O. Maler, and R. Testylier. “Accurate hybridization of nonlinear systems”. In: *Proc. of the 13th International Conference on Hybrid Systems: Computation and Control.* ACM, 2010, pp. 11–19. DOI: 10.1145/1755952.1755956.

[72] D. Li, S. Bak, and S. Bogomolov. "Reachability analysis of nonlinear systems using hybridization and dynamics scaling". In: *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2020, pp. 265–282. DOI: 10.1007/978-3-030-57628-8_16.

[73] M. Althoff, O. Stursberg, and M. Buss. "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization". In: *Proc. of the 47th Conference on Decision and Control*. IEEE. 2008, pp. 4042–4048. DOI: 10.1109/CDC.2008.4738704.

[74] M. Althoff and B. H. Krogh. "Reachability analysis of nonlinear differential-algebraic systems". In: *IEEE Transactions on Automatic Control* 59.2 (2014), pp. 371–383. DOI: 10.1109/TAC.2013.2285751.

[75] A. El-Guindy. "Control and stability of power systems using reachability analysis". Dissertation. Technische Universität München, 2017.

[76] M. Althoff. "Formal and compositional analysis of power systems using reachable sets". In: *IEEE Transactions on Power Systems* 29.5 (2014), pp. 2270–2280. DOI: 10.1109/TPWRS.2014.2306731.

[77] M. Althoff. "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets". In: *Proc. of the 16th International Conference on Hybrid Systems: Computation and Control*. ACM, 2013, pp. 173–182. DOI: 10.1145/2461328.2461358.

[78] N. Kochdumper, B. Schürmann, and M. Althoff. "Utilizing dependencies to obtain subsets of reachable sets". In: *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control*. ACM, 2020. DOI: 10.1145/3365365.3382192.

[79] E. Goubault, O. Mullier, S. Putot, and M. Kieffer. "Inner approximated reachability analysis". In: *Proc. of the 17th International Conference on Hybrid Systems: Computation and Control*. ACM, 2014, pp. 163–172. DOI: 10.1145/2562059.2562113.

[80] A. Goldsztejn and L. Jaulin. "Inner approximation of the range of vector-valued functions". In: *Reliable Computing* 14 (2010), pp. 1–23.

[81] O. Mullier, E. Goubault, M. Kieffer, and S. Putot. "General inner approximation of vector-valued functions". In: *Reliable Computing* 18 (2013), pp. 117–143.

[82] L. H. de Figueiredo and J. Stolfi. "Affine arithmetic: Concepts and applications". In: *Numerical Algorithms* 37 (2004), pp. 147–158. DOI: 10.1023/B:NUMA.0000049462.70970.b6.

[83] E. Goubault and S. Putot. "Forward inner-approximated reachability of non-linear continuous systems". In: *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 2017, pp. 1–10. DOI: 10.1145/3049797.3049811.

[84] E. Goubault and S. Putot. "Inner and outer reachability for the verification of control systems". In: *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*. ACM, 2019, pp. 11–22. DOI: 10.1145/3302504.3311794.

[85] E. Goubault and S. Putot. “Robust under-approximations and application to reachability of non-linear control systems with disturbances”. In: *IEEE Control Systems Letters* 4.4 (2020), pp. 928–933. DOI: 10.1109/LCSYS.2020.2997261.

[86] X. Chen, S. Sankaranarayanan, and E. Ábrahám. “Under-approximate flowpipes for non-linear continuous systems”. In: *Formal Methods in Computer-Aided Design*. IEEE. 2014, pp. 59–66. DOI: 10.1109/FMCAD.2014.6987596.

[87] B. Xue, Z. She, and A. Easwaran. “Under-approximating backward reachable sets by polytopes”. In: *International Conference on Computer Aided Verification*. Springer. 2016, pp. 457–476. DOI: 10.1007/978-3-319-41528-4_25.

[88] B. Xue, Z. She, and A. Easwaran. “Underapproximating backward reachable sets by semialgebraic sets”. In: *IEEE Transactions on Automatic Control* 62.10 (2017), pp. 5185–5197. DOI: 10.1109/TAC.2017.2694351.

[89] N. Kochdumper and M. Althoff. “Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems”. In: *Proc. of the 59th Conference on Decision and Control*. IEEE. 2020, pp. 2130–2137. DOI: 10.1109/CDC42340.2020.9304022.

[90] L. Yang, H. Zhang, J.-B. Jeannin, and N. Ozay. “Efficient backward reachability using the minkowski difference of constrained zonotopes”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.11 (2022), pp. 3969–3980. DOI: 10.1109/TCAD.2022.3197971.

[91] A. Girard and C. Le Guernic. “Zonotope/hyperplane intersection for hybrid systems reachability analysis”. In: *Proc. of the 11th International Workshop on Hybrid Systems: Computation and Control*. Springer, 2008, pp. 215–228. DOI: 10.1007/978-3-540-78929-1_16.

[92] M. Althoff and B. H. Krogh. “Avoiding geometric intersection operations in reachability analysis of hybrid systems”. In: *Proc. of the 15th International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 45–54. DOI: 10.1145/2185632.2185643.

[93] X. Chen, E. Ábrahám, and S. Sankaranarayanan. “Taylor model flowpipe construction for non-linear hybrid systems”. In: *Proc. of the 33rd Real-Time Systems Symposium*. IEEE. 2012, pp. 183–192. DOI: 10.1109/RTSS.2012.70.

[94] G. Frehse and R. Ray. “Flowpipe-guard intersection for reachability computations with support functions”. In: *IFAC Proceedings Volumes* 45.9 (2012), pp. 94–101. DOI: 10.3182/20120606-3-NL-3011.00053.

[95] S. Bak, S. Bogomolov, and M. Althoff. “Time-triggered conversion of guards for reachability analysis of hybrid automata”. In: *Proc. of the 15th International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2017, pp. 133–150. DOI: 10.1007/978-3-319-65765-3_8.

[96] N. Kochdumper and M. Althoff. “Reachability analysis for hybrid systems with nonlinear guard sets”. In: *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control.* ACM, 2020. DOI: 10.1145/3365365.3382194.

[97] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. “Eliminating spurious transitions in reachability with support functions”. In: *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control.* ACM, 2015, pp. 149–158. DOI: 10.1145/2728606.2728622.

[98] T. Nahhal and T. Dang. “Guided randomized simulation”. In: *Proc. of the 10th International Workshop on Hybrid Systems: Computation and Control.* Springer, 2007, pp. 731–735. DOI: 10.1007/978-3-540-71493-4_72.

[99] T. Nahhal and T. Dang. “Test coverage for continuous and hybrid systems”. In: *Proc. of the 19th International Conference on Computer Aided Verification.* Springer, 2007, pp. 219–232. DOI: 10.1007/978-3-540-73368-3_47.

[100] A. Girard and G. J. Pappas. “Verification using simulation”. In: *Proc. of the 9th International Workshop on Hybrid Systems: Computation and Control.* Springer, 2006, pp. 272–286. DOI: 10.1007/11730637_22.

[101] A. Donzé and O. Maler. “Systematic simulation using sensitivity analysis”. In: *Proc. of the 10th International Workshop on Hybrid Systems: Computation and Control.* Springer. 2007, pp. 174–189. DOI: 10.1007/978-3-540-71493-4_16.

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

[103] P. S. Duggirala, S. Mitra, and M. Viswanathan. “Verification of annotated models from executions”. In: *Proc. of the International Conference on Embedded Software.* IEEE, 2013. DOI: 10.1109/EMSOFT.2013.6658604.

[104] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. “C2E2: A verification tool for stateflow models”. In: *Proc. of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems.* Springer. 2015, pp. 68–82. DOI: 10.1007/978-3-662-46681-0_5.

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

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

[107] S. Bak and P. S. Duggirala. “HyLAA: A tool for computing simulation-equivalent reachability for linear systems”. In: *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control.* ACM. 2017, pp. 173–178. DOI: 10.1145/3049797.3049808.

[108] D. Angeli and E. D. Sontag. “Monotone control systems”. In: *IEEE Transactions on Automatic Control* 48.10 (2003), pp. 1684–1698. DOI: 10.1109/TAC.2003.817920.

[109] S. Coogan. “Mixed monotonicity for reachability and safety in dynamical systems”. In: *Proc. of the 59th Conference on Decision and Control.* IEEE, 2020, pp. 5074–5085. DOI: 10.1109/CDC42340.2020.9304391.

[110] S. Coogan and M. Arcak. “Efficient finite abstraction of mixed monotone systems”. In: *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control.* ACM, 2015, pp. 58–67. DOI: 10.1145/2728606.2728607.

[111] S. Coogan and M. Arcak. “Finite abstraction of mixed monotone systems with discrete and continuous inputs”. In: *Nonlinear Analysis: Hybrid Systems* 23 (2017), pp. 254–271. DOI: 10.1016/j.nahs.2016.04.005.

[112] L. Yang, O. Mickelin, and N. Ozay. “On sufficient conditions for mixed monotonicity”. In: *IEEE Transactions on Automatic Control* 64.12 (2019), pp. 5080–5085. DOI: 10.1109/TAC.2019.2909815.

[113] P.-J. Meyer, A. Devonport, and M. Arcak. “TIRA: Toolbox for interval reachability analysis”. In: *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control.* ACM, 2019, pp. 224–229. DOI: 10.1145/3302504.3311808.

[114] G. Frehse. “Computing maximizer trajectories of affine dynamics for reachability”. In: *Proc. of the 54th Conference on Decision and Control.* 2015, pp. 7454–7461. DOI: 10.1109/CDC.2015.7403397.

[115] G. E. Fainekos and G. J. Pappas. “Robustness of temporal logic specifications for continuous-time signals”. In: *Theoretical Computer Science* 410.42 (2009), pp. 4262–4291. DOI: https://doi.org/10.1016/j.tcs.2009.06.021.

[116] A. Donzé and O. Maler. “Robust satisfaction of temporal logic over real-valued signals”. In: *Proc. of the 8th Conference on Formal Modeling and Analysis of Timed Systems.* Springer, 2010, pp. 92–106. DOI: 10.1007/978-3-642-15297-9_9.

[117] Y. W. R. Annapureddy and G. E. Fainekos. “Ant colonies for temporal logic falsification of hybrid systems”. In: *Proc. of the 36th Annual Conference on Industrial Electronics Society.* IEEE, 2010, pp. 91–96. DOI: 10.1109/IECON.2010.5675195.

[118] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas. “Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems”. In: *Proc. of the 13th International Conference on Hybrid Systems: Computation and Control.* ACM, 2010, pp. 211–220. DOI: 10.1145/1755952.1755983.

[119] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta. “Probabilistic temporal logic falsification of cyber-physical systems”. In: *Transactions on Embedded Computing Systems* 12.2s (2013). DOI: 10.1145/2465787.2465797.

[120] H. Abbas and G. E. Fainekos. “Convergence proofs for simulated annealing falsification of safety properties”. In: *50th Annual Allerton Conference on Communication, Control, and Computing.* IEEE. 2012, pp. 1594–1601. DOI: 10.1109/Allerton.2012.6483411.

[121] S. Sankaranarayanan and G. E. Fainekos. “Falsification of temporal properties of hybrid systems using the cross-entropy method”. In: *Proc. of the 15th International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 125–134. DOI: 10.1145/2185632.2185653.

[122] A. Dokhanchi, A. Zutshi, R. T. Sriniva, S. Sankaranarayanan, and G. E. Fainekos. “Requirements driven falsification with coverage metrics”. In: *Proc. of the International Conference on Embedded Software*. 2015, pp. 31–40. DOI: 10.1109/EMSOFT.2015.7318257.

[123] Z. Ramezani, J. L. Eddeland, K. Claessen, M. Fabian, and K. Åkesson. “Multiple objective functions for falsification of cyber-physical systems”. In: *IFAC-PapersOnLine* 53.4 (2020), pp. 417–422. DOI: 10.1016/j.ifacol.2021.04.040.

[124] L. Mathesen, G. Pedrielli, and G. E. Fainekos. “Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements”. In: *Proc. of the International Conference on Automation Science and Engineering*. 2021, pp. 732–737. DOI: 10.1109/CASE49439.2021.9551474.

[125] Y. W. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. “S-TaLiRo: A tool for temporal logic falsification for hybrid systems”. In: *Proc. of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 254–257. DOI: 10.1007/978-3-642-19835-9_21.

[126] Y. Yamagata, S. Liu, T. Akazaki, Y. Duan, and J. Hao. “Falsification of cyber-physical systems using deep reinforcement learning”. In: *IEEE Transactions on Software Engineering* 47.12 (2021), pp. 2823–2840. DOI: 10.1109/TSE.2020.2969178.

[127] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh. “Verification of a cruise control system using counterexample-guided search”. In: *Control Engineering Practice* 12.10 (2004), pp. 1269–1278. DOI: 10.1016/j.conengprac.2004.04.002.

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

[129] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski. “A trajectory splicing approach to concretizing counterexamples for hybrid systems”. In: *Proc. of the 52nd Conference on Decision and Control*. IEEE, 2013, pp. 3918–3925. DOI: 10.1109/CDC.2013.6760488.

[130] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. “Multiple shooting, CEGAR-based falsification for hybrid systems”. In: *Proc. of the 14th International Conference on Embedded Software*. ACM, 2014. DOI: 10.1145/2656045.2656061.

[131] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. “Guided search for hybrid systems based on coarse-grained space abstractions”. In: *International Journal on Software Tools for Technology Transfer* 18 (2016), pp. 449–467. DOI: 10.1007/s10009-015-0393-y.

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

[133] T. T. Johnson, S. Bak, M. Caccamo, and L. Sha. "Real-time reachability for verified simplex design". In: *Transactions on Embedded Computing Systems* 15.2 (2016). DOI: `10.1145/2723871`.

[134] S. Bak, S. Bogomolov, and C. Schilling. "High-level hybrid systems analysis with Hypy". In: *Proc. of the 3rd International Workshop on Applied Verification of Continuous and Hybrid Systems*. 2016, pp. 80–90. DOI: `10.29007/4f3d`.

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

[136] S. Ray and P. S. V. Nataraj. "A matrix method for efficient computation of Bernstein coefficients." In: *Reliable Computing* 17.1 (2012), pp. 40–71.

[137] T. Dang. "Approximate reachability computation for polynomial systems". In: *Proc. of the 9th International Workshop on Hybrid Systems: Computation and Control.* Springer, 2006, pp. 138–152. DOI: `10.1007/11730637_13`.

[138] T. Dang and D. Salinas. "Image computation for polynomial dynamical systems using the Bernstein expansion". In: *Proc. of the 21st International Conference on Computer Aided Verification.* Springer, 2009, pp. 219–232. DOI: `10.1007/978-3-642-02658-4_19`.

[139] M. A. Ben Sassi, R. Testylier, T. Dang, and A. Girard. "Reachability analysis of polynomial systems using linear programming relaxations". In: *Proc. of the 10th International Symposium on Automated Technology for Verification and Analysis.* Springer, 2012, pp. 137–151. DOI: `10.1007/978-3-642-33386-6_12`.

[140] T. Dang and R. Testylier. "Reachability analysis for polynomial dynamical systems using the Bernstein expansion". In: *Reliable Computing* 17.2 (2012), pp. 128–152.

[141] T. Dreossi, T. Dang, and C. Piazza. "Parallelotope bundles for polynomial reachability". In: *Proc. of the 19th International Workshop on Hybrid Systems: Computation and Control.* ACM, 2016, pp. 297–306. DOI: `10.1145/2883817.2883838`.

[142] T. Dreossi, T. Dang, and C. Piazza. "Reachability computation for polynomial dynamical systems". In: *Formal Methods in System Design* 50.1 (2017), pp. 1–38. DOI: `10.1007/s10703-016-0266-3`.

[143] T. Dreossi. "Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems". In: *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control.* ACM, 2017, pp. 29–34. DOI: `10.1145/3049797.3049824`.

[144] E. Kim, S. Bak, and P. S. Duggirala. "Automatic dynamic parallelotope bundles for reachability analysis of nonlinear systems". In: *Formal Modeling and Analysis of Timed Systems*. Springer, 2021, pp. 50–66. DOI: 10.1007/978-3-030-85037-1_4.

[145] E. Kim and P. S. Duggirala. "Kaa: A Python implementation of reachable set computation using Bernstein polynomials". In: *Proc. of the 7th International Workshop on Applied Verification of Continuous and Hybrid Systems*. 2020, pp. 184–196. DOI: 10.29007/rs5n.

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

[147] X. Chen. "Reachability analysis of non-linear hybrid systems using Taylor models". Dissertation. RWTH Aachen University, 2015.

[148] X. Chen and S. Sankaranarayanan. "Decomposed reachability analysis for nonlinear systems". In: *Proc. of the 37th Real-Time Systems Symposium*. IEEE. 2016, pp. 13–24. DOI: 10.1109/RTSS.2016.011.

[149] X. Chen, E. Ábrahám, and S. Sankaranarayanan. "Flow*: An analyzer for non-linear hybrid systems". In: *Proc. of the 25th International Conference on Computer-Aided Verification*. LNCS 8044. Springer, 2013, pp. 258–263. DOI: 10.1007/978-3-642-39799-8_18.

[150] M. Arcak, C. Meissen, and A. Packard. *Networks of dissipative systems: Compositional certification of stability, performance, and safety*. Springer, 2016. DOI: 10.1007/978-3-319-29928-0.

[151] H. Yin, A. Packard, M. Arcak, and P. Seiler. "Reachability analysis using dissipation inequalities for uncertain nonlinear systems". In: *Systems and Control Letters* 142 (2020), p. 104736. DOI: 10.1016/j.sysconle.2020.104736.

[152] A. A. Ahmadi, G. Hall, A. Papachristodoulou, J. Saunderson, and Y. Zheng. "Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations". In: *Proc. of the 56th Conference on Decision and Control*. 2017, pp. 453–462. DOI: 10.1109/CDC.2017.8263706.

[153] H. Yin, A. Packard, M. Arcak, and P. Seiler. "Finite horizon backward reachability analysis and control synthesis for uncertain nonlinear systems". In: *American Control Conference*. 2019, pp. 5020–5026. DOI: 10.23919/ACC.2019.8814444.

[154] H. Yin, P. Seiler, and M. Arcak. "Backward reachability using integral quadratic constraints for uncertain nonlinear systems". In: *Control Systems Letters* 5.2 (2021), pp. 707–712. DOI: 10.1109/LCSYS.2020.3005315.

[155] H. Yin, M. Arcak, A. Packard, and P. Seiler. "Backward reachability for polynomial systems on a finite horizon". In: *IEEE Transactions on Automatic Control* 66.12 (2021), pp. 6025–6032. DOI: 10.1109/TAC.2021.3056611.

[156] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. “A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games”. In: *IEEE Transactions on Automatic Control* 50.7 (2005), pp. 947–957. DOI: 10.1109/TAC.2005.851439.

[157] K. Margellos and J. Lygeros. “Hamilton-Jacobi formulation for reach–avoid differential games”. In: *IEEE Transactions on Automatic Control* 56.8 (2011), pp. 1849–1861. DOI: 10.1109/TAC.2011.2105730.

[158] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. “Hamilton-Jacobi reachability: A brief overview and recent advances”. In: *Proc. of the 56th Conference on Decision and Control.* IEEE. 2017, pp. 2242–2253. DOI: 10.1109/CDC.2017.8263977.

[159] M. Chen and C. J. Tomlin. “Hamilton-Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 333–358. DOI: 10.1146/annurev-control-060117-104941.

[160] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. “Reach-avoid problems with time-varying dynamics, targets and constraints”. In: *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control.* ACM, 2015, pp. 11–20. DOI: 10.1145/2728606.2728612.

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

[162] M. Chen, S. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin. “Decomposition of reachable sets and tubes for a class of nonlinear systems”. In: *IEEE Transactions on Automatic Control* 63.11 (2018), pp. 3675–3688. DOI: 10.1109/TAC.2018.2797194.

[163] D. Lee, M. Chen, and C. J. Tomlin. “Removing leaking corners to reduce dimensionality in Hamilton-Jacobi reachability”. In: *Proc. of the International Conference on Robotics and Automation.* IEEE. 2019, pp. 9320–9326. DOI: 10.1109/ICRA.2019.8793890.

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

[165] M. Jones and M. M. Peet. “Relaxing the Hamilton Jacobi Bellman equation to construct inner and outer bounds on reachable sets”. In: *Proc. of the 58th Conference on Decision and Control.* IEEE. 2019, pp. 2397–2404. DOI: 10.1109/CDC40024.2019.9029193.

[166] B. Xue, M. Fränzle, and N. Zhan. “Under-approximating reach sets for polynomial continuous systems”. In: *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control.* ACM, 2018, pp. 51–60. DOI: 10.1145/3178126.3178133.

[167] B. Xue, M. Fränzle, and N. Zhan. “Inner-approximating reachable sets for polynomial systems with time-varying uncertainties”. In: *IEEE Transactions on Automatic Control* 65.4 (2020), pp. 1468–1483. DOI: `10.1109/TAC.2019.2923049`.

[168] I. M. Mitchell. “The flexible, extensible and efficient toolbox of level set methods”. In: *Journal of Scientific Computing* 35 (2008), pp. 300–329. DOI: `10.1007/s10915-007-9174-4`.

[169] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. “Bridging Hamilton-Jacobi safety analysis and reinforcement learning”. In: *Proc. of the International Conference on Robotics and Automation*. IEEE. 2019, pp. 8550–8556. DOI: `10.1109/ICRA.2019.8794107`.

[170] S. Herbert, J. J. Choi, S. Sanjeev, M. Gibson, K. Sreenath, and C. J. Tomlin. “Scalable learning of safety guarantees for autonomous systems using Hamilton-Jacobi reachability”. In: *Proc. of the International Conference on Robotics and Automation*. IEEE. 2021, pp. 5914–5920. DOI: `10.1109/ICRA48506.2021.9561561`.

[171] S. Bansal and C. J. Tomlin. “DeepReach: A deep learning approach to high-dimensional reachability”. In: *Proc. of the International Conference on Robotics and Automation*. IEEE. 2021, pp. 1817–1824. DOI: `10.1109/ICRA48506.2021.9561949`.

[172] S. Prajna and A. Jadbabaie. “Safety verification of hybrid systems using barrier certificates”. In: *Proc. of the 7th International Workshop on Hybrid Systems: Computation and Control*. Springer. 2004, pp. 477–492. DOI: `10.1007/978-3-540-24743-2_32`.

[173] C. Sloth, G. J. Pappas, and R. Wisniewski. “Compositional safety analysis using barrier certificates”. In: *Proc. of the 15th International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 15–24. DOI: `10.1145/2185632.2185639`.

[174] H. Kong, X. Song, D. Han, M. Gu, and J. Sun. “A new barrier certificate for safety verification of hybrid systems”. In: *The Computer Journal* 57.7 (2014), pp. 1033–1045. DOI: `10.1093/comjnl/bxt059`.

[175] L. Dai, T. Gan, B. Xia, and N. Zhan. “Barrier certificates revisited”. In: *Journal of Symbolic Computation* 80 (2017), pp. 62–86. DOI: `10.1016/j.jsc.2016.07.010`.

[176] H. Zhao, X. Zeng, T. Chen, and Z. Liu. “Synthesizing barrier certificates using neural networks”. In: *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control*. ACM. 2020. DOI: `10.1145/3365365.3382222`.

[177] Q. Zhao, X. Chen, Y. Zhang, M. Sha, Z. Yang, W. Lin, E. Tang, Q. Chen, and X. Li. “Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems”. In: *Proc. of the 24th International Conference on Hybrid Systems: Computation and Control*. ACM, 2021. DOI: `10.1145/3447928.3456638`.

[178] S. Prajna and A. Rantzer. “Convex programs for temporal verification of nonlinear dynamical systems”. In: *SIAM Journal on Control and Optimization* 46.3 (2007), pp. 999–1021. DOI: `10.1137/050645178`.

[179] P. Wieland and F. Allgöwer. “Constructive safety using control barrier functions”. In: *IFAC Proceedings Volumes* 40.12 (2007), pp. 462–467. DOI: 10.3182/20070822-3-ZA-2920.00076.

[180] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. “Control barrier function based quadratic programs for safety critical systems”. In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876. DOI: 10.1109/TAC.2016.2638961.

[181] M. Chen, J. C. Shih, and C. J. Tomlin. “Multi-vehicle collision avoidance via Hamilton-Jacobi reachability and mixed integer programming”. In: *Proc. of the 55th Conference on Decision and Control*. IEEE. 2016, pp. 1695–1700. DOI: 10.1109/CDC.2016.7798509.

[182] T. Ferrère, O. Maler, D. Ničković, and A. Pnueli. “From real-time logic to timed automata”. In: *Journal of the ACM* 66.3 (2019). DOI: 10.1145/3286976.

[183] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. “Reachset conformance testing of hybrid automata”. In: *Proc. of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 277–286. DOI: 10.1145/2883817.2883828.

[184] A. Kulmburg and M. Althoff. “On the co-NP-completeness of the zonotope containment problem”. In: *European Journal of Control* 62 (2021), pp. 84–91. DOI: 10.1016/j.ejcon.2021.06.028.

[185] M. Rungger and M. Zamani. “Accurate reachability analysis of uncertain nonlinear systems”. In: *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*. ACM, 2018, pp. 61–70. DOI: 10.1145/3178126.3178127.

A Reproduction of Publications

A.1 Adaptive Parameter Tuning for Reachability Analysis of Linear Systems

Summary Reachability algorithms require manual tuning of algorithm parameters, such as the time step size, in order to tightly enclose the reachable set. In general, the tightness of the computed approximation with respect to the exact reachable set is unknown, so one can never rule out spurious counterexamples.

In this work, we introduce a framework for tuning algorithm parameters based on the induced approximation errors in the reachable set computation. We apply the proposed parameter tuning methods to a reachability algorithm for linear systems of the form (2.2), for which we derive approximation errors in terms of the Hausdorff distance between exact partial solutions and the corresponding computed outer approximations. Furthermore, we prove that all algorithm parameters can be adapted without backtracking to fulfill any non-zero admissible error bound over the whole time horizon. To solve the verification task in Problem 1, it suffices to tune this scalar error bound.

We implement the automated reachability algorithm using zonotopes as a set representation. Our numerical evaluation shows that the novel algorithm verifies safety for several high-dimensional benchmarks at least as fast as its manually tuned counterpart.

Author contributions M.W. developed the idea of tuning algorithm parameters by bounding approximation errors, implemented the algorithm, conducted the numerical evaluation, and wrote most of the manuscript. N.K. significantly improved the technical presentation of the work and helped with the numerical evaluation. M.A. initiated the idea of automated parameter tuning, laid out the structure of the paper, and provided feedback for improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Adaptive Parameter Tuning for Reachability Analysis of Linear Systems

Mark Wetzlinger, Niklas Kochdumper, and Matthias Althoff

Abstract— **Despite the possibility to quickly compute reachable sets of large-scale linear systems, current methods are not yet widely applied by practitioners. The main reason for this is probably that current approaches are not push-button-capable and still require to manually set crucial parameters, such as time step sizes and the accuracy of the used set representation—these settings require expert knowledge. We present a generic framework to automatically find near-optimal parameters for reachability analysis of linear systems given a user-defined accuracy. To limit the computational overhead as much as possible, our methods tune all relevant parameters during runtime. We evaluate our approach on benchmarks from the ARCH competition as well as on random examples. The results show that our new framework verifies the selected benchmarks faster than manually-tuned parameters and is an order of magnitude faster compared to genetic algorithms.**

I. Introduction

Reachability analysis is one of the main techniques to formally verify the correctness of mixed discrete/continuous systems: It computes the set of reachable states of a system for a set of uncertain initial states as well as uncertain inputs. If the reachable set does not intersect the unsafe regions defined by a given safety property, that property is satisfied.

Exact reachable sets can only be computed for a limited class of systems [1]. In the remaining cases, one calculates over-approximations whose tightness is strongly influenced by algorithm parameters; unlike model parameters, these parameters have no relation to the considered model.

Due to large over-approximations resulting from poor algorithm parameters, reachability analysis may fail to prove a safety property even though the property is satisfied by the exact reachable set. We address this problem by proposing a novel generic framework to automatically tune all algorithm parameters for reachability analysis of linear continuous time-invariant systems during runtime respecting a user-defined error bound. Our framework can be used for different reachability algorithms and for different set representations.

a) State of the Art: Reachability algorithms for linear continuous systems are mainly based on the propagation of reachable sets for multiple time steps until reaching a fixed point or a user-defined time horizon. Propagation-based techniques have been extensively investigated [2]–[7] using tools such as *CORA* [8], *Flow** [9], *XSpeed* [10], *SpaceEx* [7], and *JuliaReach* [11]. A further method is to compute reachable sets using simulations [12], [13], which is implemented in the tool *HyLAA* [14]. The used set representation is another distinctive feature besides the method to compute reachable sets. Many different set representations have been researched, including polytopes [7], zonotopes [5], ellipsoids [15], griddy polyhedra [16], star sets [12], support functions [17], and constrained zonotopes [18].

Previous approaches for automated parameter tuning mainly focused on the time step size: Numerical ODE solvers often compute different solutions in parallel and decrease the time step size if the difference between solutions exceeds a certain threshold [19], [20]. Furthermore, several automated time step adaptation strategies have been developed for *guaranteed integration* methods that enclose only a single trajectory rather than a set of trajectories [21]–[23]. For reachability analysis, only a few methods automatically set the time step size. The approach in [7] chooses the time step size automatically in order to keep the error in a user-defined direction below a user-defined bound. Furthermore, the approach in [24] automatically chooses suitable time step sizes for affine systems so that the Hausdorff distance between the reachable set obtained by the exact flow and the approximated flow stays below a certain threshold. So far, there is no approach that considers all algorithm parameters.

b) Contributions: We introduce a novel generic framework that automatically tunes all algorithm parameters of a reachability algorithm so that the over-approximation error stays below a user-defined threshold. This generalized framework can be applied to many different reachability algorithms and we show that our self-parametrization approach always converges if the reachability algorithm satisfies certain requirements.

After introducing some preliminaries in Sec. II, we present our generic framework in Sec. III. In Sec. IV, we demonstrate the implementation of our framework for a specific reachability algorithm. Finally, the evaluation of our implementation on several numerical examples in Sec. V demonstrates the fully automated computation of tight over-approximations of reachable sets with only little computational overhead.

II. Preliminaries

To concisely explain the novelties of our paper, we first introduce some preliminaries.

A. Notation

Vectors are denoted by lower-case letters and matrices by upper-case letters. Square matrices of zeros are denoted by $0_n \in \mathbb{R}^{n\times n}$ and the identity matrix by $I_n \in \mathbb{R}^{n\times n}$. Given a vector $a \in \mathbb{R}^n$, a_i refers to the i-th entry. An n-dimensional interval is denoted by $\mathcal{I} = [\underline{l}, \overline{l}] \subset \mathbb{R}^n$, where $\underline{l}_i \leq \overline{l}_i$, $\forall i \in$

M. Wetzlinger, N. Kochdumper, and M. Althoff are with the Faculty of Computer Science, Technical University of Munich, Garching, Germany {m.wetzlinger@tum.de, niklas.kochdumper@tum.de, althoff@tum.de}. The authors gratefully acknowledge partial financial supports from the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.

$\{1,...,n\}$. The operators $\inf(\mathcal{I}) = \underline{l} \in \mathbb{R}^n$ and $\sup(\mathcal{I}) = \bar{l} \in \mathbb{R}^n$ return the infimum and the supremum of an interval $\mathcal{I} = [\underline{l}, \bar{l}]$. Interval matrices are denoted in boldface: $\mathbf{I} = [\underline{I}, \bar{I}] \in \mathbb{R}^{m\times n}$, where $\inf(\mathbf{I}) = \underline{I} \in \mathbb{R}^{m\times n}$ and $\sup(\mathbf{I}) = \bar{I} \in \mathbb{R}^{m\times n}$. The linear map is written without any operator between the operands, the Minkowski sum is denoted by $\oplus$, and the convex hull operator by $\texttt{conv}(\,\cdot\,)$.

B. Reachability Analysis of Linear Systems

The presented technique for adaptive self-parametrization is applied to linear time-invariant (LTI) systems with initial states bounded by $\mathcal{X}^0 \subset \mathbb{R}^n$ and inputs bounded by $\mathcal{U} \subset \mathbb{R}^n$:

$$\begin{aligned} &\dot{x}(t) = Ax(t) + u(t) \ , \\ &\text{with} \quad x(0) \in \mathcal{X}^0 \subset \mathbb{R}^n, \ \forall t : u(t) \in \mathcal{U} \subset \mathbb{R}^n \ , \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is the system matrix, $x(t) \in \mathbb{R}^n$ is the state vector, and $u(t) \in \mathbb{R}^n$ is the input vector.

The above system also encompasses systems $\dot{x}(t) = Ax(t) + B\tilde{u}(t)$, where $B \in \mathbb{R}^{m\times n}$ is the input matrix, as we can set $\mathcal{U} = \{B\tilde{u} \,|\, \tilde{u} \in \mathcal{D} \subset \mathbb{R}^m\}$. Let us introduce $\xi\big(t; x_0, u(\cdot)\big)$ as the solution of (1) to define the exact reachable set $\mathcal{R}_{\text{ex}}\big([0,t_f]\big)$ of (1) over the time horizon $t \in [0,t_f]$:

$$\mathcal{R}_{\text{ex}}\big([0,t_f]\big) = \Big\{ \xi\big(t; x_0, u(\cdot)\big) \,\Big|\, x_0 \in \mathcal{X}^0, \\ \forall \tau \in [0,t] : u(\tau) \in \mathcal{U}, \, t \in [0,t_f] \Big\} \ .$$

Due to the superposition principle, $\mathcal{R}_{\text{ex}}\big([0,t_f]\big)$ is the sum of the homogeneous solution $\mathcal{H}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big)$ and the inhomogeneous solution $\mathcal{P}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big)$ [25, (3.6)]:

$$\mathcal{R}_{\text{ex}}\big([0,t_f]\big) = \mathcal{H}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big) \oplus \mathcal{P}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big) \ ,$$

where

$$\begin{aligned} \mathcal{H}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big) &= \Big\{ e^{At}x_0 \,\Big|\, x_0 \in \mathcal{X}^0, \, t \in [0,t_f] \Big\} \ , \\ \mathcal{P}^{\mathcal{R}}_{\text{ex}}\big(t_f\big) &= \Big\{ \int_0^{t_f} e^{A(t_f-\tau)}u(\tau)\,\mathrm{d}\tau \,\Big|\, u(\tau) \in \mathcal{U} \Big\} \ . \end{aligned}$$

If $0 \in \mathcal{U}$, $\mathcal{P}^{\mathcal{R}}_{\text{ex}}\big([0,t_f]\big) = \mathcal{P}^{\mathcal{R}}_{\text{ex}}\big(t_f\big)$ and the extension for $0 \notin \mathcal{U}$ is shown in [25, Sec. 3.2.2]. To limit the over-approximation, the time horizon $[0,t_f]$ is discretized by K time steps $\Delta t_i = t_{i+1} - t_i > 0$, $i \in \{0,...,K-1\}$, where $t_f = \sum_{i=0}^{K-1} \Delta t_i$ so that $\bigcup_{i=0}^{K-1} \mathcal{R}\big([t_i,t_{i+1}]\big)$. We compute the reachable sets from the first time interval [26, Chap. 4]:

$$\begin{aligned} \mathcal{H}^{\mathcal{R}}\big([t_i,t_{i+1}]\big) &= e^{At_i}\mathcal{H}^{\mathcal{R}}\big([0,\Delta t_i]\big) \ , \\ \mathcal{P}^{\mathcal{R}}\big([0,t_{i+1}]\big) &= \mathcal{P}^{\mathcal{R}}\big([0,t_i]\big) \oplus \underbrace{e^{At_i}\mathcal{P}^{\mathcal{R}}\big([0,\Delta t_i]\big)}_{=:\,\mathcal{P}^{\mathcal{R}}([t_i,t_{i+1}])} \ . \end{aligned} \tag{2}$$

This method adapts seamlessly to varying time step sizes, making it the preferred choice for adaptive parameter tuning compared to methods propagating the reachable set from the previous time step.

III. Self-Parametrization

In this section, we introduce our novel algorithm for adaptive parameter tuning.

A. Overview of Algorithm Parameters

The reachability algorithm from Sec. II-B depends on different algorithm parameters, which we divide into:

- **Time Step Size:** The propagation in (2) can be applied to any series of Δt_i adding up to t_f. Large Δt speed up the computation, while small Δt increase the precision.
- **Propagation Parameters:** The tightness of the computed reachable sets also depends on parameters approximating the dynamics Φ_{prop}, e.g., determining the precision of computing e^{At}. We introduce the operator $\texttt{incr}(\Phi_{\text{prop}})$ which increments the parameters Φ_{prop} towards values which result in a tighter reachable set, as well as the operator $\texttt{reset}(\Phi_{\text{prop}})$ which resets Φ_{prop} to the coarsest setting.
- **Set Representation:** We denote the number of scalar values needed to describe a set by Φ_{set}. Let us introduce the operator $\texttt{decr}(\Phi_{\text{set}})$ which decreases Φ_{set}. If Φ_{set} has reached its minimum, it returns $\texttt{decr}(\Phi_{\text{set}}) = \Phi_{\text{set}}$. We denote the over-approximation of a set $\mathcal{S}$ by reducing its number of parameters to a desired value Φ_{set} by the operator $\texttt{red}\big(\mathcal{S}, \Phi_{\text{set}}\big)$.

B. Error measures

As we can only compute over-approximations, we are interested in measuring the over-approximation error. Let the reachable set $\mathcal{H}^{\mathcal{R}}$ consist of summands, e.g., $\mathcal{H}^{\mathcal{R}} = \mathcal{H}^{\mathcal{R}(1)} \oplus ... \oplus \mathcal{H}^{\mathcal{R}(\nu)}$, as shown for a concrete implementation in Sec. IV-A. This assumption is valid for almost all approaches, see e.g. [6], [7], [17], [25], [27]. We sum over all indices $\mathcal{E}$ representing exactly-computed terms to obtain $\mathcal{H}^{\mathcal{R}}_{=} = \bigoplus_{i\in\mathcal{E}}^{\nu} \mathcal{H}^{\mathcal{R}(i)}$ and the remaining terms are summed to $\mathcal{H}^{\mathcal{R}}_{+} = \bigoplus_{i\notin\mathcal{E}}^{\nu} \mathcal{H}^{\mathcal{R}(i)}$. The same is done for $\mathcal{P}^{\mathcal{R}}$, so that we obtain

$$\mathcal{H}^{\mathcal{R}}\big([0,\Delta t]\big) = \mathcal{H}^{\mathcal{R}}_{=}\big([0,\Delta t]\big) \oplus \mathcal{H}^{\mathcal{R}}_{+}\big([0,\Delta t]\big) \ , \tag{3}$$

$$\mathcal{P}^{\mathcal{R}}\big([0,\Delta t]\big) = \mathcal{P}^{\mathcal{R}}_{=}\big([0,\Delta t]\big) \oplus \mathcal{P}^{\mathcal{R}}_{+}\big([0,\Delta t]\big) \ . \tag{4}$$

We use this separation to realize a computationally efficient over-approximation of the Hausdorff distance

$$d_H(\mathcal{S}_1, \mathcal{S}_2) = \max\Big\{ \sup_{x_1\in\mathcal{S}_1} \big(\inf_{x_2\in\mathcal{S}_2} \|x_1 - x_2\|_2 \big), \\ \sup_{x_2\in\mathcal{S}_2} \big(\inf_{x_1\in\mathcal{S}_1} \|x_1 - x_2\|_2 \big) \Big\} \ .$$

Proposition 1: *Let $\mathcal{S}_= \subset \mathbb{R}^n$ and $\mathcal{S}_+ \subset \mathbb{R}^n$ with $0 \in \mathcal{S}_+$ be non-empty compact, convex sets. The Hausdorff distance between $\mathcal{S}_=$ and $\mathcal{S}_{tot} := \mathcal{S}_= \oplus \mathcal{S}_+$ can be over-approximated by*

$$d_H(\mathcal{S}_=, \mathcal{S}_{tot}) \leq \mathit{err}\big(\mathcal{S}_+\big) := r\big(\mathit{box}(\mathcal{S}_+)\big) \ , \tag{5}$$

where $\mathit{box}\big(\mathcal{S}_+\big)$ is the box enclosure of $\mathcal{S}_+$ and $r(\cdot)$ returns the radius of the smallest hypersphere centered at the origin enclosing its argument.

Proof. Since $\mathcal{S}_{\text{tot}}$ encloses $\mathcal{S}_=$, we have

$$d_H(\mathcal{S}_=, \mathcal{S}_{\text{tot}}) = \sup_{y\in\mathcal{S}_{\text{tot}}} \big(\inf_{x\in\mathcal{S}_=} \|x - y\|_2 \big) \ . \tag{6}$$

We also know that the difference between $\mathcal{S}_{\text{tot}}$ and $\mathcal{S}_{=}$ is given by $\mathcal{S}_{+}$, so that (6) can be equivalently written as

$$\sup_{y\in\mathcal{S}_{\text{tot}}} \Big(\inf_{x\in\mathcal{S}_{=}} \|x-y\|_2 \Big) \overset{0\in\mathcal{S}_+}{=} \sup_{x\in\mathcal{S}_+} \|x\|_2 = r(\mathcal{S}_+) \ .$$

Obviously, $r(\mathcal{S}_+) \leq r\big(\texttt{box}(\mathcal{S}_+)\big)$ and therefore $d_H(\mathcal{S}_{=}, \mathcal{S}_{\text{tot}}) \leq \texttt{err}\big(\mathcal{S}_+\big)$. □

Let us introduce user-defined upper bounds for errors: $\varepsilon_{\mathcal{H},\text{max}}$ for the error of the homogeneous solution $\mathcal{H}^{\mathcal{R}}$ and $\varepsilon_{\mathcal{P},\text{max}}$ for the error of the inhomogeneous solution $\mathcal{P}^{\mathcal{R}}$. Our approach ensures that these errors are not exceeded.

We now compute the errors of the i-th time step. For the error $\varepsilon_{\mathcal{H}}\big([t_i,t_{i+1}]\big)$ and $\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big)$ we use the split in (3) and apply the error measure from Prop. 1:

$$\begin{aligned}\varepsilon_{\mathcal{H}}\big([t_i,t_{i+1}]\big) &:= \texttt{err}\big(\mathcal{H}^{\mathcal{R}}_{+}\big([t_i,t_{i+1}]\big)\big) & (7)\\ &\geq d_H\big(\mathcal{H}^{\mathcal{R}}_{=}\big([t_i,t_{i+1}]\big), \mathcal{H}^{\mathcal{R}}\big([t_i,t_{i+1}]\big)\big) \ , \\ \varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) &:= \texttt{err}\big(\mathcal{P}^{\mathcal{R}}_{+}\big([t_i,t_{i+1}]\big)\big) & (8)\\ &\geq d_H\big(\mathcal{P}^{\mathcal{R}}_{=}\big([t_i,t_{i+1}]\big), \mathcal{P}^{\mathcal{R}}\big([t_i,t_{i+1}]\big)\big) \ .\end{aligned}$$

From (2), we see that the computation of $\varepsilon_{\mathcal{H}}$ does not depend on previous time intervals. Contrary, the inhomogeneous solution $\mathcal{P}^{\mathcal{R}}\big([0,t_i]\big)$ accumulates over time, see (2), and with it the error $\varepsilon_{\mathcal{P}}$:

$$\varepsilon_{\mathcal{P}}\big([0,t_{i+1}]\big) = \varepsilon_{\mathcal{P}}\big([0,t_i]\big) + \varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) \ , \qquad (9)$$

where $\varepsilon_{\mathcal{P}}\big([0,0]\big) = 0$. Let us introduce a mild assumption for the errors $\varepsilon_{\mathcal{H}}$ and $\varepsilon_{\mathcal{P}}$, which will be justified in Sec. IV-B.

Assumption 1: *(Convergence of $\varepsilon_{\mathcal{H}}$ and $\varepsilon_{\mathcal{P}}$) We neglect floating-point errors so that $\varepsilon_{\mathcal{H}}\big([t_i,t_{i+1}]\big) \to 0$ and $\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) \to 0$ for $\Delta t_i \to 0$.* □

Assumption 1 is mild, since it practically holds for all other current approaches, see e.g. [7], [8], [11]. For further derivations we need the following definition.

Definition 1: *(Superlinear decrease) The set-based evaluation $f(\mathcal{S}(t)) = \{f(x)\,|\,x\in\mathcal{S}(t)\}$ of a continuous function $f:\mathbb{R}^n\to\mathbb{R}$ decreases superlinearly if*

$$\forall\varphi\in(0,1): f\big(\mathcal{S}\big([t,t+\varphi\Delta t]\big)\big) \subseteq \varphi\, f\big(\mathcal{S}\big([t,t+\Delta t]\big)\big) \ . \ \square$$

The following proposition addresses satisfying $\varepsilon_{\mathcal{P},\text{max}}$.

Proposition 2: *Let the error $\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big)$ decrease superlinearly according to Def. 1. For any given $\varepsilon_{\mathcal{P},max} > 0$, there exists a sequence of time intervals $[t_i,t_{i+1}]$ so that*

$$\sum_i \varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) \leq \varepsilon_{\mathcal{P},max} \ .$$

Proof. We define an admissible error $\varepsilon_{\mathcal{P},\text{max},i}$ for each step:

$$\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) \leq \varepsilon_{\mathcal{P},\text{max},i} := \frac{\varepsilon_{\mathcal{P},\text{max}} - \varepsilon_{\mathcal{P}}\big([0,t_i]\big)}{t_f - t_i}\Delta t_i. \quad (10)$$

For $\Delta t_i \to 0$, $\varepsilon_{\mathcal{P},\text{max},i}$ converges to 0, as does $\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big)$ by Assumption 1. Moreover, $\varepsilon_{\mathcal{P},\text{max},i}$ decreases linearly in Δt_i, whereas $\varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big)$ decreases superlinearly by assumption. Hence, we can always obtain a Δt_i so that the inequality in (10) holds. We sum each side in (10) to

$$\varepsilon_{\mathcal{P}}\big([0,t_i]\big) \leq \sum_{j=0}^{i-1} \varepsilon_{\mathcal{P},\text{max},j} \qquad (11)$$

which is subsequently used to bound $\varepsilon_{\mathcal{P},\text{max},i}$ and $\varepsilon_{\mathcal{P},\text{max}}$:

$$\begin{aligned}\varepsilon_{\mathcal{P},\text{max},i} &\overset{(10)}{=} \big(\varepsilon_{\mathcal{P},\text{max}} - \varepsilon_{\mathcal{P}}\big([0,t_i]\big)\big)\frac{\Delta t_i}{t_f - t_i} \\ &\overset{(11)}{\leq} \Big(\varepsilon_{\mathcal{P},\text{max}} - \sum_{j=0}^{i-1}\varepsilon_{\mathcal{P},\text{max},j}\Big)\frac{\Delta t_i}{t_f - t_i} \\ \Rightarrow \varepsilon_{\mathcal{P},\text{max}} &\geq \sum_{j=0}^{i-1}\varepsilon_{\mathcal{P},\text{max},j} + \underbrace{\frac{t_f - t_i}{\Delta t_i}}_{\geq 1}\varepsilon_{\mathcal{P},\text{max},i} \\ &\geq \sum_{j=0}^{i}\varepsilon_{\mathcal{P},\text{max},j} \geq \sum_i \varepsilon_{\mathcal{P}}\big([t_i,t_{i+1}]\big) \ . \qquad \square\end{aligned}$$

The Minkowski sum in (2) typically increases the representation size of the resulting set. To counteract this growth, we enclose $\mathcal{P}^{\mathcal{R}}$ in (2) by a set specified by less parameters which over-approximates the original set by the error $\varepsilon_{\mathcal{S}}$ accumulating as the error $\varepsilon_{\mathcal{P}}$ of $\mathcal{P}^{\mathcal{R}}$:

$$\begin{aligned}&\varepsilon_{\mathcal{S}}\big([0,t_{i+1}]\big) = \varepsilon_{\mathcal{S}}\big([0,t_i]\big) + \varepsilon_{\mathcal{S}}\big([t_i,t_{i+1}]\big) \ , \qquad (12)\\ &\text{with} \quad \varepsilon_{\mathcal{S}}\big([0,0]\big) = 0 \ .\end{aligned}$$

Similarly to Prop. 2, we define an admissible error $\varepsilon_{\mathcal{S},\text{max},i}$:

$$\varepsilon_{\mathcal{S}}\big([t_i,t_{i+1}]\big) \leq \varepsilon_{\mathcal{S},\text{max},i} := \frac{\varepsilon_{\mathcal{S},\text{max}} - \varepsilon_{\mathcal{S}}\big([0,t_i]\big)}{t_f - t_i}\Delta t_i \ . \quad (13)$$

If we do not over-approximate the set representation in the i-th time step, we have $\varepsilon_{\mathcal{S}}\big([t_i,t_{i+1}]\big) = 0$.

C. Adaptive Parameter Tuning

We now present our generic framework for adaptive parameter tuning in Alg. 1. The two subroutines, Alg. 2 and Alg. 3, will be presented thereafter. First, we initialize Δt and the scaling factor μ (line 2), as well as the accumulating errors $\varepsilon_{\mathcal{P}}$ and $\varepsilon_{\mathcal{S}}$ (line 3) for the first iteration of the main loop (lines 4-11). In every iteration, we first obtain the parameters Δt_i and $\Phi_{\text{prop},i}$ (line 5) from Alg. 2. Then, we calculate the reachable sets $\mathcal{H}^{\mathcal{R}}\big([0,\Delta t_i]\big)$ and $\mathcal{P}^{\mathcal{R}}\big([0,\Delta t_i]\big)$ (line 6), which are subsequently used to obtain $\mathcal{H}^{\mathcal{R}}\big([t_i,t_{i+1}]\big)$ and $\mathcal{P}^{\mathcal{R}}\big([0,t_{i+1}]\big)$ (line 7). In Alg. 3, we obtain the parameters $\Phi_{\text{set},i}$, which are used to recalculate $\mathcal{P}^{\mathcal{R}}\big([0,t_{i+1}]\big)$ (line 8). At the end of each step, we add the homogeneous and inhomogeneous solution to obtain the reachable set $\mathcal{R}\big([t_i,t_{i+1}]\big)$ (line 9). Finally, after the calculation of the reachable set for each time interval, the reachable set of the whole time horizon $\mathcal{R}\big([0,t_f]\big)$ is obtained by unifying partial sets (line 12).

Theorem 1: *Alg. 1 terminates and the overall error stays below a user-defined error bound $\varepsilon_{max} \in \mathbb{R}^+$, assuming the reduction of the set representation in each step is optional.*

Algorithm 1 Fully automated parameter tuning

```
Input: $\varepsilon_{\mathcal{H},\text{max}}, \varepsilon_{\mathcal{P},\text{max}}, \varepsilon_{\mathcal{S},\text{max}}, t_f$
Output: $\mathcal{R}([0, t_f])$
 1: $t \leftarrow 0, i \leftarrow 0$
 2: $\mu \leftarrow 0.9, \Delta t_{-1} \leftarrow t_f\, \mu$
 3: $\varepsilon_{\mathcal{P}}([0,0]) \leftarrow 0,\ \varepsilon_{\mathcal{S}}([0,0]) \leftarrow 0$
 4: while $t < T$ do
 5:     $\Delta t_i, \Phi_{\text{prop},i}, \varepsilon_{\mathcal{P}}([0, t_{i+1}]) \leftarrow$ Alg. 2
 6:     calc. $\mathcal{H}^{\mathcal{R}}([0, \Delta t_i]), \mathcal{P}^{\mathcal{R}}([0, \Delta t_i])$ using $\Delta t_i$, $\Phi_{\text{prop},i}$
 7:     calc. $\mathcal{H}^{\mathcal{R}}([t_i, t_{i+1}]), \mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$ acc. to (2)
 8:     $\mathcal{P}^{\mathcal{R}}([0, t_{i+1}]), \varepsilon_{\mathcal{S}}([0, t_{i+1}]) \leftarrow$ Alg. 3
 9:     $\mathcal{R}([t_i, t_{i+1}]) = \mathcal{H}^{\mathcal{R}}([t_i, t_{i+1}]) \oplus \mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$
10:     $t \leftarrow t + \Delta t_i,\ i \leftarrow i + 1$
11: end while
12: $\mathcal{R}([0, t_f]) \leftarrow \bigcup_{j=0}^{i-1} \mathcal{R}([t_j, t_{j+1}])$
```

Proof. We first show that the algorithm terminates. The while-loop in Alg. 2 always terminates if

$$\forall i : \varepsilon_{\mathcal{P}}([t_i, t_{i+1}]) \leq \varepsilon_{\mathcal{P},\text{max},i}\ ,\ \varepsilon_{\mathcal{H}}([t_i, t_{i+1}]) \leq \varepsilon_{\mathcal{H},\text{max}}\ .$$

The first inequality can be satisfied as shown in the proof of Prop. 2. From line 6 in Alg. 2 and under Assumption 1, we have $\forall i : \Delta t_i \to 0 \Rightarrow \varepsilon_{\mathcal{H}}([t_i, t_{i+1}]) \to 0$. Thus, every bound $\varepsilon_{\mathcal{H},\text{max}}$ is satisfiable. The while-loop in Alg. 3 terminates as we will either exceed the admissible error bound $\varepsilon_{\mathcal{S},\text{max},i}$ during the continuous reduction or exit the loop if the set parameters Φ_{set} have been reduced as much as possible. Finally, the main loop (lines 4-11) terminates as the time t monotonically increases (line 10), eventually reaching t_f.

To prove that the error stays below a user-defined bound, we split the error ε_{max} in $\varepsilon_{\text{max}} = \varepsilon_{\mathcal{H},\text{max}} + \varepsilon_{\mathcal{P},\text{max}} + \varepsilon_{\mathcal{S},\text{max}}$. All individual bounds are satisfiable: We have already shown the satisfiability of any $\varepsilon_{\mathcal{H},\text{max}}$ in the beginning of this proof. Prop. 2 shows that any limit $\varepsilon_{\mathcal{P},\text{max}}$ is satisfiable. Lastly, any bound $\varepsilon_{\mathcal{S},\text{max}}$ can be satisfied since we can choose to not over-approximate the set representation. □

We now present two algorithms performing the adaptive parameter tuning: As an overview, Alg. 2 adapts the parameters Δt_i and $\Phi_{\text{prop},i}$, while Alg. 3 adapts the parameters $\Phi_{\text{set},i}$ concurrently to the propagation of $\mathcal{P}^{\mathcal{R}}$.

We first present Alg. 2: As the main idea, every Φ_{prop} is checked for an iteratively decreasing Δt_i until the error bounds $\varepsilon_{\mathcal{H},\text{max}}$ and $\varepsilon_{\mathcal{P},\text{max},i}$ are satisfied. Initially, we increase the time step size by division with the factor μ (line 1). Furthermore, we reset $\Phi_{\text{prop},i}$ to its coarsest setting (line 1) and calculate the admissible error $\varepsilon_{\mathcal{P},\text{max},i}$ (line 2). In the loop (lines 3-12), we increment $\Phi_{\text{prop},i}$ (line 4) and calculate the errors $\varepsilon_{\mathcal{H}}([t_i, t_{i+1}])$ and $\varepsilon_{\mathcal{P}}([t_i, t_{i+1}])$ based on the error sets $\mathcal{H}^{\mathcal{R}}_{+}([t_i, t_{i+1}])$ and $\mathcal{P}^{\mathcal{R}}_{+}([t_i, t_{i+1}])$ (lines 10-11). These errors are then compared to their respective error bounds (line 12). If $\Phi_{\text{prop},i}$ reaches $\Phi_{\text{prop,max}}$ (line 5), we decrease Δt_i, reset $\Phi_{\text{prop},i}$ (line 6), and recalculate the admissible bound $\varepsilon_{\mathcal{P},\text{max},i}$ (line 7) before restarting the loop (line 8). After the loop is finished, the accumulated error $\varepsilon_{\mathcal{P}}([0, t_{i+1}])$ is computed (line 13).

Algorithm 2 Adapt values for $\Delta t, \Phi_{\text{prop}}$

```
Input: $\varepsilon_{\mathcal{H},\text{max}}, \varepsilon_{\mathcal{P},\text{max}}, \varepsilon_{\mathcal{P}}([0, t_i]), \Delta t_{i-1}, \mu, t_f$
Output: $\Delta t_i, \Phi_{\text{prop},i}, \varepsilon_{\mathcal{P}}([0, t_{i+1}])$
 1: $\Delta t_i \leftarrow \frac{\Delta t_{i-1}}{\mu}$, reset($\Phi_{\text{prop},i}$)
 2: calc. $\varepsilon_{\mathcal{P},\text{max},i}$ acc. to (10)
 3: do
 4:     incr($\Phi_{\text{prop},i}$)
 5:     if $\Phi_{\text{prop}} \geq \Phi_{\text{prop,max}}$
 6:         $\Delta t_i \leftarrow \Delta t_i \mu$, reset($\Phi_{\text{prop},i}$)
 7:         calc. $\varepsilon_{\mathcal{P},\text{max},i}$ acc. to (10)
 8:         continue
 9:     end if
10:     calc. $\mathcal{H}^{\mathcal{R}}_{+}([t_i, t_{i+1}]), \varepsilon_{\mathcal{H}}([t_i, t_{i+1}])$
11:     calc. $\mathcal{P}^{\mathcal{R}}_{+}([t_i, t_{i+1}]), \varepsilon_{\mathcal{P}}([t_i, t_{i+1}])$
12: while $\varepsilon_{\mathcal{H}}([t_i, t_{i+1}]) > \varepsilon_{\mathcal{H},\text{max}} \wedge \varepsilon_{\mathcal{P}}([t_i, t_{i+1}]) > \varepsilon_{\mathcal{P},\text{max},i}$

13: calc. $\varepsilon_{\mathcal{P}}([0, t_{i+1}])$ acc. to (9)
```

The adaptation of Φ_{set} is shown in Alg. 3: In order to decrease the computation time, the number of parameters describing $\mathcal{P}^{\mathcal{R}}$ is iteratively reduced from a high-precision representation towards lower precision. First, we calculate the admissible error $\varepsilon_{\mathcal{S},\text{max},i}$ (line 1) for the given Δt_i. The operator param(Φ_{set}) extracts the number of stored parameters $\Phi_{\text{set},i}$ from $\mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$ (line 2). Inside the loop (lines 3-7), we iteratively decrease $\Phi_{\text{set},i}$ (line 4) and over-approximate $\mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$ by reducing the number of stored parameters to $\Phi_{\text{set},i}$ (line 5). From the difference between the original set and the set over-approximated by reduction, we calculate the induced error $\varepsilon_{\mathcal{S}}([t_i, t_{i+1}])$ (line 6) and compare it to the admissible error (line 7). After the loop, the accumulated error for the set representation $\varepsilon_{\mathcal{S}}([0, t_{i+1}])$ is computed (line 8).

Algorithm 3 Adapt values for Φ_{set}

```
Input: $\varepsilon_{\mathcal{S},\text{max}}, \varepsilon_{\mathcal{S}}([0, t_i]), \mathcal{P}^{\mathcal{R}}([0, t_{i+1}]), \Delta t_i, t_f$
Output: $\mathcal{P}^{\mathcal{R}}([0, t_{i+1}]), \varepsilon_{\mathcal{S}}([0, t_{i+1}])$
 1: calc. $\varepsilon_{\mathcal{S},\text{max},i}$ acc. to (13)
 2: $\Phi_{\text{set},i} \leftarrow$ param($\mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$)
 3: do
 4:     decr($\Phi_{\text{set},i}$)
 5:     red($\mathcal{P}^{\mathcal{R}}([0, t_{i+1}]), \Phi_{\text{set},i}$)
 6:     calc. $\varepsilon_{\mathcal{S}}([t_i, t_{i+1}])$
 7: while $\varepsilon_{\mathcal{S}}([t_i, t_{i+1}]) < \varepsilon_{\mathcal{S},\text{max},i} \vee$ decr($\Phi_{\text{set}}$) $= \Phi_{\text{set}}$
 8: calc. $\varepsilon_{\mathcal{S}}([0, t_{i+1}])$ acc. to (12)
```

The presented framework for adaptive parameter tuning can be applied to any reachability algorithm. In the next section, we will provide an example implementation and prove the assumptions underlying Theorem 1.

IV. Implementation

In this section, we present an implementation of our framework and validate our assumptions using a concrete reachability algorithm and a concrete set representation.

A. Computation of Reachable Sets

We over-approximate the exponential matrix e^{At} by a finite number of Taylor terms $\eta \in \mathbb{N}^+$ and an interval matrix $\mathbf{E}$ enclosing the remainder [28, Prop. 2].

$$e^{A\Delta t} \in \sum_{k=0}^{\eta} \frac{(A\Delta t)^k}{k!} \oplus \mathbf{E}(\Delta t, \eta) \;, \qquad (14)$$

$$\mathbf{E}(\Delta t, \eta) = [-E_{\text{abs}}(\Delta t, \eta), E_{\text{abs}}(\Delta t, \eta)] \qquad (15)$$

$$\text{with} \quad E_{\text{abs}}(\Delta t, \eta) = \Big| \sum_{k=\eta+1}^{\infty} \frac{1}{k!}\big(|A|\Delta t\big)^k \Big| \,. \qquad (16)$$

To cover all trajectories in a time interval spanned by Δt, we introduce the terms $\mathbf{F}_x$ [28, Sec. 4] and $\mathbf{F}_u$ [25, Sec. 3.2.2]:

$$\mathbf{F}_x(\Delta t, \eta) = \bigoplus_{k=2}^{\eta} [(k^{\frac{-k}{k-1}} - k^{\frac{-1}{k-1}})\Delta t^k, 0] \frac{A^k}{k!} \oplus \mathbf{E}(\Delta t, \eta) \;, \qquad (17)$$

$$\mathbf{F}_u(\Delta t, \eta) = \bigoplus_{k=2}^{\eta+1} [(k^{\frac{-k}{k-1}} - k^{\frac{-1}{k-1}})\Delta t^k, 0] \frac{A^{k-1}}{k!} \oplus \mathbf{E}(\Delta t, \eta)\,\Delta t \,. \qquad (18)$$

The reachable sets for the homogeneous and inhomogeneous solution, generally defined in (3) and (4), are computed by [25, Sect. 3.2.1-3.2.2]

$$\mathcal{H}^{\mathcal{R}}\big([0, \Delta t]\big) = \underbrace{\mathtt{conv}\big(\mathcal{X}^0, e^{A\Delta t}\mathcal{X}^0\big)}_{=\mathcal{H}^{\mathcal{R}}_{=}([0,\Delta t])} \oplus \underbrace{\mathbf{F}_x(\Delta t, \eta)\,\mathcal{X}^0 \oplus \mathbf{F}_u(\Delta t, \eta)\, c_u}_{=\mathcal{H}^{\mathcal{R}}_{+}([0,\Delta t])} \;, \qquad (19)$$

$$\mathcal{P}^{\mathcal{R}}\big([0, \Delta t]\big) = \underbrace{\sum_{k=0}^{\eta} \Big(\frac{A^k \Delta t^{k+1}}{(k+1)!}\Big)\mathcal{U}}_{=\mathcal{P}^{\mathcal{R}}_{=}([0,\Delta t])} \oplus \underbrace{\mathbf{E}(\Delta t, \eta)\,\Delta t\,\mathcal{U}}_{=\mathcal{P}^{\mathcal{R}}_{+}([0,\Delta t])}, \qquad (20)$$

with c_u being the center of the input set $\mathcal{U}$. We include the term $\mathbf{F}_u(\Delta t, \eta)\, c_u$ in the homogeneous solution as it only covers the current time interval and therefore does not accumulate over time. In the next section, we will verify the applicability of (19) and (20) for our adaptive framework.

B. Verification of Assumptions

We now want to verify Assumption 1 and Prop. 2. To obtain the error $\varepsilon_{\mathcal{H}}\big([t_i, t_{i+1}]\big)$ of the homogeneous solution $\mathcal{H}^{\mathcal{R}}\big([t_i, t_{i+1}]\big)$, we multiply (19) by e^{At_i} as required in (2) and insert the result in (7), which yields

$$\varepsilon_{\mathcal{H}}\big([t_i, t_{i+1}]\big) = \mathtt{err}\big(e^{At_i}\,\mathbf{F}_x(\Delta t_i, \eta_i)\,\mathcal{X}^0 \oplus e^{At_i}\,\mathbf{F}_u(\Delta t_i, \eta_i)\, c_u\big) \,. \qquad (21)$$

Proposition 3: *The error* $\varepsilon_{\mathcal{H}}\big([t_i, t_{i+1}]\big)$ *in* (21) *converges to 0 for* $\Delta t_i \to 0$ *and therefore satisfies Assumption 1.*

Proof. Since the operation $\mathtt{err}\big(\mathcal{S}\big)$ returns the enclosing radius of the interval over-approximation of a set $\mathcal{S}$, it suffices to show that the volume of $\mathcal{S}$ converges to 0 for $\Delta t_i \to 0$. This is the case if the interval matrices $\mathbf{F}_x$ and $\mathbf{F}_u$ converge to $[0_n, 0_n]$: Using (18), we yield $\lim_{\Delta t \to 0} \mathbf{F}_u(\Delta t, \eta) = [0_n, 0_n]$. For $\mathbf{F}_x$, we have that $\lim_{\Delta t \to 0} \mathbf{E}(\Delta t, \eta) \overset{(15)}{=} [0_n, 0_n]$ since $\lim_{\Delta t \to 0} E_{\text{abs}}(0, \eta) \overset{(16)}{=} 0_n$. By plugging this in (17), we immediately see that $\mathbf{F}_x(\Delta t, \eta) = [0_n, 0_n]$ for $\Delta t \to 0$. □

To obtain the error $\varepsilon_{\mathcal{P}}\big([t_i, t_{i+1}]\big)$ of the inhomogeneous solution $\mathcal{P}^{\mathcal{R}}\big([t_i, t_{i+1}]\big)$, we multiply (20) by e^{At_i} as required in (2) and insert the result in (8), which gives us

$$\varepsilon_{\mathcal{P}}\big([t_i, t_{i+1}]\big) = \mathtt{err}\big(e^{At_i}\,\mathbf{E}(\Delta t_i, \eta_i)\,\Delta t_i\,\mathcal{U}\big) \,. \qquad (22)$$

Proposition 4: *The error* $\varepsilon_{\mathcal{P}}\big([t_i, t_{i+1}]\big)$ *in* (22) *converges to 0 for* $\Delta t_i \to 0$ *and therefore satisfies Assumption 1.*

Proof. Equivalently to the proof of Prop. 3, we show that the volume of the resulting set converges to 0 for $\Delta t_i \to 0$. Since Δt_i appears as a multiplicative factor, it holds that $\lim_{\Delta t_i \to 0} e^{At_i}\,\mathbf{E}(\Delta t_i, \eta_i)\,\Delta t_i\,\mathcal{U} = [0_n, 0_n]$. □

We introduce the following lemma for the subsequent derivations:

Lemma 1: *The size of* $\mathbf{E}(\Delta t, \eta)$ *decreases superlinearly with respect to* Δt*:*

$$\forall \varphi \in (0,1) : \mathbf{E}(\varphi\Delta t, \eta) \subseteq \varphi\,\mathbf{E}(\Delta t, \eta) \,.$$

Proof. Using $\mathbf{E}(\Delta t, \eta) \overset{(15)}{=} [-E_{\text{abs}}(\Delta t, \eta), E_{\text{abs}}(\Delta t, \eta)]$, it is sufficient to show that each entry of $E_{\text{abs}}(\Delta t, \eta)$ decreases superlinearly with respect to Δt:

$$\forall \varphi \in (0,1) : E_{\text{abs}}(\varphi\Delta t, \eta) \overset{(16)}{=} \Big| \sum_{k=\eta+1}^{\infty} \frac{1}{k!}\big(|A|\varphi\Delta t\big)^k \Big| \le \varphi \Big| \sum_{k=\eta+1}^{\infty} \frac{1}{k!}\big(|A|\Delta t\big)^k \Big| = \varphi\, E_{\text{abs}}(\Delta t, \eta) \,. \quad □$$

Theorem 2: *The error* $\varepsilon_{\mathcal{P}}\big([t_i, t_{i+1}]\big)$ *in* (22) *decreases superlinearly according to Def. 1:*

$$\forall \varphi \in (0,1) : \varepsilon_{\mathcal{P}}\big([t_i, t_i + \varphi\Delta t_i]\big) \le \varphi\,\varepsilon_{\mathcal{P}}\big([t_i, t_i + \Delta t_i]\big) \,.$$

Proof: Since $\mathtt{err}\big(\cdot\big)$ is defined by the enclosing radius r according to Prop. 1, it suffices to show that the enclosing radius decreases superlinearly with respect to Δt_i:

$$\forall \varphi \in (0,1)\colon r(\mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i{+}\varphi\Delta t_i]\big)) \le \varphi\, r(\mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i{+}\Delta t_i]\big)).$$

This condition is satisfied if

$$\forall \varphi \in (0,1)\colon \mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i + \varphi\Delta t_i]\big) \subseteq \varphi\,\mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i + \Delta t_i]\big) \,.$$

Using the definition of $\mathcal{P}^{\mathcal{R}}_{+}\big([0, \Delta t]\big)$ in (20) and Lemma 1 it holds that

$$\mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i + \varphi\Delta t_i]\big) \overset{(20)}{=} (\varphi\,\Delta t_i)\, e^{At_i} \underbrace{\mathbf{E}(\varphi\Delta t_i, \eta_i)}_{\overset{\text{Lemma 1}}{\subseteq} \varphi\,\mathbf{E}(\Delta t_i, \eta_i)} \mathcal{U} \subseteq \varphi\,\underbrace{\big(\varphi\Delta t_i\, e^{At_i}\,\mathbf{E}(\Delta t_i, \eta_i)\mathcal{U}\big)}_{\overset{(20)}{=} \varphi\,\mathcal{P}^{\mathcal{R}}_{+}([t_i, t_i+\Delta t_i])} \overset{\varphi\in(0,1)}{\subseteq} \varphi\,\mathcal{P}^{\mathcal{R}}_{+}\big([t_i, t_i + \Delta t_i]\big). \;□$$

The cut-off value η_{max} can be automatically obtained according to [29] to truncate the power series in (14). If η_{max} does not satisfy the current error bounds, we proceed to smaller values for Δt which are guaranteed to eventually satisfy the error by Theorem 1.

Following the above derivations, the presented implementation satisfies Theorem 1. Hence, it can be used to perform reachability analysis for LTI systems while remaining below a user-defined error bound.

C. Set Representation

It remains to choose a set representation. The work in [30] shows that zonotopes are optimal and that one should add support functions if the initial set is not a zonotope. Since we only use zonotopes as initial sets, we only use them:

Definition 2: *(Zonotopes) Given a center $c \in \mathbb{R}^n$ and an arbitrary number $\gamma \in \mathbb{N}$ of generator vectors $g^{(1)}, ..., g^{(\gamma)} \in \mathbb{R}^n$, a zonotope is defined as [27, Def. 1]*

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n \,\middle|\, x = c + \sum_{i=1}^{\gamma} \beta_i \cdot g^{(i)},\ -1 \leq \beta_i \leq 1 \right\} .$$

The order of a zonotope is $\rho = \frac{\gamma}{n}$. □

Following Def. 2, we have that $\Phi_{\text{set}} = \rho$. We iteratively decrease the zonotope order by decrementing the total number of generators describing $\mathcal{P}^{\mathcal{R}}([0, t_{i+1}])$ as shown in [31], which also provides us with $\varepsilon_{\mathcal{S}}([t_i, t_{i+1}])$. Therefore, we define the operator $\texttt{decr}(\Phi_{\text{set}}) : n\gamma \leftarrow n(\gamma - 1)$. In the next section, we apply the presented implementation.

V. Numerical Examples

We have implemented our approach in *MATLAB*. To extensively test our approach, we perform several investigations: First, we measure the computational overhead caused by our parameter tuning. Next, Alg. 1 is evaluated on benchmark systems and compared to manually-tuned algorithm parameters. Finally, we compare Alg. 1 with a genetic algorithm searching for algorithm parameters. Since the genetic algorithm requires a lot of memory, we used an Intel Xeon Gold 6136 3.00GHz processor and 768GB of DDR4 2666/3273MHz memory, all other computations have been performed using an Intel i3 processor with 8GB memory. For the genetic algorithm and the overhead measurement, we used the subsequently introduced randomly-generated systems because the results might vary depending on the investigated system.

a) Random Generation of Systems: We randomly picked complex conjugate pairs of eigenvalues from a uniform distribution over the range of $[-1, 1]$ for the real part and $[-\mathrm{i}, \mathrm{i}]$ for the imaginary part without loss of generality, since the characteristics of the solution only depends on the ratio of real and imaginary values. The state-space form for these eigenvalues is computed and subsequently rotated by a random orthogonal matrix of increasing sparsity for higher dimensions. Furthermore, the initial set $\mathcal{X}^0$ and the input set $\mathcal{U}$ are given by hypercubes centered at $(10, ..., 10)^T$ with edge length 0.5, and $(1, ..., 1)^T$ with edge length 0.1, respectively. Lastly, we set $\varepsilon_{\text{max}} = 0.05$ and $t_f = 3\text{s}$.

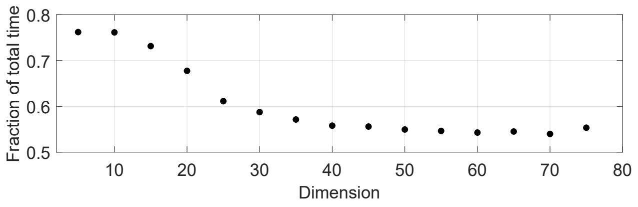


Fig. 1: Time consumption by parameter tuning in relation to total execution time using 50 randomly-generated systems per dimension.

b) Computational Overhead: First, we investigate the computational overhead caused by the continuous adaptation of all algorithm parameters. For this purpose, we measured the time Alg. 1 spends on the adaptation and the set propagation over 50 randomly-generated systems of dimensions 5 to 75. Fig. 1 shows the fraction of the total time spent on the parameter tuning: The overhead remains manageable even for high-dimensional systems, settling at about the same time as used for the set propagation. This is achieved by the computational efficiency of Prop. 1. The overhead is more than compensated by the adaptive adjustment of the algorithm parameters as discussed at the end of this section. Also, the common practice of trial and error requires tuning times that exceed computation times by several factors.

c) ARCH benchmarks: Next, we have applied our algorithm to the 48-dimensional Building (BLD) benchmark and the 273-dimensional International Space Station (ISS) benchmark, both from the ARCH competition [32]. These systems are manually tuned for the competition by executing many runs to optimize the algorithm parameters with respect to the computation time while still satisfying all specifications. For a fair comparison, we set ε_{max} to the highest possible value that still verifies the given specification. We compare our results to the tool *CORA* [8] since it is also implemented in *MATLAB*.

Table I shows a comparison between Alg. 1 and *CORA* in terms of the computation time, the number of steps, and the minimum and maximum values for Δt. Alg. 1 adapts the values of the algorithm parameters depending on the current system behavior by enlarging the time step size in regions where this only moderately increases the over-approximation. The resulting range can be observed by the large differences between Δt_{min} and Δt_{max}. Consequently, the total number of steps decreases making less computations necessary than in the case of fixed algorithm parameters as used by *CORA* and shown in Table I. The range for Δt in both building benchmarks by *CORA* is explained by the switching between two manually-tuned time steps.

Fig. 2 shows the reachable sets for the benchmark ISSF01 using different values of ε_{max}: The smaller the defined error bound, the tighter are the reachable sets. We also recognize the linear increase of the admissible error bound over time as the computed sets differ more towards the end of the time horizon. Fig. 3 shows the evolution of all algorithm

TABLE I: Results of ARCH benchmarks for Alg. 1 and *CORA*.

Benchmark	**Alg. 1**				**CORA**		
	$\varepsilon_{\max}$	Time	Steps	$[\Delta t_{\min}, \Delta t_{\max}]$	Time	Steps	$[\Delta t_{\min}, \Delta t_{\max}]$
BLDC01	$2 \cdot 10^{-3}$	4.0s	839	$[0.0081, 0.0448]$	5.5s	2400	$[0.0020, 0.0100]$
BLDF01	$6 \cdot 10^{-3}$	5.6s	818	$[0.0081, 0.0597]$	6.0s	2400	$[0.0020, 0.0100]$
ISSC01	5.6	21.7s	189	$[0.0704, 0.1371]$	22.8s	1000	$[0.0200, 0.0200]$
ISSF01	$2 \cdot 10^{-3}$	295s	1216	$[0.0059, 0.0395]$	849s	2000	$[0.0100, 0.0100]$

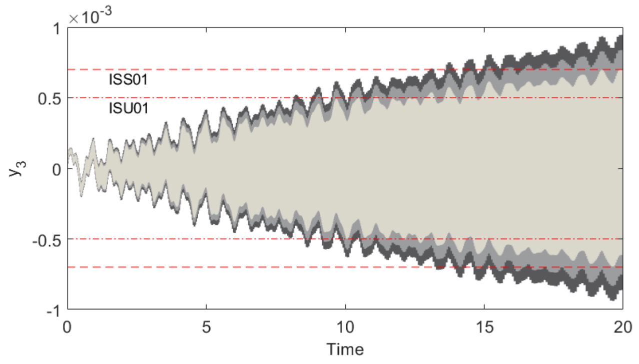


Fig. 2: Benchmark ISSF01 with the specifications ISS01, ISU01, and the reachable sets $\mathcal{R}([0,T])$ in dark gray ($\varepsilon_{\max} = 20 \cdot 10^{-3}$), light gray ($\varepsilon_{\max} = 10 \cdot 10^{-3}$), and ivory ($\varepsilon_{\max} = 2 \cdot 10^{-3}$).

parameters corresponding to the different values of $\varepsilon_{\max}$: A higher value for $\varepsilon_{\max}$ yields a larger initial Δt and a smaller total number of steps. Note that the switching between previously-computed values of Δt does not add any computations since the sets are read from memory once they are computed. The number of Taylor terms η is chosen jointly with Δt and increases towards the end of the time horizon to facilitate larger time step sizes. We also observe that the zonotope order ρ reaches a higher maximum for smaller values of $\varepsilon_{\max}$ since in that case we cannot reduce as much as for larger $\varepsilon_{\max}$.

d) Genetic Algorithm Comparison: Finally, we want to compare our approach to a genetic algorithm searching for Δt, η, and ρ. To this end, we use the *MATLAB* built-in genetic algorithm function. While the parameters η and ρ are fixed, we model the time step size by a polynomial up to order 2: $\Delta t(t) = a + bt + ct^2$. We restrict these parameters by the ranges $\eta \in [1, 10]$, $\rho \in [2, 1000]$, $a \in [0.0003, 0.3], b \in [-0.1, 0.1], c \in [-0.033, 0.033]$. The chosen bounds for a, b, and c prevent Δt from too drastic growth or shrinkage, thereby focussing on suitable curves of Δt. Higher orders did not provide any benefits.

In order to establish a level playing field, we terminate once the obtained reachable set is within the box enclosure of the reachable set of the adaptive algorithm enlarged by 10%. For computational efficiency interval over-approximations were used for this comparison. The cost function is chosen as the maximum distance to the enlarged adaptive reachable set over all dimensions.

The parameters specific to the genetic algorithm have been set as follows: We enable an infinite number of generations with a maximum of 3 stall generations. We aim to speed up the convergence by setting only 10 members per generation as the evaluation of a single member is costly in higher dimensions. For the members of the next generation, we use a standard crossover fraction of 0.75 and set the elite count to 1, carrying the best solution over to the next generation.

We applied Alg. 1 and the genetic algorithm on 50 randomly-generated systems per dimension. Table II compares the average computation time over varying dimensions of Alg. 1 to the time the genetic algorithms takes until convergence. The results show that Alg. 1 outspeeds all genetic algorithms. Since Alg. 1 tunes the algorithm parameters during runtime, we only need a single iteration for the computation of the reachable set. Contrary, the genetic algorithms run over many generations repeatedly computing the reachable set while iteratively improving the solution by means of the cost function. This process is far more time-consuming than the overhead caused by the adaptive parameter tuning. The genetic algorithm using the constant polynomial for Δt is faster than the higher-order polynomials as they re-compute auxiliary reachable sets due to the non-constant time step size.

TABLE II: Computation time for Alg. 1 and the genetic algorithm (GA) averaged over 50 randomly-generated systems per dimension.

	Dimension						
	5	10	15	20	25	30	40
Alg. 1	0.14s	0.22s	0.40s	0.58s	1.0s	1.9s	6.7s
GA (order: 0)	1.6s	3.4s	7.0s	10s	20s	30s	50s
GA (order: 1)	4.1s	9.4s	13s	21s	28s	40s	70s
GA (order: 2)	5.1s	10s	14s	21s	42s	58s	97s

e) Discussion: Our framework can also be applied to other computations of reachable sets and other set representations. In order to guarantee convergence and termination for all $\varepsilon_{\max} \in \mathbb{R}$, Theorem 1 has to hold for the applied error terms, similarly as shown in Sec. IV-B for the presented implementation: A tool developer has to modify $\Phi_{\text{prop,max}}$ and Φ_{set} which are, e.g., the number of template directions when using template polyhedra. A corresponding error term for the set representation has to be defined.

The choice of $\texttt{err}(\cdot)$ in (5) does not add much over-approximation as can be observed from comparing the ranges for the time step size $[\Delta t_{\min}, \Delta t_{\max}]$ to the manually-tuned Δt by *CORA* in Table I, where we see that Alg. 1 chooses a similar time step size, yielding comparable results both in terms of the tightness and the computational efficiency.

The only remaining parameter for the practitioner to set is the error $\varepsilon_{\max}$. As shown in Fig. 2, increasing the value of $\varepsilon_{\max}$ results in a more over-approximative reachable set and vice versa. Thus, the setting of $\varepsilon_{\max}$ is intuitive and can easily be adjusted for any system.

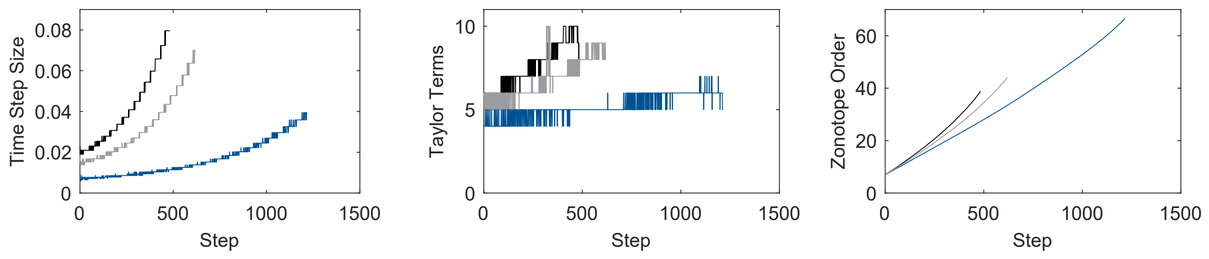


Fig. 3: Benchmark ISSF01: $\Delta t, \eta, \rho$ for different $\varepsilon_{\max}$: black ($\varepsilon_{\max} = 20 \cdot 10^{-3}$), gray ($\varepsilon_{\max} = 10 \cdot 10^{-3}$), and blue ($\varepsilon_{\max} = 2 \cdot 10^{-3}$).

VI. Conclusion

In this paper, we presented a novel generic framework to automatically tune all algorithm parameters which is a major problem of present reachability algorithms. The presented algorithm enables a fully-automated computation of the reachable set whose error is below a user-defined error. Previous work only considered the tuning of time parameters. An example implementation has shown to outperform manually-tuned algorithm parameters on benchmarks and provides better results than genetic algorithms searching for algorithm parameters on randomly-generated systems of varying dimensions. The extension of the presented framework to nonlinear systems will be considered in the future.

References

[1] G. Lafferriere and et al., "Symbolic reachability computation for families of linear vector fields," *Journal of Symbolic Computation*, vol. 32, no. 3, pp. 231–253, 2001.

[2] M. Althoff and et al., "Reachability analysis of linear systems with uncertain parameters and inputs," in *Proc. of the 46th IEEE Conference on Decision and Control*, pp. 726–732, 2007.

[3] X. Chen, *Reachability Analysis of Non-Linear Hybrid Systems using Taylor Models*. PhD thesis, Fachgruppe Informatik, RWTH Aachen University, 2015.

[4] A. Gurung and et al., "Parallel reachability analysis of hybrid systems in XSpeed," *International Journal on Software Tools for Technology Transfer*, vol. 21, no. 4, pp. 401–423, 2019.

[5] A. Girard and et al., "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Hybrid Systems: Computation and Control*, LNCS 3927, pp. 257–271, Springer, 2006.

[6] S. Bogomolov and et al., "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*, pp. 41–50, 2018.

[7] G. Frehse and et al., "SpaceEx: Scalable verification of hybrid systems," in *Proc. of the 23rd International Conference on Computer Aided Verification*, LNCS 6806, pp. 379–395, Springer, 2011.

[8] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pp. 120–151, 2015.

[9] X. Chen and et al., "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification*, pp. 258–263, Springer, 2013.

[10] R. Ray and et al., "XSpeed: Accelerating reachability analysis on multi-core processors," in *Haifa Verification Conference*, pp. 3–18, Springer, 2015.

[11] S. Bogomolov and et al., "JuliaReach: a toolbox for set-based reachability," in *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*, pp. 39–44, ACM, 2019.

[12] S. Bak and P. S. Duggirala, "Simulation-equivalent reachability of large linear systems with inputs," in *Proc. of 29th International Conference on Computer Aided Verification*, pp. 401–420, 2017.

[13] P. S. Duggirala and M. Viswanathan, "Parsimonious, simulation based verification of linear systems," in *Proc. of 28th International Conference on Computer Aided Verification*, pp. 477–494, 2016.

[14] S. Bak and P. S. Duggirala, "HyLAA: A tool for computing simulation-equivalent reachability for linear systems," in *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*, pp. 173–178, 2017.

[15] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Hybrid Systems: Computation and Control*, LNCS 1790, pp. 202–214, Springer, 2000.

[16] E. Asarin and et al., "Approximate reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control*, pp. 20–31, Springer, 2000.

[17] A. Girard and C. Le Guernic, "Efficient reachability analysis for linear systems using support functions," in *Proc. of the 17th IFAC World Congress*, pp. 8966–8971, 2008.

[18] J. K. Scott and et al., "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.

[19] L. Lapidus and J. H. Seinfeld, *Numerical solution of ordinary differential equations*. Academic press, 1971.

[20] U. M. Ascher and et al., *Numerical solution of boundary value problems for ordinary differential equations*. SIAM, 1994.

[21] M. Kerbl, "Stepsize strategies for inclusion algorithms for ODE's," *Computer Arithmetic, Scientific Computation, and Mathematical Modelling, IMACS Annals on Computing and Appl. Math*, vol. 12, pp. 437–452, 1991.

[22] W. Rufeger and E. Adams, "A step size control for Lohner's enclosure algorithm for ordinary differential equations with initial conditions," in *Mathematics in Science and Engineering*, vol. 189, pp. 283–299, Elsevier, 1993.

[23] N. S. Nedialkov, *Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation.* Dissertation, University of Toronto, 2000.

[24] P. Prabhakar and M. Viswanathan, "A dynamic algorithm for approximate flow computations," in *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control*, pp. 133–142, ACM, 2011.

[25] M. Althoff, *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. Dissertation, Technische Universität München, 2010.

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

[27] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, LNCS 3414, pp. 291–305, Springer, 2005.

[28] M. Althoff and et al., "Reachable set computation for uncertain time-varying linear systems," in *Hybrid Systems: Computation and Control*, pp. 93–102, 2011.

[29] T. A. Bickart, "Matrix exponential: Approximation by truncated power series," *Proceedings of the IEEE*, vol. 56, no. 5, pp. 872–873, 1968.

[30] M. Althoff and G. Frehse, "Combining zonotopes and support functions for efficient reachability analysis of linear systems," in *Proc. of the 55th IEEE Conference on Decision and Control*, pp. 7439–7446, 2016.

[31] A.-K. Kopetzki and et al., "Methods for order reduction of zonotopes," in *Proc. of the 56th IEEE Conference on Decision and Control*, pp. 5626–5633, 2017.

[32] M. Althoff and et al., "ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics," in *Proc. of the 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, pp. 14–40, 2019.

A.2 Fully Automated Verification of Linear Systems Using Inner- and Outer-Approximations of Reachable Sets

Summary The work in Appendix A.1 suffers from two major shortcomings: First, the computed approximation error is non-exhaustive, and second, the admissible error bound still has to be manually tuned for successful verification.

In this work, we address these issues as follows: We rigorously derive all induced approximation errors in the reachability algorithm for linear systems of the form (2.2). Consequently, we obtain a bound on the Hausdorff distance between the exact reachable set and the computed outer approximation for any algorithm parameter setting. Next, we prove that the algorithm parameters can be adapted such that any non-zero admissible error bound is fulfilled over the whole time horizon. The outer approximation and the error bound are then used to compute an inner approximation of the exact reachable set. By automatically refining the admissible error bound, we devise a fully automated verification algorithm that is capable of verifying or falsifying any safety specification that does not require the exact reachable set, thereby solving the verification task in Problem 1.

A comparison on challenging benchmarks shows that the proposed verification algorithm can verify or falsify safety similarly fast as state-of-the-art approaches that still rely on manual tuning of algorithm parameters. However, our verification algorithm only requires the model and the model parameters to return a conclusive result on safety within a single run. This automated procedure represents a substantial improvement in the verification of safety specifications for linear systems and constitutes a leap toward industrial application.

Author contributions M.W. derived the approximation errors, integrated the adaptive parameter tuning into the reachability algorithm, implemented the reachability algorithm, conducted parts of the numerical evaluation, and wrote most of the manuscript. N.K. described and implemented the verification algorithm and conducted parts of the numerical evaluation. S.B. and M.A. provided feedback for improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Fully Automated Verification of Linear Systems Using Inner- and Outer-Approximations of Reachable Sets

Mark Wetzlinger, Niklas Kochdumper, Stanley Bak, and Matthias Althoff

***Abstract*—Reachability analysis is a formal method to guarantee safety of dynamical systems under the influence of uncertainties. A substantial bottleneck of all reachability algorithms is the necessity to adequately tune specific algorithm parameters, such as the time step size, which requires expert knowledge. In this work, we solve this issue with a fully automated reachability algorithm that tunes all algorithm parameters internally such that the reachable set enclosure respects a user-defined approximation error bound in terms of the Hausdorff distance to the exact reachable set. Moreover, this bound can be used to extract an inner-approximation of the reachable set from the outer-approximation using the Minkowski difference. Finally, we propose a novel verification algorithm that automatically refines the accuracy of the outer-approximation and inner-approximation until specifications given by time-varying safe and unsafe sets can be verified or falsified. The numerical evaluation demonstrates that our verification algorithm successfully verifies or falsifies benchmarks from different domains without requiring manual tuning.**

***Index Terms*—Formal verification, reachability analysis, linear systems, set-based computing.**

I. Introduction

DEPLOYING cyber-physical systems in safety-critical environments requires formal verification techniques to ensure correctness with respect to the desired functionality, as failures can lead to severe economic or ecological consequences and loss of human life. One of the main techniques to provide safety guarantees is reachability analysis, which predicts all possible future system behaviors under uncertainty in the initial state and input. Reachability analysis has already been successfully applied in a wide variety of applications, such as analog/mixed-signal circuits [1], power systems [2], robotics [3], system biology [4], aerospace applications [5], and autonomous driving [6]. The most common verification tasks for these applications are reach-avoid problems, where one aims to prove that the system reaches a goal set while avoiding unsafe sets. A substantial bottleneck of all verification algorithms is the manual tuning of certain algorithm parameters, which requires expert knowledge. To overcome this limitation, we present the first fully automated verification algorithm for linear time-invariant systems.

A. State of the Art

The exact reachable set can only be computed in rare special cases [7]. Therefore, one usually computes tight outer-approximations or inner-approximations of the reachable set instead, which can be used to either prove or disprove safety, respectively. While there exist many different approaches, e.g., stochastic techniques [8] or data-driven/learning methods [9], [10], the following review focuses on model-based reachability analysis of linear systems.

Most work is concerned with computing outer-approximations, where the most prominent approaches for linear systems are based on set propagation [11]–[13]. These methods evaluate the analytical solution for linear systems in a set-based manner and iteratively propagate the resulting reachable sets forward in time. One can propagate the sets from the previous step or the initial set. The latter method avoids the so-called wrapping effect [14], i.e., the amplification of outer-approximation errors over subsequent steps, but deals less efficiently with time-varying inputs. An alternative to set propagation is to compute the reachable set using widened trajectories from simulation runs [15], [16]. Moreover, special techniques have been developed recently to facilitate the analysis of high-dimensional systems. One strategy is to decompose the system into several decoupled/weakly-coupled blocks to reduce the computational effort [17], [18], while another group computes the reachable set in a lower-dimensional Krylov subspace that captures the dominant dynamical behavior [19], [20]. Inner-approximation algorithms have been researched for systems with piecewise-constant inputs based on set propagation [21], piecewise-affine systems based on linear matrix inequalities [22], and time-varying linear systems based on ellipsoidal inner-approximations to parametric integrals [23]. Other approaches compute inner-approximations by extraction from outer-approximations [24] or for projected dimensions [25]—despite being designed for nonlinear dynamics, these

This paragraph of the first footnote will contain the date on which you submitted your paper for review. This work was supported by the European Research Council (ERC) project justITSELF under grant agreement No 817629, by the German Research Foundation (DFG) project ConVeY under grant number GRK 2428, and by the Air Force Office of Scientific Research and the Office of Naval Research under award number FA9550-19-1-0288, FA9550-21-1-0121, FA9550-22-1-0450 and N00014-22-1-2156.

Mark Wetzlinger and Matthias Althoff are with the Department of Computer Science, Technical University of Munich, 85748 Garching, Germany (e-mail: {m.wetzlinger, althoff}@tum.de). Niklas Kochdumper and Stanley Bak are with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA (e-mail: {niklas.kochdumper, stanley.bak}@stonybrook.edu.

works can still compete with the aforementioned specialized approaches for linear systems due to their recency.

Concerning the set representations for reachability analysis, early approaches used polyhedra or template polyhedra [26], which are limited to low-dimensional systems due to the exponential increase in the representation size; ellipsoids [23] were also used but result in a conservative approximation as they are not closed under Minkowski sum. More recent approaches use support functions [14], zonotopes [11], or their combination [27], for which all relevant set operations can be computed accurately and efficiently. A related representation is star sets [28], where constraints are imposed on a linear combination of base vectors.

Some approaches explicitly address the requirement of tightening the computed outer-approximation for successful verification. Many of them are based on counterexample-guided abstraction refinement (CEGAR), where either the model [1], [29] or the set representation [30], [31] is refined. A more recent approach [32] utilizes the relation between all algorithm parameters and the tightness of the reachable sets, proposing individual parameter refinement in fixed discrete steps to yield tighter results. Another method [33] refines the tightness of the reachable set enclosure as much as the real-time constraints allow for re-computation in order to choose between a verified but conservative controller and an unverified counterpart with better performance.

Common reachability tools for linear systems are *CORA* [34], *Flow** [35], *HyDRA* [36], *HyLAA* [37], *JuliaReach* [38], *SpaceEx* [13], and *XSpeed* [39]. These tools still require manual tuning of algorithm parameters to obtain tight approximations. Since this requires expert knowledge about the underlying algorithms, the usage of reachability analysis is currently mainly limited to academia. Another issue is that the unknown distance between the computed outer-approximation and the exact reachable set, which may result in so-called spurious counterexamples. Recently, first steps towards automated parameter tuning have been taken: A rather brute-force method [40] proposes to recompute the reachable set from scratch, where the parameter values are refined using fixed scaling factors after each run. However, recomputation is a computationally demanding procedure and does not exploit information about the specific system dynamics. Another work [41] tunes the time step size by approximating the flow below a user-defined error bound but is limited to affine systems. The most sophisticated approach [13], [42] tunes the time step size by iterative refinement to eventually satisfy a user-defined error bound between the exact reachable set and the computed outer-approximation. Since this error bound is limited to manually selected directions, the obtained information may differ significantly depending on their choice. In conclusion, there does not yet exist a fully automated parameter tuning algorithm for linear systems that satisfies an error bound in terms of the Haussdorf distance to the exact reachable set.

B. Overview

This work is structured as follows: After introducing some preliminaries in Sec. II, we provide a mathematical formulation of the problem statement in Sec. III. The main body (Secs. IV-VI) builds upon our previous results [43], where we tuned the algorithm parameters based on non-rigorous approximation error bounds using a naive tuning strategy. Neither inner-approximations nor automated verification/falsification were considered in [43]. In detail, this article contributes the following novelties:

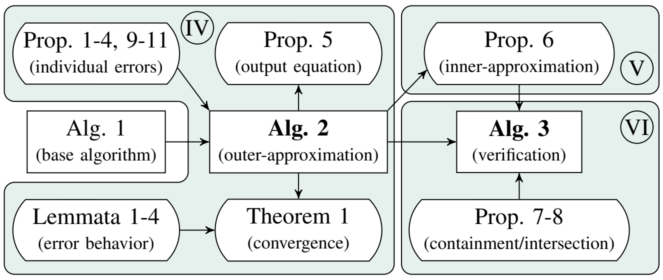


Fig. 1. Overview of theoretical contributions in Secs. IV-VI.

- We derive a rigorous approximation error for the reachable set, based on which we provide an *automated reachability algorithm* (Alg. 2) that adaptively tunes all algorithm parameters so that any desired error bound in terms of the Hausdorff distance between the exact reachable set and the computed outer-approximation is respected at all times (see Sec. IV).
- We show how to efficiently extract an *inner-approximation* from the previously computed outer-approximation (see Sec. V).
- We introduce an *automated verifier* (Alg. 3), which iteratively refines the accuracy of the outer- and inner-approximations until the specifications given by time-varying safe/unsafe sets can be either proven or disproven (see Sec. VI).

Fig. 1 depicts the relation of individual contributions in Secs. IV-VI. Finally, we demonstrate the performance of the proposed algorithms on a variety of numerical examples in Sec. VII and discuss directions for future work in Sec. VIII.

II. Preliminaries

A. Notation

Scalars and vectors are denoted by lowercase letters, matrices are denoted by uppercase letters. Given a vector $v \in \mathbb{R}^n$, $v_{(i)}$ represents the i-th entry and $\left\| v \right\|_p$ its p-norm. Similarly, for a matrix $M \in \mathbb{R}^{m \times n}$, $M_{(i,\cdot)}$ refers to the i-th row and $M_{(\cdot,j)}$ to the j-th column. The identity matrix of dimension n is denoted by I_n, the concatenation of two matrices M_1, M_2 by $[M_1 \; M_2]$, and we use $\mathbf{0}$ and $\mathbf{1}$ to represent vectors and matrices of proper dimension containing only zeros or ones, respectively. The operation $\operatorname{diag}(v)$ returns a square matrix with the vector v on its diagonal. Exact sets are denoted by standard calligraphic letters $\mathcal{S}$, outer-approximations by $\widehat{\mathcal{S}}$, and inner-approximations by $\check{\mathcal{S}}$. The empty set is represented by $\emptyset$. Moreover, we write v for the set $\{v\}$ consisting only of the point v. We refer to the radius of the smallest hypersphere centered at the origin and enclosing a set $\mathcal{S}$ by $\operatorname{rad}(\mathcal{S})$. The operation $\operatorname{box}(\mathcal{S})$ denotes the tightest axis-aligned interval outer-approximation of $\mathcal{S}$. Interval matrices are denoted by bold calligraphic letters: $\boldsymbol{\mathcal{M}} = [\underline{M}, \overline{M}] = \{M \in \mathbb{R}^{m \times n} \mid \underline{M} \leq$

$\underline{M} \leq \overline{M}\}$, where the inequality is evaluated element-wise. Intervals are a special case of interval matrices, where the lower and upper bounds are vectors. Additionally, we define an n-dimensional hyperball centered at the origin by $\mathcal{B}_\varepsilon = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq \varepsilon\} \subset \mathbb{R}^n$ with respect to the Euclidean norm. The floor operation $\lfloor x \rfloor$ returns the next smaller integer for a scalar x. For clarify, arguments of functions are sometimes omitted. Finally, we use $f(t) \sim \mathcal{O}(t^a)$ to represent that the function $f(t)$ approaches its limit value (0 or infinity in this work) as fast or faster than t^a.

B. Definitions

All sets are assumed to be compact, convex, and bounded. In this work, we represent outer-approximations of reachable sets with zonotopes [11, Def. 1]:

Definition 1 (Zonotope): *Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n\times\gamma}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is*

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

The zonotope order is defined as $\rho := \frac{\gamma}{n}$ and we use the shorthand $\mathcal{Z} = \langle c, G\rangle_Z$.

Moreover, we represent inner-approximations of reachable sets with constrained zonotopes [44, Def. 3]:

Definition 2 (Constrained zonotope): *Given a vector $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n\times\gamma}$, a constraint matrix $A \in \mathbb{R}^{h\times\gamma}$, and a constraint offset $b \in \mathbb{R}^h$, a constrained zonotope $\mathcal{CZ} \subset \mathbb{R}^n$ is*

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

We use the shorthand $\mathcal{CZ} = \langle c, G, A, b\rangle_{CZ}$.

Finally, unsafe sets and safe sets are represented by polytopes [45, Sec. 1.1]:

Definition 3 (Polytope): *Given a constraint matrix $C \in \mathbb{R}^{a\times n}$ and a constraint offset $d \in \mathbb{R}^a$, the halfspace representation of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is*

$$\mathcal{P} := \big\{ x \in \mathbb{R}^n \;\big|\; Cx \leq d \big\}.$$

Equivalently, one can use the vertex representation

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

where $\{v_1, \dots, v_s\} \in \mathbb{R}^n$ are the polytope vertices. We use the shorthands $\mathcal{P} = \langle C, d\rangle_H$ and $\mathcal{P} = \langle [v_1 \dots v_s]\rangle_V$.

Given the sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n, \mathcal{S}_3 \subset \mathbb{R}^m$ and a matrix $M \in \mathbb{R}^{w\times n}$, we require the set operations linear map $M\mathcal{S}_1$, Cartesian product $\mathcal{S}_1 \times \mathcal{S}_3$, Minkowski sum $\mathcal{S}_1 \oplus \mathcal{S}_2$, Minkowski difference $\mathcal{S}_1 \ominus \mathcal{S}_2$, and linear combination $\mathrm{comb}(\mathcal{S}_1, \mathcal{S}_2)$, which are defined as

$$M\mathcal{S}_1 = \{Ms \mid s \in \mathcal{S}_1\}, \tag{1}$$

$$\mathcal{S}_1 \times \mathcal{S}_3 = \{[s_1^\top\ s_3^\top]^\top \mid s_1 \in \mathcal{S}_1, s_3 \in \mathcal{S}_3\}, \tag{2}$$

$$\mathcal{S}_1 \oplus \mathcal{S}_2 = \{s_1 + s_2 \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}, \tag{3}$$

$$\mathcal{S}_1 \ominus \mathcal{S}_2 = \{s \mid s \oplus \mathcal{S}_2 \subseteq \mathcal{S}_1\}, \tag{4}$$

$$\begin{aligned}\mathrm{comb}(\mathcal{S}_1, \mathcal{S}_2) = \big\{ &\lambda s_1 + (1-\lambda)s_2 \mid \\ &s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2, \lambda \in [0,1] \big\}.\end{aligned} \tag{5}$$

For zonotopes $\mathcal{Z}_1 = \langle c_1, G_1\rangle_Z, \mathcal{Z}_2 = \langle c_2, G_2\rangle_Z \subset \mathbb{R}^n$, linear map and Minkowski sum can be computed as [12, Eq. (2.1)]

$$M\mathcal{Z}_1 = \langle Mc_1, MG_1\rangle_Z, \tag{6}$$

$$\mathcal{Z}_1 \oplus \mathcal{Z}_2 = \langle c_1 + c_2, [G_1\ G_2]\rangle_Z. \tag{7}$$

Since the convex hull represents an enclosure of the linear combination, we furthermore obtain [12, Eq. (2.2)]

$$\begin{aligned}\mathrm{comb}(\mathcal{Z}_1, \mathcal{Z}_2) \subseteq \big\langle 0.5(c_1 + c_2),\ \big[&0.5(c_1 - c_2) \\ 0.5(G_1 + G_2^{(1)})\ \ &0.5(G_1 - G_2^{(1)})\ \ G_2^{(2)}\big]\big\rangle_Z\end{aligned} \tag{8}$$

with

$$G_2^{(1)} = [G_{2(\cdot,1)}\ \dots\ G_{2(\cdot,\gamma_1)}],\ G_2^{(2)} = [G_{2(\cdot,\gamma_1+1)}\ \dots\ G_{2(\cdot,\gamma_2)}],$$

where we assume without loss of generality that $\mathcal{Z}_2$ has more generators than $\mathcal{Z}_1$ so that we can write the formula in a compact form. Later on in Sec. V, we show that the Minkowski difference of two zonotopes can be represented as a constrained zonotope. The multiplication $\boldsymbol{\mathcal{I}}\,\mathcal{Z}$ of an interval matrix $\boldsymbol{\mathcal{I}}$ with a zonotope $\mathcal{Z}$ can be outer-approximated as specified in [46, Thm. 4], and the operation $\mathrm{box}(\mathcal{Z})$ returns the axis-aligned box outer-approximation according to [12, Prop. 2.2]. The representation size of a zonotope can be decreased with zonotope order reduction. While our reachability algorithm is compatible with all common reduction techniques [47], we focus on Girard's method [11, Sec. 3.4] for simplicity:

Definition 4 (Zonotope order reduction): *Given a zonotope $\mathcal{Z} = \langle c, G\rangle_Z \subset \mathbb{R}^n$ and a desired zonotope order $\rho_d \geq 1$, the operation $\mathrm{reduce}(\mathcal{Z}, \rho_d) \supseteq \mathcal{Z}$ returns an enclosing zonotope with order smaller or equal to ρ_d:*

$$\mathrm{reduce}(\mathcal{Z}, \rho_d) = \langle c, [G_{keep}\ G_{red}]\rangle_Z,$$

with

$$G_{keep} = [G_{(\cdot,\pi_{\chi+1})}\ \dots\ G_{(\cdot,\pi_\gamma)}],\quad G_{red} = \mathrm{diag}\Big(\sum_{i=1}^{\chi} \big|G_{(\cdot,\pi_i)}\big|\Big),$$

where $\pi_1, \dots, \pi_\gamma$ are the indices of the sorted generators

$$\|G_{(\cdot,\pi_1)}\|_1 - \|G_{(\cdot,\pi_1)}\|_\infty \leq \dots \leq \|G_{(\cdot,\pi_\gamma)}\|_1 - \|G_{(\cdot,\pi_\gamma)}\|_\infty,$$

and $\chi = \gamma - \lfloor(\rho_d - 1)n\rfloor$ is the number of reduced generators.

For distances between sets, we use the Hausdorff distance:

Definition 5 (Hausdorff distance): *For two compact sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathbb{R}^n$, the Hausdorff distance with respect to the Euclidean norm is defined as*

$$\begin{aligned}d_H(\mathcal{S}_1, \mathcal{S}_2) = \max\Big\{ &\max_{s_1\in\mathcal{S}_1}\Big(\min_{s_2\in\mathcal{S}_2} \|s_1 - s_2\|_2\Big), \\ &\max_{s_2\in\mathcal{S}_2}\Big(\min_{s_1\in\mathcal{S}_1} \|s_1 - s_2\|_2\Big)\Big\}.\end{aligned} \tag{9}$$

Using a hyperball $\mathcal{B}_\varepsilon$ of radius ε, an alternative definition is

$$d_H(\mathcal{S}_1, \mathcal{S}_2) = \varepsilon \Leftrightarrow \mathcal{S}_2 \subseteq \mathcal{S}_1 \oplus \mathcal{B}_\varepsilon \wedge \mathcal{S}_1 \subseteq \mathcal{S}_2 \oplus \mathcal{B}_\varepsilon. \tag{10}$$

Moreover, we will frequently use the operator

$$\operatorname{err}(\mathcal{S}) := \operatorname{rad}(\operatorname{box}(\mathcal{S})). \tag{11}$$

As an immediate consequence of (11), we obtain

$$d_H\big(\mathbf{0},\mathcal{S}\big) \overset{\mathbf{0}\in\mathcal{S}}{\leq} \operatorname{err}(\mathcal{S})\,, \tag{12}$$

which states that the Hausdorff distance between a set and the origin can be bounded by the radius of an enclosing hyperball.

III. Problem Formulation

We consider linear time-invariant systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + p, \tag{13}$$
$$y(t) = Cx(t) + Wv(t) + q, \tag{14}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{\ell\times n}$, $W \in \mathbb{R}^{\ell\times o}$, where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input, $y(t) \in \mathbb{R}^\ell$ is the output, $v(t) \in \mathbb{R}^o$ is a measurement error, $p \in \mathbb{R}^n$ is a constant input, and $q \in \mathbb{R}^\ell$ is a constant offset on the output. The initial state $x(t_0)$ is uncertain within the initial set $\mathcal{X}^0 \subset \mathbb{R}^n$, the input $u(t)$ is uncertain within the input set $\mathcal{U} \subset \mathbb{R}^m$, and $v(t)$ is uncertain within the set of measurement errors $\mathcal{V} \subset \mathbb{R}^o$. In this work, we assume that $\mathcal{X}^0, \mathcal{U}$, and $\mathcal{V}$ are represented by zonotopes. Using $\mathcal{U} = \langle c_u, G_u\rangle_Z$, we define the vector $\tilde{u} = Bc_u + p \in \mathbb{R}^n$ and the set $\mathcal{U}_0 = \langle \mathbf{0}, BG_u\rangle_Z \subset \mathbb{R}^n$ for later derivations. Please note that we assume a constant vector $\tilde{u}$ to keep the presentation simple, but the extension to time-varying inputs $\tilde{u}(t)$ is straightforward. Without loss of generality, we set the initial time to $t_0 = 0$ and the time horizon to $[0, t_{\text{end}}]$. The reachable set is defined as follows:

Definition 6 (Reachable set): *Let us denote the solution to* (13) *for the initial state* $x(0)$ *and the input signal* $u(\cdot)$ *by* $\xi(t; x(0), u(\cdot))$. *Given an initial set* $\mathcal{X}^0$ *and an input set* $\mathcal{U}$, *the reachable set at time* $t \geq 0$ *is*

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

We denote the time-point reachable set at time $t = t_k$ *by* $\mathcal{R}(t_k)$ *and the time-interval reachable set over* $\tau_k \in [t_k, t_{k+1}]$ *by* $\mathcal{R}(\tau_k) := \bigcup_{t\in[t_k,t_{k+1}]} \mathcal{R}(t)$.

Since the exact reachable set as defined in Def. 6 cannot be computed for general linear systems [7], we aim to compute tight outer-approximations $\widehat{\mathcal{R}}(t) \supseteq \mathcal{R}(t)$ and inner-approximations $\check{\mathcal{R}}(t) \subseteq \mathcal{R}(t)$ instead.

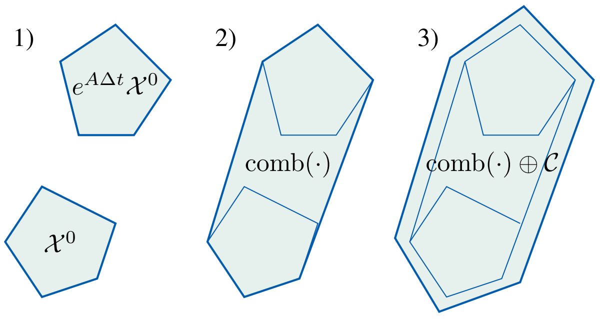


Fig. 2. Visualization of the three steps required to compute the homogeneous solution of the first time interval, adapted from [12, Fig. 3.1].

Algorithm 1 Reachability algorithm (manual tuning)

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0 = \langle c_x, G_x\rangle_Z$, input set $\mathcal{U} = \langle c_u, G_u\rangle_Z$, time horizon t_{end}, time step size Δt dividing the time horizon into an integer number of steps, truncation order η, zonotope order ρ

Ensure: Outer-approximation of the reachable set $\widehat{\mathcal{R}}([0, t_{\text{end}}])$

```
 1: $t_0 \leftarrow 0, \tilde{u} \leftarrow Bc_u + p, \mathcal{U}_0 \leftarrow \langle \mathbf{0}, BG_u\rangle_Z$
 2: $\mathcal{H}(t_0) \leftarrow \mathcal{X}^0, \mathcal{P}^u(t_0) \leftarrow \mathbf{0}, \widehat{\mathcal{P}}^{\mathcal{U}}(t_0) \leftarrow \mathbf{0}$
 3: $\mathcal{P}^u(\Delta t) \leftarrow$ Eq. (19), $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t) \leftarrow$ Eq. (20)
 4: for $k \leftarrow 0$ to $\frac{t_{\text{end}}}{\Delta t} - 1$ do
 5:     $t_{k+1} \leftarrow t_k + \Delta t, \tau_{k+1} \leftarrow [t_k, t_{k+1}]$
 6:     $\mathcal{P}^u(t_{k+1}) \leftarrow \mathcal{P}^u(t_k) \oplus e^{At_k}\mathcal{P}^u(\Delta t)$
 7:     $\mathcal{H}(t_{k+1}) \leftarrow e^{At_{k+1}}\mathcal{X}^0 + \mathcal{P}^u(t_{k+1})$
 8:     $\mathcal{C} \leftarrow \boldsymbol{\mathcal{F}}(\Delta t,\eta)\mathcal{H}(t_k) \oplus \boldsymbol{\mathcal{G}}(\Delta t,\eta)\tilde{u}$      ▷ see (15)-(16)
 9:     $\widehat{\mathcal{H}}(\tau_k) \leftarrow \operatorname{comb}\big(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})\big) \oplus \mathcal{C}$
10:     $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}) \leftarrow \operatorname{reduce}\big(\widehat{\mathcal{P}}^{\mathcal{U}}(t_k) \oplus e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t), \rho\big)$
11:     $\widehat{\mathcal{R}}(\tau_k) \leftarrow \widehat{\mathcal{H}}(\tau_k) \oplus \widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$
12: end for
13: $\widehat{\mathcal{R}}([0,t_{\text{end}}]) \leftarrow \bigcup_{k=0}^{\frac{t_{\text{end}}}{\Delta t}-1} \widehat{\mathcal{R}}(\tau_k)$
```

Our automated tuning approach is based on Alg. 1, which is a slight modification of the wrapping-free propagation-based reachability algorithm [14]. Fundamentally, one exploits the superposition principle of linear systems by separately computing the homogeneous and particular solutions, which are then combined in Line 11 to yield the overall reachable set for each time interval. The homogeneous solution can be enclosed using the following three steps visualized in Fig. 2:

1) Compute the time-point solution by propagating the initial set with the exponential matrix $e^{A\Delta t}$.
2) Approximate the time-interval solution with the linear combination (8), which would only enclose straight-line trajectories.
3) Account for the curvature of the trajectories by enlarging the linear combination with the set $\mathcal{C}$.

The curvature enclosure $\mathcal{C}$ is computed in Line 8 of Alg. 1 using the interval matrices [12, Sec. 3.2]

$$\boldsymbol{\mathcal{F}}(\Delta t,\eta) = \bigoplus_{i=2}^{\eta} \mathcal{I}_i(\Delta t)\,\frac{A^i}{i!} \oplus \boldsymbol{\mathcal{E}}(\Delta t,\eta) \tag{15}$$

$$\boldsymbol{\mathcal{G}}(\Delta t,\eta) = \bigoplus_{i=2}^{\eta+1} \mathcal{I}_i(\Delta t)\,\frac{A^{i-1}}{i!} \oplus \boldsymbol{\mathcal{E}}(\Delta t,\eta)\Delta t \tag{16}$$

$$\text{with}\quad \mathcal{I}_i(\Delta t) = \big[\big(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}}\big)\Delta t^i, 0\big], \tag{17}$$

where the interval matrix $\boldsymbol{\mathcal{E}}(\Delta t_k, \eta_k)$ represents the remainder of the exponential matrix [12, Eq. (3.2)]:

$$\boldsymbol{\mathcal{E}}(\Delta t,\eta) = [-E(\Delta t,\eta), E(\Delta t,\eta)],$$
$$E(\Delta t,\eta) = e^{|A|\Delta t} - \sum_{i=0}^{\eta} \frac{\big(|A|\Delta t\big)^i}{i!}. \tag{18}$$

To increase the tightness, the particular solution $\mathcal{P}^u(\Delta t)$ due to the constant input $\tilde{u}$ [12, Eq. (3.7)],

$$\mathcal{P}^u(\Delta t) = A^{-1}(e^{A\Delta t} - I_n)\,\tilde{u}, \tag{19}$$

is added to the homogeneous solution in Line 7. If the matrix A is not invertible, we can integrate A^{-1} in the power series of the exponential matrix to compute $\mathcal{P}^u(\Delta t)$. The particular solution due to the time-varying input within the set $\mathcal{U}_0$ can be enclosed by [12, Eq. (3.7)]

$$\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t) = \bigoplus_{i=0}^{\eta} \frac{A^i \Delta t^{i+1}}{(i+1)!}\,\mathcal{U}_0 \oplus \boldsymbol{\mathcal{E}}(\Delta t, \eta)\Delta t\,\mathcal{U}_0. \tag{20}$$

Finally, the reachable set $\widehat{\mathcal{R}}([0, t_{\text{end}}])$ for the entire time horizon is given by the union of the sets for individual time-interval reachable sets according to Line 13.

As for all other state-of-the-art reachability algorithms [11]–[14], [21], [43], the main disadvantage of Alg. 1 is that the tightness of the computed reachable set $\widehat{\mathcal{R}}(t)$ is unknown and heavily depends on the chosen time step size Δt, truncation order η, and zonotope order ρ. In this work, we solve both issues by automatically tuning these algorithm parameters such that the Hausdorff distance between the computed enclosure $\widehat{\mathcal{R}}(t)$ and the exact reachable set $\mathcal{R}(t)$ remains below a desired threshold $\varepsilon_{\max}$ at all times:

$$\text{Tune } \Delta t, \eta, \rho \quad \text{s.t.} \quad \forall t \in [0, t_{\text{end}}] : d_H\big(\mathcal{R}(t), \widehat{\mathcal{R}}(t)\big) \leq \varepsilon_{\max}.$$

We will further utilize this result to efficiently extract an inner-approximation of the reachable set (Sec. V) and construct a fully automated verification algorithm (Sec. VI).

IV. Automated Parameter Tuning

Let us now present our approach for the automated tuning of algorithm parameters. While Alg. 1 uses fixed values for Δt, η, and ρ, we tune different values Δt_k, η_k, and ρ_k in each step k based on the induced outer-approximation error in order to satisfy the error bound at all times. To achieve this, we first derive closed-form expressions describing how the individual errors depend on the values of each parameter in Sec. IV-A. Next, we present our automated parameter tuning algorithm in Sec. IV-B and prove its convergence in Sec. IV-C. Finally, we discuss further improvements to the algorithm in Sec. IV-D and describe the extension to output sets in Sec. IV-E.

A. Error Measures

Several sources of outer-approximation errors exist in Alg. 1:

1) **Affine dynamics (Line 9)**: The time-interval solution $\widehat{\mathcal{H}}(\tau_k)$ of the affine dynamics contains errors originating from enclosing the linear combination by zonotopes and the set $\mathcal{C}$ accounting for the curvature of trajectories.
2) **Particular solution (Lines 10-11)**: Using the outer-approximation $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$ based on (20) for the particular solution due to the input set $\mathcal{U}_0$ induces another error. Moreover, the Minkowski addition of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$ to $\widehat{\mathcal{H}}(\tau_k)$ is outer-approximative as it ignores dependencies in time.
3) **Zonotope order reduction (Line 10)**: The representation size of the particular solution $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$ has to be reduced, which induces another error.

In this subsection, we derive upper bounds for all these errors in terms of the Hausdorff distance between an exact set $\mathcal{S}$ and the computed outer-approximation $\widehat{\mathcal{S}}$. We will use two different albeit related error notations: New errors induced in step k are denoted by $\Delta\varepsilon_k^*(\Delta t_k, \eta_k)$ and $\Delta\varepsilon_k^*(\rho_k)$ to emphasize the dependence on the respective algorithm parameters (using $*$ as a placeholder for various superscripts). Some errors for single time steps add up over time. We accumulate all previous errors $\varepsilon_k^*(\cdot)$ until time t_k by

$$\varepsilon_{k+1}^* = \varepsilon_k^* + \Delta\varepsilon_k^*(\cdot), \quad \varepsilon_0^* := 0. \tag{21}$$

Let us now derive closed-form expressions for all errors.

1) Affine Dynamics: We start by determining the error contained in the computed outer-approximation $\widehat{\mathcal{H}}(\tau_k)$ of the time-interval solution for the affine dynamics $\dot{x}(t) = Ax(t) + \tilde{u}$:

Proposition 1 (Affine dynamics error): *Given the set $\mathcal{H}(t_k) = \langle c_h, G_h \rangle_Z$ with $G_h \in \mathbb{R}^{n \times \gamma_h}$, the Hausdorff distance between the exact time-interval solution of the affine dynamics*

$$\mathcal{H}(\tau_k) = \big\{ e^{At}x(t_k) + A^{-1}(e^{A(t-t_k)} - I_n)\tilde{u} \;\big|\; t \in \tau_k,\, x(t_k) \in \mathcal{H}(t_k) \big\}$$

and the corresponding outer-approximation

$$\widehat{\mathcal{H}}(\tau_k) = \text{comb}\big(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})\big) \oplus \mathcal{C} \tag{22}$$

in Line 9 of Alg. 1 is bounded by

$$d_H\big(\mathcal{H}(\tau_k), \widehat{\mathcal{H}}(\tau_k)\big) \leq \Delta\varepsilon_k^h(\Delta t_k, \eta_k) := 2\,\text{err}(\mathcal{C}) + \sqrt{\gamma_h}\,\big\|G_h^{(-)}\big\|_2, \tag{23}$$

where $G_h^{(-)} = (e^{A\Delta t_k} - I_n)G_h$.

Proof. An inner-approximation of $\mathcal{H}(\tau_k)$ is given by

$$\check{\mathcal{H}}(\tau_k) = \text{comb}\big(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})\big) \ominus \mathcal{B}_\mu \ominus \mathcal{C} \subseteq \mathcal{H}(\tau_k),$$

where we additionally have to subtract a hyperball of radius $\mu = \sqrt{\gamma_h}\,\big\|G_h^{(-)}\big\|_2$ bounding the Hausdorff distance between the exact linear combination (5) and the zonotope enclosure (8) according to Prop. 9 in Appendix A. The error $\Delta\varepsilon_k^h(\Delta t_k, \eta_k)$ bounds the distance between $\check{\mathcal{H}}(\tau_k)$ and the outer-approximation $\widehat{\mathcal{H}}(\tau_k)$ in (22). □

2) Particular Solution: We first account for the error induced by enclosing the exact time-point solution $\mathcal{P}^{\mathcal{U}}(t_{k+1})$ with the outer-approximation $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$:

Proposition 2 (Time-point error in particular solution): *The Hausdorff distance between the exact particular solution*

$$\mathcal{P}^{\mathcal{U}}(t_{k+1}) = \left\{ \int_0^{t_{k+1}} e^{A(t_{k+1}-\theta)}u(\theta)\,\mathrm{d}\theta \;\middle|\; u(\theta) \in \mathcal{U}_0 \right\}$$

and the recursively computed outer-approximation

$$\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}) = \widehat{\mathcal{P}}^{\mathcal{U}}(t_k) \oplus e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$$

with $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ computed according to (20) is bounded by

$$\varepsilon_{k+1}^{\mathcal{U}} = \varepsilon_k^{\mathcal{U}} + \Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k, \eta_k), \tag{24}$$

where the error $\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k, \eta_k)$ for one time step is bounded *by*

$$\begin{aligned}&\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)\\ &\quad := \mathrm{err}\left(e^{At_k}\left(\Big(\sum_{i=1}^{\eta_k}\tilde{A}_i\Big)\mathcal{U}_0\oplus\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\,\mathcal{U}_0\right)\right)\\ &\qquad + \mathrm{err}\left(e^{At_k}\left(\bigoplus_{i=1}^{\eta_k}\tilde{A}_i\mathcal{U}_0\oplus\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\,\mathcal{U}_0\right)\right)\end{aligned}\tag{25}$$

with $\tilde{A}_i=\frac{A^i\Delta t_k^{i+1}}{(i+1)!}$ *and* $\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)$ *from* (18).

Proof. The error $\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$ for one time step is given by the Hausdorff distance between the computed outer-approximation and an inner-approximation obtained by considering constant inputs according to Prop. 11 in Appendix A. The overall error $\varepsilon^{\mathcal{U}}_{k+1}$ follows by error propagation as in (21). □

Since $0\in\mathcal{U}_0$, the added set due to uncertain inputs $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\theta)$ is equal to $\mathbf{0}$ at the beginning of each time step ($\theta=0$) and monotonically grows towards the set $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ at the end of the time step ($\theta=\Delta t_k$) as shown in Fig. 3 on the left. The Minkowski sum in Line 9 of Alg. 1 ignores this dependency on time, inducing another outer-approximation error:

Proposition 3 (Time-interval error in particular solution): *The maximum Hausdorff distance at any time* $t\in\tau_k=[t_k,t_{k+1}]$ *between the exact time-interval particular solution*

$$\mathcal{P}^{\mathcal{U}}(t)=\left\{\int_0^t e^{A(t-\theta)}u(\theta)\mathrm{d}\theta\;\middle|\;u(\theta)\in\mathcal{U}_0\right\}$$

and the outer-approximation $\forall t\in\tau_k:\mathcal{P}^{\mathcal{U}}(t)\subseteq\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$ *is bounded by*

$$\max_{t\in[t_k,t_{k+1}]} d_H\big(\mathcal{P}^{\mathcal{U}}(t),\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})\big)\leq\varepsilon^{\mathcal{U}}_k+\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k)$$

with $\varepsilon^{\mathcal{U}}_k$ *from Prop. 2 and the additional error*

$$\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k):=\mathrm{err}\Big(e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\Big).\tag{26}$$

Proof. Since $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\theta)$ grows monotonically with θ, the maximum deviation over the time interval τ_k occurs at $t=t_k$, where the actual additional set would be $\mathbf{0}$, but instead $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ is used. Therefore, the error is

$$d_H\big(\mathbf{0},e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\big)\overset{(12)}{\leq}\mathrm{err}\Big(e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\Big),$$

which corresponds to the size of the additional set. □

3) Zonotope Order Reduction: The zonotope order reduction of the particular solution $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$ in Line 10 of Alg. 1 induces another error. To determine this reduction error, we first split the particular solution $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ into two parts

$$\begin{aligned}e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)&\overset{(20)}{=}e^{At_k}\left(\bigoplus_{i=0}^{\eta}\tilde{A}_i\mathcal{U}_0\oplus\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\,\mathcal{U}_0\right)\\ &=e^{At_k}\underbrace{\Delta t_k\,\mathcal{U}_0}_{=:\widehat{\mathcal{P}}^{\mathcal{U}}_0(\Delta t_k)}\oplus\underbrace{e^{At_k}\left(\bigoplus_{i=1}^{\eta}\tilde{A}_i\mathcal{U}_0\oplus\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t\,\mathcal{U}_0\right)}_{=:\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(\Delta t_k)}\end{aligned}\tag{27}$$

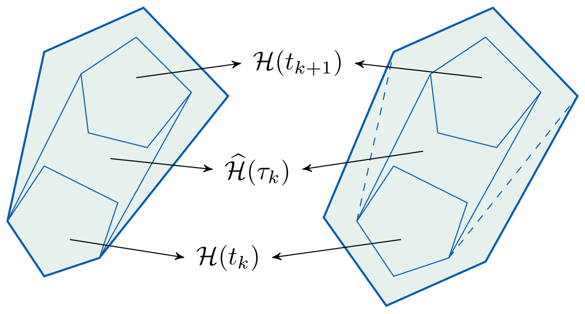


Fig. 3. Reachable set computation using the correct particular time-interval solution $\widehat{\mathcal{P}}^{\mathcal{U}}(\tau_k)$ (left) and an outer-approximation $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})\supseteq\widehat{\mathcal{P}}^{\mathcal{U}}(\tau_k)$ (right).

with $\tilde{A}_i$ defined as in Prop. 2. We exploit that the error $\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$ in (25) is unaffected by using the box outer-approximation $\mathrm{box}(e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(\Delta t_k))$ instead of $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(\Delta t_k)$ since the box outer-approximation is also used in the computation of $\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$. Therefore, we can always reduce $\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_{k+1})$ to a box (which has zonotope order 1) as that reduction error is already contained in $\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$. Consequently, we only have to determine the reduction error of $\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1})$:

Proposition 4 (Zonotope order reduction error): *The Hausdorff distance between the particular solution* $\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1})$ *computed without any reduction and its iteratively reduced counterpart* $\mathrm{reduce}\big(\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1}),\rho_k\big)$ *is bounded by*

$$\varepsilon^{r}_{k+1}=\varepsilon^{r}_k+\Delta\varepsilon^{r}_k(\rho_k),\tag{28}$$

where the error $\Delta\varepsilon^{r}_k(\rho_k)$ *for one time step is bounded by*

$$\begin{aligned}&d_H\big(\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1}),\mathrm{reduce}\big(\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1}),\rho_k\big)\big)\\ &\quad\leq\Delta\varepsilon^{r}_k(\rho_k):=\mathrm{err}(\langle\mathbf{0},G_{red}\rangle_Z),\end{aligned}\tag{29}$$

where G_{red} *defined as in Def. 4 contains the generators selected for reduction.*

Proof. The error $\Delta\varepsilon^{\mathrm{r}}_k(\rho_k)$ for one time step is given by the box enclosure of the zonotope formed by the generators selected for reduction. Using the error propagation formula (21), we then obtain the overall error $\varepsilon^{\mathrm{r}}_{k+1}$ in (28). □

4) Summary: The derived error terms allow us to compute an upper bound for the outer-approximation error contained in the time-point solution and time-interval solution:

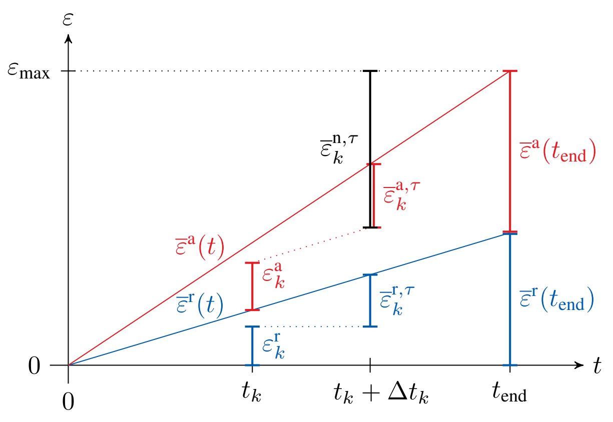
$$d_H\big(\mathcal{R}(t_k),\widehat{\mathcal{R}}(t_k)\big)\leq\varepsilon^{\mathcal{U}}_k+\varepsilon^{\mathrm{r}}_k,\tag{30}$$

$$\begin{aligned}&d_H\big(\mathcal{R}(\tau_k),\widehat{\mathcal{R}}(\tau_k)\big)\leq\varepsilon^{x}_k\\ &\quad:=\Delta\varepsilon^{\mathrm{h}}_k(\Delta t_k,\eta_k)+\varepsilon^{\mathcal{U}}_k+\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k)+\varepsilon^{\mathrm{r}}_{k+1},\end{aligned}\tag{31}$$

where we use $\varepsilon^{\mathcal{U}}_k$ instead of $\varepsilon^{\mathcal{U}}_{k+1}$ in (31), since the difference $\varepsilon^{\mathcal{U}}_{k+1}-\varepsilon^{\mathcal{U}}_k\overset{(24)}{=}\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$ is already included in $\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k)$. As usually only time-interval reachable sets are required for formal verification, we will only use the time-interval error ε^{x}_k in our automated parameter tuning algorithm.

B. Automated Tuning Algorithm

Using the error terms derived in the previous subsection, we now present an algorithm that tunes Δt_k, η_k, and ρ_k automatically such that the Hausdorff distance between the



ε
$\varepsilon_{\max}$
0
$\bar{\varepsilon}^{\mathrm{n},\tau}_k$
$\bar{\varepsilon}^{\mathrm{a},\tau}_k$
$\bar{\varepsilon}^{\mathrm{a}}(t_{\mathrm{end}})$
$\bar{\varepsilon}^{\mathrm{a}}(t)$
$\varepsilon^{\mathrm{a}}_k$
$\bar{\varepsilon}^{\mathrm{r}}(t)$
$\bar{\varepsilon}^{\mathrm{r},\tau}_k$
$\bar{\varepsilon}^{\mathrm{r}}(t_{\mathrm{end}})$
$\varepsilon^{\mathrm{r}}_k$
t
0
t_k
$t_k+\Delta t_k$
t_{end}

Fig. 4. The errors $\varepsilon^{\mathrm{a}}_k$ and $\varepsilon^{\mathrm{r}}_k$ until t_k and the bounds $\bar{\varepsilon}^{\mathrm{a}}(t)$ and $\bar{\varepsilon}^{\mathrm{r}}(t)$ yield the individual error bounds $\bar{\varepsilon}^{\mathrm{n},\tau}_k$, $\bar{\varepsilon}^{\mathrm{a},\tau}_k$, $\bar{\varepsilon}^{\mathrm{r},\tau}_k$ for the current step k.

exact reachable set $\mathcal{R}([0,t_{\mathrm{end}}])$ and the computed enclosure $\widehat{\mathcal{R}}([0,t_{\mathrm{end}}])$ is below the error bound $\varepsilon_{\max}$. As different types of errors require different strategies for parameter tuning, we divide the derived errors into three categories:

1) **Non-accumulating error** $\Delta\varepsilon^{\mathrm{n}}_k(\Delta t_k,\eta_k)$: Since the errors $\Delta\varepsilon^{\mathrm{h}}_k(\Delta t_k,\eta_k)$ and $\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k)$ only affect the current step, we define the non-accumulating error by
$$\Delta\varepsilon^{\mathrm{n}}_k(\Delta t_k,\eta_k) := \Delta\varepsilon^{\mathrm{h}}_k(\Delta t_k,\eta_k) + \Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k). \tag{32}$$
2) **Accumulating error** $\varepsilon^{\mathrm{a}}_k$: The particular solution $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$ accumulates over time, yielding
$$\varepsilon^{\mathrm{a}}_k := \varepsilon^{\mathcal{U}}_k, \qquad \Delta\varepsilon^{\mathrm{a}}_k(\Delta t_k,\eta_k) := \Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k), \tag{33}$$
for the overall accumulating error and the accumulating error for one time step.
3) **Reduction error** $\varepsilon^{\mathrm{r}}_k$: The representation size of the particular solution $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$ is iteratively reduced (Line 10), which induces an accumulating error (28). Despite its accumulation, we do not add this error to $\varepsilon^{\mathrm{a}}_k$ since it does not directly depend on the time step size Δt_k.

We have to manage these errors over time so that the resulting set $\widehat{\mathcal{R}}(t)$ respects the error bound $\varepsilon_{\max}$ at all times. Therefore, we partition $\varepsilon_{\max}$ into individual admissible errors $\bar{\varepsilon}^{\mathrm{n},\tau}_k$, $\bar{\varepsilon}^{\mathrm{a},\tau}_k$, and $\bar{\varepsilon}^{\mathrm{r},\tau}_k$ for each step, which is visualized in Fig. 4:

1) **Reduction error bound** $\bar{\varepsilon}^{\mathrm{r},\tau}_k$: We limit the reduction error by a linearly increasing bound
$$\bar{\varepsilon}^{\mathrm{r}}(t) = \frac{t}{t_{\mathrm{end}}}\zeta\varepsilon_{\max}, \qquad \zeta\in[0,1). \tag{34}$$
Thus, the additional error $\Delta\varepsilon^{\mathrm{r}}_k(\rho_k)$ in step k is bounded by
$$\bar{\varepsilon}^{\mathrm{r},\tau}_k = \bar{\varepsilon}^{\mathrm{r}}(t_k+\Delta t_k) - \varepsilon^{\mathrm{r}}_k, \tag{35}$$
i.e., the difference between the bound at time $t_k+\Delta t_k$ and the accumulated error until t_k. While our algorithm works for arbitrary values ζ, we present a heuristic for choosing ζ later in Sec. IV-D.
2) **Accumulating error bound** $\bar{\varepsilon}^{\mathrm{a},\tau}_k$: Similarly, we limit the accumulating error by another linearly increasing bound
$$\bar{\varepsilon}^{\mathrm{a}}(t) = \frac{t}{t_{\mathrm{end}}}(1-\zeta)\varepsilon_{\max}, \tag{36}$$
so that we have $\bar{\varepsilon}^{\mathrm{r}}(t_{\mathrm{end}}) + \bar{\varepsilon}^{\mathrm{a}}(t_{\mathrm{end}}) = \varepsilon_{\max}$. Analogously to (35), the bound for the additional error $\Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$ is
$$\bar{\varepsilon}^{\mathrm{a},\tau}_k = \bar{\varepsilon}^{\mathrm{a}}(t_k+\Delta t_k) - \varepsilon^{\mathrm{a}}_k. \tag{37}$$
3) **Non-accumulating error bound** $\bar{\varepsilon}^{\mathrm{n},\tau}_k$: Finally, we obtain the bound for the non-accumulating error $\Delta\varepsilon^{\mathrm{n}}_k(\Delta t_k,\eta_k)$ by subtracting the other two bounds from $\varepsilon_{\max}$:
$$\bar{\varepsilon}^{\mathrm{n},\tau}_k = \varepsilon_{\max} - \bar{\varepsilon}^{\mathrm{r}}(t_k+\Delta t_k) - \varepsilon^{\mathrm{a}}_k. \tag{38}$$
Note that we only subtract $\varepsilon^{\mathrm{a}}_k$ instead of $\bar{\varepsilon}^{\mathrm{a}}(t_k+\Delta t_k)$ for the accumulating error since the accumulating error $\Delta\varepsilon^{\mathrm{a}}_k(\Delta t_k,\eta_k) = \Delta\varepsilon^{\mathcal{U}}_k(\Delta t_k,\eta_k)$ for the current step is already accounted for by the error $\Delta\varepsilon^{\mathcal{U},\tau}_k(\Delta t_k,\eta_k)$, which is according to (32) part of the non-accumulating error. This also guarantees us a non-zero bound for $\bar{\varepsilon}^{\mathrm{n},\tau}_k$ in the last step even though $\bar{\varepsilon}^{\mathrm{a}}(t_{\mathrm{end}}) + \bar{\varepsilon}^{\mathrm{r}}(t_{\mathrm{end}}) = \varepsilon_{\max}$.

The tuning strategies for the parameters are as follows:

- **Time step size** Δt_k: We initialize Δt_k by its previous value Δt_{k-1}, or by t_{end} as an initial guess for the first step. To keep the presentation simple, we iteratively halve this value until the error bounds are satisfied; a more sophisticated tuning method is described in Sec. IV-D.
- **Truncation order** η_k: We tune η_k simultaneously with the computation of $\boldsymbol{\mathcal{F}}(\Delta t_k,\eta_k)$ and $\boldsymbol{\mathcal{G}}(\Delta t_k,\eta_k)$, for which the idea proposed in [48, Sec. 3.1] is reused: The partial sums
$$\boldsymbol{\mathcal{T}}^{(j)} = \bigoplus_{i=1}^{j}\mathcal{I}_i\frac{A^i}{i!} \tag{39}$$
in the computation of $\boldsymbol{\mathcal{F}}(\Delta t_k,\eta_k)$ in (15) are successively compared until the relative change in the Frobenius norm of $\boldsymbol{\mathcal{T}}^{(j)}$ computed according to [49, Thm. 10] is smaller than 10^{-10}. As this bound is relative, we can ensure convergence independently of the scale of the system, since the size of the additional terms decreases exponentially for $i\to\infty$.
- **Zonotope order** ρ_k: We iteratively increase the order ρ_k until the error $\Delta\varepsilon^{\mathrm{r}}_k(\rho_k)$ is smaller than the error bound $\bar{\varepsilon}^{\mathrm{r},\tau}_k$. A more efficient method compared to this naive implementation is to directly integrate the search for a suitable order into the zonotope order reduction.

The resulting automated tuning algorithm is shown in Alg. 2: In the repeat-until loop (Lines 6-15), we first decrease the time step size Δt_k (Line 7) and tune the truncation order η_k (Lines 9-12) until the respective error bounds $\bar{\varepsilon}^{\mathrm{a},\tau}_k$ and $\bar{\varepsilon}^{\mathrm{n},\tau}_k$ for the accumulating and non-accumulating errors are safisfied. After this loop, we compute the particular solution due to the input set $\mathcal{U}_0$ and tune the zonotope order ρ_k (Lines 20-22) yielding the reduction error $\Delta\varepsilon^{\mathrm{r}}_k(\rho_k)$. Afterwards, we compute the solution to the affine dynamics (Lines 25-28) and finally obtain the reachable set of the current time interval (Line 30).

The runtime complexity of Alg. 2 is $\mathcal{O}(n^3)$ as for the base algorithm, Alg. 1. For an initial set $\mathcal{X}^0$ with zonotope order ρ_X and an input set with zonotope order ρ_U, the space complexity for the k-th set $\widehat{\mathcal{R}}(\tau_k)$ is bounded by $\mathcal{O}\big(n^2(\rho_X+k\rho_U)\big)$ and the space complexity for Alg. 2 then follows by summing over all individual steps.

Algorithm 2 Reachability algorithm (automated tuning)

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0 = \langle c_x, G_x \rangle_Z$, input set $\mathcal{U} = \langle c_u, G_u \rangle_Z$, time horizon t_{end}, error bound ε_{max}

Ensure: Outer-approximation of the reachable set $\widehat{\mathcal{R}}([0, t_{\text{end}}])$

```
1:  $k \leftarrow 0, t_0 \leftarrow 0, \Delta t_{-1} \leftarrow t_{\text{end}}, \mathcal{H}(t_0) \leftarrow \mathcal{X}^0$
2:  $\mathcal{P}^u(t_0) \leftarrow \langle \mathbf{0}, [\,] \rangle_Z, \widehat{\mathcal{P}}^{\mathcal{U}}_0(t_0) \leftarrow \langle \mathbf{0}, [\,] \rangle_Z, \widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_0) \leftarrow \langle \mathbf{0}, [\,] \rangle_Z$
3:  $\tilde{u} \leftarrow Bc_u + p, \mathcal{U}_0 \leftarrow \langle \mathbf{0}, BG_u \rangle_Z$
4:  while $t_k < t_{\text{end}}$ do
5:      $\Delta t_k \leftarrow 2\Delta t_{k-1}$
6:      repeat
7:          $\Delta t_k \leftarrow \frac{1}{2}\Delta t_k,\ t_{k+1} \leftarrow t_k + \Delta t_k$
8:          $\eta_k \leftarrow 0, \boldsymbol{\mathcal{T}}^{(\eta_k)} = \mathbf{0}$
9:          repeat
10:             $\eta_k \leftarrow \eta_k + 1$
11:             $\boldsymbol{\mathcal{T}}^{(\eta_k)} \leftarrow \boldsymbol{\mathcal{T}}^{(\eta_k - 1)} \oplus \mathcal{I}_{\eta_k} \frac{A^{\eta_k}}{\eta_k!}$   ▷ see (17), (39)
12:         until $1 - \left\|\boldsymbol{\mathcal{T}}^{(\eta_k - 1)}\right\|_F / \left\|\boldsymbol{\mathcal{T}}^{(\eta_k)}\right\|_F \leq 10^{-10}$
13:         $\Delta\varepsilon^{\text{n}}_k(\Delta t_k, \eta_k), \Delta\varepsilon^{\text{a}}_k(\Delta t_k, \eta_k) \leftarrow$ (32), (33)
14:         $\bar{\varepsilon}^{\text{a},\tau}_k, \bar{\varepsilon}^{\text{n},\tau}_k \leftarrow$ (37), (38)
15:     until $\Delta\varepsilon^{\text{a}}_k(\Delta t_k, \eta_k) \leq \bar{\varepsilon}^{\text{a},\tau}_k \wedge \Delta\varepsilon^{\text{n}}_k(\Delta t_k, \eta_k) \leq \bar{\varepsilon}^{\text{n},\tau}_k$
16:     $\widehat{\mathcal{P}}^{\mathcal{U}}_0(\Delta t_k), \widehat{\mathcal{P}}^{\mathcal{U}}_\infty(\Delta t_k) \leftarrow$ (27)
17:     $\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_{k+1}) \leftarrow \widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_k) \oplus \text{box}\Big(e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(\Delta t_k)\Big)$
18:     $\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1}) \leftarrow \widehat{\mathcal{P}}^{\mathcal{U}}_0(t_k) \oplus e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}_0(\Delta t_k)$
19:     $\bar{\varepsilon}^{\text{r},\tau}_k \leftarrow$ (35), $\rho_k \leftarrow 0$
20:     repeat
21:         $\rho_k \leftarrow \rho_k + 1, \Delta\varepsilon^{\text{r}}_k(\rho_k) \leftarrow$ Prop. 4
22:     until $\Delta\varepsilon^{\text{r}}_k(\rho_k) \leq \bar{\varepsilon}^{\text{r},\tau}_k$
23:     $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}) \leftarrow \text{reduce}\big(\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{k+1}), \rho_k\big) \oplus \widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_{k+1})$
24:     $\mathcal{P}^u(\Delta t_k) \leftarrow$ (19)
25:     $\mathcal{P}^u(t_{k+1}) \leftarrow \mathcal{P}^u(t_k) \oplus e^{At_k}\mathcal{P}^u(\Delta t_k)$
26:     $\mathcal{H}(t_{k+1}) \leftarrow e^{At_{k+1}}\mathcal{X}^0 + \mathcal{P}^u(t_{k+1})$
27:     $\mathcal{C} \leftarrow \boldsymbol{\mathcal{F}}(\Delta t_k, \eta_k)\widehat{\mathcal{H}}(t_k) \oplus \boldsymbol{\mathcal{G}}(\Delta t_k, \eta_k)\tilde{u}$ ▷ see (15),(16)
28:     $\widehat{\mathcal{H}}(\tau_k) \leftarrow \text{comb}\big(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})\big) \oplus \mathcal{C}$
29:     $\varepsilon^{\text{a}}_{k+1}, \varepsilon^{\text{r}}_{k+1} \leftarrow$ (33), (28)
30:     $\widehat{\mathcal{R}}(\tau_k) \leftarrow \widehat{\mathcal{H}}(\tau_k) \oplus \widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1})$
31:     $k \leftarrow k + 1$
32: end while
33: return $\widehat{\mathcal{R}}([0, t_{\text{end}}]) \leftarrow \bigcup_{j=0}^{k-1} \widehat{\mathcal{R}}(\tau_j)$
```

C. Proof of Convergence

While Alg. 2 guarantees to return a reachable set $\widehat{\mathcal{R}}([0, t_{\text{end}}])$ satisfying the error bound ε_{max} by construction, it remains to show that the algorithm terminates in finite time. To respect the linearly increasing bound for the accumulating error, we have to show that this error decreases faster than linearly with the time step size Δt_k; thus, by successively halving the time step size, we will always find a time step size so that the error bound is satisfied. Using Lemmas 1-4 from Appendix B, we now formulate our main theorem:

Theorem 1 (Convergence): *Alg. 2 terminates in finite time for arbitrary error bounds $\varepsilon_{max} > 0$.*

Proof. By Lemma 2, the additional accumulating error $\Delta\varepsilon^{\text{a}}_k(\Delta t_k, \eta_k)$ decreases quadratically with Δt_k. Thus, we are guaranteed to find a time step size that satisfies the linearly decreasing bound $\bar{\varepsilon}^{\text{a},\tau}_k$ by successively halving Δt_k. The non-accumulating error $\Delta\varepsilon^{\text{n}}_k(\Delta t_k, \eta_k)$ decreases at least linearly with Δt_k according to Lemmas 3-4. Since the error bound $\bar{\varepsilon}^{\text{n},\tau}_k$ approaches a constant value greater than 0 for $\Delta t_k \to 0$, we are therefore always able to safisfy $\bar{\varepsilon}^{\text{n},\tau}_k$ by reducing the time step size. The additional reduction error $\Delta\varepsilon^{\text{r}}_k(\rho_k)$ can be set to 0 by simply omitting the reduction, which trivially satisfies any bound $\bar{\varepsilon}^{\text{r},\tau}_k$. □

Our adaptive algorithm Alg. 2 must be based on a wrapping-free reachability algorithm to guarantee convergence as successive propagation with $e^{A\Delta t_k}$ would eliminate the required faster-than-linear decrease of the accumulating error.

D. Improved Tuning Methods

While Alg. 2 is guaranteed to converge, there is still room for improvement regarding the computation time. Hence, we present enhanced methods for adapting the time step size Δt and the choice of ζ in (34) determining the amount of error that is allocated for reduction.

1) Time Step Size: Ideally, the chosen time step size Δt_k fulfills the resulting error bounds as tightly as possible. To this end, we replace the naive adaptation of Δt_k in Line 7 of Alg. 2 by regression: We use the previously obtained error values as data points to define linear and quadratic approximation functions modeling the behavior of the error over Δt_k, depending on the asymptotic behavior of the respective errors according to Lemmas 1-4 from Appendix B. We then compute an estimate of the time step size required to satisfy the error bounds based on the approximation functions. This estimate is then refined until the error bounds are satisfied.

2) Reduction Error Allocation: The second major improvement is to pre-compute a near-optimal value for the parameter ζ in (34) using a heuristic that aims to minimize the zonotope order of the resulting reachable sets. Our heuristic is based on the following observation: For increasing values of ζ, more margin is allocated to the reduction error and less margin to the accumulating and non-accumulating errors. Thus, the total number of steps increases because the algorithm has to select smaller time step sizes, yielding a higher zonotope order. At the same time, the zonotope order can be lowered more due to the larger reduction error margin. We now want to determine the optimal value of ζ balancing these two effects.

We first estimate the zonotope order of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{\text{end}})$ using the number of time steps if reduction is completely omitted. Let us denote the total number of steps for Alg. 2 without reduction ($\zeta = 0$) by k'_0, and the zonotope order of the input set $\mathcal{U}_0$ by $\rho_{\mathcal{U}}$. In each step, the set $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}_0(\Delta t_k) = e^{At_k}\Delta t_k \mathcal{U}_0$ is added to $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$, which iteratively increases the zonotope of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_k)$ by $\rho_{\mathcal{U}}$. Due to the linear decrease of the total error (see Lemmas 2-4 in Appendix B), using the value $\zeta = 0.5$ at most doubles the number of steps compared to $\zeta = 0$. Therefore, the zonotope order of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{\text{end}}) = \widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{\text{end}}) \oplus \widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_{\text{end}})$ can be estimated as

$$\rho^+(\zeta) = \frac{1}{1-\zeta} k'_0 \rho_{\mathcal{U}} + 1, \tag{40}$$

if no order reduction takes place, where the summands represent the orders of $\widehat{\mathcal{P}}^{\mathcal{U}}_0(t_{\text{end}})$ and $\widehat{\mathcal{P}}^{\mathcal{U}}_\infty(t_{\text{end}})$, respectively.

Next, we estimate the zonotope order of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{\text{end}})$ for a non-zero value $\zeta > 0$ yielding a non-zero error margin $\overline{\varepsilon}^{\text{r}}(t) > 0$ that is used to reduce the order of $\widehat{\mathcal{P}}^{\mathcal{U}}(t_{\text{end}})$. Using a fixed time step size Δt yielding an integer number $k' = \frac{t_{\text{end}}}{\Delta t}$ of time steps, we compute the sequence

$$\forall j \in \{1, ..., k'+1\} : \tilde{\varepsilon}^{\text{r}}_j = \text{err}\Big(e^{A(j-1)\Delta t} \Delta t \, \mathcal{U}_0\Big),$$

which estimates the maximum reduction error in each time step. Let us introduce the ordering π which permutes $\{1, ..., k'+1\}$ such that $\tilde{\varepsilon}^{\text{r}}_{\pi_1} < ... < \tilde{\varepsilon}^{\text{r}}_{\pi_{k'+1}}$. To mimic the accumulation of the reduction error, we introduce the cumulative sum over all $\tilde{\varepsilon}^{\text{r}}_j$ ordered by π:

$$\forall j \in \{1, ..., k'+1\} : \sigma_j = \sum_{i=1}^{j+1} \tilde{\varepsilon}^{\text{r}}_{\pi_i}.$$

The maximum reducible order $\rho^-(\zeta)$ exploits the reduction error bound $\overline{\varepsilon}^{\text{r}}(t_{\text{end}}) = \zeta \varepsilon_{\max}$ as much as possible:

$$\begin{aligned} &\rho^-(\zeta) = j^* \rho_{\mathcal{U}}, \\ &\text{where } j^* = \operatorname*{arg\,max}_{j \in \{1,...,k'+1\}} \sigma_j \leq \zeta \varepsilon_{\max}. \end{aligned} \tag{41}$$

Finally, we combine the two parts (40) and (41) describing the counteracting influences to obtain the following heuristic:

$$\zeta = \operatorname*{arg\,min}_{\zeta \in [0,1)} \rho^+(\zeta) - \rho^-(\zeta). \tag{42}$$

Since this is a scalar optimization problem, we use a fine grid of different values for $\zeta \in [0,1)$ to estimate the optimal value.

E. Extension to Output Sets

We now show how to extend the proposed algorithm to outputs $y(t)$. The output set $\mathcal{Y}(t)$ can be computed by evaluating the output equation (14) in a set-based manner:

$$\mathcal{Y}(t) = C\mathcal{R}(t) \oplus W\mathcal{V} + q. \tag{43}$$

For the outer-approximation error of the output set, we have to account for the linear transformation with the matrix C:

Proposition 5: *Consider a linear system of the form* (13)-(14). *Given the error* ε^x_k (31) *of the reachable set* $\widehat{\mathcal{R}}(\tau_k)$, *the corresponding output set* $\widehat{\mathcal{Y}}(\tau_k)$ *has an error of*

$$d_H\big(\mathcal{Y}(\tau_k), \widehat{\mathcal{Y}}(\tau_k)\big) \leq \varepsilon^y_k := \varepsilon^x_k \big\|C\big\|_2. \tag{44}$$

Proof. For the error in the state $x(t)$, we have

$$d_H\big(\mathcal{R}(\tau_k), \widehat{\mathcal{R}}(\tau_k)\big) \leq \varepsilon^x_k \overset{(10)}{\Rightarrow} \widehat{\mathcal{R}}(\tau_k) \subseteq \mathcal{R}(\tau_k) \oplus \mathcal{B}_\varepsilon,$$

where the hyperball $\mathcal{B}_\varepsilon$ has radius $\varepsilon = \varepsilon^x_k$. Applying the output equation (14) to the right-hand side yields

$$\begin{aligned} &C\widehat{\mathcal{R}}(\tau_k) \oplus W\mathcal{V} + q \subseteq C\mathcal{R}(\tau_k) \oplus C\mathcal{B}_\varepsilon \oplus W\mathcal{V} + q \\ \overset{(43)}{\Leftrightarrow} \quad &\widehat{\mathcal{Y}}(\tau_k) \subseteq \mathcal{Y}(\tau_k) \oplus C\mathcal{B}_\varepsilon. \end{aligned}$$

The error in $\widehat{\mathcal{Y}}(\tau_k)$ is therefore given by the radius of the smallest sphere enclosing the set $C\mathcal{B}_\varepsilon$, i.e.,

$$\begin{aligned} \text{rad}(C\mathcal{B}_\varepsilon) &= \text{rad}\left(\{Cz \mid z^\top z \leq \varepsilon\}\right) \\ &= \max_{\|z\|_2 \leq \varepsilon} \big\|Cz\big\|_2 = \varepsilon \max_{\|z\|_2 \leq 1} \big\|Cz\big\|_2 = \varepsilon \big\|C\big\|_2, \end{aligned}$$

where $\big\|C\big\|_2$ is the largest singular value of C. □

V. Inner-Approximations

As shown in Sec. IV, the outer-approximation $\widehat{\mathcal{R}}(t)$ computed by Alg. 2 has a Hausdorff distance of at most $\varepsilon_{\max}$ to the exact reachable set $\mathcal{R}(t)$. Consequently, an inner-approximation $\check{\mathcal{R}}(t) \subseteq \mathcal{R}(t)$ can be computed by the Minkowski difference $\check{\mathcal{R}}(t) = \widehat{\mathcal{R}}(t) \ominus \mathcal{B}_\varepsilon$ of the outer-approximation and the hyperball $\mathcal{B}_\varepsilon$ with radius $\varepsilon = \varepsilon_{\max}$. Note that one can also replace $\varepsilon_{\max}$ by the computed error from Alg. 2 to obtain a tighter inner-approximation. Unfortunately, there exists no closed formula for the Minkowski difference of a zonotope and a hyperball. Therefore, we first enclose $\mathcal{B}_\varepsilon$ with a polytope $\mathcal{P} \supseteq \mathcal{B}_\varepsilon$ since the Minkowski difference of a zonotope and a polytope can be computed efficiently if the resulting set is represented by a constrained zonotope:

Proposition 6 (Minkowski difference): *Given a zonotope* $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ *and a polytope* $\mathcal{P} = \langle [v_1 \ \dots \ v_s] \rangle_V \subset \mathbb{R}^n$, *their Minkowski difference can be represented by the constrained zonotope*

$$\mathcal{Z} \ominus \mathcal{P} = \langle c - v_1, [G \ \mathbf{0}], A, b \rangle_{CZ},$$

where

$$A = \begin{bmatrix} G & -G & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ G & \mathbf{0} & \dots & -G \end{bmatrix}, \quad b = \begin{bmatrix} v_1 - v_2 \\ \vdots \\ v_1 - v_s \end{bmatrix}.$$

Proof. According to [50, Lemma 1], the Minkowski difference with a polytope as minuend can be computed as

$$\mathcal{Z} \ominus \mathcal{P} = \bigcap_{i \in \{1,...,s\}} (\mathcal{Z} - v_i) = (\mathcal{Z} - v_1) \cap \ldots \cap (\mathcal{Z} - v_s).$$

Using the equation for the intersection of constrained zonotopes in [44, Eq. (13)], we obtain for the first intersection

$$\begin{aligned} &(\mathcal{Z} - v_1) \cap (\mathcal{Z} - v_2) \\ &\quad = \langle c - v_1, G, [\,], [\,] \rangle_{CZ} \cap \langle c - v_2, G, [\,], [\,] \rangle_{CZ} \\ &\overset{[44,\text{ Eq. (13)}]}{=} \langle c - v_1, G, [G \ -G], v_1 - v_2 \rangle_{CZ}. \end{aligned}$$

Repeated application of [44, Eq. (13)] yields the claim. □

For general polytopes the number of vertices increases exponentially with the system dimension. To keep the computational complexity small, we enclose the hyperball $\mathcal{B}_\varepsilon$ by a cross-polytope $\langle \varepsilon \sqrt{n} [-I_n \ I_n] \rangle_V \supseteq \mathcal{B}_\varepsilon$, which is a special type of polytope with only $2n$ vertices. Since the Hausdorff distance between the hyperball and the enclosing cross-polytope is $(\sqrt{n} - 1)\varepsilon$, we scale the error bound $\varepsilon_{\max}$ by the factor $1/\sqrt{n}$ before executing Alg. 2 in order to obtain an inner-approximation with a maximum Hausdorff distance of $\varepsilon_{\max}$ to the exact reachable set.

VI. Automated Verification

One core application of reachability analysis is the verification of safety specifications. Based on our automated parameter tuning approach, we introduce a fully automated verification algorithm for linear systems, which iteratively refines the tightness of the reachable set inner-approximation and outer-approximation until a given specification can be verified or falsified. We consider specifications of the form

$$\forall t \in [0, t_{\text{end}}]: \left(\bigwedge_{i=1}^{r} \mathcal{R}(t) \subseteq \mathcal{G}_i \right) \wedge \left(\bigwedge_{i=1}^{w} \mathcal{R}(t) \cap \mathcal{F}_i = \emptyset \right)$$

defined by a list of safe sets $\{\mathcal{G}_1, \dots, \mathcal{G}_r\} \subset \mathbb{R}^n$ and a list of unsafe sets $\{\mathcal{F}_1, \dots, \mathcal{F}_w\} \subset \mathbb{R}^n$, both specified as polytopes in halfspace representation. While we omit the dependence on time here for simplicity, the extension to time-varying safe sets and unsafe sets is straightforward. To check if the reachable set satisfies the specification, we need to perform containment and intersection checks on zonotopes and constrained zonotopes:

Proposition 7 (Containment check): *Given a polytope $\mathcal{P} = \langle C, d\rangle_H \subset \mathbb{R}^n$ and a constrained zonotope $\mathcal{CZ} = \langle c, G, A, b\rangle_{CZ} \subset \mathbb{R}^n$, we have*

$$\mathcal{CZ} \subseteq \mathcal{P} \;\Leftrightarrow\; \underbrace{\max\left(\nu_1, \dots, \nu_a\right)}_{\nu} \leq 0, \tag{45}$$

where each linear program

$$\begin{aligned} \forall i \in \{1, \dots, a\}: \quad \nu_i = \max_{\alpha \in \mathbb{R}^{\gamma}} \; & C_{(i,\cdot)} c + C_{(i,\cdot)} G \alpha - d_{(i)} \\ \text{s.t.} \;\; & \alpha \in [-\mathbf{1}, \mathbf{1}], \; A\alpha = b. \end{aligned}$$

computes the distance to a single polytope halfspace.

Proof. In general, a set $\mathcal{S} \subset \mathbb{R}^n$ is contained in a polytope if it is contained in all polytope halfspaces. The linear program above evaluates the support function (see [14, Def. 1]) of $\mathcal{S}$ along the normal vector of each halfspace, which has to be smaller or equal to the corresponding offset to prove containment [51, Corollary 13.1.1]. □

For a zonotopic in-body $\mathcal{Z} = \langle c, G\rangle_Z \subset \mathbb{R}^n$, there is a closed-form solution ([51, Corollary 13.1.1] with [14, Prop. 1]):

$$\mathcal{Z} \subseteq \mathcal{P} \;\Leftrightarrow\; \underbrace{\max\left(Cc - d + \sum_{i=1}^{\gamma} |CG_{(\cdot,i)}| \right)}_{\nu} \leq 0. \tag{46}$$

Next, we consider intersection checks:

Proposition 8 (Intersection check): *A polytope $\mathcal{P} = \langle C, d\rangle_H \subset \mathbb{R}^n$ and a constrained zonotope $\mathcal{CZ} = \langle c, G, A, b\rangle_{CZ} \subset \mathbb{R}^n$ intersect if $\nu \leq 0$ computed by the linear program*

$$\begin{aligned} \nu = \min_{x \in \mathbb{R}^n,\, \alpha \in \mathbb{R}^{\gamma},\, \delta \in \mathbb{R}} \; & \delta \\ \text{s.t.} \;\; & \forall i \in \{1, ..., a\}: C_{(i,\cdot)} x - d_{(i)} \leq \delta, \\ & x = c + G\alpha, \; A\alpha = b, \; \alpha \in [-\mathbf{1}, \mathbf{1}]. \end{aligned}$$

Proof. If $\forall i \in \{1, ..., a\}: C_{(i,\cdot)} x - d_{(i)} \leq \delta \leq 0$, then there exists a point $x \in \mathcal{CZ}$ that is also contained in $\mathcal{P}$. □

Algorithm 3 Automated verification

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0 = \langle c_x, G_x\rangle_Z$, input set $\mathcal{U} = \langle c_u, G_u\rangle_Z$, time horizon t_{end}, specification defined by a list of safe sets $\mathcal{G}_1, \dots, \mathcal{G}_r$ and a list of unsafe sets $\mathcal{F}_1, \dots, \mathcal{F}_w$

Ensure: Specification satisfied (true) or violated (false)

```
 1: $\varepsilon_{\max} \leftarrow$ estimated from simulations
 2: repeat
 3:     $\widehat{\mathcal{R}}(t) \leftarrow$ comp. with Alg. 2 using error bound $\varepsilon_{\max}$
 4:     $\check{\mathcal{R}}(t) \leftarrow \widehat{\mathcal{R}}(t) \ominus \mathcal{B}_\varepsilon$                ▷ see Sec. V
 5:     $\widehat{\nu}_G, \check{\nu}_G \leftarrow -\infty$, $\widehat{\nu}_F, \check{\nu}_F \leftarrow \infty$
 6:     for $j \leftarrow 1$ to $r$ do
 7:         $\nu \leftarrow$ distance from $\widehat{\mathcal{R}}(t) \subseteq \mathcal{G}_j$          ▷ see (46)
 8:         $\widehat{\nu}_G \leftarrow \max(\widehat{\nu}_G, \nu)$
 9:         $\nu \leftarrow$ distance from $\check{\mathcal{R}}(t) \subseteq \mathcal{G}_j$          ▷ see (45)
10:         $\check{\nu}_G \leftarrow \max(\check{\nu}_G, \nu)$
11:     end for
12:     for $j \leftarrow 1$ to $w$ do
13:         $\nu \leftarrow$ distance from $\widehat{\mathcal{R}}(t) \cap \mathcal{F}_j = \emptyset$   ▷ see Prop. 8
14:         $\widehat{\nu}_F \leftarrow \min(\widehat{\nu}_F, \nu)$
15:         $\nu \leftarrow$ distance from $\check{\mathcal{R}}(t) \cap \mathcal{F}_j = \emptyset$   ▷ see Prop. 8
16:         $\check{\nu}_F \leftarrow \min(\check{\nu}_F, \nu)$
17:     end for
18:     $\nu \leftarrow \min(-\check{\nu}_G, \check{\nu}_F)$
19:     if $\widehat{\nu}_G \geq 0$ then
20:         $\nu \leftarrow \min(\nu, \widehat{\nu}_G)$
21:     end if
22:     if $\widehat{\nu}_F \leq 0$ then
23:         $\nu \leftarrow \min(\nu, -\widehat{\nu}_F)$
24:     end if
25:     $\varepsilon_{\max} \leftarrow \max\left(0.1\,\varepsilon_{\max}, \min(\nu, 0.9\,\varepsilon_{\max})\right)$
26: until $\left((\widehat{\nu}_G \leq 0) \wedge (\widehat{\nu}_F > 0)\right) \vee (\check{\nu}_G > 0) \vee (\check{\nu}_F \leq 0)$
27: return $(\widehat{\nu}_G \leq 0) \wedge (\widehat{\nu}_F > 0)$
```

Since a zonotope is just a special case of a constrained zonotope, Prop. 8 can also be used to check if a zonotope intersects a polytope. For both Prop. 7 and Prop. 8, ν is a good estimate for the Hausdorff distance between the sets if polytopes with normalized halfspace normal vectors are used. We utilize this in our automated verification algorithm to estimate the accuracy that is required to verify or falsify the specification.

The overall verification algorithm is summarized in Alg. 3: We first obtain an initial guess for the error bound $\varepsilon_{\max}$ in Line 1 by simulating trajectories for a finite set of points from $\mathcal{X}^0$. The repeat-until loop (Lines 2-26) then refines the inner- and outer-approximations of the reachable set by iteratively decreasing the error bound $\varepsilon_{\max}$ until the specifications can be verified or falsified. In particular, we first compute the outer- and inner-approximation (Lines 3-4). Next, we perform the containment and intersection checks with the safe and unsafe sets (Lines 6-17) and store the corresponding distances $\widehat{\nu}_G, \check{\nu}_G, \widehat{\nu}_F, \check{\nu}_F$. Using these distances, we determine the minimum distance (Lines 18-24), which is then used to update the error bound $\varepsilon_{\max}$ (Line 25). Since ν is only an estimate, we also restrict the updated error bound to the interval $[0.1\,\varepsilon_{\max}, 0.9\,\varepsilon_{\max}]$ to guarantee convergence and avoid values that are too small. Finally, the specifications are satisfied if the outer-approximation of the reachable set is contained in all safe sets ($\widehat{\nu}_G \leq 0$) and does not intersect any unsafe sets ($\widehat{\nu}_F > 0$). On the other hand, the specifications are falsified if the inner-approximation of the reachable set is not contained in all safe sets ($\check{\nu}_G > 0$) or intersects an unsafe set ($\check{\nu}_F \leq 0$). Improvements for Alg. 3 which we omitted here for simplicity include using the computed error from Alg. 2 instead of the error bound $\varepsilon_{\max}$, omitting re-computation as well as containment and intersection checks for time intervals that are already verified, and only computing inner-approximations if the corresponding outer-approximation is not yet verified.

The space complexity of Alg. 3 is dominated by the inner-approximation $\check{\mathcal{R}}(\tau_k)$, which is $\mathcal{O}(n^4)$ following Prop. 6. Assuming a conservative bound of $\mathcal{O}(p^{3.5})$ for a linear program with p variables according to [52], the runtime complexity of Alg. 3 is $\mathcal{O}(n^7)$ since we evaluate linear programs in Props. 7-8 with $\mathcal{O}(n^2)$ variables, respectively.

VII. Numerical Examples

Let us now demonstrate the performance of our adaptive tuning approach and our verification algorithm. We integrated both algorithms into the MATLAB toolbox CORA [34], and they will be made publicly available with the 2023 release[1]. All computations are carried out on a 2.59GHz quad-core i7 processor with 32GB memory.

A. Electrical Circuit

To showcase the general concept of our approach, we first consider the deliberately simple example of an electric circuit consisting of a resistance $R = 2\Omega$, a capacitor with capacity $C = 1.5\text{mF}$, and a coil with inductance $L = 2.5\text{mH}$:

$$\begin{bmatrix} \dot{u}_C(t) \\ \dot{i}_L(t) \end{bmatrix} = \begin{bmatrix} -\frac{1}{RC} & \frac{1}{C} \\ -\frac{1}{L} & 0 \end{bmatrix} \begin{bmatrix} u_C(t) \\ i_L(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u_I(t),$$

where the state is defined by the voltage at the capacitor $u_C(t)$ and the current at the coil $i_L(t)$. The initial set is $\mathcal{X}^0 = [1,3]\text{V} \times [3,5]\text{A}$, the input voltage to the circuit $u_I(t)$ is uncertain within the set $\mathcal{U} = [-0.1, 0.1]\text{V}$, and the time horizon is $t_{\text{end}} = 2\text{s}$. As shown in Fig. 5, the inner- and outer-approximations computed using Alg. 2 and Sec. V converge to the exact reachable set with decreasing error bounds. The computation times are 0.26s for $\varepsilon_{\max} = 0.04$, 0.41s for $\varepsilon_{\max} = 0.02$, and 0.55s for $\varepsilon_{\max} = 0.01$.

B. ARCH Benchmarks

Next, we evaluate our verification algorithm on benchmarks from the 2021 ARCH competition [53], where state-of-the-art reachability tools compete with one another to solve challenging verification tasks. We consider all linear continuous-time systems, which are the building benchmark (BLD) describing the movement of an eight-story hospital

[1]available at `https://cora.in.tum.de`

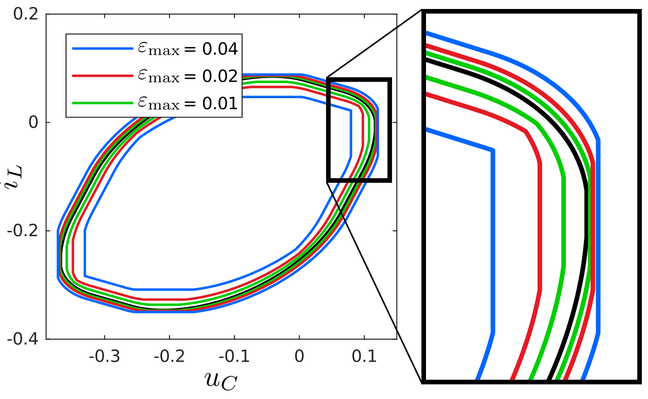


Fig. 5. Inner- and outer-approximations of the final reachable set $\mathcal{R}(t_{\text{end}})$ for the electric circuit using different error bounds $\varepsilon_{\max}$, with the exact reachable set is shown in black.

building, the International Space Station (ISS) benchmark modeling a service module of the ISS, the Heat 3D benchmark (HEAT) representing a spatially discretized version of the heat equation, and the clamped beam benchmark (CB) monitoring oscillations of a beam. The results in Tab. I demonstrate that our fully automated verification algorithm correctly verifies all safe benchmarks without being significantly slower than state-of-the-art tools that require extensive parameter tuning by experts. Please note that we compare only to the computation time of other tools achieved by the optimal run with expert-tuned algorithm parameters, disregarding the significant amount of time required for tuning. Moreover, our algorithm also successfully falsifies the two unsafe benchmarks, where our computation time is slightly worse because the other tools do not explicitly falsify these benchmarks but only test if they cannot be verified, which is considerably easier.

C. Autonomous Car

Finally, we show that our verification algorithm can handle complex verification tasks featuring time-varying specifications. To this end, we consider the benchmark proposed in [54], where the task is to verify that a planned reference trajectory $x_{\text{ref}}(t)$ tracked by a feedback controller is robustly safe despite disturbances and measurement errors. The nonlinear vehicle model in [54, Eq. (3)] is replaced by a linear point mass model, which yields the closed-loop system

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

with $A = [\mathbf{0}\ [I_2\ \mathbf{0}]^\top]$, $B = [\mathbf{0}\ I_2]^\top$, and feedback matrix $K \in \mathbb{R}^{2\times 4}$. The initial set is $\mathcal{X}^0 = (x_0 + \mathcal{V}) \times x_0$ and the set of uncertain inputs is $\mathcal{U} = u_{\text{ref}}(t) \times \mathcal{W} \times \mathcal{V}$, where $\mathcal{W} \subset \mathbb{R}^2$ and $\mathcal{V} \subset \mathbb{R}^4$ are the sets of disturbances and measurement errors taken from [54, Sec. 3], and the initial state $x_0 \in \mathbb{R}^4$ and control inputs for the reference trajectory $u_{\text{ref}}(t) \in \mathbb{R}^2$ are specific to the considered traffic scenario. To compute occupied space of the car, we apply affine arithmetic [55] to evaluate the nonlinear map in [54, Eq. (4)], where we determine the orientation of the car from the direction of the velocity vector.

For verification, we consider the traffic scenario *BEL_Putte-*

TABLE I

COMPARISON OF COMPUTATION TIMES ON THE ARCH BENCHMARKS, WHERE n IS THE SYSTEM DIMENSION, m IS THE NUMBER OF INPUTS, AND ℓ IS THE OUTPUT DIMENSION. FOR OUR APPROACH WE ADDITIONALLY SPECIFY THE NUMBER OF REFINEMENT ITERATIONS OF ALG. 3. THE COMPUTATION TIMES OF THE OTHER TOOLS ARE TAKEN FROM [53].

Benchmark					Our approach		Time comparison			
Identifier	n	m	ℓ	Safe?	Time	Iterations	CORA	HyDRA	JuliaReach	SpaceEx
HEAT01	125	0	1	✓	2.2s	2	2.2s	13.2s	0.13s	4.2s
HEAT02	1000	0	1	✓	59s	1	9.3s	160s	32s	—
CBC01	201	0	1	✓	28s	1	7.1s	—	1.4s	312.78s
CBF01	200	1	1	✓	144s	2	30s	—	12s	318.88s
BLDC01-BDS01	49	0	1	✓	1.7s	1	2.9s	0.426s	0.0096s	1.6s
BLDF01-BDS01	48	1	1	✓	2.1s	1	3.3s	—	0.012s	1.8s
ISSC01-ISS02	273	0	3	✓	4.3s	1	1.3s	—	1.4s	29s
ISSC01-ISU02	273	0	3	✗	10s	4	0.072s	—	1.4s	29s
ISSF01-ISS01	270	3	3	✓	75s	2	59s	—	10s	49s
ISSF01-ISU01	270	3	3	✗	191s	3	38s	—	10s	48s

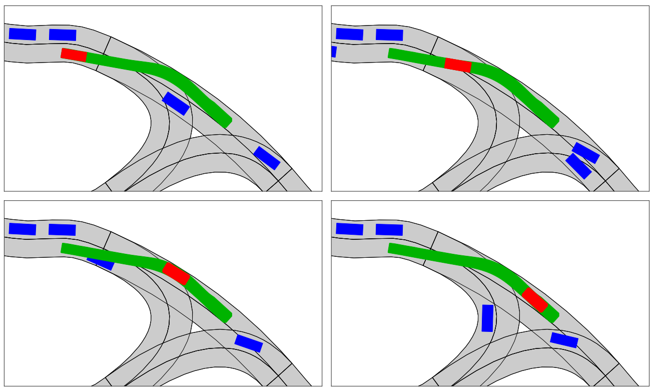


Fig. 6. Traffic scenario at times 0s, 1s, 2s, and 3s, where the reachable set for the whole time horizon, the reachable set for the current time point, and the other traffic participants are shown.

4_2_T-1 from the CommonRoad database[2]. The unsafe sets $\mathcal{F}_i$ for the verification task are given by the road boundary and the occupancy space of other traffic participants. Since the road boundary is non-convex, we use triangulation to represent it as the union of 460 convex polytopes. The occupancy spaces of other traffic participants over time intervals of length 0.1s are represented by polytopes, which results in 170 time-varying unsafe sets for the six vehicles in the scenario. The safe set $\mathcal{G}_i$ is given by the constraint that the absolute acceleration should stay below $11.5\,\mathrm{m\,s^{-2}}$, which we inner-approximate by a polytope with 20 halfspaces. Even for this complex verification task, Alg. 3 only requires 64s and two refinements of the error bound $\varepsilon_{\max}$ to prove that the reference trajectory is robustly safe (see Fig. 6).

VIII. Discussion

Despite the convincing results of our automated verification algorithm in Sec. VII, there is still potential for improvement: According to Tab. I, other reachability tools solve high-dimensional benchmarks often faster than our approach since they apply tailored algorithms, such as block-decomposition [18] and Krylov subspace methods [19], [20]. Therefore, a natural next step is to extend our concept of automated parameter tuning via error analysis to these specialized algorithms to accelerate the verification of high-dimensional systems. In addition, since many reachability algorithms for nonlinear systems [56], [57] are based on reachability analysis for linear systems, another intriguing research direction is the extension to nonlinear systems. Verification algorithms based on implicit set representations, such as support functions [58], also present an interesting comparison to our proposed method, which computes explicit sets.

Moreover, for systems where some states have no initial uncertainty and are not influenced by uncertain inputs, our proposed algorithm cannot falsify the system since the computed inner-approximation of the reachable set will always be empty. An example is the autonomous car in Sec. VII-C, where the states corresponding to the reference trajectory are not subject to any uncertainty. Fortunately, these cases are easy to detect and one can use classical safety falsification techniques, such as Monte Carlo methods [59], Bayesian optimization [60], or cross-entropy techniques [61] instead. In general, combining safety falsification methods with our algorithm may accelerate the falsification process and provide the user with a concrete counterexample in the form of a falsifying trajectory.

Finally, while our algorithm already supports the general case of specifications defined by time-varying safe sets and unsafe sets, future work could include an extension to temporal logic specifications. This may be realized by a conversion to reachset temporal logic [62], a particular type of temporal logic that can be evaluated on reachable sets directly. Another possibility is to convert temporal logic specifications to an acceptance automaton [63], which can then be combined with the linear system via parallel composition [64].

IX. Conclusion

In this work, we propose a paradigm shift for reachability analysis of linear systems: Instead of requiring the user to manually tune algorithm parameters such as the time step size, our approach automatically adapts all parameters internally

[2]available at `https://commonroad.in.tum.de/scenarios`

such that the computed outer-approximation respects a desired maximum distance to the exact reachable set. Building on this result, we then extract an inner-approximation of the reachable set directly from the outer-approximation using the Minkowski difference, which finally enables us to design a sound verification algorithm that automatically refines the inner- and outer-approximations until specifications given by time-varying safe and unsafe sets can either be verified or falsified. An evaluation on benchmarks representing the current limits for state-of-the-art reachability tools demonstrates that our approach is competitive regarding the computation time, even for high-dimensional systems. Overall, the autonomy of our approach enables non-experts to verify or falsify safety specifications for linear systems in reasonable time.

Appendix A: Additional Propositions

For the proof of Prop. 1 we require the error induced by outer-approximating the linear combination for zonotopes:

Proposition 9: *Given the homogeneous solution $\mathcal{H}(t_k) = \langle c_h, G_h \rangle_Z$ with $G_h \in \mathbb{R}^{n \times \gamma_h}$ and the particular solution $\mathcal{P}^u(\Delta t_k) = c_p$, the Hausdorff distance between the exact linear combination*

$$\mathcal{S} = \mathrm{comb}\big(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})\big)$$

with $\mathcal{H}(t_{k+1}) = e^{A\Delta t_k}\mathcal{H}(t_k) + \mathcal{P}^u(\Delta t_k)$ and the corresponding zonotope outer-approximation $\widehat{\mathcal{S}}$ computed using (8) is bounded by

$$d_H\big(\mathcal{S}, \widehat{\mathcal{S}}\big) \leq \sqrt{\gamma_h}\ \big\|G_h^{(-)}\big\|_2,$$

where $G_h^{(-)} = (e^{A\Delta t_k} - I_n)G_h$.

Proof. We insert the homogeneous and particular solutions into (5) and shift the interval of λ from $[0,1]$ to $[-1,1]$, which yields

$$\begin{aligned}\mathcal{S} = \Bigg\{ &\lambda\Big(c_h + \sum_{i=1}^{\gamma_h} G_{h(\cdot,i)}\alpha_i\Big) + (1-\lambda)\Big(e^{A\Delta t_k} \\ &\Big(c_h + \sum_{i=1}^{\gamma_h} G_{h(\cdot,i)}\alpha_i\Big) + c_p\Big) \ \Bigg|\ \alpha_i \in [-1,1], \lambda \in [0,1]\Bigg\}, \\ = \Bigg\{ &0.5\Big(\big(I_n + e^{A\Delta t_k}\big)c_h + c_p\Big) \\ &+ 0.5\lambda\Big(\big(e^{A\Delta t_k} - I_n\big)c_h + c_p\Big) + 0.5\sum_{i=1}^{\gamma_h} G^{(+)}_{h(\cdot,i)}\alpha_i \\ &+ 0.5\sum_{i=1}^{\gamma_h} G^{(-)}_{h(\cdot,i)}\alpha_i\lambda \ \Bigg|\ \alpha_i, \lambda \in [-1,1]\Bigg\},\end{aligned}$$

with $G_h^{(+)} = (e^{A\Delta t_k} + I_n)G_h$ and $G_h^{(-)} = (e^{A\Delta t_k} - I_n)G_h$. In order to represent this exact linear combination $\mathcal{S}$ as a zonotope, we have to substitute the bilinear factors $\alpha_i\lambda$ in the last term by additional linear factors $\omega_i \in [-1,1]$. By neglecting the dependency between α and λ, we obtain the outer-approximation $\widehat{\mathcal{S}}$ of the exact linear combination $\mathcal{S}$. Due to the obvious containment $\mathcal{S} \subseteq \widehat{\mathcal{S}}$, the formula for the Hausdorff distance in (9) simplifies to

$$d_H\big(\mathcal{S}, \widehat{\mathcal{S}}\big) = \max_{\widehat{s} \in \widehat{\mathcal{S}}} \min_{s \in \mathcal{S}} \big\|\widehat{s} - s\big\|_2.$$

Exploiting the identical factors before and after conversion, all terms but one cancel out and we obtain

$$\begin{aligned}\max_{\widehat{s} \in \widehat{\mathcal{S}}} \min_{s \in \mathcal{S}} \big\|\widehat{s} - s\big\|_2 &\leq \max_{\substack{\omega_i \in [-1,1] \\ \alpha_i \in [-1,1] \\ \lambda \in [-1,1]}} 0.5\Big\|\sum_{i=1}^{\gamma_h} G^{(-)}_{h(\cdot,i)}(\omega_i - \alpha_i\lambda)\Big\|_2 \\ &\leq 0.5\big\|G_h^{(-)}\big\|_2 \max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \big\|(\omega - \alpha\lambda)\big\|_2 \end{aligned} \qquad (47)$$

with $\alpha = [\alpha_1\ \dots\ \alpha_{\gamma_h}]^\top$, $\omega = [\omega_1\ \dots\ \omega_{\gamma_h}]^\top$, and $\varphi = [\omega\ \alpha\ \lambda]^\top$. According to the Bauer Maximum Principle, we may assume that the maximum is attained at a point which satisfies the constraints

$$\forall i \in \{1, \dots, \gamma_h\}: \omega_i^2 = 1, \alpha_i^2 = 1, \quad \text{and} \quad \lambda^2 = 1$$

for the maximization term in (47). Consequently, we obtain

$$\begin{aligned}&\max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \big\|\omega - \alpha\lambda\big\|_2^2 = \max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \omega^\top\omega + \alpha^\top\alpha\,\lambda^2 - 2\lambda\,\omega^\top\alpha \\ &\leq \max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \omega^\top\omega + \max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \alpha^\top\alpha\,\lambda^2 + \max_{\varphi \in [-\mathbf{1},\mathbf{1}]} -2\lambda\,\omega^\top\alpha \\ &= \gamma_h + \gamma_h + 2\gamma_h = 4\gamma_h,\end{aligned}$$

which implies $\max_{\varphi \in [-\mathbf{1},\mathbf{1}]} \big\|\omega - \alpha\lambda\big\|_2 \leq 2\sqrt{\gamma_h}$. We insert this result into (47) to obtain the error

$$d_H\big(\mathcal{S}, \widehat{\mathcal{S}}\big) \leq 0.5\big\|G_h^{(-)}\big\|_2 2\sqrt{\gamma_h} = \sqrt{\gamma_h}\big\|G_h^{(-)}\big\|_2,$$

which concludes the proof. □

To derive the error $\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k, \eta_k)$ for one time step contained in the particular solution $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ as used in Prop. 2, we require the following proposition:

Proposition 10: *For a compact set $\mathcal{S} \subset \mathbb{R}^n$ and two matrices $M_1, M_2 \in \mathbb{R}^{m \times n}$, we have*

$$d_H\big((M_1 + M_2)\mathcal{S}, M_1\mathcal{S}\big) \leq d_H\big(\mathbf{0}, M_2\mathcal{S}\big).$$

Proof. For the left-hand side, we choose $s_1 = s_2$ for both min-operations in the definition (9) of the Hausdorff distance:

$$\begin{aligned}&d_H\big((M_1 + M_2)\mathcal{S}, M_1\mathcal{S}\big) \\ &\overset{(9)}{=} \max\Big\{ \max_{s_1 \in \mathcal{S}}\big(\min_{s_2 \in \mathcal{S}} \big\|(M_1 + M_2)s_1 - M_1 s_2\big\|_2\big), \\ &\qquad \max_{s_2 \in \mathcal{S}}\big(\min_{s_1 \in \mathcal{S}} \big\|(M_1 + M_2)s_1 - M_1 s_2\big\|_2\big)\Big\} \\ &\leq \max\Big\{ \max_{s_1 \in \mathcal{S}} \big\|(M_1 + M_2)s_1 - M_1 s_1\big\|_2, \\ &\qquad \max_{s_2 \in \mathcal{S}} \big\|(M_1 + M_2)s_2 - M_1 s_2\big\|_2\Big\} \\ &= \max_{s \in \mathcal{S}} \big\|M_2 s\big\|_2.\end{aligned}$$

The right-hand side evaluates trivially to

$$d_H\big(\mathbf{0}, M_2\mathcal{S}\big) = \max_{s \in \mathcal{S}} \big\|M_2 s\big\|_2,$$

which combined with the result above yields the claim. □

Using Prop. 10, we obtain the following error bound:

Proposition 11: *The Hausdorff distance between the propa-*

gated exact particular solution

$$e^{At_k}\mathcal{P}^{\mathcal{U}}(\Delta t_k) = \left\{ e^{At_k}\int_0^{\Delta t_k} e^{A(\Delta t_k-\theta)}u(\theta)\,\mathrm{d}\theta \,\middle|\, u(\theta)\in\mathcal{U}_0 \right\}$$

and the outer-approximation $e^{At_k}\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ *with* $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ *from* (20) *is bounded by* (25).

Proof. We obtain a tight bound for the error by computing the distance between an inner-approximation $\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ and the outer-approximation $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ in (20). By considering uncertain but constant inputs we compute an inner-approximation $\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k) \subseteq \mathcal{P}^{\mathcal{U}}(\Delta t_k)$ as

$$\begin{aligned}\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k) &= \int_0^{\Delta t_k} e^{A(\Delta t_k-\theta)}\mathrm{d}\theta\,\mathcal{U}_0 \\ &= \left(\sum_{i=0}^{\infty}\frac{A^i\Delta t_k^{i+1}}{(i+1)!}\right)\mathcal{U}_0 = \left(\Delta t_k I_n + \sum_{i=1}^{\infty}\tilde{A}_i\right)\mathcal{U}_0\end{aligned}$$

where $\tilde{A}_i = \frac{A^i\Delta t_k^{i+1}}{(i+1)!}$. Applying Prop. 10 with $M_1 = \Delta t_k I_n$ and $M_2 = \sum_{i=1}^{\infty}\tilde{A}_i$, we have

$$\begin{aligned}d_H\big(\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k), \Delta t_k\mathcal{U}_0\big) &\leq d_H\left(\mathbf{0}, \left(\sum_{i=1}^{\infty}\tilde{A}_i\right)\mathcal{U}_0\right) \\ &\overset{(12)}{\leq} \operatorname{err}\left(\left(\sum_{i=1}^{\infty}\tilde{A}_i\right)\mathcal{U}_0\right) \\ &\overset{(18)}{\leq} \operatorname{err}\left(\left(\sum_{i=1}^{\eta}\tilde{A}_i\right)\mathcal{U}_0 \oplus \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\mathcal{U}_0\right).\end{aligned}$$

From (20), we have the trivial containment $\Delta t_k\mathcal{U}_0 \subset \widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)$ and thus

$$\begin{aligned}&d_H\big(\Delta t_k\mathcal{U}, \widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\big) \\ &\quad\leq \operatorname{err}\left(\bigoplus_{i=1}^{\eta}\left(\tilde{A}_i\mathcal{U}_0\right) \oplus \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\mathcal{U}_0\right).\end{aligned}$$

We use the triangle inequality to combine the last two results:

$$\begin{aligned}d_H\big(\mathcal{P}^{\mathcal{U}}(\Delta t_k), \widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\big) &\leq d_H\big(\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k), \widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\big) \\ &\leq d_H\big(\check{\mathcal{P}}^{\mathcal{U}}(\Delta t_k), \Delta t_k\mathcal{U}_0\big) + d_H\big(\Delta t_k\mathcal{U}_0, \widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t_k)\big) \\ &\leq \operatorname{err}\left(\left(\sum_{i=1}^{\eta}\tilde{A}_i\right)\mathcal{U}_0 \oplus \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\mathcal{U}_0\right) \\ &\quad + \operatorname{err}\left(\bigoplus_{i=1}^{\eta}\left(\tilde{A}_i\mathcal{U}_0\right) \oplus \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k\mathcal{U}_0\right).\end{aligned}$$

Including the mapping by e^{At_k} then yields the final result. □

Appendix B: Additional Lemmata

For the proof of Theorem 1, we require results about the limit behavior of the error terms, which we derive here. First, we examine the remainder of the exponential matrix:

Lemma 1: *The remainder of the exponential matrix* $\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)$ *in* (18) *satisfies*

$$\lim_{\Delta t_k\to 0}\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k) = \mathbf{0} \quad \textit{and} \quad \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k) \sim \mathcal{O}\left(\Delta t_k^{\eta_k+1}\right).$$

Proof. For the limit value we obtain from (18)

$$\begin{aligned}&\lim_{\Delta t_k\to 0} E(\Delta t_k,\eta_k) = e^{|A|0} - \sum_{i=0}^{\eta_k}\frac{1}{i!}\big(|A|0\big)^i = \mathbf{0}, \\ &\lim_{\Delta t_k\to 0}\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k) = \lim_{\Delta t_k\to 0}[-E(\Delta t_k,\eta_k), E(\Delta t_k,\eta_k)] = \mathbf{0},\end{aligned}$$

and the asymptotic behavior follows from

$$E(\Delta t_k,\eta_k) = e^{|A|\Delta t_k} - \sum_{i=0}^{\eta_k}\frac{\big(|A|\Delta t_k\big)^i}{i!} = \sum_{i=\eta_k+1}^{\infty}\frac{\big(|A|\Delta t_k\big)^i}{i!},$$

which yields $\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k) \sim \mathcal{O}\left(\Delta t_k^{\eta_k+1}\right)$. □

Next, we consider the error of the particular solution:

Lemma 2: *The error* $\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k,\eta_k)$ *in* (25) *satisfies*

$$\lim_{\Delta t_k\to 0}\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k,\eta_k) = 0 \quad \textit{and} \quad \Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k,\eta_k) \sim \mathcal{O}\big(\Delta t_k^2\big).$$

Proof. Using the definition of the error $\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k,\eta_k)$ according to (25) with $\tilde{A}_i = \frac{A^i\Delta t_k^{i+1}}{(i+1)!}$ and Lemma 1, we have

$$\begin{aligned}&\lim_{\Delta t_k\to 0}\tilde{A}_i = 0, && \tilde{A}_i \sim \mathcal{O}\big(\Delta t_k^2\big), \\ &\lim_{\Delta t_k\to 0}\boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k = 0, && \boldsymbol{\mathcal{E}}(\Delta t_k,\eta_k)\Delta t_k \sim \mathcal{O}\left(\Delta t_k^{\eta_k+2}\right),\end{aligned}$$

where we exploit that the minimal index is $i = 1$. It is then straightforward to obtain the limit and asymptotic behavior for $\Delta\varepsilon_k^{\mathcal{U}}(\Delta t_k,\eta_k)$. □

While we require a faster than linear decrease in Δt_k for the accumulating error, a linear decrease is sufficient for the non-accumulating error as it only affects a single time step:

Lemma 3: *The error* $\Delta\varepsilon_k^{h}(\Delta t_k,\eta_k)$ *in* (23) *satisfies*

$$\lim_{\Delta t_k\to 0}\Delta\varepsilon_k^{h}(\Delta t_k,\eta_k) = 0 \quad \textit{and} \quad \Delta\varepsilon_k^{h}(\Delta t_k,\eta_k) \sim \mathcal{O}(\Delta t_k).$$

Proof. We examine the two individual terms of $\Delta\varepsilon_k^{\mathrm{h}}(\Delta t_k,\eta_k) = 2\operatorname{err}(\mathcal{C}) + \sqrt{\gamma_h}\,\big\|G_h^{(-)}\big\|_2$ separately:

$$\begin{aligned}\lim_{\Delta t_k\to 0}\sqrt{\gamma_h}\,\big\|G_h^{(-)}\big\|_2 &= \lim_{\Delta t_k\to 0}\sqrt{\gamma_h}\,\big\|(e^{A\Delta t_k} - I_n)G_h\big\|_2 \\ &= \sqrt{\gamma_h}\,\big\|(e^{\mathbf{0}} - I_n)G_h\big\|_2 = \sqrt{\gamma_h}\,\big\|\mathbf{0}\big\|_2 = 0.\end{aligned}$$

To analyze the asymptotic behavior, it suffices to look at $(e^{A\Delta t_k} - I_n)G_h$ as the matrix G_h and the factor γ_h do not depend on the time step size: Since $(e^{A\Delta t_k} - I_n) \sim \mathcal{O}(\Delta t_k)$, we consequently obtain

$$\sqrt{\gamma_h}\,\big\|G_h^{(-)}\big\|_2 \sim \mathcal{O}(\Delta t_k).$$

For the limit behavior of the term $2\operatorname{err}(\mathcal{C})$, we have

$$\begin{aligned}&\lim_{\Delta t_k\to 0}\mathcal{I}_i(\Delta t_k) \overset{(17)}{=} \mathcal{I}_i(0) = \left[\big(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}}\big)0^i, 0\right] = 0, \\ &\lim_{\Delta t_k\to 0}\boldsymbol{\mathcal{F}}(\Delta t_k,\eta_k) \overset{(15)}{=} \bigoplus_{i=2}^{\eta_k}\mathcal{I}_i(0)\,\frac{A^i}{i!} \oplus \boldsymbol{\mathcal{E}}(0,\eta_k) = \mathbf{0}, \\ &\lim_{\Delta t_k\to 0}\boldsymbol{\mathcal{G}}(\Delta t_k,\eta_k) \overset{(16)}{=} \bigoplus_{i=2}^{\eta_k+1}\mathcal{I}_i(0)\,\frac{A^i}{i!} \oplus \boldsymbol{\mathcal{E}}(0,\eta_k)0 = \mathbf{0},\end{aligned}$$

which entails

$$\lim_{\Delta t_k \to 0} 2 \underbrace{\left(\mathrm{err}(\boldsymbol{\mathcal{F}}(0,\eta)\mathcal{H}(t_k)) + \mathrm{err}(\boldsymbol{\mathcal{G}}(0,\eta)\tilde{u})\right)}_{\mathrm{err}(\mathcal{C})} = 0.$$

Moreover, we have $\mathcal{I}_i(\Delta t_k) \sim \mathcal{O}\big(\Delta t_k^2\big)$ as the minimal index is $i = 2$, so that we obtain in combination with Lemma 1

$$2(\mathrm{err}(\boldsymbol{\mathcal{F}}(0,\eta)\mathcal{H}(t_k)) + \mathrm{err}(\boldsymbol{\mathcal{G}}(0,\eta)\tilde{u})) \sim \mathcal{O}\big(\Delta t_k^2\big)\,.$$

It is then straightforward to obtain the limit and asymptotic behavior for $\Delta\varepsilon_k^{\mathrm{h}}(\Delta t_k, \eta_k)$. □

Lemma 4: *The error* $\Delta\varepsilon_k^{\mathcal{U},\tau}(\Delta t_k, \eta_k)$ *in* (26) *satisfies*

$$\lim_{\Delta t_k \to 0} \Delta\varepsilon_k^{\mathcal{U},\tau}(\Delta t_k, \eta_k) = 0 \text{ and } \Delta\varepsilon_k^{\mathcal{U},\tau}(\Delta t_k, \eta_k) \sim \mathcal{O}(\Delta t_k)\,.$$

Proof. For the limit, we insert $\Delta t_k = 0$ into (20) to obtain

$$\lim_{\Delta t_k \to 0} \Delta\varepsilon_k^{\mathcal{U},\tau}(\Delta t_k, \eta_k) = \mathrm{err}\Big(e^{A t_k}\widehat{\mathcal{P}}^{\mathcal{U}}(0)\Big)$$
$$= \mathrm{err}\left(e^{A t_k}\left(\bigoplus_{i=0}^{\eta_k} \frac{A^i 0^{i+1}}{(i+1)!}\mathcal{U}_0 \oplus \boldsymbol{\mathcal{E}}(0,\eta_k)0\,\mathcal{U}_0\right)\right) = 0.$$

According to Lemma 1, we have $\boldsymbol{\mathcal{E}}(\Delta t_k, \eta_k) \sim \mathcal{O}(\Delta t_k)$, and with the first term of the sum above, we obtain $\Delta\varepsilon_k^{\mathcal{U},\tau}(\Delta t_k, \eta_k) \sim \mathcal{O}(\Delta t_k)$. □

References

[1] G. Frehse, B. H. Krogh, and R. A. Rutenbar, "Verifying analog oscillator circuits using forward/backward abstraction refinement," in *Proc. of the Design Automation & Test in Europe Conference*. IEEE, 2006, pp. 257–262.

[2] H. N. V. Pico and D. C. Aliprantis, "Voltage ride-through capability verification of wind turbines with fully-rated converters using reachability analysis," *IEEE Transactions on Energy Conversion*, vol. 29, no. 2, pp. 392–405, 2014.

[3] S. Lengagne, N. Ramdani, and P. Fraisse, "Planning and fast replanning safe motions for humanoid robots," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1095–1106, 2011.

[4] S. Kaynama et al., "Computing the viability kernel using maximal reachable sets," in *Proc. of the 15th International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 55–64.

[5] K. Hobbs et al., "Space debris collision detection using reachability," in *Proc. of the 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2018, pp. 218–228.

[6] S. Vaskov et al., "Guaranteed safe reachability-based trajectory design for a high-fidelity model of an autonomous passenger vehicle," in *Proc. of the American Control Conference*, 2019, pp. 705–710.

[7] 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, 2018.

[8] A. Vinod, B. HomChaudhuri, and M. Oishi, "Forward stochastic reachability analysis for uncontrolled linear systems using Fourier transforms," in *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 2017, pp. 35–44.

[9] A. Devonport, F. Yang, L. El Ghaoui, and M. Arcak, "Data-driven reachability analysis with christoffel functions," in *Proc. of the 60th Conference on Decision and Control*, 2021, pp. 5067–5072.

[10] A. Thorpe, K. Ortiz, and M. Oishi, "Learning approximate forward reachable sets using separating kernels," in *Learning for Dynamics and Control*, 2021, pp. 201–212.

[11] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *8th International Workshop on Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305.

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

[13] G. Frehse et al., "SpaceEx: Scalable verification of hybrid systems," in *Proc. of the 23rd International Conference on Computer Aided Verification*. Springer, 2011, pp. 379–395.

[14] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.

[15] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *10th International Workshop on Hybrid Systems: Computation and Control*. Springer, 2007, pp. 174–189.

[16] T. Dang et al., "Sensitive state-space exploration," in *Proc. of the 47th Conference on Decision and Control*. IEEE, 2008, pp. 4049–4054.

[17] S. Kaynama and M. Oishi, "Complexity reduction through a schur-based decomposition for reachability analysis of linear time-invariant systems," *International Journal of Control*, vol. 84, no. 1, pp. 165–179, 2011.

[18] S. Bogomolov et al., "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*. ACM, 2018, pp. 41–50.

[19] M. Althoff, "Reachability analysis of large linear systems with uncertain inputs in the Krylov subspace," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 477–492, 2020.

[20] S. Bak, H.-D. Tran, and T. T. Johnson, "Numerical verification of affine systems with up to a billion dimensions," in *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*. ACM, 2019, pp. 23–32.

[21] A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *9th International Workshop on Hybrid Systems: Computation and Control*. Springer, 2006, pp. 257–271.

[22] A. Hamadeh and J. Goncalves, "Reachability analysis of continuous-time piecewise affine systems," *Automatica*, vol. 44, no. 12, pp. 3189–3194, 2008.

[23] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *3rd International Workshop on Hybrid Systems: Computation and Control*. Springer, 2000, pp. 202–214.

[24] N. Kochdumper and M. Althoff, "Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems," in *Proc. of the 59th Conference on Decision and Control*. IEEE, 2020, pp. 2130–2137.

[25] E. Goubault and S. Putot, "Inner and outer reachability for the verification of control systems," in *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*. ACM, 2019, pp. 11–22.

[26] E. Asarin et al., "Approximate reachability analysis of piecewise-linear dynamical systems," in *3rd International Workshop on Hybrid Systems: Computation and Control*. Springer, 2000, pp. 20–31.

[27] M. Althoff and G. Frehse, "Combining zonotopes and support functions for efficient reachability analysis of linear systems," in *Proc. of the 55th Conference on Decision and Control*. IEEE, 2016, pp. 7439–7446.

[28] P. S. Duggirala and M. Viswanathan, "Parsimonious, simulation based verification of linear systems," in *Proc. of the 28th International Conference on Computer Aided Verification*. Springer, 2016, pp. 477–494.

[29] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh, "Verification of a cruise control system using counterexample-guided search," *Control Engineering Practice*, vol. 12, no. 10, pp. 1269–1278, 2004.

[30] S. Bogomolov et al., "Guided search for hybrid systems based on coarse-grained space abstractions," *International Journal on Software Tools for Technology Transfer*, vol. 18, pp. 449–467, 2016.

[31] ——, "Counterexample-guided refinement of template polyhedra," in *23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2017, pp. 589–606.

[32] S. Schupp and E. Ábrahám, "Efficient dynamic error reduction for hybrid systems reachability analysis," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2018, pp. 287–302.

[33] T. T. Johnson, S. Bak, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 2, 2016.

[34] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, pp. 120–151.

[35] X. Chen et al., "Flow*: An analyzer for non-linear hybrid systems," in *Proc. of the 25th International Conference Computer-Aided Verification*. Springer, 2013, pp. 258–263.

[36] S. Schupp et al., "HyPRO: A C++ library of state set representations for hybrid systems reachability analysis," in *NASA Formal Methods Symposium*. Springer, 2017, pp. 288–294.

[37] S. Bak and P. S. Duggirala, "HyLAA: A tool for computing simulation-equivalent reachability for linear systems," in *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 2017, pp. 173–178.

[38] S. Bogomolov et al., "JuliaReach: a toolbox for set-based reachability," in *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*. ACM, 2019, pp. 39–44.

[39] R. Ray et al., "XSpeed: Accelerating reachability analysis on multi-core processors," in *Haifa Verification Conference*. Springer, 2015, pp. 3–18.

[40] S. Bak, S. Bogomolov, and C. Schilling, "High-level hybrid systems analysis with Hypy," in *Proc. of the Workshop on Applied Verification of Continuous and Hybrid Systems*, 2016, pp. 80–90.

[41] P. Prabhakar and M. Viswanathan, "A dynamic algorithm for approximate flow computations," in *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control*. ACM, 2011, pp. 133–142.

[42] G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," in *Proc. of the 16th International Conference on Hybrid Systems: Computation and Control*. ACM, 2013, pp. 203–212.

[43] M. Wetzlinger, N. Kochdumper, and M. Althoff, "Adaptive parameter tuning for reachability analysis of linear systems," in *Proc. of the 59th Conference on Decision and Control*. IEEE, 2020, pp. 5145–5152.

[44] J. K. Scott et al., "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.

[45] G. M. Ziegler, *Lectures on polytopes*. Springer Science & Business Media, 2012.

[46] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of linear systems with uncertain parameters and inputs," in *Proc. of the 46th Conference on Decision and Control*. IEEE, 2007, pp. 726–732.

[47] X. Yang and J. K. Scott, "A comparison of zonotope order reduction techniques," *Automatica*, vol. 95, pp. 378–384, 2016.

[48] M. Wetzlinger, A. Kulmburg, and M. Althoff, "Adaptive parameter tuning for reachability analysis of nonlinear systems," in *Proc. of the 24th International Conference on Hybrid Systems: Computation and Control*. ACM, 2021.

[49] R. Farhadsefat, J. Rohn, and T. Lotfi, "Norms of interval matrices," Academy of Sciences of the Czech Republic, Institute of Computer Science, Tech. Rep., 2011.

[50] M. Althoff, "On computing the Minkowski difference of zonotopes," *arXiv preprint arXiv:1512:02794v3*, 2022.

[51] R. Rockafellar, *Convex analysis*. Princeton university press, 1972, vol. 2.

[52] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. of the 16th annual ACM symposium on Theory of computing*, 1984, pp. 302–311.

[53] M. Althoff et al., "ARCH-COMP21 category report: continuous and hybrid systems with linear continuous dynamics," in *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2021, pp. 1–31.

[54] N. Kochdumper, P. Gassert, and M. Althoff, "Verification of collision avoidance for CommonRoad traffic scenarios," in *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2021, pp. 184–194.

[55] L. H. de Figueiredo and J. Stolfi, "Affine arithmetic: Concepts and applications," *Numerical Algorithms*, vol. 37, pp. 147–158, 2004.

[56] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proc. of the 47th Conference on Decision and Control*. IEEE, 2008, pp. 4042–4048.

[57] D. Li, S. Bak, and S. Bogomolov, "Reachability analysis of nonlinear systems using hybridization and dynamics scaling," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2020, pp. 265–282.

[58] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff, "Fully-automated verification of linear systems using reachability analysis with support functions," in *Proc. of the 26th International Conference on Hybrid Systems: Computation and Control*. ACM, 2023.

[59] H. Abbas et al., "Probabilistic temporal logic falsification of cyber-physical systems," *Transactions on Embedded Computing Systems*, vol. 12, no. 2s, 2013.

[60] L. Mathesen, G. Pedrielli, and G. Fainekos, "Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements," in *Proc. of the International Conference on Automation Science and Engineering*, 2021, pp. 732–737.

[61] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proc. of the 15th International Conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 125–134.

[62] H. Roehm et al., "STL model checking of continuous and hybrid systems," in *Proc. of the International Symposium on Automated Technology for Verification and Analysis*, 2016, pp. 412–427.

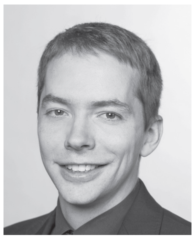
[63] O. Maler, D. Nickovic, and A. Pnueli, "From MITL to timed automata," in *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*, 2006, pp. 274–289.

[64] G. Frehse et al., "A toolchain for verifying safety properties of hybrid automata via pattern templates," in *Proc. of the American Control Conference*, 2018, pp. 2384–2391.

Mark Wetzlinger received the B.S. degree in Engineering Sciences in 2017 jointly from Universität Salzburg, Austria and Technische Universität München, Germany, and the M.S. degree in Robotics, Cognition and Intelligence in 2019 from Technische Universität München, Germany. He is currently pursuing the Ph.D. degree in computer science at Technische Universität München, Germany. His research interests include formal verification of linear and nonlinear continuous systems, reachability analysis, adaptive parameter tuning, and model order reduction.

Niklas Kochdumper received the B.S. degree in Mechanical Engineering in 2015, the M.S. degree in Robotics, Cognition and Intelligence in 2017, and the Ph.D. degree in computer science in 2022, all from Technische Universität München, Germany. He is currently a postdoctoral researcher at Stony Brook University, USA. His research interests include formal verification of continuous and hybrid systems, reachability analysis, computational geometry, controller synthesis, and neural network verification.

Stanley Bak is an assistant professor in computer science at Stony Brook University in Stony Brook, NY, USA. He received the B.S. degree in computer science from Rensselaer Polytechnic Institute in 2007, and the M.S. degree and Ph.D. degree both in computer science from the University of Illinois at Urbana-Champaign in 2009 and 2013. His research interests include verification and testing methods for cyber-physical systems and neural networks.



Matthias Althoff is an associate professor in computer science at Technische Universität München, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from Technische Universität München, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

A.3 Fully-Automated Verification of Linear Systems Using Reachability Analysis with Support Functions

Summary For verification, one has to prove that there is no intersection between the reachable set and the unsafe set, which does not necessarily require an explicit representation of the reachable set. In this work, we significantly accelerate the solution to the verification task in Problem 1 for linear systems of the form (2.2) by reducing the reachable set computation to its relevant parts for the intersection check with the unsafe set.

Our main idea is to only examine the extent of the reachable set toward directions of interest, which are determined by the safety specifications. To this end, we combine the computational efficiency of support function evaluations for reachability analysis with automated methods for algorithm parameter tuning. For specifications represented by halfspaces, this allows us to quickly determine whether or not the reachable set intersects the unsafe set. In specific cases where the homogeneous and particular solutions can be pre-computed, the runtime complexity of the propagation is quadratic in the state dimension. Furthermore, we verify safety specifications represented by arbitrary convex unsafe sets via the Gilbert-Johnson-Keerthi algorithm, which is also based on efficient support function evaluations.

Our numerical evaluation demonstrates the immense impact of using support function reachability for verification: The proposed verification algorithm automatically verifies and falsifies benchmarks orders of magnitude faster than state-of-the-art approaches and even scales to large system dimensions that could not be analyzed before. Similarly to the verification algorithm in Appendix A.2, no manual parameter tuning is required to obtain a conclusive result on safety.

Author contributions M.W. initiated the idea of using adaptive parameter tuning with support function-based reachability for accelerated verification, implemented the algorithm, conducted parts of the numerical evaluation, and wrote most of the manuscript. N.K. extended the approach to verifying arbitrary convex sets, implemented the algorithm, and conducted parts of the numerical evaluation. S.B. suggested using the Gilbert-Johnson-Keerthi algorithm and provided feedback for improving the manuscript together with M.A.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

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

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

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

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

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

ABSTRACT

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

KEYWORDS

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

ACM Reference Format:
Mark Wetzlinger, Niklas Kochdumper, Stanley Bak, and Matthias Althoff. 2023. Fully-Automated Verification of Linear Systems Using Reachability Analysis with Support Functions. In *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3575870.3587121

1 INTRODUCTION

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

1.1 Related work

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

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

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

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

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

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

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

1.2 Contributions

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

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

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

2 PRELIMINARIES

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

2.1 Notation

We denote scalars and vectors by lowercase letters and matrices by uppercase letters. Given a vector $s \in \mathbb{R}^n$, $s_{(i)}$ represents the i-th entry; given a matrix $M \in \mathbb{R}^{m\times n}$, $M_{(i,\cdot)}$ and $M_{(\cdot,j)}$ refer to the i-th row and the j-th column, respectively. We use $\mathbf{0}$ and $\mathbf{1}$ for vectors and matrices of proper dimension containing only zeros or ones, as well as I_n to denote the identity matrix of dimension n. The concatenation of two matrices M_1, M_2 is written as $[M_1\ M_2]$ and $\text{diag}(s)$ returns a square matrix with the vector s on its main diagonal and zeros otherwise. Exact sets are denoted by standard calligraphic letters $\mathcal{S}$, outer-approximations by $\widehat{\mathcal{S}}$, and inner-approximations by $\check{\mathcal{S}}$. Moreover, we overload the vector notation s to also denote the set $\{s\}$ consisting only of the point s and we abbreviate $-I_n\,\mathcal{S}$ to $-\mathcal{S}$. Intervals are represented by $[a,b], a, b \in \mathbb{R}^n$, where $a \leq b$ holds element-wise. Interval matrices extend intervals by using matrices for the lower and upper bounds: $\boldsymbol{\mathcal{M}} = [\underline{M}, \overline{M}] = \{M \in \mathbb{R}^{m\times n} \mid \underline{M} \leq M \leq \overline{M}\}$, where the inequality is again evaluated element-wise. The operations $\mathrm{center}(\mathcal{S})$ and $\mathrm{box}(\mathcal{S})$ return the volumetric center and the tightest enclosing interval of $\mathcal{S}$, respectively. The sign function $\mathrm{sgn}(x)$ returns $-1, 0$, and 1 for the input ranges $x < 0, x = 0$, and $x > 0$, respectively. We use $\mathcal{O}(\cdot)$ to denote the big O notation.

2.2 Set representations and operations

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

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

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

Note that the support vector is not necessarily unique. □

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

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

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

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

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

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

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

Polytopes can be represented in halfspace or vertex representation:

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

$$\mathcal{P} := \left\{ x \in \mathbb{R}^n \;\middle|\; Hx \leq f \right\}.$$

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

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

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

Another common convex set representation are ellipsoids:

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

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

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

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

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

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

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

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

$$\rho(M\,\mathcal{S}_1, \ell) = \rho(\mathcal{S}_1, M^\top \ell), \tag{1}$$

$$\rho(\mathcal{S}_1 \oplus \mathcal{S}_2, \ell) = \rho(\mathcal{S}_1, \ell) + \rho(\mathcal{S}_2, \ell), \tag{2}$$

$$\rho(\mathrm{conv}(\mathcal{S}_1, \mathcal{S}_2), \ell) = \max\{\rho(\mathcal{S}_1, \ell), \rho(\mathcal{S}_2, \ell)\}. \tag{3}$$

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

$$\begin{aligned} M\,\mathcal{Z}_1 &= \langle M c_1, M G_1\rangle_Z, \\ \mathcal{Z}_1 \oplus \mathcal{Z}_2 &= \langle c_1 + c_2, [G_1\ G_2]\rangle_Z, \end{aligned}$$

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

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

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

3 PROBLEM STATEMENT

We consider linear time-invariant systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + p, \tag{5a}$$

$$y(t) = Cx(t) + Wv(t) + q, \tag{5b}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

4 REACHABILITY ANALYSIS

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

4.1 Outer-approximations of reachable sets

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

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

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

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

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

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

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

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

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

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

Let us now introduce the approximation models for computing outer-approximations of $\mathcal{H}(\tau_0)$ and $\mathcal{P}(\Delta t)$ in order to evaluate (8)-(9). We use the correction hull approximation model [1, Sec. 3.2], which represents a good trade-off between fast but inaccurate first-order models [30, Sec. 3], [40, Eq. (2)] and the accurate but slow forward-backward method [27, Sec. 3.1], [25, Sec. 2.4]. The affine time-interval solution is computed according to [1, Eq. (3.10)] by

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

which translates to the support function evaluation

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

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

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

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

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

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

using the interval matrices [1, Sec. 3.2]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2 Inner-approximations of reachable sets

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

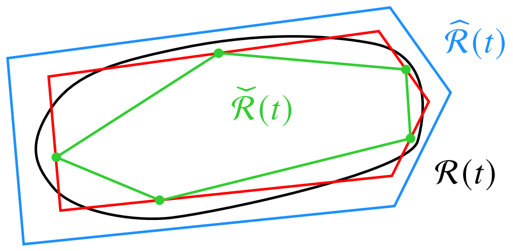


Figure 1: The halfspace polytope $\langle H, f\rangle_H$ (in red) with $H = [\ell_1 \dots \ell_w]^\top$, $f = [\rho(\breve{\mathcal{R}}(t), \ell_1) \dots \rho(\breve{\mathcal{R}}(t), \ell_w)]^\top$ constructed from lower bounds of the support function is in general *not* an inner-approximation of the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

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

4.3 Reachability algorithm

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

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

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

5 VERIFICATION

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

Algorithm 1 Reachability analysis using support functions

```
Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0$, input set $\mathcal{U}$, direction $\ell$, time horizon $t_{\mathrm{end}}$, time step size $\Delta t$, truncation order $\eta$
Ensure: Sequence of upper and lower bounds for the time-point solutions $\rho(\widehat{\mathcal{R}}(t_k), \ell), \rho(\breve{\mathcal{R}}(t_k), \ell)$ and time-interval solutions $\rho(\widehat{\mathcal{R}}(\tau_k), \ell), \rho(\breve{\mathcal{R}}(\tau_k), \ell)$ in direction $\ell$
 1: $t_0 \leftarrow 0$, $d_0 \leftarrow \ell$, $\rho(\mathcal{H}(t_0), \ell) \leftarrow \rho(\mathcal{X}^0, \ell)$
 2: $\tilde{u} \leftarrow B\,\mathrm{center}(\mathcal{U}) + p$, $\mathcal{U}_0 \leftarrow B(\mathcal{U} - \mathrm{center}(\mathcal{U}))$
 3: $\rho(\mathcal{P}^u(t_0), \ell) \leftarrow 0$, $\rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_0), \ell) \leftarrow 0, \rho(\breve{\mathcal{P}}^{\mathcal{U}}(t_0), \ell) \leftarrow 0$
 4: $\mathcal{P}^u(\Delta t) \leftarrow$ Eq. (12), $\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t) \leftarrow$ Eq. (16), $\breve{\mathcal{P}}^{\mathcal{U}}(\Delta t) \leftarrow$ Eq. (17)
 5: $C \leftarrow \mathcal{F}\mathcal{X}^0 \oplus \mathcal{G}\tilde{u}$                ▷ see Eq. (14)-(15)
 6: for $k \leftarrow 0$ to $\frac{t_{\mathrm{end}}}{\Delta t} - 1$ do
 7:     $t_{k+1} \leftarrow t_k + \Delta t$, $\tau_k \leftarrow [t_k, t_{k+1}]$
 8:     $d_{k+1} \leftarrow (e^{A\Delta t})^\top d_k$
 9:     $\mathcal{P}^u(t_{k+1}) \leftarrow e^{A\Delta t}\mathcal{P}^u(t_k) + \mathcal{P}^u(\Delta t)$
10:     $\rho(\mathcal{H}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{X}^0, d_{k+1}) + \rho(\mathcal{P}^u(t_{k+1}), \ell)$
        Upper bounds:
11:     $\rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell) \leftarrow \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_k), \ell) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(\Delta t), d_k)$
12:     $\rho(\widehat{\mathcal{R}}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{H}(t_{k+1}), \ell) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
13:     $\rho(\widehat{\mathcal{R}}(\tau_k), \ell) \leftarrow \max\{\rho(\mathcal{H}(t_k), \ell), \rho(\mathcal{H}(t_{k+1}), \ell)\}$
                    $+\rho(C, d_k) + \rho(\widehat{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
        Lower bounds:
14:     $\rho(\breve{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell) \leftarrow \rho(\breve{\mathcal{P}}^{\mathcal{U}}(t_k), \ell) + \rho(\breve{\mathcal{P}}^{\mathcal{U}}(\Delta t), d_k)$
15:     $\rho(\breve{\mathcal{R}}(t_{k+1}), \ell) \leftarrow \rho(\mathcal{H}(t_{k+1}), \ell) + \rho(\breve{\mathcal{P}}^{\mathcal{U}}(t_{k+1}), \ell)$
16:     $\rho(\breve{\mathcal{R}}(\tau_k), \ell) \leftarrow \max\{\rho(\mathcal{H}(t_k), \ell), \rho(\mathcal{H}(t_{k+1}), \ell)\},$
                    $+\rho(\breve{\mathcal{P}}^{\mathcal{U}}(t_k), \ell)$
17: end for
```

5.1 Verifying halfspace specifications

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

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

from the computation of $\mathcal{F}$ in (14). We increase η until the relative change between $\mathcal{T}^{(j)}$ and $\mathcal{T}^{(j+1)}$ in the Frobenius norm computed

Algorithm 2 Verification algorithm (halfspace specifications)

Require: Linear system $\dot{x} = Ax + Bu + p$, initial set $\mathcal{X}^0$, input set $\mathcal{U}$, time horizon t_{end}, safe set $\mathcal{K} = \langle H, f \rangle_H$
Ensure: Safe $(\forall t \in [0, t_{\text{end}}] : \mathcal{R}(t) \subseteq \mathcal{K})$ or unsafe $(\exists t \in [0, t_{\text{end}}] : \mathcal{R}(t) \nsubseteq \mathcal{K})$

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

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

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

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

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

5.2 Verifying arbitrary convex unsafe sets

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

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

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

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

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

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

In the verification setting, the set $\mathcal{S}_1$ is the reachable set $\mathcal{R}(t)$ and the set $\mathcal{S}_2$ is a convex unsafe set $\mathcal{L}$, so that $\mathcal{S} = \mathcal{R}(t) \oplus (-\mathcal{L})$. Our verification algorithm is summarized in Alg. 3: In each iteration we first compute upper and lower bounds of the support function in the current direction ℓ_i (Line 3), based on which we try to disprove that the sets intersect by checking if $\rho(\mathcal{S}, \ell_i) \leq \rho(\widehat{\mathcal{R}}(t), \ell_i) + \rho(\mathcal{L}, -\ell_i) < 0$ holds (Line 4). If true, we found a separating hyperplane between $\mathcal{R}(t)$ and $\mathcal{L}$, which proves that the system is safe. Otherwise, we use the corresponding support vector of the lower bound to check if we can prove that the sets intersect (Line 12). In case safety can neither

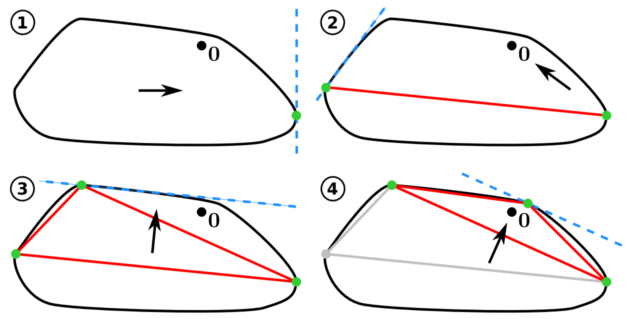


Figure 2: Schematic visualization of the iterations of the GJK algorithm for a set $\mathcal{S} = \mathcal{S}_1 \oplus (-\mathcal{S}_2)$.

be proven nor disproven based on the current direction ℓ_i, we choose a new direction $\ell_{i+1} = h$. If the polytope $\widecheck{\mathcal{P}}$ constructed from the support vectors is already non-degenerate ($i > n+1$, Line 18), the new direction is chosen as the normal vector of the polytope facet that is closest to the origin (Line 19), which can be determined by solving the quadratic program $\min_{x \in \widecheck{\mathcal{P}}} \|x\|_2^2$ to obtain the point in $\widecheck{\mathcal{P}}$ closest to the origin. Otherwise, we apply Gram-Schmidt orthogonalization to construct a direction orthogonal to the space spanned by the support vectors and pointing towards the origin (Line 23).

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

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

Improvements to accelerate the verification include parallelizing the computations for the different $(\mathcal{R}(\tau_k),\mathcal{L}_j)$-pairs, re-using the support functions computed for the current pair to disprove intersections for other pairs, initializing the first search direction based on the distance between the unsafe set and a simulated trajectory, and selecting the new search direction based on the polytope $\widehat{\mathcal{P}}$

Algorithm 3 Verification algorithm (arbitrary convex unsafe sets)

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

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

instead of $\widecheck{\mathcal{P}}$ once we know that the inner-approximation cannot be used to disprove safety (`falsifiable` = `false` in Alg. 3).

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

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

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

6 NUMERICAL EXAMPLES

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

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

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

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

6.1 ARCH benchmarks

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

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

6.2 Frequency and voltage control

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

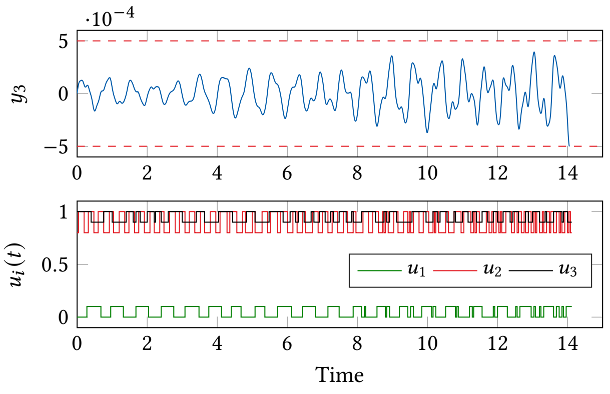


Figure 3: Specifications for ISSF01-ISU01 in red (dashed) and falsifying trajectory in blue (top); input trajectory $u(t) \in \mathbb{R}^3$ generating the falsifying trajectory (bottom).

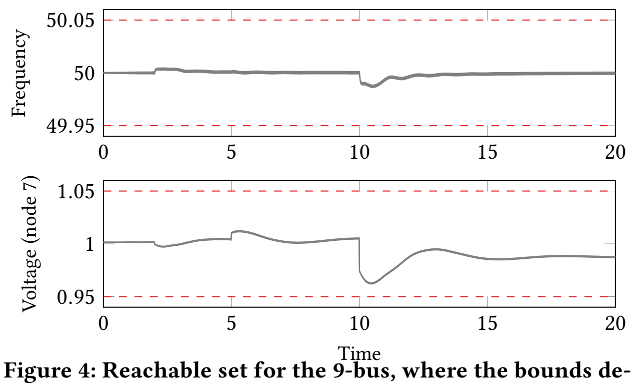


Figure 4: Reachable set for the 9-bus, where the bounds defined by the specification are depicted by the dashed red lines.

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

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

6.3 Quadcopter example

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

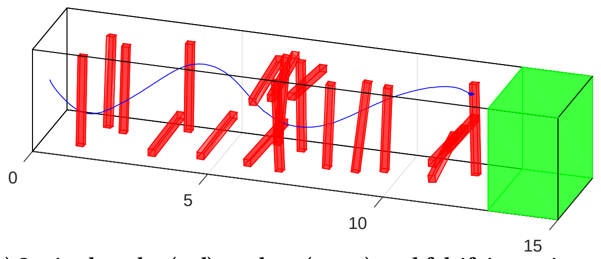
$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_{\text{ref}}(t) \end{bmatrix} = \begin{bmatrix} A + BK & -BK \\ \mathbf{0} & A \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\text{ref}}(t) \end{bmatrix} + \begin{bmatrix} B & B & BK \\ B & \mathbf{0} & \mathbf{0} \end{bmatrix} u(t) + \begin{bmatrix} p \\ p \end{bmatrix}$$

$$y(t) = \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} x(t) \\ x_{\text{ref}}(t) \end{bmatrix} + v(t)$$

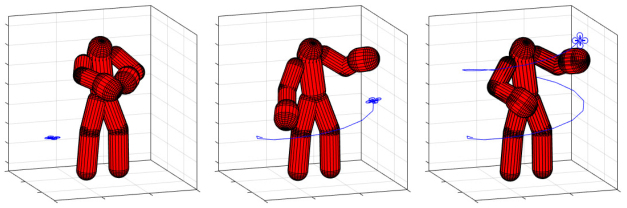
with $A = [\mathbf{0}\ [I_3\ \mathbf{0}]^\top]$, $B = [\mathbf{0}\ I_3]^\top$, $C = [I_3\ \mathbf{0}]$, $p = -9.81 \cdot \mathbf{1} \in \mathbb{R}^6$. The feedback matrix $K \in \mathbb{R}^{3\times 6}$ is determined by applying an LQR control approach with state weighting matrix $Q = I_6$ and input weighting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

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

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



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



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

Figure 5: Quadcoptor verification task.

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

7 CONCLUSION

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

ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial supports from the research training group ConVeY funded by the German Research Foundation under grant GRK 2428 and the project justITSELF funded by the European Research Council (ERC) under grant agreement No 817629. This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force or the United States Navy.

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[49] S. R. Schepp, J. Thumm, S. B. Liu, and M. Althoff. 2022. SaRA: A tool for safe human-robot coexistence and collaboration through reachability analysis. In

Proc. of the International Conference on Robotics and Automation. 4312–4317.

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

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

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

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

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

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

A.4 Backward Reachability Analysis of Perturbed Continuous-Time Linear Systems Using Set Propagation

Summary The verification tasks formulated in Problems 2 and 3 can be solved using minimal and maximal backward reachable sets as defined in Definition 6, respectively. Here, we must prove the existence of a control input trajectory for linear systems of the form (2.1) to steer the trajectory away or toward a target set.

This work proposes four novel algorithms for the computation of minimal and maximal time-point and time-interval backward reachable sets. We compute inner and outer approximations by combining various set representations—polytopes, zonotopes, and constrained zonotopes. Exploiting their respective advantages results in a polynomial runtime complexity in the state dimension for all proposed backward reachability algorithms. In some cases, the approximation error with respect to the exact backward reachable set can be reduced to zero along all or specific directions.

Our numerical evaluation shows similar tightness of our computed inner and outer approximations compared to the sets obtained using Hamilton-Jacobi reachability. Then, we compute the minimal and maximal backward reachable set for a 6D ground collision avoidance scenario and a 12D quadrotor system within seconds, thereby outspeeding Hamilton-Jacobi reachability by orders of magnitude. Finally, we demonstrate the scalability of our novel algorithms by analyzing a benchmark of over a thousand states for the first time.

Author contributions M.W. initiated the idea of using set propagation methods to compute backward reachable sets, designed all backward reachability algorithms, proved their soundness, implemented the code, and conducted the experiments. M.A. contributed ideas to the design of the backward reachability algorithms and provided feedback for improving the manuscript.

TUM Graduate School This publication is *not* a core publication, cf. Article 7, section 3 TUM Doctoral Regulations (PromO).

Backward Reachability Analysis of Perturbed Continuous-Time Linear Systems Using Set Propagation

Mark Wetzlinger and Matthias Althoff

***Abstract*— Backward reachability analysis computes the set of states that reach a target set under the competing influence of control input and disturbances. Depending on their interplay, the backward reachable set either represents all states that can be steered into the target set or all states that cannot avoid entering it—the corresponding solutions can be used for controller synthesis and safety verification, respectively. A popular technique for backward reachable set computation solves Hamilton-Jacobi-Isaacs equations, which scales exponentially with the state dimension due to gridding the state space. In this work, we instead use set propagation techniques to design backward reachability algorithms for linear time-invariant systems. Crucially, the proposed algorithms scale only polynomially with the state dimension. Our numerical examples demonstrate the tightness of the obtained backward reachable sets and show an overwhelming improvement of our proposed algorithms over state-of-the-art methods regarding scalability, as systems with well over a hundred states can now be analyzed.**

***Index Terms*— Formal verification, reachability analysis, linear systems, set-based computing.**

I. Introduction

The use of autonomous systems in safety-critical scenarios requires formal verification techniques to rigorously prove safe operation at all times in the presence of uncertainties. One popular method is backward reachability analysis, which computes the set of states that reach a given target set under a certain interplay between control input and disturbance. This so-called *two-player game* can be set up in two different ways, depending on the meaning of the target set.

If the target set represents an unsafe set, one utilizes the notion of *minimal* reachability [1, Sec. 4.2]: The minimal backward reachable set contains all states that cannot avoid entering the target set regardless of the chosen control input. Consequently, all states within the backward reachable set are deemed unsafe and thus should also be avoided. In case an exact solution cannot be obtained, we resort to computing outer approximations to maintain safety. A common example is obstacle avoidance: The target set represents the obstacle and the minimal backward reachable set contains all states from which one cannot avoid hitting the obstacle.

If the target set represents a goal set, the concept of *maximal* reachability [1, Sec. 4.1] is applicable: The maximal backward reachable set contains all states from which we can steer into the target set despite worst-case disturbances. Note that any initial state only requires to reach the target set by a single control input trajectory to become part of the backward reachable set. To ensure that all contained initial states can definitely be steered into the target set, we require an inner approximation if the exact solution cannot be computed. Maximal backward reachability is closely related to controller synthesis: The backward reachable set contains all states for which a controller exists such that the target set is reachable.

In this article, we compute minimal and maximal backward reachable sets for continuous-time linear time-invariant (LTI) systems. As there are many similar definitions of backward reachable sets as well as related concepts, we postpone the literature review to Section IV. This allows us to use the preliminary information from Sections II and III for a more concise overview. Our contributions are as follows:

- An inner and outer approximation for the time-point minimal backward reachable set (Section V-A).
- An outer approximation of the time-interval minimal backward reachable set (Section V-B).
- An inner and outer approximation for the time-point maximal backward reachable set (Section VI-A).
- An inner approximation for the time-interval maximal backward reachable set (Section VI-B).

Crucially, all proposed algorithms scale only polynomially with respect to the state dimension. Additionally, we discuss the approximation errors of each computed set. Our evaluation in Section VII is followed by closing remarks in Section VIII.

II. Preliminaries

We introduce some general notation, basics of set-based arithmetic, and fundamentals on forward reachability analysis required for the main body of this article.

A. Notation

The set of real numbers is denoted by $\mathbb{R}$, the set of natural numbers without zero is denoted by $\mathbb{N}$, and the subset $\{a, a+1, ..., b\} \subset \mathbb{N}$ for $0 < a < b$, is denoted by $\mathbb{N}_{[a,b]}$. We

This paragraph of the first footnote will contain the date on which you submitted your paper for review. This work was supported by the European Research Council (ERC) project justITSELF under grant agreement No 817629 and by the German Research Foundation (DFG) project ConVeY under grant number GRK 2428.

Mark Wetzlinger and Matthias Althoff are with the Department of Computer Science, Technical University of Munich, 85748 Garching, Germany (e-mail: {m.wetzlinger, althoff}@tum.de).

denote scalars and vectors by lowercase letters and matrices by uppercase letters. For a vector $s \in \mathbb{R}^n$, $\|s\|_p$ returns its p-norm and $s_{(i)}$ represents its ith entry; for a matrix $M \in \mathbb{R}^{m\times n}$, $M_{(i,\cdot)}$ refers to the ith row and $M_{(\cdot,j)}$ to the jth column. The operation $\operatorname{diag}(s)$ returns a square matrix with the vector s on its main diagonal. Horizontal concatenation of two properly-sized matrices M_1 and M_2 is denoted by $[M_1\ M_2]$ and the identity matrix of dimension n by I_n. Furthermore, we use $\mathbf{0}$ and $\mathbf{1}$ to represent vectors and matrices of proper dimension containing only zeros or ones. We denote exact sets by standard calligraphic letters $\mathcal{S}$, inner approximations by $\breve{\mathcal{S}}$, and outer approximations by $\widehat{\mathcal{S}}$. We write the set $\{-s|s \in \mathcal{S}\}$ as $-\mathcal{S}$ and represent the empty set by $\emptyset$. An interval is defined by $\mathcal{I} = [a, b] = \{x \in \mathbb{R}^n \,|\, a \leq x \leq b\}$, where the inequality is evaluated element-wise. Interval matrices extend intervals by using matrices as lower and upper limits and are denoted in bold calligraphic letters, e.g. $\boldsymbol{\mathcal{I}}$. The operations $\texttt{cen}(\mathcal{S})$ and $\texttt{box}(\mathcal{S})$ compute the volumetric center and tightest axis-aligned interval outer approximation of the set $\mathcal{S}$, respectively. Additionally, we introduce the hyperball $\mathcal{B}_\varepsilon = \{x \in \mathbb{R}^n \,|\, \|x\|_2 \leq \varepsilon\}$. Finally, we use $f(x) \in \mathcal{O}(g(x))$ to denote the big O notation.

B. Set-Based Arithmetic

Let us introduce the convex sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ as well as the matrix $M \in \mathbb{R}^{m\times n}$ to formally define a linear map, Minkowski sum, Minkowski difference, intersection, and convex hull:

$$M\mathcal{S}_1 := \{Ms_1 \,|\, s_1 \in \mathcal{S}_1\}, \tag{1}$$
$$\mathcal{S}_1 \oplus \mathcal{S}_2 := \{s_1 + s_2 \,|\, s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}, \tag{2}$$
$$\mathcal{S}_1 \ominus \mathcal{S}_2 := \{s \,|\, \{s\} \oplus \mathcal{S}_2 \subseteq \mathcal{S}_1\}, \tag{3}$$
$$\mathcal{S}_1 \cap \mathcal{S}_2 := \{s \,|\, s \in \mathcal{S}_1 \wedge s \in \mathcal{S}_2\}, \tag{4}$$
$$\begin{aligned}\texttt{conv}(\mathcal{S}_1, \mathcal{S}_2) := \{\lambda s_1 + (1-\lambda)s_2 \,|\\ s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2, \lambda \in [0,1]\}.\end{aligned} \tag{5}$$

Convex sets can be implicitly described by their support function:

Definition 1 (Support function [2, Sec. 2]). *For a compact set $\mathcal{S} \subset \mathbb{R}^n$ and a vector $\ell \in \mathbb{R}^n$, the support function $\rho : \mathbb{R}^n \to \mathbb{R}$ is*

$$\rho(\mathcal{S}, \ell) := \max_{s\in\mathcal{S}} \ell^\top s. \qquad \square$$

For support functions, we require the identities [3, Eq. (3)]

$$\rho(M\mathcal{S}, \ell) = \rho\big(\mathcal{S}, M^\top \ell\big)\,, \tag{6}$$
$$\rho(\mathcal{S}_1 \oplus \mathcal{S}_2, \ell) = \rho(\mathcal{S}_1, \ell) + \rho(\mathcal{S}_2, \ell)\,, \tag{7}$$
$$\rho(\mathcal{S}_1 \ominus \mathcal{S}_2, \ell) = \rho(\mathcal{S}_1, \ell) - \rho(\mathcal{S}_2, \ell)\,. \tag{8}$$

Next, we will introduce the three set representations required for our backward reachability algorithms. The runtime complexity of each operation is summarized in Table I, where we assume a runtime complexity of $\mathcal{O}(q^3)$ for the evaluation of a linear program with q variables according to [4]. We start with polytopes.

Definition 2 (Polytope [5, Sec. 1.1]). *A polytope $\mathcal{P} \subset \mathbb{R}^n$ in halfspace representation is described using $h \in \mathbb{N}$ linear inequalities defined by the matrix $H \in \mathbb{R}^{h\times n}$ and the vector $d \in \mathbb{R}^h$:*

$$\mathcal{P} := \big\{s \in \mathbb{R}^n \;\big|\; Hs \leq d\big\}.$$

We use the shorthand $\mathcal{P} = \langle H, d\rangle_H$. $\square$

A set $\mathcal{S} \subset \mathbb{R}^n$ can be enclosed with a polytope composed by a finite number $h \in \mathbb{N}$ of support function evaluations

$$\begin{aligned}&\mathcal{S} \subseteq \langle H, d\rangle_H,\\ &\text{where } \forall j \in \mathbb{N}_{[1,h]} : d_{(j)} = \rho\Big(\mathcal{S}, H_{(j,\cdot)}^\top\Big)\end{aligned} \tag{9}$$

and set equality holds if and only if the normal vectors of all faces of $\mathcal{S}$ are the row vectors of $H \in \mathbb{R}^{h\times n}$. Polytopes are closed under all aforementioned set operations (1)-(5) [6, Tab. 1]. We will, however, only make use of the linear map with an invertible matrix $M \in \mathbb{R}^{n\times n}$ and Minkowski difference [7, Thm. 2.2]:

$$M\mathcal{P} = \langle HM^{-1}, d\rangle_H, \tag{10}$$
$$\mathcal{P} \ominus \mathcal{S} = \langle H, \tilde{d}\rangle_H, \tag{11}$$
$$\text{where } \forall j \in \mathbb{N}_{[1,h]} : \tilde{d}_{(j)} = d_{(j)} - \rho\Big(\mathcal{S}, H_{(j,\cdot)}^\top\Big)\,.$$

An enclosing interval $\texttt{box}(\mathcal{P})$ can be computed by evaluating $2n$ support functions (linear programs) for all positive and negative axis-aligned directions. Next, we introduce zonotopes.

Definition 3 (Zonotope [8, Def. 1]). *Given a center $c \in \mathbb{R}^n$ and $\gamma \in \mathbb{N}$ generators stored as columns in the matrix $G \in \mathbb{R}^{n\times\gamma}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is*

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

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

For zonotopes, we require the linear map with a matrix $M \in \mathbb{R}^{m\times n}$ and Minkowski sum computed as [9, Eq. (2.1)]

$$M\mathcal{Z} = \langle Mc, MG\rangle_Z, \tag{12}$$
$$\mathcal{Z}_1 \oplus \mathcal{Z}_2 = \langle c_1 + c_2, [G_1\ G_2]\rangle_Z, \tag{13}$$

and the support function in a direction $\ell \in \mathbb{R}^n$ [10, Prop. 1]:

$$\rho(\mathcal{Z}, \ell) = \ell^\top c + \sum_{i=1}^{\gamma} |\ell^\top G_{(\cdot,i)}|. \tag{14}$$

The multiplication of an interval matrix $\boldsymbol{\mathcal{M}} = [L, U]$ with a zonotope $\mathcal{Z}$ can be tightly enclosed by [11, Thm. 4]

$$\boldsymbol{\mathcal{M}}\mathcal{Z} \subseteq \big\langle M_c c, \big[M_c G\ \operatorname{diag}\big(M_r \nu\big)\big]\big\rangle_Z, \tag{15}$$
$$M_c = \tfrac{1}{2}(L+U), M_r = \tfrac{1}{2}(U-L), \nu = |c| + \sum_{i=1}^{\gamma} |G_{(\cdot,i)}|.$$

Constrained zonotopes extend zonotopes by introducing equality constraints on the factors.

Definition 4 (Constrained zonotope [12, Def. 3]). *Given a vector $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n\times\gamma}$, a constraint matrix $K \in \mathbb{R}^{h\times\gamma}$, and a constraint offset $l \in \mathbb{R}^h$, a*

TABLE I

RUNTIME COMPLEXITY OF SET OPERATIONS FOR n-DIMENSIONAL SETS, WHERE THE POLYTOPE $\mathcal{P}$ HAS $h \in \mathbb{N}$ CONSTRAINTS, THE CONSTRAINED ZONOTOPE $\mathcal{CZ}$ AND THE ZONOTOPE $\mathcal{Z}$ HAVE $\gamma \in \mathbb{N}$ GENERATORS, AND $\ell \in \mathbb{R}^n$ IS A VECTOR.

Operation	Complexity	Operation	Complexity
$M\mathcal{P}$	$\mathcal{O}(hn^2)$	$\mathcal{Z}_1 \oplus \mathcal{Z}_2$	$\mathcal{O}(n)$
$M\mathcal{Z}$	$\mathcal{O}(n^2\gamma)$	$\mathcal{CZ}_1 \oplus \mathcal{CZ}_2$	$\mathcal{O}(n)$
$M\mathcal{CZ}$	$\mathcal{O}(n^2\gamma)$	$\mathcal{P} \ominus \mathcal{S}$	$\mathcal{O}(h\rho(\mathcal{S},\ell))$
$\boldsymbol{\mathcal{M}}\mathcal{Z}$	$\mathcal{O}(n^2\gamma)$	$\mathcal{S} \subseteq \mathcal{P}$	$\mathcal{O}(h\rho(\mathcal{S},\ell))$
$\boldsymbol{\mathcal{M}}\mathcal{CZ}$	$\mathcal{O}(n^2\gamma)$	$\mathcal{CZ} \cap \mathcal{P}$	$\mathcal{O}(hn^3)$
$\rho(\mathcal{P},\ell)$	$\mathcal{O}(n^3)$	$\texttt{conv}(\mathcal{CZ}_1,\mathcal{CZ}_2)$	$\mathcal{O}(n)$
$\rho(\mathcal{Z},\ell)$	$\mathcal{O}(n\gamma)$	$\texttt{box}(\mathcal{P})$	$\mathcal{O}(n^4)$
$\rho(\mathcal{CZ},\ell)$	$\mathcal{O}(\gamma^3)$	$\texttt{CZ}(\mathcal{P})$	$\mathcal{O}(n^4 + hn^3)$

constrained zonotope $\mathcal{CZ} \subset \mathbb{R}^n$ is

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

We use the shorthand $\mathcal{CZ} = \langle c, G, K, l\rangle_{CZ}$. □

For constrained zonotopes, we require the linear map with a matrix $M \in \mathbb{R}^{m\times n}$ and Minkowski sum [12, Prop. 1]:

$$M\mathcal{CZ} = \langle Mc, MG, K, l\rangle_{CZ},$$

$$\mathcal{CZ}_1 \oplus \mathcal{CZ}_2 = \left\langle c_1 + c_2, [G_1\ G_2], \begin{bmatrix} K_1 & \mathbf{0} \\ \mathbf{0} & K_2 \end{bmatrix}, \begin{bmatrix} l_1 & \mathbf{0} \\ \mathbf{0} & l_2 \end{bmatrix} \right\rangle_{CZ}.$$

The intersection of a constrained zonotope with a polytope $\mathcal{P} = \langle H, d\rangle_H$ can be computed via sequential intersection with each halfspace $\langle H_{(j,\cdot)}, d_{(j)}\rangle_H, j \in \mathbb{N}_{[1,h]}$ [13, Thm. 1]

$$\mathcal{CZ} \cap \langle H_{(j,\cdot)}, d_{(j)}\rangle_H = \left\langle c, [G\ \mathbf{0}], \begin{bmatrix} K & \mathbf{0} \\ H_{(j,\cdot)}G & \frac{1}{2}(d_{(j)} - o) \end{bmatrix}, \begin{bmatrix} l \\ \frac{1}{2}(d+o) - H_{(j,\cdot)}c \end{bmatrix} \right\rangle_{CZ}, \tag{16}$$

where $o = -\rho\Big(\mathcal{CZ}, -H_{(j,\cdot)}^\top\Big)$

evaluates the support function of the constrained zonotope using linear programming. The exact conversion of a polytope $\mathcal{P} = \langle H, d\rangle_H$ to a constrained zonotope [12, Thm. 1] is computed by

$$\begin{aligned} &\langle c, G, K, l\rangle_{CZ} = \texttt{CZ}(\mathcal{P}), \\ &\text{where } \langle c, G\rangle_Z \supseteq \mathcal{P} \text{ and } K, l \text{ from } \langle c, G\rangle_Z \cap \mathcal{P}, \end{aligned} \tag{17}$$

that is, we first enclose the polytope $\mathcal{P}$ by a zonotope $\langle c, G\rangle_Z$, e.g., $\texttt{box}(\mathcal{P})$, and then intersect this zonotope with each halfspace of $\mathcal{P}$ as in (16) to obtain the constraint matrix K and constraint offset l. The convex hull can be computed according to [13, Thm. 5] and the multiplication with an interval matrix $\boldsymbol{\mathcal{M}}\mathcal{CZ}$ follows from (15). All introduced set operations scale polynomially in the set dimension, which will enable our backward reachability algorithms to run in polynomial time.

C. Forward Reachable Set Computation

Our backward reachability algorithms are partly based on established knowledge from forward reachability analysis:

Definition 5 (Forward reachable set)**.** *For an LTI system of the form $\dot{x}(t) = Ax(t) + u(t)$, let the solution trajectory at time $t \in \mathbb{R}$ for an initial state $x_0 \in \mathbb{R}^n$ and an input trajectory $u(\cdot) : \mathbb{R} \to \mathbb{R}^n$ be denoted by $\xi(t; x_0, u(\cdot))$. Given an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ and an input set $\mathcal{U} \subset \mathbb{R}^n$, the forward reachable set at time $t \geq 0$ is*

$$\mathcal{R}(t) := \{\xi(t; x_0, u(\cdot)) \,|\, x_0 \in \mathcal{X}_0, \forall\theta \in [0,t]\colon u(\theta) \in \mathcal{U}\}. \quad \square$$

Next, we briefly recall the computation of the homogeneous time-interval solution and the particular solution, which can be computed separately due to the well-known superposition principle of linear systems.

1) Homogeneous time-interval solution: Given two successive homogeneous time-point solutions $\mathcal{H}(t_k), \mathcal{H}(t_{k+1}) \subset \mathbb{R}^n$, we enclose all trajectories over the interval $\tau_k = [t_k, t_{k+1}]$ of length $\Delta t = t_{k+1} - t_k \geq 0$ by [9, Sec. 3.2]

$$\mathcal{H}(\tau_k) \subseteq \texttt{conv}(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})) \oplus \boldsymbol{\mathcal{F}}\mathcal{H}(t_k), \tag{18}$$

where the interval matrix $\boldsymbol{\mathcal{F}}$ is [9, Prop. 3.1]

$$\boldsymbol{\mathcal{F}} = \bigoplus_{i=2}^{\eta} \left[(i^{\frac{-i}{i-1}} - i^{\frac{-1}{i-1}})\Delta t^i, 0\right] \frac{A^i}{i!} \oplus \boldsymbol{\mathcal{E}}, \tag{19}$$

where $A \in \mathbb{R}^{n\times n}$ is the system matrix in Definition 5 and the interval matrix $\boldsymbol{\mathcal{E}}$ is the remainder of the exponential matrix [9, Eq. (3.2)]:

$$\begin{aligned} \boldsymbol{\mathcal{E}} &= [-E(\Delta t, \eta), E(\Delta t, \eta)], \\ E(\Delta t, \eta) &= e^{|A|\Delta t} - \sum_{i=0}^{\eta} \frac{\big(|A|\Delta t\big)^i}{i!}. \end{aligned} \tag{20}$$

An inner approximation of the homogeneous time-interval solution can be computed by [14, Prop. 1]

$$\mathcal{H}(\tau_k) \supseteq \texttt{conv}(\mathcal{H}(t_k), \mathcal{H}(t_{k+1})) \ominus \boldsymbol{\mathcal{F}}\mathcal{H}(t_k) \ominus \mathcal{B}_\mu, \tag{21}$$

$$\text{where } \mu = \sqrt{\gamma}\, \|(e^{A\Delta t} - I_n)G\|_2 \tag{22}$$

uses the generator matrix $G \in \mathbb{R}^{n\times\gamma}$ of $\mathcal{H}(t_k) = \langle c, G\rangle_Z$.

2) Particular solution: Let the particular solution for time-varying inputs within a set $\mathcal{S}$ be denoted by $\mathcal{Z}_\mathcal{S} \subset \mathbb{R}^n$. We compute an outer approximation $\widehat{\mathcal{Z}}_\mathcal{S}$ and an inner approximation $\check{\mathcal{Z}}_\mathcal{S}$ as [9, Eq. (3.7)]

$$\mathcal{Z}_\mathcal{S} \subseteq \widehat{\mathcal{Z}}_\mathcal{S}(\Delta t) := \bigoplus_{i=1}^{\eta} \frac{A^i \Delta t^{i+1}}{(i+1)!}\, \mathcal{S} \oplus \boldsymbol{\mathcal{E}}\Delta t\, \mathcal{S}, \tag{23}$$

$$\mathcal{Z}_\mathcal{S} \supseteq \check{\mathcal{Z}}_\mathcal{S}(\Delta t) := A^{-1}(e^{A\Delta t} - I_n)\, \mathcal{S}. \tag{24}$$

The value for $\eta \in \mathbb{N}$ in (20) and (23) can be automatically determined as shown in [14]. For (24), we can integrate the term A^{-1} in the power series of the exponential matrix $e^{A\Delta t}$ if the matrix A is not invertible. The particular solution can be propagated by [15, Eq. (6)]

$$\mathcal{Z}_\mathcal{S}(t_{k+1}) = \mathcal{Z}_\mathcal{S}(t_k) \oplus e^{At_k}\mathcal{Z}_\mathcal{S}(\Delta t), \tag{25}$$

which avoids the wrapping effect [16]. The proposition below examines the approximation error of the particular solutions:

Proposition 1 (Convergence of particular solution [14])**.** *The outer approximation $\widehat{\mathcal{Z}}_\mathcal{S}(t)$ and inner approximation $\check{\mathcal{Z}}_\mathcal{S}(t)$*

of the particular solution propagated by (25) *converge to the exact particular solution*

$$\mathcal{Z}_{\mathcal{S}}(t) := \left\{ \int_0^t e^{A(t-\theta)} s(\theta)\, \mathrm{d}\theta \,\middle|\, s(\theta) \in \mathcal{S} \right\}$$

in the limit $\Delta t \to 0$ *used in* (23) *and* (24)*, respectively.*

Proof. See Appendix. □

For a piecewise constant trajectory $s \in \mathbb{R}^{1\times\omega}$ over $\omega \in \mathbb{N}$ steps

$$s = \left[s(t_0)\ s(t_1)\ \dots\ s(t_{\omega-1}) \right],$$

the particular solution $\mathcal{Z}_s(\tau_k) \subset \mathbb{R}^n$ over a time interval τ_k can be outer approximated by [9, Prop. 3.2]

$$\widehat{\mathcal{Z}}_s(\tau_k) = \bigoplus_{j=0}^{k-1} e^{At_{k-1-j}} \left(A^{-1}(e^{A\Delta t} - I_n)\, s(t_j) \right) \oplus \boldsymbol{\mathcal{G}}\{s(t_k)\}, \tag{26}$$

where the interval matrix $\boldsymbol{\mathcal{G}}$ is [9, Eq. (3.9)]

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

with $\boldsymbol{\mathcal{E}}$ as in (20). To enclose the particular solution $\mathcal{Z}_{\mathcal{S}}(\tau_k) \subset \mathbb{R}^n$ over a time interval τ_k, we first split the set $\mathcal{S}$ into two parts [9, Sec. 3.2.2]: $\mathcal{S} = \mathcal{S}_0 \oplus \{s\}$ with $s = \mathtt{cen}(\mathcal{S})$. Since $\{\mathbf{0}\} \in \mathcal{S}_0$, we have $\widehat{\mathcal{Z}}_{\mathcal{S}_0}(\tau_k) \subseteq \widehat{\mathcal{Z}}_{\mathcal{S}_0}(t_{k+1})$ and thus

$$\mathcal{Z}_{\mathcal{S}}(\tau_k) \subseteq \widehat{\mathcal{Z}}_{\mathcal{S}}(\tau_k) = \widehat{\mathcal{Z}}_{\mathcal{S}_0}(t_{k+1}) \oplus \widehat{\mathcal{Z}}_s(\tau_k), \tag{27}$$

where the set $\widehat{\mathcal{Z}}_{\mathcal{S}_0}(t_{k+1})$ is propagated using (25), and the set $\widehat{\mathcal{Z}}_s(\tau_k)$ is computed using (26).

III. Problem Statement

We consider LTI systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + Ew(t), \tag{28}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $A \in \mathbb{R}^{n\times n}$ is the state matrix, $B \in \mathbb{R}^{n\times m}$ is the input matrix, and $E \in \mathbb{R}^{n\times r}$ is the disturbance matrix. The control input $u(t) \in \mathbb{R}^m$ and the disturbance $w(t) \in \mathbb{R}^r$ are bounded by the sets $\mathcal{U} \subset \mathbb{R}^m$ and $\mathcal{W} \subset \mathbb{R}^r$, respectively, which we assume to be zonotopes. We use $\mathbb{U}$ to denote the set of all input trajectories $u(\cdot)$ for which $\forall t \in [0, t_{\text{end}}] : u(t) \in \mathcal{U}$ holds and analogously $\mathbb{W}$ for the set of all disturbances trajectories $w(\cdot)$. A solution to (28) at time t starting from the initial state $x_0 \in \mathbb{R}^n$ using an input trajectory $u(\cdot) \in \mathbb{U}$ and a disturbance trajectory $w(\cdot) \in \mathbb{W}$ is written as $\xi(t; x_0, u(\cdot), w(\cdot))$. We will denote the particular solutions (23)-(25) due to the sets $B\mathcal{U}$ and $E\mathcal{W}$ at time t by $\mathcal{Z}_{\mathcal{U}}(t)$ and $\mathcal{Z}_{\mathcal{W}}(t)$, respectively.

In general, backward reachability analysis aims to compute the set of states that reach a target set $\mathcal{X}_{\text{end}} \subset \mathbb{R}^n$ after a certain elapsed time t (time-point backward reachable set) or at any time within the interval $\tau = [t_0, t_{\text{end}}]$ (time-interval backward reachable set). We assume the target set $\mathcal{X}_{\text{end}} \subset \mathbb{R}^n$ to be represented as a polytope. Let us first define the *minimal* backward reachable set, where the target set is composed of unsafe states:

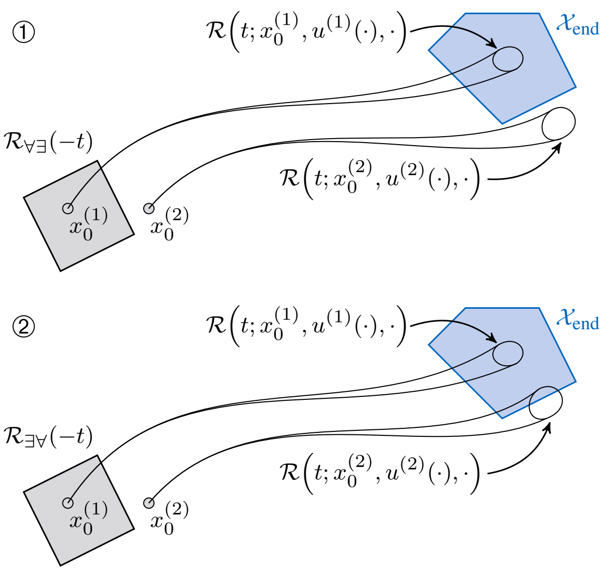

①
$\mathcal{R}\left(t; x_0^{(1)}, u^{(1)}(\cdot), \cdot\right)$
$\mathcal{X}_{\text{end}}$
$\mathcal{R}_{\forall\exists}(-t)$
$\mathcal{R}\left(t; x_0^{(2)}, u^{(2)}(\cdot), \cdot\right)$
$x_0^{(1)}$
$x_0^{(2)}$
②
$\mathcal{R}\left(t; x_0^{(1)}, u^{(1)}(\cdot), \cdot\right)$
$\mathcal{X}_{\text{end}}$
$\mathcal{R}_{\exists\forall}(-t)$
$\mathcal{R}\left(t; x_0^{(2)}, u^{(2)}(\cdot), \cdot\right)$
$x_0^{(1)}$
$x_0^{(2)}$

Fig. 1. Target set $\boldsymbol{\mathcal{X}}_{\text{end}}$ with ① minimal backward reachable set $\boldsymbol{\mathcal{R}}_{\forall\exists}(-t)$ and ② maximal backward reachable set $\boldsymbol{\mathcal{R}}_{\exists\forall}(-t)$ as well as initial states $\boldsymbol{x}_0$ with corresponding forward reachable sets $\boldsymbol{\mathcal{R}}(t)$ for different input signals $\boldsymbol{u}(\cdot)$ and disturbance signals $\boldsymbol{w}(\cdot)$.

Definition 6 (Minimal backward reachable set)**.** *The time-point minimal backward reachable set [17, Def. 2]*

$$\mathcal{R}_{\forall\exists}(-t) := \left\{ x_0 \in \mathbb{R}^n \,\middle|\, \forall u(\cdot) \in \mathbb{U}\ \exists w(\cdot) \in \mathbb{W} : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{end} \right\} \tag{29}$$

contains all states, where for all input trajectories $u(\cdot) \in \mathbb{U}$ there is at least one disturbance trajectory $w(\cdot) \in \mathbb{W}$ so that the state trajectory will end up in the target set $\mathcal{X}_{end}$ after time t. The time-interval minimal backward reachable set [17, Def. 4]

$$\mathcal{R}_{\forall\exists}(-\tau) := \left\{ x_0 \in \mathbb{R}^n \,\middle|\, \forall u(\cdot) \in \mathbb{U}\ \exists w(\cdot) \in \mathbb{W}\ \exists t \in \tau : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{end} \right\} \tag{30}$$

requires the state to pass through $\mathcal{X}_{end}$ anytime in the time interval τ. □

Case ① in Figure 1 illustrates the time-point set (29): For all states within the minimal backward reachable set $\mathcal{R}_{\forall\exists}(-t)$, such as $x_0^{(1)}$, the target set $\mathcal{X}_{\text{end}}$ is unavoidable regardless of the input signal $u^{(1)}(\cdot)$. For any initial state outside $\mathcal{R}_{\forall\exists}(-t)$ like $x_0^{(2)}$, there is at least one input signal $u^{(2)}(\cdot)$, for which there is no disturbance signal such that the corresponding forward reachable set intersects $\mathcal{X}_{\text{end}}$.

In the following definition of the *maximal* backward reachable set, the target set represents a goal set into which we want to steer the state despite worst-case disturbances.

Definition 7 (Maximal backward reachable set)**.** *The time-point maximal backward reachable set [17, Def. 1]*

$$\mathcal{R}_{\exists\forall}(-t) := \left\{ x_0 \in \mathbb{R}^n \,\middle|\, \exists u(\cdot) \in \mathbb{U}\ \forall w(\cdot) \in \mathbb{W} : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{end} \right\} \tag{31}$$

contains all states, where one input trajectory $u(\cdot)$ can steer the state trajectory into the target set $\mathcal{X}_{end}$ for all potential *disturbances $w(\cdot)$. The time-interval maximal backward reachable set [17, Def. 3]*

$$\mathcal{R}_{\exists\forall}(-\tau) := \big\{x_0 \in \mathbb{R}^n \,\big|\, \exists u(\cdot) \in \mathbb{U}\ \forall w(\cdot) \in \mathbb{W}\ \exists t \in \tau : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{end}\big\} \quad (32)$$

requires the state to pass through $\mathcal{X}_{end}$ anytime in the time interval τ. □

Case ② in Figure 1 illustrates the time-point set (31): For all states within the maximal backward reachable set $\mathcal{R}_{\exists\forall}(-t)$, such as $x_0^{(1)}$, there exists an input signal $u^{(1)}(\cdot)$ reaching the target set regardless of the disturbance. In contrast, the forward reachable set of an initial state outside of $\mathcal{R}_{\exists\forall}(-t)$ like $x_0^{(2)}$ is not fully contained in the target set for any input signal $u^{(2)}(\cdot)$.

Let us briefly highlight an important consequence of the two-player game notion in backward reachability analysis:

Proposition 2 (Union [1, Prop. 2])**.** *The union of time-point solutions is a subset of the corresponding time-interval solution, i.e.,*

$$\bigcup_{t\in\tau} \mathcal{R}_{\forall\exists}(-t) \subseteq \mathcal{R}_{\forall\exists}(-\tau), \ \bigcup_{t\in\tau} \mathcal{R}_{\exists\forall}(-t) \subseteq \mathcal{R}_{\exists\forall}(-\tau).$$

Proof. The reason is the order of quantifiers [1, Prop. 2]. □

For the runtime complexity analysis of our proposed algorithms in Sections V and VI, we assume the following:

Assumption 1 (Fixed parameters)**.** *The truncation order η in* (23)*, the number of propagation steps ω, and the number of halfspaces h constraining the target set $\mathcal{X}_{end}$ are fixed.* □

In the next section, we review the state of the art in backward reachability analysis.

IV. Related Work

There exists a wide range of different yet similar definitions labeled *backward reachable set*. The following literature review discusses the various types in order of increasing complexity. We will include approaches in discrete and continuous time as well as for linear and nonlinear dynamics, where uniqueness of solution trajectories and sufficient differentiability are assumed.

A. Autonomous Systems

Let us briefly consider autonomous systems $\dot{x} = f(x)$, where the backward reachable set is equal to the forward reachable set for the time-inverted dynamics $\dot{x} = -f(x)$ using the target set $\mathcal{X}_{\text{end}}$ as the initial set. If the target set represents an unsafe set, one can use established forward reachability algorithms for computing outer approximations of linear systems [10], [18] and nonlinear systems [19], [20]. If on the other hand, the target set is a goal set, we require to compute an inner approximation, for which there also exist many methods for linear systems [3], [18] as well as nonlinear systems [21]–[25]. Since this very special case is not the focus of our work, we do not discuss the different approaches here and instead refer the interested reader to the overviews in the cited literature.

B. Hamilton-Jacobi Reachability

A well-established framework for computing minimal and maximal reachable sets is commonly referred to as *Hamilton-Jacobi (HJ) reachability*: It is based on the proof that the reachable set of a continuous-time dynamical system is the zero sublevel set of the Hamilton-Jacobi-Isaacs partial differential equation (PDE) [26, Thm. 2]. The value function of the sublevel set is evaluated over a gridded state space, thus the computation scales exponentially with the system dimension [27]. Still, the framework is very versatile, covering the general case of nonlinear dynamics with all variations of competing inputs and disturbances as presented in our subsequent overview of minimal and maximal reachability.

C. Minimal Reachability

1) Unperturbed Case: The unperturbed minimal backward reachable set is defined by

$$\mathcal{R}_{\forall}(-\tau) := \big\{x_0 \in \mathbb{R}^n \,\big|\, \forall u(\cdot) \in \mathbb{U}\ \exists t \in \tau : \xi(t; x_0, u(\cdot), \mathbf{0}) \in \mathcal{X}_{\text{end}}\big\}. \quad (33)$$

The scalability issue of HJ reachability has first been tackled for time-point solutions by decomposing the dynamics into subsystems and reconstructing the full solution thereafter [28], which was later generalized to time-interval solutions [29]. However, these approaches did not provide rigorous results for cases with conflicting controls between subspaces, which was later addressed [30].

2) Perturbed Case: An approach for decoupled dynamics has been presented in [31]. In the context of systems coupled by multi-agent interaction, the decoupled computation has been augmented by a higher-level control using mixed integer programming [32]. Moreover, a deep neural network has been trained to output the value function describing the reachable set, which improves the scalability but invalidates all safety guarantees [33]. Other ideas to improve performance include warm-starting and adaptive grid sampling [34].

D. Maximal Reachability

1) Unperturbed Case: The unperturbed maximal backward reachable set is defined by

$$\mathcal{R}_{\exists}(-\tau) := \big\{x_0 \in \mathbb{R}^n \,\big|\, \exists u(\cdot) \in \mathbb{U}\ \exists t \in \tau : \xi(t; x_0, u(\cdot), \mathbf{0}) \in \mathcal{X}_{\text{end}}\big\}. \quad (34)$$

This is equivalent to the forward reachable set for the time-inverted dynamics $\dot{x} = -f(x)$ using the target set as the start set [35, Lemma 2]. As a consequence, all algorithms computing inner approximations for dynamical systems with inputs are applicable, including [23] and [36, Sec. 4.3.3] reviewed in Section IV-A. Another approach rescales an initial guess until the forward reachable set is contained in the target set [37]. For polynomial systems, sum-of-squares (SOS) optimization can be used to compute polynomial lower (and upper) bounds on the reachable set [35].

2) Perturbed Case: Algorithms using set propagation exist for both linear and nonlinear discrete-time systems, with ellipsoids [38] or zonotopes [39], [40] as a set representation.

The original HJ reachability was introduced in [26] for the set $\mathcal{R}_{\exists\forall}(-t)$, with extensions such as decoupling approaches [17] attempting to alleviate the computational burden. For dissipative control-affine nonlinear systems, one can compute the backward reachable sets via SOS programming [41], followed by synthesizing a controller to steer the states into the target set. This algorithm has been improved by merging both steps into one, including accommodation of control saturation [42]. An extension covers a more general class of perturbations represented by integral quadratic constraints [43].

An extended definition requires the trajectories to remain within a state constraint set $\bar{\mathcal{X}} \subset \mathbb{R}^n$ at all times:

$$\begin{aligned}\mathcal{R}_{\exists\forall,\bar{\mathcal{X}}}(-\tau) := \big\{x_0 \in \mathbb{R}^n \,\big|\, &\exists u(\cdot) \in \mathbb{U}\ \forall w(\cdot) \in \mathbb{W} \\ &\exists t \in \tau : \xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}, \\ &\forall t' \in [0, t_{\text{end}}] : \xi(t'; x_0, u(\cdot), w(\cdot)) \in \bar{\mathcal{X}}\big\},\end{aligned}$$

where t_{end} is the upper bound of the time interval τ. HJ reachability supports this definition [44]—including a time-varying state constraint set [45]—as do SOS approaches by solving a single semi-definite program [46].

E. Related Concepts

A related concept is the viability or discriminating kernel:

Definition 8 (Viability/Discriminating kernel [47, Def. 6], [48, Def. 2])**.** *The* discriminating *kernel of a set $\mathcal{K} \subset \mathbb{R}^n$ is*

$$\begin{aligned}\mathcal{D}(\tau, \mathcal{K}) := \big\{x_0 \in \mathcal{K} \,\big|\, &\forall w(\cdot) \in \mathbb{W}\ \exists u(\cdot) \in \mathbb{U}\ \forall t \in \tau : \\ &\xi(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{K}\big\}.\end{aligned}$$

It contains all initial states in $\mathcal{K}$, where for all potential disturbances $w(\cdot)$ there exists an input trajectory $u(\cdot)$ to keep the state in $\mathcal{K}$ over the time interval τ. Omitting the disturbance $w(\cdot)$ yields the viability *kernel $\mathcal{V}(\mathcal{K})$.* □

Inner approximations of the viability kernel for linear systems can be computed using ellipsoids [47], [49] or polytopes [50] as a set representation. The ellipsoidal methods have later been extended to computing the discriminating kernel in [48].

Another perspective is the computation of *forward* minimal/maximal reachable sets, where the quantifiers are equal to Definitions 6 and 7, but the start set is given instead of the target set. To this end, Kaucher arithmetic has been applied [51] as well as contraction of an outer approximation computed using Taylor models [23].

Our overview of the related literature shows that Definitions 6 and 7 represent general cases for minimal and maximal backward reachable sets. In the next sections, we present the first propagation-based approach to compute inner and outer approximations of these sets for systems of the form in (28).

V. Minimal Backward Reachability Analysis

In this section, we compute inner and outer approximations of the time-point minimal backward reachable set $\mathcal{R}_{\forall\exists}(-t)$ (29) in Section V-A as well as an outer approximation of the time-interval minimal backward reachable set $\mathcal{R}_{\forall\exists}(-\tau)$ (30) in Section V-B. We will also show that the runtime complexity of the presented algorithms is polynomial in the state dimension n and discuss the approximation errors of the computed sets. Finally, we briefly highlight simplifications for the unperturbed cases $\mathcal{R}_{\forall}(-t)$ and $\mathcal{R}_{\forall}(-\tau)$ defined in (33).

A. Time-Point Solution

We base our computations of the time-point solution $\mathcal{R}_{\forall\exists}(-t)$ on the following proposition:

Proposition 3 (Minimal time-point backward reachable set)**.** *The backward reachable set $\mathcal{R}_{\forall\exists}(-t)$ defined in* (29) *can be computed by*

$$\mathcal{R}_{\forall\exists}(-t) = e^{-At}\big((\mathcal{X}_{end} \oplus -\mathcal{Z}_{\mathcal{W}}(t)) \ominus \mathcal{Z}_{\mathcal{U}}(t)\big). \tag{35}$$

Proof. This is a continuization of the discrete-time case proven in [38, Thm. 2.4]. □

Note that the above proposition holds independently of the chosen set representations. Next, we show how to compute approximations in polynomial time under the assumption that the target set $\mathcal{X}_{\text{end}}$ is given as a polytope, while the particular solutions $\mathcal{Z}_{\mathcal{W}}(t)$ and $\mathcal{Z}_{\mathcal{U}}(t)$ are represented by zonotopes.

1) Outer Approximation: The main difficulty in evaluating (35) is the Minkowski sum of a polytope in halfspace representation and a zonotope, for which there exists no known polynomial-time algorithm[1]. We overestimate the influence of the disturbance by $\widehat{\mathcal{Z}}_{\mathcal{W}}(t) \supseteq \mathcal{Z}_{\mathcal{W}}(t)$ using (23) and underestimate the influence of the control input by $\check{\mathcal{Z}}_{\mathcal{U}}(t) \subseteq \mathcal{Z}_{\mathcal{U}}(t)$ using (24). The following proposition provides a scalable yet outer approximative evaluation for the Minkowski sum of a polytope in halfspace representation and a zonotope:

Proposition 4. *(Outer approximation of Minkowski sum) Given a polytope $\mathcal{P} = \langle H, d\rangle_H \subset \mathbb{R}^n$ with h constraints and a zonotope $\mathcal{Z} \subset \mathbb{R}^n$, their Minkowski sum can be enclosed by*

$$\begin{aligned}&\mathcal{P} \oplus \mathcal{Z} \subseteq \mathcal{P} \,\widehat{\oplus}\, \mathcal{Z} := \langle H, d + \tilde{d}\rangle_H, \\ &\forall j \in \mathbb{N}_{[1,h]} : \tilde{d}_{(j)} = \rho\Big(\mathcal{Z}, H_{(j,\cdot)}^\top\Big),\end{aligned} \tag{36}$$

where we introduce the operator $\widehat{\oplus}$ to distinguish this operation from the exact Minkowski sum. The runtime complexity is $\mathcal{O}(hn\gamma)$.

Proof. See Appendix. □

The resulting set in (36) can be further tightened by additional support function evaluations[2]. Using Proposition 4, we obtain an outer approximation of (35) by

$$\begin{aligned}\mathcal{R}_{\forall\exists}(-t) &\overset{(35)}{=} e^{-At}\big((\mathcal{X}_{\text{end}} \oplus -\mathcal{Z}_{\mathcal{W}}(t)) \ominus \mathcal{Z}_{\mathcal{U}}(t)\big) \\ &\overset{(23),(24)}{\subseteq} e^{-At}\big((\mathcal{X}_{\text{end}} \oplus -\widehat{\mathcal{Z}}_{\mathcal{W}}(t)) \ominus \check{\mathcal{Z}}_{\mathcal{U}}(t)\big) \\ &\overset{\text{Proposition 4}}{\subseteq} e^{-At}\big((\mathcal{X}_{\text{end}} \,\widehat{\oplus}\, -\widehat{\mathcal{Z}}_{\mathcal{W}}(t)) \ominus \check{\mathcal{Z}}_{\mathcal{U}}(t)\big) =: \widehat{\mathcal{R}}_{\forall\exists}(-t),\end{aligned} \tag{37}$$

[1]Other polytope representations, e.g., the Z-representation [36, Sec. 3.3], allow for computing the Minkowski sum with a zonotope in polynomial time, but we require the halfspace representation for the subsequent computation of the Minkowski difference.

[2]In fact, incorporating all infinite directions $\ell \in \mathbb{R}^n$ with $\|\ell\|_2 = 1$ would return the exact result $\mathcal{P} \oplus \mathcal{Z}$ since any compact convex set is uniquely determined by the intersection of the support functions in all directions [2].

resulting in a polytope representing $\widehat{\mathcal{R}}_{\forall\exists}(-t)$.

2) Inner Approximation: We now underestimate the influence of the disturbance by $\check{\mathcal{Z}}_{\mathcal{W}}(t) \subseteq \mathcal{Z}_{\mathcal{W}}(t)$ and overestimate the influence of the control input by $\widehat{\mathcal{Z}}_{\mathcal{U}}(t) \supseteq \mathcal{Z}_{\mathcal{U}}(t)$. Using the following re-ordering relation for the compact, convex, nonempty sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \subset \mathbb{R}^n$ [39, Lemma 1(i)]

$$(\mathcal{S}_1 \oplus \mathcal{S}_2) \ominus \mathcal{S}_3 \supseteq (\mathcal{S}_1 \ominus \mathcal{S}_3) \oplus \mathcal{S}_2, \tag{38}$$

we can inner approximate (35) by

$$\begin{aligned} \mathcal{R}_{\forall\exists}(-t) &\overset{(35)}{=} e^{-At}\Big((\mathcal{X}_{\text{end}} \oplus -\mathcal{Z}_{\mathcal{W}}(t)) \ominus \mathcal{Z}_{\mathcal{U}}(t)\Big) \\ &\overset{(23),(24)}{\supseteq} e^{-At}\Big((\mathcal{X}_{\text{end}} \oplus -\check{\mathcal{Z}}_{\mathcal{W}}(t)) \ominus \widehat{\mathcal{Z}}_{\mathcal{U}}(t)\Big) \\ &\overset{(38)}{\supseteq} e^{-At}\Big(\texttt{CZ}(\mathcal{X}_{\text{end}} \ominus \widehat{\mathcal{Z}}_{\mathcal{U}}(t)) \oplus -\check{\mathcal{Z}}_{\mathcal{W}}(t)\Big) =: \check{\mathcal{R}}_{\forall\exists}(-t), \end{aligned} \tag{39}$$

where we evaluate the Minkowski difference $\mathcal{X}_{\text{end}} \ominus \widehat{\mathcal{Z}}_{\mathcal{U}}(t)$ by (11) and convert the resulting polytope to a constrained zonotope using (17) to efficiently evaluate the Minkowski sum with $-\check{\mathcal{Z}}_{\mathcal{W}}(t)$.

3) Runtime Complexity: Under Assumption 1 and following Table I, the outer approximative Minkowski sum from Proposition 4, the Minkowski difference, and the linear map in the computation of the outer approximation $\widehat{\mathcal{R}}_{\forall\exists}(-t)$ are all $\mathcal{O}(n^3)$, while the computation of the inner approximation $\check{\mathcal{R}}_{\forall\exists}(-t)$ is dominated by the conversion to a constrained zonotope, which is $\mathcal{O}(n^4)$.

4) Approximation Error: Both approximations have a non-zero approximation error even in the limit $\Delta t \to 0$ due to using Proposition 4 and the re-ordering in (38), respectively. For the more important outer approximation $\widehat{\mathcal{R}}_{\forall\exists}(-t)$, the approximation error can be made arbitrarily small in all directions selected for the evaluation of Proposition 4 as the particular solutions converge to their exact counterparts in the limit $\Delta t \to 0$ by Proposition 1.

5) Unperturbed Case: Let us briefly discuss the unperturbed case where $\mathcal{W} = \{\mathbf{0}\}$. Here, we compute the backward reachable set defined in (33) with $\tau = t$, for which (37) and (39) simplify accordingly. We can compute inner and outer approximations in $\mathcal{O}(n^3)$, whose approximation error depends on the approximation of the particular solution, which can be made arbitrarily small according to Proposition 1.

B. Time-Interval Solution

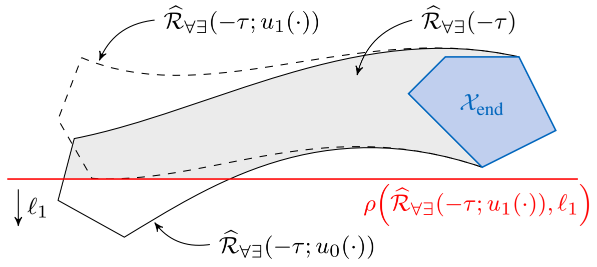
For the time-interval solution $\mathcal{R}_{\forall\exists}(-\tau)$ as defined in (30), we compute an outer approximation enclosing all states that cannot avoid entering the unsafe target set $\mathcal{X}_{\text{end}}$. We reformulate the definition in (30) to obtain

$$\mathcal{R}_{\forall\exists}(-\tau) = \bigcap_{u^*(\cdot)\in\mathbb{U}} \mathcal{R}_{\exists}(-\tau; u^*(\cdot)), \tag{40}$$

where

$$\mathcal{R}_{\exists}(-\tau; u^*(\cdot)) := \Big\{x_0 \in \mathbb{R}^n \,\Big|\, \exists w(\cdot) \in \mathbb{W}\ \exists t \in \tau : \xi(t; x_0, u^*(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}\Big\} \tag{41}$$

is the forward reachable set for the time-inverted dynamics using a single input trajectory $u^*(\cdot) \in \mathbb{U}$, which is equivalent



$\widehat{\mathcal{R}}_{\forall\exists}(-\tau; u_1(\cdot))$
$\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$
$\mathcal{X}_{\text{end}}$
ℓ_1
$\rho\Big(\widehat{\mathcal{R}}_{\forall\exists}(-\tau; u_1(\cdot)), \ell_1\Big)$
$\widehat{\mathcal{R}}_{\forall\exists}(-\tau; u_0(\cdot))$

Fig. 2. Computation of an outer approximation of the minimal backward reachable set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$ by intersection of multiple backward reachable sets for specific input trajectories (shown for two trajectories $u_0(\cdot), u_1(\cdot)$): The set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau; u_0(\cdot))$ is intersected with the halfspace constructed by the support function of the other set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau; u_1(\cdot))$ in the direction ℓ_1, while the input aims to maximize the extent of the set in the direction $-\ell_1$.

to (34) by replacing u by w. Consequently, the set $\mathcal{R}_{\forall\exists}(-\tau)$ is the intersection of the sets $\mathcal{R}_{\exists}(-\tau; u(\cdot))$ for all potential input trajectories $u(\cdot) \in \mathbb{U}$. Next, we show how to compute an outer approximation of (40).

1) Outer Approximation: The reachable set $\mathcal{R}_{\exists}(-\tau; u^*(\cdot))$ in (41) can be enclosed using standard methods [9, Sec. 3.2]:

$$\widehat{\mathcal{R}}_{\exists}(-\tau; u^*(\cdot)) = \bigcup_{k\in\{0,\dots,\omega-1\}} \widehat{\mathcal{R}}_{\exists}(-\tau_k; u^*(\cdot)) \tag{42}$$

$$\begin{aligned} \widehat{\mathcal{R}}_{\exists}(-\tau_k; u^*(\cdot)) = \texttt{conv}(&e^{-At_{k+1}}\texttt{CZ}(\mathcal{X}_{\text{end}}), e^{-At_k}\texttt{CZ}(\mathcal{X}_{\text{end}})) \\ &\oplus \boldsymbol{\mathcal{F}} e^{-At_{k+1}}\texttt{CZ}(\mathcal{X}_{\text{end}}) \oplus -\widehat{\mathcal{Z}}_{\mathcal{W}}(-\tau_k) \\ &\oplus -\widehat{\mathcal{Z}}_{u}(-\tau_k), \end{aligned} \tag{43}$$

where ω is the number of time intervals in $\tau = \tau_0 \cup \ldots \cup \tau_{\omega-1}$.

Our main idea is to compute an outer approximation of the reachable set for a single input trajectory and enclose the reachable sets for a finite number of other input trajectories by a polytope before evaluating the intersection in (40). Figure 2 illustrates this process: First, we compute an outer approximation of $\mathcal{R}_{\exists}(-\tau; u_0(\cdot))$ by (43) for the center input trajectory $\forall t \in \tau : u(t) = \texttt{cen}(\mathcal{U})$ denoted by $u_0(\cdot)$. Next, we intersect this solution with a polytope $\mathcal{P} = \langle N, p\rangle_H$ constructed via support function evaluations of reachable sets $\mathcal{R}_{\exists}(-\tau; u_j(\cdot))$ computed using a finite number of other input trajectories $u_j(\cdot) \in \mathbb{U}, j \in \mathbb{N}_{[1,q]}$ in a set of directions $\{\ell_1, \dots, \ell_q\}$. These input trajectories are chosen such that the extent of the reachable set $\mathcal{R}_{\exists}(-\tau; u_j(\cdot))$ is maximized in the opposite direction $-\ell$. We evaluate the support function of the corresponding additional reachable set in each direction $\ell_j, j \in \mathbb{N}_{[1,q]}$, i.e.,

$$\begin{aligned} &\rho\Big(\widehat{\mathcal{R}}_{\exists}(-\tau_k; u_j(\cdot)), \ell_j\Big) \\ &= \max\Big\{\rho\big(\mathcal{X}_{\text{end}}, (e^{-At_k})^\top \ell\big), \rho\big(\mathcal{X}_{\text{end}}, (e^{-At_{k+1}})^\top \ell\big)\Big\} \\ &\quad + \rho\Big(\widehat{\mathcal{Z}}_{\mathcal{W}}(-\tau_k), \ell\Big) - \rho\Big(\check{\mathcal{Z}}_{\mathcal{U}}(-t_k), -\ell\Big), \end{aligned} \tag{44}$$

and take the maximum value over all ω steps to construct the polytope $\mathcal{P}$ for intersection with $\widehat{\mathcal{R}}_{\exists}(-\tau; u_0(\cdot))$. Next, we prove that the outlined procedure indeed computes an outer approximation:

Theorem 1 (Time-interval minimal backward reachable set)**.** *Let the subset $\breve{\mathbb{U}} \subset \mathbb{U}$ be composed of $q \in \mathbb{N}$ input trajectories. The time-interval minimal backward reachable set* (30) *can be outer approximated by*

$$\widehat{\mathcal{R}}_{\forall\exists}(-\tau) = \bigcup_{k\in\{0,\dots,\omega-1\}} \left(\widehat{\mathcal{R}}_{\exists}(-\tau_k; u_0(\cdot)) \cap \langle N, p\rangle_H\right) \quad (45)$$

where $\widehat{\mathcal{R}}_{\exists}(-\tau; u_0(\cdot))$ is computed by (42) *using the center trajectory $u_0(\cdot)$, for which $\forall t \in \tau: u(t) = \texttt{cen}(\mathcal{U})$ holds, and*

$$\begin{aligned} &\forall j \in \mathbb{N}_{[1,q]} : N_{(j,\cdot)} = \ell_j^\top, \\ &\forall j \in \mathbb{N}_{[1,q]} : p_{(j)} = \min_{i\in\{1,\dots,q\}} \rho\Big(\breve{\mathcal{R}}_{\exists}(-\tau; u_i(\cdot)), \ell_j\Big) \end{aligned} \quad (46)$$

constructs a polytope via support function evaluations of the outer approximation $\widehat{\mathcal{R}}_{\exists}(-\tau; u_i(\cdot))$ of the backward reachable set (41) *using each of the q input trajectories in $\breve{\mathbb{U}}$.*

Proof. See Appendix. □

Algorithm 1 implements Theorem 1: In the main loop, we iteratively compute an explicit outer approximation $\widehat{\mathcal{R}}_{\exists}(-\tau; u^*(\cdot))$ of time-inverted dynamics $\dot{\tilde{x}}(t) = -A\tilde{x}(t) - B\texttt{cen}(\mathcal{U}) - Ew(t)$, where we account for the center input trajectory $\forall t \in \tau : u^*(t) = \texttt{cen}(\mathcal{U})$. Furthermore, we choose the maximizing directions for the other input trajectories $u(\cdot) \in \breve{\mathbb{U}}$ in positive and negative axis directions and propagate the corresponding support functions of an outer approximation for the time-inverted dynamics (lines 12-14). Ultimately, the intersection of the constructed polytope with the outer approximation $\widehat{\mathcal{R}}_{\exists}(-\tau)$ (line 16) yields the outer approximation of the time-interval minimal backward reachable set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$.

2) Runtime Complexity: Under Assumption 1, the dominating operations are the conversion of the target set $\mathcal{X}_{\text{end}}$ to a constrained zonotope in line 3 and the intersection in line 16, which are both $\mathcal{O}(n^4)$ according to Table I. All other operations are $\mathcal{O}(n^3)$.

3) Approximation Error: The approximation error of the intermediate result $\widehat{\mathcal{R}}_{\exists}(-\tau; u^*(\cdot))$ in (42) converges to 0 for $\Delta t \to 0$, since the particular solution $\widehat{\mathcal{Z}}_{\mathcal{W}}(t)$ converges to the exact solution $\mathcal{Z}_{\mathcal{W}}(t)$ by Proposition 1, the error term $\boldsymbol{\mathcal{F}} e^{-At_{k+1}}\texttt{CZ}(\mathcal{X}_{\text{end}})$ converges to $\{\mathbf{0}\}$ as $\lim_{\Delta t\to 0} \boldsymbol{\mathcal{F}} = [\mathbf{0}, \mathbf{0}]$ [14, Lemma 3], and all sets are closed under the applied set operations. For a zero approximation error everywhere, one would have to consider all combinations of directions of the support function of the particular solution $\mathcal{Z}_{\mathcal{U}}(t)$ and directions, in which to compute the intersection with $\widehat{\mathcal{R}}_{\exists}(-\tau)$.

4) Unperturbed Case: Setting $\mathcal{W} = \{\mathbf{0}\}$ removes every occurrence of the particular solution $\widehat{\mathcal{Z}}_{\mathcal{W}}(t)$ in Algorithm 2. This leaves the runtime complexity and approximation error unchanged.

VI. Maximal Backward Reachability Analysis

In this section, we compute an inner and an outer approximation of the time-point maximal backward reachable set $\mathcal{R}_{\exists\forall}(-t)$ (31) in Section VI-A as well as an inner approximation of the time-interval maximal backward reachable set $\mathcal{R}_{\exists\forall}(-\tau)$ (32) in Section VI-B. For all computed sets, we

Algorithm 1 Minimal time-interval backward reachable set

Require: Linear system $\dot{x} = Ax+Bu+Ew$, target set $\mathcal{X}_{\text{end}} = \langle H, d\rangle_H$, input set $\mathcal{U} = \langle c_u, G_u\rangle_Z$, disturbance set $\mathcal{W} = \langle c_w, G_w\rangle_Z$, time interval $\tau = [t_0, t_{\text{end}}]$, steps $\omega \in \mathbb{N}$

Ensure: Outer approximation of the time-interval backward reachable set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$

```
1:  $\Delta t \leftarrow (t_{\text{end}} - t_0)/\omega$, $w \leftarrow \texttt{cen}(\mathcal{W}) + \texttt{cen}(\mathcal{U})$
2:  $\mathcal{W}_0 \leftarrow \langle \mathbf{0}, G_w\rangle_Z$, $\mathcal{U}_0 \leftarrow \langle \mathbf{0}, G_u\rangle_Z$
3:  $\boldsymbol{\mathcal{F}} \leftarrow$ Eq. (19), $\mathcal{CZ} \leftarrow \texttt{CZ}(\mathcal{X}_{\text{end}})$    ▷ see (17)
4:  $N \leftarrow [I_n \ -I_n]^\top$, $q \leftarrow 2n$, $\forall j \in \mathbb{N}_{[1,q]}$: $p_{(j)} \leftarrow \infty$
5:  pre-compute $\widehat{\mathcal{Z}}_{\mathcal{W}_0}(-\Delta t)$ and $\widehat{\mathcal{Z}}_{\mathcal{W}_0}(-t_0)$    ▷ see (23), (25)
6:  $\forall j \in \mathbb{N}_{[1,q]}$: pre-compute $\rho\Big(\widehat{\mathcal{Z}}_{\mathcal{W}_0}(-t_0), N_{(j,\cdot)}^\top\Big)$ and
        $\rho\Big(\breve{\mathcal{Z}}_{\mathcal{U}_0}(-t_0), -N_{(j,\cdot)}^\top\Big)$    ▷ see (6), (7), (25)
7:  for $k \leftarrow 0$ to $\omega - 1$ do
8:      $t_{k+1} \leftarrow t_k + \Delta t$, $\tau_k \leftarrow [t_k, t_{k+1}]$, $\widehat{\mathcal{Z}}_w(-\tau_k) \leftarrow$ Eq. (26)
9:      $\widehat{\mathcal{Z}}_{\mathcal{W}_0}(-t_{k+1}) \leftarrow \widehat{\mathcal{Z}}_{\mathcal{W}_0}(-t_k) \oplus e^{-At_k}\widehat{\mathcal{Z}}_{\mathcal{W}_0}(-\Delta t)$
10:     $\widehat{\mathcal{Z}}_{\mathcal{W}}(-\tau_k) \leftarrow \widehat{\mathcal{Z}}_{\mathcal{W}_0}(-t_{k+1}) \oplus \widehat{\mathcal{Z}}_w(-\tau_k)$
11:     $\widehat{\mathcal{R}}_{\exists}(-\tau_k) \leftarrow \texttt{conv}(e^{-At_{k+1}}\mathcal{CZ}, e^{-At_k}\mathcal{CZ})$
            $\oplus \boldsymbol{\mathcal{F}} e^{-At_{k+1}}\mathcal{CZ} \oplus -\widehat{\mathcal{Z}}_{\mathcal{W}}(-\tau_k)$
12:     $\forall j \in \mathbb{N}_{[1,q]}$: propagate $\rho\Big(\widehat{\mathcal{Z}}_{\mathcal{W}}(-\tau_k), N_{(j,\cdot)}^\top\Big)$ and
            $\rho\Big(\breve{\mathcal{Z}}_{\mathcal{U}_0}(-t_{k+1}), -N_{(j,\cdot)}^\top\Big)$    ▷ see (6), (7)
13:     $\forall j \in \mathbb{N}_{[1,q]}$: $\rho\Big(\breve{\mathcal{R}}_{\exists}(-\tau_k), N_{(j,\cdot)}^\top\Big) \leftarrow$ Eq. (44)
14:     $\forall j \in \mathbb{N}_{[1,q]}$: $p_{(j)} \leftarrow \min\Big\{p_{(j)}, \rho\Big(\breve{\mathcal{R}}_{\exists}(-\tau_k), N_{(j,\cdot)}^\top\Big)\Big\}$
15: end for
16: $\widehat{\mathcal{R}}_{\forall\exists}(-\tau) = \bigcup_{k=0}^{\omega-1} \widehat{\mathcal{R}}_{\exists}(-\tau_k) \cap \langle N, p\rangle_H$    ▷ see (16)
```

show that the runtime complexity is polynomial in the state dimension n and discuss the approximation errors. Finally, we also compute the unperturbed cases $\mathcal{R}_{\exists}(-t)$ and $\mathcal{R}_{\exists}(-\tau)$ defined by (34).

A. Time-Point Solution

We base the computation of the backward reachable set $\mathcal{R}_{\exists\forall}(-t)$ on the following proposition:

Proposition 5 (Maximal time-point backward reachable set)**.** *The backward reachable set $\mathcal{R}_{\exists\forall}(-t)$ defined in* (31) *can be computed by*

$$\mathcal{R}_{\exists\forall}(-t) = e^{-At}\Big((\mathcal{X}_{end} \ominus \mathcal{Z}_{\mathcal{W}}(t)) \oplus -\mathcal{Z}_{\mathcal{U}}(t)\Big). \quad (47)$$

Proof. This is a continuization of the discrete-time case proven in [38, Thm. 2.4]. □

The formula above holds independently of the chosen set representations. Next, we compute approximations of (47) in polynomial time assuming a polytopic target set $\mathcal{X}_{\text{end}}$ and zonotopic particular solutions $\mathcal{Z}_{\mathcal{W}}(t)$ and $\mathcal{Z}_{\mathcal{U}}(t)$.

1) Outer and Inner Approximation: While the Minkowski difference $\mathcal{X}_{\text{end}} \ominus \mathcal{Z}_{\mathcal{W}}(t)$ in (47) can be evaluated efficiently using (11), the following Minkowski sum of the resulting polytope with the zonotope $-\mathcal{Z}_{\mathcal{U}}(t)$ is prohibitively slow. We overcome this issue by converting the polytope $\mathcal{X}_{\text{end}} \ominus \mathcal{Z}_{\mathcal{W}}(t)$

to a constrained zonotope using (17). For an outer approximation, we underestimate the influence of the disturbance by $\check{\mathcal{Z}}_{\mathcal{W}}(t) \subseteq \mathcal{Z}_{\mathcal{W}}(t)$ and overestimate the influence of the control input by $\widehat{\mathcal{Z}}_{\mathcal{U}}(t) \supseteq \mathcal{Z}_{\mathcal{U}}(t)$:

$$\begin{aligned}\mathcal{R}_{\exists\forall}(-t) &= e^{-At}\big((\mathcal{X}_{\text{end}} \ominus \mathcal{Z}_{\mathcal{W}}(t)) \oplus -\mathcal{Z}_{\mathcal{U}}(t)\big)\\ &\overset{(23),(24)}{\subseteq} e^{-At}\big(\mathtt{CZ}(\mathcal{X}_{\text{end}} \ominus \check{\mathcal{Z}}_{\mathcal{W}}(t)) \oplus -\widehat{\mathcal{Z}}_{\mathcal{U}}(t)\big) =: \widehat{\mathcal{R}}_{\exists\forall}(-t)\end{aligned} \tag{48}$$

and vice versa to compute an inner approximation:

$$\begin{aligned}\mathcal{R}_{\exists\forall}(-t) &= e^{-At}\big((\mathcal{X}_{\text{end}} \ominus \mathcal{Z}_{\mathcal{W}}(t)) \oplus -\mathcal{Z}_{\mathcal{U}}(t)\big)\\ &\overset{(23),(24)}{\supseteq} e^{-At}\big(\mathtt{CZ}(\mathcal{X}_{\text{end}} \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(t)) \oplus -\check{\mathcal{Z}}_{\mathcal{U}}(t)\big) =: \check{\mathcal{R}}_{\exists\forall}(-t).\end{aligned} \tag{49}$$

2) Runtime Complexity: Under Assumption 1, the dominating operation in (48) and (49) is the conversion to a constrained zonotope, which is $\mathcal{O}(n^4)$, as the Minkowski sums, Minkowski differences, and linear maps are at most $\mathcal{O}(n^3)$.

3) Approximation Error: Since the respective sets are closed under the applied operations, the entire approximation error is incurred by the outer and inner approximation of the particular solutions. Since these approximations converge to their exact counterparts in the limit $\Delta t \to 0$ by Proposition 1, the approximation errors of $\widehat{\mathcal{R}}_{\exists\forall}(-t)$ in (48) and $\check{\mathcal{R}}_{\exists\forall}(-t)$ in (49) also converge to 0 as $\Delta t \to 0$.

4) Unperturbed Case: For the unperturbed case with $\mathcal{W} = \{\mathbf{0}\}$, the formulae (48) and (49) for outer and inner approximation simplify accordingly to compute $\mathcal{R}_{\exists}(-t)$, see (34) with $\tau = t$. This yields the same runtime complexity and behavior of the approximation error in the limit $\Delta t \to 0$ as above.

B. Time-Interval Solution

For the time-interval solution $\mathcal{R}_{\exists\forall}(-\tau)$ as defined in (32), we want to compute an inner approximation so that all states are guaranteed to reach the target set $\mathcal{X}_{\text{end}}$. Our main idea is to inner approximate the union of time-point solutions $\bigcup_{t\in\tau} \mathcal{R}_{\exists\forall}(-t)$, which by Proposition 2 is an inner approximation of the time-interval solution $\mathcal{R}_{\exists\forall}(-\tau)$. We now show how to compute this inner approximation in polynomial time.

1) Inner Approximation: We require the following lemma:

Lemma 1 (Distributivity of Minkowski difference over convex hull)**.** *For three compact, convex, and nonempty sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \subset \mathbb{R}^n$, we have*

$$\mathtt{conv}(\mathcal{S}_1 \ominus \mathcal{S}_3, \mathcal{S}_2 \ominus \mathcal{S}_3) \subseteq \mathtt{conv}(\mathcal{S}_1, \mathcal{S}_2) \ominus \mathcal{S}_3.$$

Proof. See Appendix. □

Next, we exploit the superposition principle to inner approximate the union of time-point solutions over a time interval τ:

Theorem 2 (Maximal time-interval backward reachable set)**.** *The union of maximal time-point backward reachable sets*

$$\bigcup_{t\in\tau_k} \mathcal{R}_{\exists\forall}(-t) = \big\{e^{-At}\big((\mathcal{X}_{end} \ominus \mathcal{Z}_{\mathcal{W}}(t)) \oplus -\mathcal{Z}_{\mathcal{U}}(t)\big) \,\big|\, t \in \tau_k\big\} \tag{50}$$

over $\tau_k = [t_k, t_{k+1}]$ can be inner approximated by

$$\begin{aligned}\check{\mathcal{R}}_{\exists\forall}(-\tau_k) = e^{-At_{k+1}}\big(&-\check{\mathcal{Z}}_{\mathcal{U}}(t_k) \oplus\\ &\mathtt{conv}\big(\mathtt{CZ}(\mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(\tau_k)),\\ &\mathtt{CZ}(e^{A\Delta t}\mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(\tau_k))\big)\big),\end{aligned} \tag{51}$$

where all variables are computed as introduced in Section II-C. The union over all ω steps, that is,

$$\check{\mathcal{R}}_{\exists\forall}(-\tau) = \bigcup_{k\in\{0,\dots,\omega-1\}} \check{\mathcal{R}}_{\exists\forall}(-\tau_k),$$

is an inner approximation of the time-interval backward reachable set $\mathcal{R}_{\exists\forall}(-\tau)$ in (32) over the time interval $\tau = [t_0, t_{end}]$.

Proof. See Appendix. □

Algorithm 2 Maximal time-interval backward reachable set

Require: Linear system $\dot{x} = Ax+Bu+Ew$, target set $\mathcal{X}_{\text{end}} = \langle H, d\rangle_H$, input set $\mathcal{U} = \langle c_u, G_u\rangle_Z$, disturbance set $\mathcal{W} = \langle c_w, G_w\rangle_Z$, time interval $\tau = [t_0, t_{\text{end}}]$, steps $\omega \in \mathbb{N}$
Ensure: Inner approximation of the time-interval backward reachable set $\check{\mathcal{R}}_{\exists\forall}(-\tau)$

```
1:  $\Delta t \leftarrow (t_{\text{end}} - t_0)/\omega$, $w \leftarrow \mathtt{cen}(\mathcal{W})$, $\mathcal{W}_0 \leftarrow \langle \mathbf{0}, G_w\rangle_Z$
2:  pre-compute $\check{\mathcal{Z}}_{\mathcal{U}}(t_0)$ and $\widehat{\mathcal{Z}}_{\mathcal{W}_0}(t_0)$   ▷ see (23), (24), (25)
3:  $\mu \leftarrow \sqrt{\gamma}\,\|(e^{A\Delta t} - I_n)G\|_2$   ▷ $G$ and $\gamma$ from $\mathtt{box}(\mathcal{X}_{\text{end}})$
4:  $\mathcal{P}_1 \leftarrow \mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu$   ▷ see (19), (21)
5:  $\mathcal{P}_2 \leftarrow e^{A\Delta t}\mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu$   ▷ see (19), (21)
6:  for $k \leftarrow 0$ to $\omega - 1$ do
7:      $t_{k+1} \leftarrow t_k + \Delta t$, $\tau_k \leftarrow [t_k, t_{k+1}]$
8:      $\check{\mathcal{Z}}_{\mathcal{U}}(t_{k+1}) \leftarrow \check{\mathcal{Z}}_{\mathcal{U}}(t_k) \oplus e^{At_k}\check{\mathcal{Z}}_{\mathcal{U}}(\Delta t)$
9:      $\widehat{\mathcal{Z}}_{\mathcal{W}_0}(t_{k+1}) \leftarrow \widehat{\mathcal{Z}}_{\mathcal{W}_0}(t_k) \oplus e^{At_k}\widehat{\mathcal{Z}}_{\mathcal{W}_0}(\Delta t)$
10:     $\widehat{\mathcal{Z}}_w(\tau_k) \leftarrow$ Eq. (26), $\widehat{\mathcal{Z}}_{\mathcal{W}}(\tau_k) \leftarrow \widehat{\mathcal{Z}}_{\mathcal{W}_0}(t_{k+1}) \oplus \widehat{\mathcal{Z}}_w(\tau_k)$
11:     $\mathcal{CZ} \leftarrow \mathtt{conv}((\mathtt{CZ}(\mathcal{P}_1 \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(\tau_k)), \mathtt{CZ}(\mathcal{P}_2 \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(\tau_k)))$
12:     $\check{\mathcal{R}}_{\exists\forall}(-\tau_k) \leftarrow e^{-At_{k+1}}(\mathcal{CZ} \oplus -\check{\mathcal{Z}}_{\mathcal{U}}(t_k))$
13: end for
14: $\check{\mathcal{R}}_{\exists\forall}(-\tau) = \bigcup_{k=0}^{\omega-1} \check{\mathcal{R}}_{\exists\forall}(-\tau_k)$
```

Algorithm 2 implements Theorem 2, where we explicitly consider the more general case of a time interval $\tau = [t_0, t_{\text{end}}]$ with $t_0 > 0$: We pre-compute the particular solutions $\check{\mathcal{Z}}_{\mathcal{U}}(t)$ and $\widehat{\mathcal{Z}}_{\mathcal{W}}(t)$ until time t_0 in line 2 and pre-compute the polytopes $\mathcal{P}_1, \mathcal{P}_2$ (lines 4-5) that are used for inner approximating the time-interval homogeneous solution, see (21). The main loop computes all individual backward reachable sets $\check{\mathcal{R}}_{\exists\forall}(-\tau_k)$ following Theorem 2, whose union (line 14) yields the inner approximation of the time-interval maximal backward reachable set $\check{\mathcal{R}}_{\exists\forall}(-\tau)$.

2) Runtime Complexity: Under Assumption 1 and following Table I, only the operation $\mathtt{box}(\mathcal{X}_{\text{end}})$ is $\mathcal{O}(n^4)$, as the two following insights allow us to remove all other linear programs from Algorithm 2, which occur in line 11:

1) According to [12, Thm. 3], the exact conversion operation $\mathtt{CZ}(\mathcal{P})$ in (17) works with *any* enclosure of $\mathcal{P}$. Hence, we can use the pre-computed set $\mathtt{box}(\mathcal{X}_{\text{end}})$ in

all steps as

$$\forall t \in \tau, \forall i \in \{1,2\} : \mathcal{P}_i \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(t) \subseteq \texttt{box}(\mathcal{X}_{\text{end}}).$$

2) The conversion $\texttt{CZ}(\mathcal{P})$ only requires the support function evaluation of $\mathcal{P}$ for the intersection (16). Hence, we apply the identity (8) to obtain for $i \in \{1,2\}$:

$$\rho\Big(\mathcal{P}_i \ominus \widehat{\mathcal{Z}}_{\mathcal{W}}(t), -\ell\Big) = \rho(\mathcal{P}_i, -\ell) - \rho\Big(\widehat{\mathcal{Z}}_{\mathcal{W}}(t), -\ell\Big).$$

As $\mathcal{P}_1$ and $\mathcal{P}_2$ are constant, this simplifies to the efficient evaluation of the support function of $\widehat{\mathcal{Z}}_{\mathcal{W}}(t_{k+1})$.

As a consequence, increasing the number of steps ω and thereby improving the tightness is only $\mathcal{O}(n^3)$.

3) Approximation Error: By Proposition 1, the particular solutions $\widehat{\mathcal{Z}}_{\mathcal{W}}(t_{k+1})$ and $\check{\mathcal{Z}}_{\mathcal{U}}(t_k)$ converge to their exact counterparts at time t_k in the limit $\Delta t \to 0$. Moreover, the sets $\mathcal{P}_1$ and $\mathcal{P}_2$ converge to $\mathcal{X}_{\text{end}}$ as $\lim_{\Delta t \to 0} \boldsymbol{\mathcal{F}} = [\mathbf{0}, \mathbf{0}]$ by [14, Lemma 1] and $\lim_{\Delta t \to 0} \mu \overset{(22)}{=} 0$. Consequently, the computed individual time-interval solutions $\check{\mathcal{R}}_{\exists\forall}(-\tau_k)$ converge to the exact time-point solution $\mathcal{R}_{\exists\forall}(-t_k)$ in the limit $\Delta t \to 0$. However, a non-zero approximation error remains even in the limit as the union of time-point solutions is an inner-approximation of the time-interval solution by Proposition 2.

4) Unperturbed Case: As mentioned in Section IV-D, the unperturbed case is equivalent to computing the forward reachable set as defined in Definition 5 for the time-inverted dynamics $\dot{x}(t) = -Ax(t) - Bu(t)$. For $\mathcal{W} = \{\mathbf{0}\}$, Algorithm 2 simplifies to computing an inner approximation of this forward reachable set in $\mathcal{O}(n^4)$. In contrast to above, the approximation error of the unperturbed case does indeed converge to 0 in the limit $\Delta t \to 0$ [14, Thm. 1].

VII. Numerical Examples

We implemented our algorithms using the MATLAB toolbox CORA [52] for set-based computing and MOSEK[3] for solving linear programs. All computations are carried out on a 2.60GHz six-core i7 processor with 32GB RAM.

A. Pursuit-Evasion Game

First, we compare the results with the Python implementation[4] of the state-of-the-art Hamilton-Jacobi reachability analysis on a 4D pursuit-evasion game defined by the double integrator dynamics with [31, Eq. (24)]

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \; E = \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}.$$

The state is comprised of the relative positions and velocities in the horizontal and vertical plane, while the control inputs and disturbances represent the corresponding accelerations of Player 1 and Player 2, respectively. We examine both minimal and maximal reachability: In the minimal case, we look for all initial states from which Player 1 cannot avoid collision with Player 2, whereas the maximal case finds all states enabling Player 1 to catch Player 2. The target set $\mathcal{X}_{\text{end}} = \langle \mathbf{0}, I_4 \rangle_Z$ defines a collision between the players and we consider a time horizon of $\tau = [0,1]$. For the computation of the minimal time-interval backward reachable set, we use

$$\mathcal{U}_{\min} = \left\langle \begin{bmatrix} 0 \\ \frac{1}{8} \end{bmatrix}, \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{8} \end{bmatrix} \right\rangle_Z, \; \mathcal{W}_{\min} = \left\langle \begin{bmatrix} \frac{1}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \right\rangle_Z$$

while the maximal backward reachable set is computed for

$$\mathcal{U}_{\max} = \left\langle \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix}, \begin{bmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \right\rangle_Z, \; \mathcal{W}_{\max} = \left\langle \begin{bmatrix} \frac{1}{10} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{1}{10} & 0 \\ 0 & \frac{1}{10} \end{bmatrix} \right\rangle_Z$$

giving different steering capacities to each player.

A total number of $\omega = 100$ steps is used for the evaluation of both Algorithm 1 and Algorithm 2. The grid for the HJ reachability covers the plotted domain of $[-1.5, 1.5]$ along all four dimensions and consists of n_{grid} grid points in each dimension. Recall that we want an inner approximation of the maximal backward reachable set $\check{\mathcal{R}}_{\exists\forall}(-\tau)$ and an outer approximation of the minimal backward reachable set $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$: Hence, we plot only the grid points with a negative value function to represent $\check{\mathcal{R}}_{\exists\forall}(-\tau)$; for $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$, we plot all grid points with a negative value function evaluation as well as their neighbors in all directions (also diagonally) with nonnegative values.

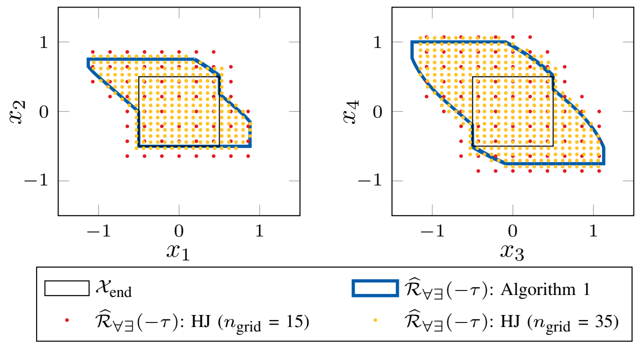


Fig. 3. Projections of the time-interval minimal backward reachable set for the pursuit-evasion game in Section VII-A using $\boldsymbol{\mathcal{U}}_{\mathbf{max}}$ and $\boldsymbol{\mathcal{W}}_{\mathbf{max}}$.

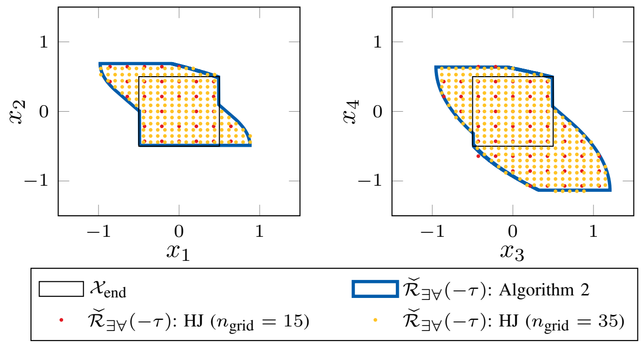


Fig. 4. Projections of the time-interval maximal backward reachable set for the pursuit-evasion game in Section VII-A using $\boldsymbol{\mathcal{U}}_{\mathbf{max}}$ and $\boldsymbol{\mathcal{W}}_{\mathbf{max}}$.

Figure 3 and Figure 4 show projections of the minimal and maximal backward reachable set, respectively, computed by Algorithm 1 and Algorithm 2, as well as via HJ reachability using $n_{\text{grid}} \in \{15, 35\}$ grid points per dimension. The plotted

[3]Available at `https://www.mosek.com`.

[4]Available at https://github.com/StanfordASL/hj_reachability.

grid points indicate that the outer approximation $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$ tightens with finer sampling, while the inner approximation $\breve{\mathcal{R}}_{\exists\forall}(-\tau)$ widens. The computation of $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$ using Algorithm 1 takes 0.27s and the computation of $\breve{\mathcal{R}}_{\exists\forall}(-\tau)$ using Algorithm 2 takes 2.2s. For the minimal and maximal HJ reachability, the time required to evaluate all grid points is similar: 1.6s for $n_{\text{grid}} = 15$ and 58s for $n_{\text{grid}} = 35$. A finer sampling of $n_{\text{grid}} = 55$ grid points per dimension results in computation times of over ten minutes due to the exponential increase in the total number of grid points.

The pursuit-evasion game shows similarly tight results obtained by our proposed algorithms compared to HJ reachability. While the runtime complexity of our proposed algorithms only scales linearly with the number of time steps, the computation time of HJ reachability strongly depends on the partitioning on the grid, as it suffers from the curse of dimensionality. Furthermore, the grid has to cover the domain of the backward reachable set, which ultimately requires knowledge about the solution before computing it. This is not the case for our proposed backward reachability algorithms.

B. Ground Collision Avoidance

Next, we examine the computation of the minimal backward reachable set by analyzing the effects of altering the input or disturbance capacities. To this end, we use a linearized longitudinal model of a quadrotor [53, Eq. (42)]

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -d_0 & -d_1 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K & 0 \\ 0 & 0 \\ 0 & n_0 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

with $g = 9.81, d_0 = 70, d_1 = 17, K = 0.89/1.4$, and $n_0 = 55$. In order, the states represent the horizontal position, vertical position, horizontal velocity, vertical velocity, roll, and roll velocity. For our ground collision avoidance scenario, we want to avoid any state $x_2 \leq 0.1$ with a negative velocity $x_4 \leq 0$. Since the target set $\mathcal{X}_{\text{end}}$ has to be bounded, we constrain the other states inspired by [53, Sec. 6.1]:

$$\mathcal{X}_{\text{end}} = \left\langle \begin{bmatrix} 0 & \frac{1}{20} & 0 & -\frac{1}{2} & 0 & 0 \end{bmatrix}^\top, \operatorname{diag}\begin{bmatrix} \frac{1}{2} & \frac{1}{20} & 1 & \frac{1}{2} & \frac{\pi}{15} & \frac{\pi}{2} \end{bmatrix} \right\rangle_Z.$$

The control inputs are the total normalized thrust and the desired roll angle, while the disturbances represent linearization errors. Inspired by [53, Eq. (45)], we bound these values by

$$\begin{aligned} \mathcal{U} &= \left\langle \begin{bmatrix} \frac{g}{K} & 0 \end{bmatrix}^\top, \operatorname{diag}\begin{bmatrix} \zeta\frac{3}{2} & \frac{\pi}{6} \end{bmatrix} \right\rangle_Z \\ \mathcal{W} &= \left\langle \begin{bmatrix} 0 & 0 \end{bmatrix}^\top, \operatorname{diag}\begin{bmatrix} 0.2760\varphi & 0.3668 \end{bmatrix} \right\rangle_Z \end{aligned}$$

where the scaling factors $\zeta \in \mathbb{R}$ and $\varphi \in \mathbb{R}$ allow us to design cases with different input and disturbance capacities.

Figure 5 shows the time-interval minimal backward reachable sets for $\tau = [0, 0.5]$ computed using Algorithm 1 with $\omega = 200$ steps for three different pairs of ζ and φ:

- $\widehat{\mathcal{R}}^{(1)}_{\forall\exists}(-\tau)$: $\zeta^{(1)} = 1$, $\varphi^{(1)} = 10$
- $\widehat{\mathcal{R}}^{(2)}_{\forall\exists}(-\tau)$: $\zeta^{(2)} = 1$, $\varphi^{(2)} = 1$
- $\widehat{\mathcal{R}}^{(3)}_{\forall\exists}(-\tau)$: $\zeta^{(3)} = 2$, $\varphi^{(3)} = 1$

The computations times are 3.8s, 3.4s, and 1.9s for the three cases. As expected, the projections show that $\widehat{\mathcal{R}}^{(1)}_{\forall\exists}(-\tau) \supset \widehat{\mathcal{R}}^{(2)}_{\forall\exists}(-\tau) \supset \widehat{\mathcal{R}}^{(3)}_{\forall\exists}(-\tau)$ since the input capacity increases via $\zeta^{(1)} \leq \zeta^{(2)} \leq \zeta^{(3)}$ and the size of the disturbance set $\mathcal{W}$ decreases as $\varphi^{(1)} \geq \varphi^{(2)} \geq \varphi^{(3)}$. In the leftmost projection, we see that $\widehat{\mathcal{R}}^{(1)}_{\forall\exists}(-\tau)$ extends furthest in $\pm x_1$ and $\pm x_3$ because the disturbance w_1 is larger than in the other cases and forces more states to enter the target set. As indicated by the middle and rightmost plots, an increase of the input capacity of u_1 for $\widehat{\mathcal{R}}^{(3)}_{\forall\exists}(-\tau)$ allows more states to avoid the target set $\mathcal{X}_{\text{end}}$ in comparison to $\widehat{\mathcal{R}}^{(2)}_{\forall\exists}(-\tau)$, which is affected by the same disturbance set. Also note that the leftmost projections of $\widehat{\mathcal{R}}^{(2)}_{\forall\exists}(-\tau)$ and $\widehat{\mathcal{R}}^{(3)}_{\forall\exists}(-\tau)$ are identical because the input u_1 neither directly nor indirectly influences these dimensions. Moreover, the middle plot shows that no state with positive vertical velocity x_4 that is unable to avoid the target set. In summary, the chosen example nicely demonstrates the effect of different input capacities and disturbances on the size of the minimal backward reachable set.

C. Terminal Set Reachability

In this subsection, we analyze the computation of the maximal backward reachable set. To this end, we use a 12-dimensional quadrotor system linearized about the hover condition [54, Sec. 2]. The state matrix $A \in \mathbb{R}^{12\times 12}$ and the input matrix $B \in \mathbb{R}^{12\times 4}$ are given in [54, Appendix A], while the disturbance matrix $E \in \mathbb{R}^{12\times 3}$ is all-zero except for $E_{(4,1)} = E_{(5,2)} = E_{(6,3)} = 1$ as in [55, Sec. V-D]. To highlight the relation of maximal reachability with controller synthesis, we choose a so-called *safe terminal set* [56, Sec. IV-A] as our target set: For each state in the safe terminal set, there exists a stabilizing controller such that the state remains in the safe terminal set at the next time step and, by induction, for all times. Our maximal backward reachable set contains all states that can be steered into the safe terminal set despite worst-case disturbances.

Using the approach in [56] implemented in the MATLAB toolbox AROC [57], we obtain the safe terminal set $\langle \mathbf{0}, G\rangle_Z$ whose generator matrix G (tabulated in the Appendix in Figure 7) is square and full-rank. Hence, the set $\langle \mathbf{0}, G\rangle_Z$ is a parallelotope and can be easily converted into the polytope $\mathcal{X}_{\text{end}}$ as required by Algorithm 2. We define the input set and disturbance set as [55, Sec. V-D]

$$\begin{aligned} \mathcal{U} &= \left\langle \begin{bmatrix} -3.715 & 0 & 0 & 0 \end{bmatrix}^\top, \operatorname{diag}\begin{bmatrix} 6.095 & \zeta_{(1)} & \zeta_{(2)} & \zeta_{(3)} \end{bmatrix} \right\rangle_Z \\ \mathcal{W} &= \left\langle \mathbf{0}, \operatorname{diag}\begin{bmatrix} \varphi_{(1)} & \varphi_{(2)} & \varphi_{(3)} \end{bmatrix} \right\rangle_Z \end{aligned}$$

where the scaling factors $\zeta \in \mathbb{R}^3$ and $\varphi \in \mathbb{R}^3$ allow us to compare the backward reachable sets obtained for different input capacities and disturbances.

We compute the time-interval maximal backward reachable set for a time horizon $\tau = [0, 1]$ using $\omega = 500$ steps. Figure 6 shows various projections for different values of ζ and φ:

- $\breve{\mathcal{R}}^{(1)}_{\exists\forall}(-\tau)$: $\zeta^{(1)} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}^\top$, $\varphi^{(1)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$
- $\breve{\mathcal{R}}^{(2)}_{\exists\forall}(-\tau)$: $\zeta^{(2)} = \begin{bmatrix} 1 & 0.75 & 0.25 \end{bmatrix}^\top$, $\varphi^{(2)} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$
- $\breve{\mathcal{R}}^{(3)}_{\exists\forall}(-\tau)$: $\zeta^{(3)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top$, $\varphi^{(3)} = \begin{bmatrix} 0.05 & 0.025 & 0.01 \end{bmatrix}^\top$

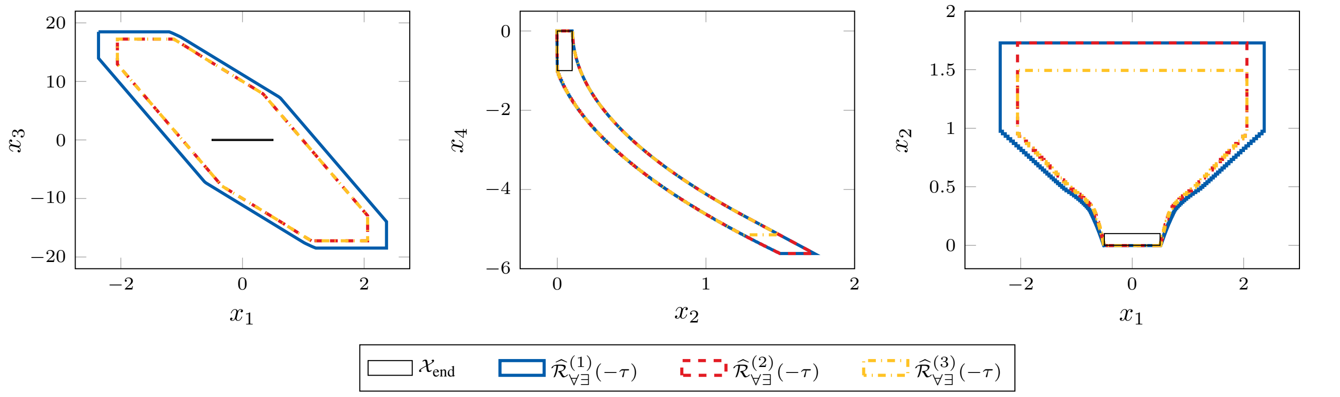


Fig. 5. Projections of the time-interval minimal backward reachable set for the ground collision avoidance scenario in Section VII-B.

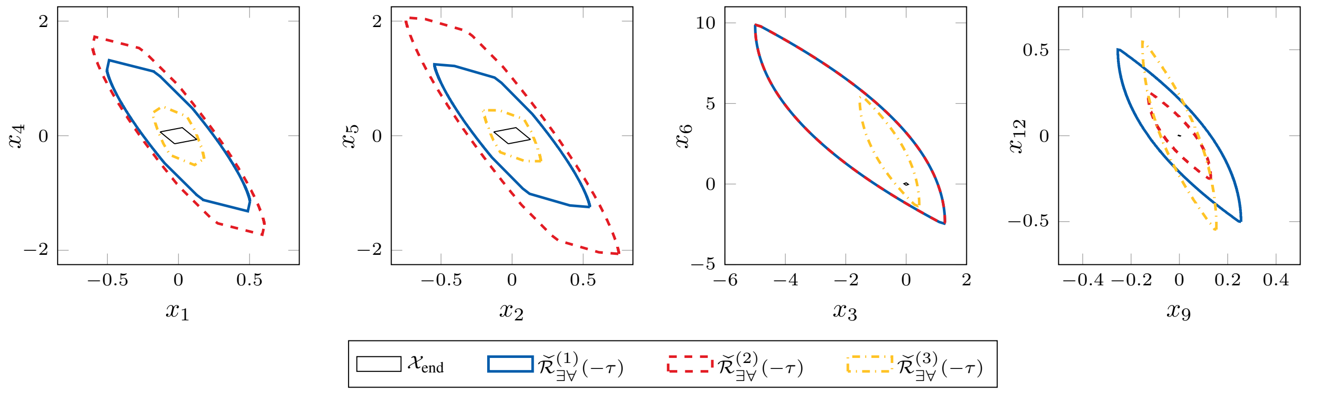


Fig. 6. Projections of the time-interval maximal backward reachable set for the quadrotor system in Section VII-C.

The computation time is 4.1s in all three cases. First of all, we notice that the backward reachable sets are symmetric with respect to the origin along some dimensions, which is caused by the symmetry of the input set and the disturbance set—except for $u_{(1)} \in [-9.81, 2.38]$, which becomes apparent in the projection onto the x_3-x_6 axes. The sets $\breve{\mathcal{R}}^{(1)}_{\exists\forall}(-\tau)$ and $\breve{\mathcal{R}}^{(2)}_{\exists\forall}(-\tau)$ are computed without any disturbance. Hence, the backward reachable set $\breve{\mathcal{R}}^{(2)}_{\exists\forall}(-\tau)$ encloses $\breve{\mathcal{R}}^{(1)}_{\exists\forall}(-\tau)$ in the first two projections from the left since the contributing part of the input set $\mathcal{U}$ is larger: $i \in \{1,2\} : \zeta^{(1)}_{(i)} > \zeta^{(2)}_{(i)}$. In constrast, the opposite containment holds true for the rightmost projection as $\zeta^{(1)}_{(3)} > \zeta^{(2)}_{(3)}$. This follows our expectation because more input capacity can steer additional states into the target set, thereby enlarging the maximal backward reachable set.

The computation of $\breve{\mathcal{R}}^{(3)}_{\exists\forall}(-\tau)$ takes a non-zero disturbance into account, but also increases the input capacity compared to $\breve{\mathcal{R}}^{(2)}_{\exists\forall}(-\tau)$: The first three projections show a much smaller backward reachable set as the additional input capacity is outweighed by the disturbance. In contrast, the relatively small disturbance $\varphi^{(3)}_{(3)}$ does not affect the reachable set size in the rightmost projection too much but results in a slightly rotated set in comparison with $\breve{\mathcal{R}}^{(1)}_{\exists\forall}(-\tau)$ in combination with the increased input capacity. In summary, this example demostrates well how different input capacities and disturbances affect the size of the maximal reachable set.

D. Scalability Analysis

Finally, we analyze the scalability of our backward reachability algorithms. To this end, we choose the scalable platoon benchmark [58], whose dynamics are given in [58, Eq. (9)], where we choose $\gamma = 2$ as in [58, Sec. 2.4]. For a number of trucks θ, the state vector is $x(t) = \left[x^{(1)}(t)^\top \;\; \dots \;\; x^{(v)}(t)^\top\right]^\top \in \mathbb{R}^{3\theta}$ with $x^{(j)} = \left[e^{(j)}(t) \;\; \dot{e}^{(j)}(t) \;\; a^{(j)}(t)\right]^\top$, where $e^{(j)}(t)$ is the relative position between truck $j-1$ and j shifted by a safe distance, $\dot{e}^{(j)}(t)$ is the relative velocity between truck $j-1$ and j, and $a^{(j)}(t)$ is the acceleration of the jth truck. The input $u(t) \in \mathbb{R}^\theta$ concatenates the individual input accelerations $u^{(j)}$ for all θ trucks. The disturbance $w(t) \in \mathbb{R}$ is the acceleration of the leading truck.

We use $t = 2$ and $\tau = [0, 2]$ for the time-point and time-interval backward reachable sets, respectively, and $\omega = 100$ as the total number of steps. The target set is composed of the Cartesian product of the individual target sets for each truck, which are $\mathcal{X}_{\text{end}} = [-20, 0]\mathrm{m} \times [3, 10]\mathrm{m\,s^{-1}} \times [1, 5]\mathrm{m\,s^{-2}}$ in the minimal case and $\mathcal{X}_{\text{end}} = [0, 20]\mathrm{m} \times [-1.5, 1.5]\mathrm{m\,s^{-1}} \times [-1, 1]\mathrm{m\,s^{-2}}$ in the maximal case. In both cases, the input acceleration of each truck is bounded by $[-5, 1]\mathrm{m\,s^{-2}}$ and the acceleration of the leading truck by $[-0.5, 0.5]\mathrm{m\,s^{-2}}$.

In Table II, we show the computation time for evaluating all four time-point and time-interval backward reachable sets for an increasing number of trucks θ. Unsurprisingly, the computa-

TABLE II
COMPUTATION TIMES FOR PLATOON BENCHMARK FOR INCREASING STATE DIMENSION n AND INPUT DIMENSION m.

n	m	$\widehat{\mathcal{R}}_{\forall\exists}(-t)$	$\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$	$\check{\mathcal{R}}_{\exists\forall}(-t)$	$\check{\mathcal{R}}_{\exists\forall}(-\tau)$
15	5	0.02s	0.27s	0.13s	0.24s
30	10	0.02s	0.35s	0.18s	0.36s
51	17	0.03s	0.66s	0.27s	0.71s
99	33	0.05s	2.0s	0.52s	2.4s
150	50	0.11s	4.1s	0.98s	4.4s
300	100	0.42s	15s	3.4s	20s
600	200	2.3s	76s	19s	92s
999	333	8.8s	—	70s	—
2001	667	76s	—	—	—

tion of $\widehat{\mathcal{R}}_{\forall\exists}(-t)$ is always fastest since it is the only algorithm that scales with $\mathcal{O}(n^3)$. Second is the other time-point solution $\widehat{\mathcal{R}}_{\exists\forall}(-t)$ due to only one operation being $\mathcal{O}(n^4)$. Compared to the time-point solutions, the computation of both time-interval solutions is more time-consuming, largely due to the numerous linear programs and concatenation of large zonotope generator matrices. In summary, this evaluation of the scalable platoon benchmark demonstrates the polynomial runtime complexity in the state dimension of all our backward reachability algorithms, enabling the analysis of very high-dimensional linear systems.

E. Discussion

Let us now address some critical aspects regarding our proposed backward reachability algorithms: First of all, the target set $\mathcal{X}_{\text{end}}$ has to be represented as a polytope. While the manual design of polytopes is quite intuitive, the target set may come from another algorithm and thus be represented by a different set representation, which needs to be converted to a polytope. For minimal reachability, one wants an outer approximation of the original set, whereas maximal reachability requires the converted polytope to be contained in the original set—both cases can be handled via optimization, e.g., using support function evaluations (9) to obtain an enclosing polytope.

As discussed in the respective subsections, the approximation errors of all backward reachable sets except the time-point maximal backward set are non-zero even in the limit $\Delta t \to 0$. Still, one can tighten the time-point and time-interval minimal backward reachable sets in arbitrary directions by additional support function evaluations. These directions will be chosen according to the demands of the considered application scenario. For the time-interval maximal backward reachable set, the approximation error entirely depends on the tightness of the containment in Proposition 2. For large disturbances, the forward reachable set of a given initial state may not be contained within the target set at any specific point in time, but still pass through the target set over a time interval. This would make that initial state part of the time-interval solution, while excluding it from the time-point solution. Further investigation into this issue is required to formally capture the notion of one set passing through another, which is different from both containment and intersection.

Since we know that there exists a control input to steer each state of the maximal backward reachable set into the target set, a natural next step is the extraction of such a controller as in [39, Sec. IV-B.2)]. The sets in our work are limited to feed-forward controllers because we consider the effects of the control input and disturbance separately. Instead, one can also skip backward reachability and directly synthesize a controller, which is a well-researched topic for linear continuous-time systems offering a wide range of different approaches.

VIII. Conclusion

This article presents the first backward reachability algorithms using set propagation techniques for perturbed continuous-time linear systems. The proposed algorithms cover minimal and maximal reachability and compute both time-point and time-interval solutions. The runtime complexity of all algorithms is polynomial in the state dimension. Our evaluation shows tight results and how changes in the input and disturbance set affect the size of the resulting backward rechable set. Furthermore, we examined the scalability of our algorithms by analyzing systems with well over a hundred states within seconds, which significantly improves the state of the art in backward reachability analysis.

Appendix

Proof of Proposition 1:
The approximation error in $\widehat{\mathcal{Z}}_{\mathcal{S}}(t)$ as propagated by (25) is given by the sum of the approximation errors induced by the additional terms $e^{At_k}\widehat{\mathcal{Z}}_{\mathcal{S}}(\Delta t)$ [14, Proposition 2]. Each of these additionally induced approximation errors converges to 0 for $\Delta t \to 0$ [14, Lemma 2]. Thus, the total approximation error in $\widehat{\mathcal{Z}}_{\mathcal{S}}(t)$ also converges to 0 in the limit $\Delta t \to 0$. The same reasoning holds also for the inner approximation $\check{\mathcal{Z}}_{\mathcal{S}}(t)$ as the approximation errors in [14, Proposition 2] are measured in terms of the Hausdorff distance between the outer and inner approximation [14, Proposition 11]. □

Proof of Proposition 4:
We insert $\mathcal{P} \oplus \mathcal{Z}$ into (9) to obtain

$$\begin{aligned}
&\mathcal{P} \oplus \mathcal{Z} \subseteq \langle H, \tilde{d} \rangle_H, \\
&\forall j \in \mathbb{N}_{[1,h]} : \tilde{d}_{(j)} = \rho\Big(\mathcal{P} \oplus \mathcal{Z}, H_{(j,\cdot)}^\top\Big) \\
&\qquad\qquad = \rho\Big(\mathcal{P}, H_{(j,\cdot)}^\top\Big) + \rho\Big(\mathcal{Z}, H_{(j,\cdot)}^\top\Big) \\
&\qquad\qquad = d_{(j)} + \rho\Big(\mathcal{Z}, H_{(j,\cdot)}^\top\Big) .
\end{aligned}$$

The runtime complexity follows from the h support function evaluations of $\mathcal{Z}$ (14). □

Proof of Theorem 1:
By considering only a finite subset of input signals $\check{\mathbb{U}} \subset \mathbb{U}$, we obtain an outer approximation:

$$\begin{aligned}
\mathcal{R}_{\forall\exists}(-\tau) &\overset{(30)}{=} \big\{x_0 \in \mathbb{R}^n \,\big|\, \forall u(\cdot) \in \mathbb{U}\ \exists w(\cdot) \in \mathbb{W}\ \exists t \in \tau : \\
&\qquad x(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}\big\} \\
&\subseteq \big\{x_0 \in \mathbb{R}^n \,\big|\, \forall u(\cdot) \in \check{\mathbb{U}}\ \exists w(\cdot) \in \mathbb{W}\ \exists t \in \tau : \\
&\qquad x(t; x_0, u(\cdot), w(\cdot)) \in \mathcal{X}_{\text{end}}\big\} \\
&= \bigcap_{u^* \in \check{\mathbb{U}}} \mathcal{R}_{\exists}(-\tau; u^*(\cdot)) =: \mathcal{S}_1 .
\end{aligned} \qquad (52)$$

Let us denote the input trajectory $\forall t \in \tau : u(t) = \mathtt{cen}(\mathcal{U})$ by u_0 and the other q input trajectories in $\breve{\mathbb{U}}$ by $u_1, ..., u_q$. To evaluate $\mathcal{S}_1$ in (52), we compute an outer approximation of $\mathcal{R}_\exists(-\tau; u_0)$ that also encloses $\mathcal{R}_{\forall\exists}(-\tau)$ since

$$\mathcal{R}_{\forall\exists}(-\tau) \overset{(52)}{\subseteq} \mathcal{R}_\exists(-\tau; u_0) \overset{(42)}{\subseteq} \widehat{\mathcal{R}}_\exists(-\tau; u_0).$$

Second, we incorporate all other input trajectories in $\breve{\mathbb{U}}$:

$$\begin{aligned}\mathcal{S}_1 &= \bigcap_{j\in\{0,...,q\}} \mathcal{R}_\exists(-\tau; u_j) \\ &\overset{(43)}{\subseteq} \left(\widehat{\mathcal{R}}_\exists(-\tau_0; u_0) \cup ... \cup \widehat{\mathcal{R}}_\exists(-\tau_{\omega-1}; u_0)\right) \\ &\qquad \cap \mathcal{R}_\exists(-\tau; u_1) \cap ... \cap \mathcal{R}_\exists(-\tau; u_q) \\ &\overset{(42)}{\subseteq} \left(\widehat{\mathcal{R}}_\exists(-\tau_0; u_0) \cup ... \cup \widehat{\mathcal{R}}_\exists(-\tau_{\omega-1}; u_0)\right) \\ &\qquad \cap \widehat{\mathcal{R}}_\exists(-\tau; u_1) \cap ... \cap \widehat{\mathcal{R}}_\exists(-\tau; u_q) =: \mathcal{S}_2.\end{aligned} \quad (53)$$

We enclose each additional set by the polytope constructed using support function evaluations in the directions $\ell_1, ..., \ell_q$:

$$\forall j \in \mathbb{N}_{[1,q]} : \widehat{\mathcal{R}}_\exists(-\tau; u_j) \overset{(9)}{\subseteq} \langle N, p^{(j)} \rangle_H$$
$$\text{with } N = [\ell_1 ... \ell_q]^\top, \forall i \in \mathbb{N}_{[1,q]} : p^{(j)}_{(i)} = \rho\Big(\widehat{\mathcal{R}}_\exists(-\tau; u_j), \ell_j\Big). \quad (54)$$

We insert this in (53) to obtain

$$\begin{aligned}\mathcal{S}_2 &\overset{(54)}{\subseteq} \left(\widehat{\mathcal{R}}_\exists(-\tau_0; u_0) \cup ... \cup \widehat{\mathcal{R}}_\exists(-\tau_{\omega-1}; u_0)\right) \\ &\qquad \cap \langle N, p^{(1)} \rangle_H \cap ... \cap \langle N, p^{(q)} \rangle_H \\ &= \left(\widehat{\mathcal{R}}_\exists(-\tau_0; u_0) \cup ... \cup \widehat{\mathcal{R}}_\exists(-\tau_{\omega-1}; u_0)\right) \cap \langle N, p \rangle_H,\end{aligned}$$

where p is the minimum value as in (46). Finally, distributing the intersection over the union yields $\widehat{\mathcal{R}}_{\forall\exists}(-\tau)$ in (45). □

Proof of Lemma 1:
We plug into the definitions of the Minkowski difference (3) and convex hull (5):

$$\begin{aligned}&\mathtt{conv}(\mathcal{S}_1 \ominus \mathcal{S}_3, \mathcal{S}_2 \ominus \mathcal{S}_3) \oplus \mathcal{S}_3 \\ &= \{\lambda a + (1-\lambda) b + c \,|\, \lambda \in [0,1], a \oplus \mathcal{S}_3 \subseteq \mathcal{S}_1, \\ &\qquad b \oplus \mathcal{S}_3 \subseteq \mathcal{S}_2, c \in \mathcal{S}_3\} \\ &= \{\lambda(a+c) + (1-\lambda)(b+c) \,|\, \lambda \in [0,1], a \oplus \mathcal{S}_3 \subseteq \mathcal{S}_1, \\ &\qquad b \oplus \mathcal{S}_3 \subseteq \mathcal{S}_2, c \in \mathcal{S}_3\} \\ &\subseteq \{\lambda s_1 \oplus (1-\lambda) s_2 \,|\, \lambda \in [0,1], s_1 \in a \oplus \mathcal{S}_3 \subseteq \mathcal{S}_1, \\ &\qquad s_2 \in b \oplus \mathcal{S}_3 \subseteq \mathcal{S}_2\} \\ &\subseteq \mathtt{conv}(\mathcal{S}_1, \mathcal{S}_2),\end{aligned}$$

from which it follows that

$$\mathtt{conv}(\mathcal{S}_1 \ominus \mathcal{S}_3, \mathcal{S}_2 \ominus \mathcal{S}_3) \subseteq \mathtt{conv}(\mathcal{S}_1, \mathcal{S}_2) \ominus \mathcal{S}_3,$$

since $\mathcal{S} \oplus \mathcal{S}_3 \ominus \mathcal{S}_3 = \mathcal{S}$ holds [39, Lemma 1(iii)]. □

Proof of Theorem 2:
A single time-interval solution $\mathcal{R}_{\exists\forall}(-\tau_k)$ over $\tau_k = [t_k, t_{k+1}]$ covering part of the union in (50) can be expressed by

$$\mathcal{S}_1 := e^{-At_{k+1}}\big((\mathcal{H}(\tau_0) \ominus \mathcal{Z}_\mathcal{W}(\tau_k)) \oplus -\mathcal{Z}_\mathcal{U}(\tau_k)\big).$$

For the particular solutions over $\tau_k = [t_k, t_{k+1}]$, we have

$$\breve{\mathcal{Z}}_\mathcal{U}(t_k) \subseteq \mathcal{Z}_\mathcal{U}(\tau_k), \ \mathcal{Z}_\mathcal{W}(\tau_k) \subseteq \widehat{\mathcal{Z}}_\mathcal{W}(\tau_k),$$

which are computed using (24)-(27). Consequently, we obtain

$$\mathcal{S}_1 \supseteq e^{-At_{k+1}}\big((\mathcal{H}(\tau_0) \ominus \widehat{\mathcal{Z}}_\mathcal{W}(\tau_k)) \oplus -\breve{\mathcal{Z}}_\mathcal{U}(t_k)\big) =: \mathcal{S}_2.$$

Let us now plug in the inner approximation of the homogeneous time-interval solution (21):

$$\begin{aligned}\mathcal{S}_2 \supseteq e^{-At_{k+1}}\big(&\mathtt{conv}(\mathcal{X}_{\text{end}}, e^{A\Delta t}\mathcal{X}_{\text{end}}) \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \\ &\ominus \mathcal{B}_\mu \ominus \widehat{\mathcal{Z}}_\mathcal{W}(\tau_k) \oplus -\breve{\mathcal{Z}}_\mathcal{U}(t_k)\big) =: \mathcal{S}_3.\end{aligned}$$

Note that we enclose $\mathcal{X}_{\text{end}}$ by $\mathtt{box}(\mathcal{X}_{\text{end}})$ to evaluate the multiplication with the interval matrix $\boldsymbol{\mathcal{F}}$ using (15) and compute μ as in (22) using the generator matrix of $\mathtt{box}(\mathcal{X}_{\text{end}})$. We now apply Lemma 1 and convert the two polytopes of the convex hull operation to constrained zonotopes by (17) to efficiently evaluate the following Minkowski sum with $-\breve{\mathcal{Z}}_\mathcal{U}(t_k)$, resulting in

$$\begin{aligned}\mathcal{S}_3 \supseteq e^{-At_{k+1}}\big(&-\breve{\mathcal{Z}}_\mathcal{U}(t_k) \oplus \\ &\mathtt{conv}\big(\mathtt{CZ}(\mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu \ominus \widehat{\mathcal{Z}}_\mathcal{W}(\tau_k)), \\ &\mathtt{CZ}(e^{A\Delta t}\mathcal{X}_{\text{end}} \ominus \boldsymbol{\mathcal{F}}\,\mathtt{box}(\mathcal{X}_{\text{end}}) \ominus \mathcal{B}_\mu \ominus \widehat{\mathcal{Z}}_\mathcal{W}(\tau_k))\big)\big) \\ =:&\ \breve{\mathcal{R}}_{\exists\forall}(-\tau_k).\end{aligned}$$

Thus, each set $\breve{\mathcal{R}}_{\exists\forall}(-\tau_k)$ is an inner approximation of the union of time-point solutions over τ_k, which in turn is an inner approximation of the time-interval solution $\mathcal{R}_{\exists\forall}(-\tau_k)$ by Proposition 2:

$$\breve{\mathcal{R}}_{\exists\forall}(-\tau_k) \subseteq \bigcup_{t\in\tau_k} \mathcal{R}_{\exists\forall}(-t) \overset{\text{Proposition 2}}{\subseteq} \mathcal{R}_{\exists\forall}(-\tau_k).$$

Extending this reasoning to all ω consecutive time intervals yields the claim. □

Acknowledgment

Many thanks to my colleagues Adrian Kulmburg, Tobias Ladner, Lukas Schäfer, and Victor Gaßmann for their help in the formalization of some proofs, the installation of the HJ reachability toolbox, the design of the numerical examples, and the discussion of the algorithms.

References

[1] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *Proc. of the International Workshop on Hybrid Systems: Computation and Control*, Springer, 2007, pp. 428–443.

[2] A. Girard and C. Le Guernic, "Efficient reachability analysis for linear systems using support functions," *IFAC Proceedings Volumes*, vol. 41, no. 2, 2008.

[3] G. Frehse, "Computing maximizer trajectories of affine dynamics for reachability," in *Proc. of the 54th Conference on Decision and Control*, 2015, pp. 7454–7461.

[4] P. M. Vaidya, "An algorithm for linear programming which requires $\mathcal{O}(((M+n)N^2 + (M+n)^{1.5}n)L)$ arithmetic operations," in *Proc. of the 19th Annual Symposium on Theory of Computing*, ACM, 1987, pp. 29–38.

[5] G. M. Ziegler, *Lectures on polytopes*. Springer Science & Business Media, 2012.

[6] 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.

$$G = \begin{bmatrix} -0.0042 & 0.0455 & 0.0064 & -0.0694 & 0 & 0 & 0.0001 & -0.0004 & 0 & 0 & -0.0002 & -0.0004 \\ 0.0455 & 0.0042 & 0.0694 & 0.0064 & 0 & 0 & 0.0004 & 0.0001 & 0 & 0 & -0.0004 & 0.0002 \\ 0 & 0 & 0 & 0 & -0.0370 & 0.0377 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0086 & -0.0924 & 0.0031 & -0.0331 & 0 & 0 & 0.0008 & -0.0022 & 0 & 0 & -0.0003 & -0.0006 \\ -0.0924 & -0.0086 & 0.0331 & 0.0031 & 0 & 0 & 0.0022 & 0.0008 & 0 & 0 & -0.0006 & 0.0003 \\ 0 & 0 & 0 & 0 & 0.0491 & 0.0284 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0044 & -0.0004 & 0.0083 & 0.0008 & 0 & 0 & 0.0088 & 0.0032 & 0 & 0 & 0.0046 & -0.0023 \\ 0.0004 & -0.0044 & 0.0008 & -0.0083 & 0 & 0 & 0.0032 & -0.0088 & 0 & 0 & 0.0023 & 0.0046 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0045 & -0.0005 & 0 & 0 \\ -0.0091 & -0.0008 & 0.0071 & 0.0007 & 0 & 0 & -0.0244 & -0.0088 & 0 & 0 & 0.0016 & -0.0008 \\ 0.0008 & -0.0091 & 0.0007 & -0.0071 & 0 & 0 & -0.0088 & 0.0244 & 0 & 0 & 0.0008 & 0.0016 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0019 & -0.0011 & 0 & 0 \end{bmatrix}$$

Fig. 7. Generator matrix $\boldsymbol{G}$ of the safe terminal set $\langle \mathbf{0}, \boldsymbol{G} \rangle_Z$ for the quadrotor system in Section VII-C computed using the approach in [56].

[7] I. Kolmanovsky and E. G. Gilbert, “Theory and computation of disturbance invariant sets for discrete-time linear systems,” *Mathematical Problems in Engineering*, vol. 4, 1998.

[8] A. Girard, “Reachability of uncertain linear systems using zonotopes,” in *Proc. of the 8th International Workshop on Hybrid Systems: Computation and Control*, Springer, 2005, pp. 291–305.

[9] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” Dissertation, Technische Universität München, 2010.

[10] C. Le Guernic and A. Girard, “Reachability analysis of linear systems using support functions,” *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.

[11] M. Althoff, O. Stursberg, and M. Buss, “Reachability analysis of linear systems with uncertain parameters and inputs,” in *Proc. of the 46th Conference on Decision and Control*, IEEE, 2007, pp. 726–732.

[12] J. K. Scott, D. M. Raimondo, G. R. Marseglia, *et al.*, “Constrained zonotopes: A new tool for set-based estimation and fault detection,” *Automatica*, vol. 69, pp. 126–136, 2016.

[13] V. Raghuraman and J. P. Koeln, “Set operations and order reductions for constrained zonotopes,” *Automatica*, vol. 139, p. 110 204, 2022.

[14] M. Wetzlinger, N. Kochdumper, S. Bak, *et al.*, “Fully automated verification of linear systems using inner and outer approximations of reachable sets,” *IEEE Transactions on Automatic Control*, vol. Early Access, pp. 1–16, 2023.

[15] G. Frehse, C. Le Guernic, A. Donzé, *et al.*, “SpaceEx: Scalable verification of hybrid systems,” in *Proc. of the 23rd International Conference on Computer Aided Verification*, ser. LNCS 6806, Springer, 2011, pp. 379–395.

[16] C. Le Guernic, “Reachability analysis of hybrid systems with linear continuous dynamics,” Dissertation, Université Joseph-Fourier - Grenoble I, 2009.

[17] M. Chen and C. J. Tomlin, “Hamilton-Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 333–358, 2018.

[18] A. Girard, C. Le Guernic, and O. Maler, “Efficient computation of reachable sets of linear time-invariant systems with inputs,” in *Proc. of the 9th International Workshop on Hybrid Systems: Computation and Control*, Springer, 2006, pp. 257–271.

[19] M. Althoff, O. Stursberg, and M. Buss, “Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization,” in *Proc. of the 47th Conference on Decision and Control*, IEEE, 2008, pp. 4042–4048.

[20] X. Chen, “Reachability analysis of non-linear hybrid systems using Taylor models,” Dissertation, RWTH Aachen University, 2015.

[21] B. Xue, Z. She, and A. Easwaran, “Underapproximating backward reachable sets by semialgebraic sets,” *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5185–5197, 2017.

[22] E. Goubault and S. Putot, “Forward inner-approximated reachability of non-linear continuous systems,” in *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*, ACM, 2017, pp. 1–10.

[23] E. Goubault and S. Putot, “Robust under-approximations and application to reachability of non-linear control systems with disturbances,” *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 928–933, 2020.

[24] X. Chen, S. Sankaranarayanan, and E. Ábrahám, “Under-approximate flowpipes for non-linear continuous systems,” in *Formal Methods in Computer-Aided Design*, IEEE, 2014, pp. 59–66.

[25] N. Kochdumper and M. Althoff, “Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems,” in *Proc. of the 59th Conference on Decision and Control*, IEEE, 2020, pp. 2130–2137.

[26] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, “A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games,” *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[27] S. Bansal, M. Chen, S. Herbert, *et al.*, “Hamilton-Jacobi reachability: A brief overview and recent advances,” in *Proc. of the 56th Conference on Decision and Control*, IEEE, 2017, pp. 2242–2253.

[28] M. Chen, S. Herbert, and C. J. Tomlin, “Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition,” in *Proc. of the International Conference on Robotics and Automation*, IEEE, 2017, pp. 87–92.

[29] M. Chen, S. Herbert, M. S. Vashishtha, *et al.*, “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.

[30] D. Lee, M. Chen, and C. J. Tomlin, “Removing leaking corners to reduce dimensionality in Hamilton-Jacobi reachability,” in *Proc. of the International Conference on Robotics and Automation*, IEEE, 2019, pp. 9320–9326.

[31] M. Chen and C. J. Tomlin, “Exact and efficient Hamilton-Jacobi reachability for decoupled systems,” in *Proc. of the 54th Conference on Decision and Control*, IEEE, 2015, pp. 1297–1303.

[32] M. Chen, J. C. Shih, and C. J. Tomlin, “Multi-vehicle collision avoidance via Hamilton-Jacobi reachability and mixed integer programming,” in *Proc. of the 55th Conference on Decision and Control*, IEEE, 2016, pp. 1695–1700.

[33] S. Bansal and C. J. Tomlin, “DeepReach: A deep learning approach to high-dimensional reachability,” in *Proc. of the International Conference on Robotics and Automation*, IEEE, 2021, pp. 1817–1824.

[34] S. Herbert, J. J. Choi, S. Sanjeev, *et al.*, “Scalable learning of safety guarantees for autonomous systems using Hamilton-Jacobi reachability,” in *Proc. of the International Conference on Robotics and Automation*, IEEE, 2021, pp. 5914–5920.

[35] M. Jones and M. M. Peet, “Relaxing the Hamilton Jacobi Bellman equation to construct inner and outer bounds on reachable sets,” in *Proc. of the 58th Conference on Decision and Control*, IEEE, 2019, pp. 2397–2404.

[36] N. Kochdumper, “Extensions of polynomial zonotopes and their application to verification of cyber-physical systems,” Dissertation, Technische Universität München, 2022.

[37] B. Schürmann, M. Klischat, N. Kochdumper, *et al.*, “Formal safety net control using backward reachability analysis,” *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 5698–5713, 2021.

[38] A. A. Kurzhanskiy and P. Varaiya, “Reach set computation and control synthesis for discrete-time dynamical systems with disturbances,” *Automatica*, vol. 47, no. 7, pp. 1414–1426, 2011.

[39] 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.

[40] L. Yang, H. Zhang, J.-B. Jeannin, *et al.*, “Efficient backward reachability using the minkowski difference of constrained zonotopes,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3969–3980, 2022.

[41] H. Yin, A. Packard, M. Arcak, *et al.*, “Finite horizon backward reachability analysis and control synthesis for uncertain nonlinear systems,” in *American Control Conference*, 2019, pp. 5020–5026.

[42] H. Yin, M. Arcak, A. Packard, *et al.*, “Backward reachability for polynomial systems on a finite horizon,” *IEEE Transactions on Automatic Control*, vol. 66, no. 12, pp. 6025–6032, 2021.

[43] H. Yin, P. Seiler, and M. Arcak, “Backward reachability using integral quadratic constraints for uncertain nonlinear systems,” *Control Systems Letters*, vol. 5, no. 2, pp. 707–712, 2021.

[44] K. Margellos and J. Lygeros, “Hamilton-Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management,” *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1849–1861, 2011.

[45] J. F. Fisac, M. Chen, C. J. Tomlin, *et al.*, “Reach-avoid problems with time-varying dynamics, targets and constraints,” in *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control*, ACM, 2015, pp. 11–20.

[46] B. Xue, M. Fränzle, and N. Zhan, “Inner-approximating reachable sets for polynomial systems with time-varying uncertainties,” *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1468–1483, 2020.

[47] S. Kaynama, M. Oishi, I. M. Mitchell, *et al.*, “The continual reachability set and its computation using maximal reachability techniques,” in *Proc. of the 50th Conference on Decision and Control and European Control Conference*, IEEE, 2011, pp. 6110–6115.

[48] S. Kaynama, I. M. Mitchell, M. Oishi, *et al.*, “Scalable safety-preserving robust control synthesis for continuous-time linear systems,” *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 3065–3070, 2015.

[49] S. Kaynama, J. Maidens, M. Oishi, *et al.*, “Computing the viability kernel using maximal reachable sets,” in *Proc. of the 15th international conference on Hybrid Systems: Computation and Control*, ACM, 2012, pp. 55–64.

[50] J. N. Maidens, S. Kaynama, I. M. Mitchell, *et al.*, “Lagrangian methods for approximating the viability kernel in high-dimensional systems,” *Automatica*, vol. 49, no. 7, pp. 2017–2029, 2013.

[51] E. Goubault and S. Putot, “Inner and outer reachability for the verification of control systems,” in *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*, ACM, 2019, pp. 11–22.

[52] M. Althoff, “An introduction to CORA 2015,” in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, pp. 120–151.

[53] I. M. Mitchell, J. Budzis, and A. Bolyachevets, “Invariant, viability and discriminating kernel under-approximation via zonotope scaling,” *arXiv preprint arXiv:1901.01006*, 2019.

[54] S. Kaynama and C. J. Tomlin, “Benchmark: Flight envelope protection in autonomous quadrotors,” in *Workshop on Applied Verification of Continuous and Hybrid Systems*, 2014.

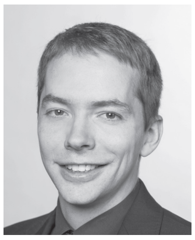
[55] F. Gruber and M. Althoff, “Scalable robust safety filter with unknown disturbance bounds,” *IEEE Transactions on Automatic Control*, pp. 1–15, 2023.

[56] F. Gruber and M. Althoff, “Computing safe sets of linear sampled-data systems,” *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 385–390, 2021.

[57] N. Kochdumper, F. Gruber, B. Schürmann, *et al.*, “AROC: A toolbox for automated reachset optimal controller synthesis,” in *Proc. of the 24th International Conference on Hybrid Systems: Computation and Control*, ACM, 2021.

[58] I. Ben Makhlouf and S. Kowalewski, “Optimizing safe control of a network platoon of trucks using reachability,” in *ARCH14-15. 1st and 2nd International Workshop on Applied Verification for Continuous and Hybrid Systems*, EasyChair, 2015, pp. 169–179.

MARK WETZLINGER received the B.S. degree in Engineering Sciences in 2017 jointly from Universität Salzburg, Austria and Technische Universität München, Germany, and the M.S. degree in Robotics, Cognition and Intelligence in 2019 from Technische Universität München, Germany. He is currently pursuing the Ph.D. degree in computer science at Technische Universität München, Germany. His research interests include formal verification of linear and nonlinear continuous systems, reachability analysis, adaptive parameter tuning, and model order reduction.



MATTHIAS ALTHOFF is an associate professor in computer science at Technische Universität München, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from Technische Universität München, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

A.5 Adaptive Parameter Tuning for Reachability Analysis of Nonlinear Systems

Summary Using reachable sets to solve the verification task in Problem 1 is difficult since the approximation error may grow uncontrollably, resulting in an excessively large outer approximation. This issue is caused by the wrapping effect—the amplification of approximation errors over time—which is unavoidable for all current propagation-based reachability algorithms for nonlinear systems. Only well-tuned algorithm parameters can mitigate the wrapping effect so that the computed reachable sets are tight enough for verification.

In this work, we automate the tuning of algorithm parameters for a reachability algorithm based on on-the-fly abstraction. Following an analysis of the influence of the time step size on the abstraction error, we design an estimation function for the reachable set size after a finite time horizon. Scalar optimization over the estimated reachable set size yields the optimal time step size for the current time step. Another critical parameter is the abstraction order up to which we Taylor expand the right-hand side of the differential equation: In each time step, we check whether the abstraction error changes more than a fixed threshold compared to the previous time step. If this is the case, we compare the abstraction errors for both abstraction orders and switch to the other order if the difference exceeds a fixed threshold.

The proposed automated reachability algorithm successfully analyzes a wide variety of benchmark systems. We obtain similar performance in a single run compared to manually tuned state-of-the-art tools for reachability and verification of nonlinear systems. If we also consider the many runs of trial and error necessary for manual parameter tuning, the comparison would greatly favor our novel approach.

Author contributions M.W. developed the idea of parameter tuning using a local analysis of the abstraction errors, implemented the algorithm, conducted the numerical evaluation, and wrote most of the manuscript. A.K. proved Lemma 3.1 and Theorem 3.2 and contributed to improving the rigor of the presentation. M.A. suggested ideas for parameter tuning and provided feedback for improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Adaptive Parameter Tuning for Reachability Analysis of Nonlinear Systems

Mark Wetzlinger
Technische Universität München
85748 Garching bei München
Germany
m.wetzlinger@tum.de

Adrian Kulmburg
Technische Universität München
85748 Garching bei München
Germany
adrian.kulmburg@tum.de

Matthias Althoff
Technische Universität München
85748 Garching bei München
Germany
althoff@tum.de

ABSTRACT

Reachability analysis fails to produce tight reachable sets if certain algorithm parameters are poorly tuned, such as the time step size or the accuracy of the set representation. The tuning is especially difficult in the context of nonlinear systems where over-approximation errors accumulate over time due to the so-called wrapping effect, often requiring expert knowledge. In order to widen the applicability of reachability analysis for practitioners, we propose the first adaptive parameter tuning approach for reachability analysis of nonlinear continuous systems tuning all algorithm parameters. Our modular approach can be applied to different reachability algorithms as well as various set representations. Finally, an evaluation on numerous benchmark systems shows that the adaptive parameter tuning approach efficiently computes very tight enclosures of reachable sets.

CCS CONCEPTS

• **General and reference** → **Verification**; • **Mathematics of computing** → **Ordinary differential equations**.

KEYWORDS

Reachability analysis, nonlinear systems, parameter tuning.

ACM Reference Format:
Mark Wetzlinger, Adrian Kulmburg, and Matthias Althoff. 2021. Adaptive Parameter Tuning for Reachability Analysis of Nonlinear Systems. In *24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447928.3456643

1 INTRODUCTION

Reachability analysis provably guarantees avoiding unsafe states of mixed discrete/continuous systems for a set of uncertain initial states and uncertain inputs. Since exact reachable sets can only be computed for a limited number of system classes [36], reachability algorithms compute over-approximations to establish soundness. The performance of these algorithms heavily relies on the correct setting of algorithm parameters—a safety property may not be verified although it is satisfied by the exact reachable set. We consider the automated tuning of algorithm parameters to be a crucial next step in the development of reachability analysis. A full automation would enable non-experts and practitioners to use reachability analysis supporting the development of safer products. This paper advances in this direction by automatically tuning all algorithm parameters for state-space abstracted reachability analysis of nonlinear systems.

Related Work. The computation of reachable sets for nonlinear systems can be divided into four groups: First, there are approaches for invariant generation; any invariant set containing the initial set is also a reachable set [34, 40, 43]. Second, there exist optimization-based approaches which treat reachability analysis by solving an optimization problem [19, 44]. Third, other approaches abstract the solution space: The work in [24] uses validated simulations for the construction of bounded flowpipes. Taylor models computed by using the Picard iteration were initially proposed in [29, 42] and later extended to include uncertain inputs [15]. Finally, there are approaches abstracting the state space by differential inclusions, such as the abstraction of nonlinear systems by a hybrid automaton with linear dynamics [7, 8, 28, 39]. Other methods linearize the nonlinear dynamics on-the-fly [6, 20, 21]—a concept that has been extended to polynomial abstractions of nonlinear dynamics [3] resulting in a tighter enclosure of the exact reachable set. In this paper, we present an automatic parameter tuning approach for state-space abstracted reachability algorithms. Many aforementioned methods have been realized by tools: For nonlinear systems, there is Ariadne [11], C2E2 [23], CORA [4], DynIBEX [22], Flow* [16], Isabelle/HOL [30], and JuliaReach [13].

While there is almost no work on finding a suitable time step size for reachability analysis of nonlinear systems, this problem is well studied for numerical integration of ordinary differential equations (ODEs): A common method applied in numerical ODE solvers is to compute solutions with different precision in parallel and adapt the time step size according to the difference between the solutions [9, 37]. In order to enclose a single trajectory, guaranteed integration methods provide several automated time step size control strategies [31, 45, 48]. Since reachability analysis considers a set of uncertain initial states as well as uncertain inputs, automatic parameter tuning is much more difficult than for classical and validated integration.

Concerning reachability analysis, there are approaches automatically tuning algorithm parameters for linear systems: In [25], the time step is adapted in each step in order to satisfy a linearly increasing user-defined error bound. The approach in [46] adaptively determines the time step size by approximating the actual flow within a user-defined error bound. Recently, an approach to adapt all algorithm parameters in reachability analysis of linear systems has been developped [51], using over-approximation measures related to the Hausdorff distance to enable users to tune the desired accuracy. For nonlinear systems, the work in [14] adaptively tunes the time step size within a user-defined range, according to a numeric threshold condition, which in turn has to be defined by the user for each system analysis.

Contributions. We introduce the first approach that automatically tunes all algorithm parameters for reachability analysis of nonlinear systems. After introducing some preliminaries in Sec. 2, we present our novel automated parameter tuning approach in Sec. 3. Each parameter is tuned individually due to the modular structure of our method, thus providing a very flexible integration in the reachability algorithm. Furthermore, our proposed tuning during runtime without backtracking improves the computational efficiency. Finally, the evaluation on numerical examples in Sec. 4 demonstrates the practical usability of our tuning methods, followed by concluding remarks in Sec. 5.

2 PRELIMINARIES

In this section, we give an overview of reachability analysis for nonlinear systems based on state-space abstraction. This will serve as a basis for our adaptive tuning methods.

2.1 Notation

Vectors are denoted by lower-case letters, matrices by upper-case letters, and sets by upper-case calligraphic letters. An all-zero vector of proper dimension is denoted by $\mathbf{0}$. Given a vector $v \in \mathbb{R}^n$, v_i refers to the i-th entry and the absolute value $|v| \in \mathbb{R}^n$ is computed element-wise. For a matrix $M \in \mathbb{R}^{n \times p}$, m_{ij} refers to the entry in the i-th row and j-th column. The concatenation of two matrices is denoted by $[M_1 \; M_2]$. An n-dimensional axis-aligned box is denoted by $\mathcal{B} = [a, b] \subset \mathbb{R}^n$, where $a_i \leq b_i, \forall i \in \{1, ..., n\}$. The diameter and the absolute value of a box are respectively defined by $d(\mathcal{B}) := b - a \in \mathbb{R}^n$ and $\mathtt{abs}(\mathcal{B}) := [-c, c] \subset \mathbb{R}^n$, where $c = \max\{|a|, |b|\}$ is evaluated element-wise [1, eq. (10)]. As an abbreviation, we denote the Cartesian product of identical lower and upper limits for n dimensions by $[a, b]^n$. Interval matrices are denoted by upper-case boldface letters: $\mathbf{I} = [P, Q] \in \mathbb{R}^{m \times n}$, where $p_{ij} \leq q_{ij}, \forall i \in \{1, ..., m\}, \forall j \in \{1, ..., n\}$. The Minkowski addition is denoted by $\oplus$. The operations $\mathtt{center}(\mathcal{S})$, $\mathtt{box}(\mathcal{S})$, and $\mathtt{vol}(\mathcal{S})$ return the geometric center, the smallest box over-approximation, and the volume of a set $\mathcal{S} \subset \mathbb{R}^n$, respectively. Furthermore, the projection onto the i-th axis is denoted by $\mathcal{S}_i = e_i^\top \mathcal{S}$, where e_i is the i-th basis vector, and the convex hull of two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ is written as $\mathtt{conv}(\mathcal{S}_1, \mathcal{S}_2)$. The floor operator $\lfloor k \rfloor$ rounds k down to the next smaller integer number, $\mathrm{sgn}(\cdot)$ denotes the sign function, and $\|\cdot\|_F$ the Frobenius norm.

2.2 Reachability Analysis of Nonlinear Systems

The presented techniques for automated parameter adaptation are applied to nonlinear systems

$$\dot{x}(t) = f(x(t), u(t)) \,, \quad x(t) \in \mathbb{R}^n, \; u(t) \in \mathbb{R}^m \,, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a sufficiently smooth nonlinear function, $x(t) \in \mathbb{R}^n$ is the state vector, and $u(t) \in \mathbb{R}^m$ is the input vector. Let us introduce $\xi(t; x_0, u(\cdot))$ as the solution of (1) at time t for the initial point $x_0 = x(0)$. Then, the exact reachable set $\mathcal{R}_{\mathrm{ex}}([0, t_K])$ of (1) over the time horizon $t \in [0, t_K]$ is defined as

$$\mathcal{R}_{\mathrm{ex}}([0, t_K]) = \Big\{ \xi(t; x_0, u(\cdot)) \,\Big|\, x_0 \in \mathcal{X}^0, \; t \in [0, t_K], \\ \forall \tau \in [0, t] : u(\tau) \in \mathcal{U} \Big\} \,,$$

with the initial set $\mathcal{X}^0 \subset \mathbb{R}^n$ and the input set $\mathcal{U} \subset \mathbb{R}^m$. In this work, we use state-space abstraction, where the nonlinear dynamics in (1) are abstracted by a Taylor series of order κ at an expansion point z^* [3, eq. (2)] so that

$$\dot{x}_i \in \sum_{\nu=0}^{\kappa} \frac{\left((z(t) - z^*)^T \nabla\right)^\nu f_i(\hat{z})}{\nu!} \Bigg|_{\hat{z}=z^*} \oplus \mathcal{L}_i(t) \,, \tag{2}$$

using the extended vector $z = [x^T u^T]^T \in \mathbb{R}^{n+m}$ and the Nabla operator $\nabla = \sum_{i=1}^{n+m} e_i \frac{\partial}{\partial z_i}$, where e_i are orthogonal unit vectors. The Lagrange remainder $\mathcal{L}_i$ is defined by [3, eq. (2)]

$$\mathcal{L}_i = \left\{ \frac{\left((z(t) - z^*)^T \nabla\right)^{\kappa+1} f_i(\hat{z})}{(\kappa+1)!} \,\middle|\, \hat{z} = z^* + \alpha(z(t) - z^*), \alpha \in [0, 1] \right\} , \tag{3}$$

which is evaluated using range-bounding techniques such as interval arithmetic [12]. The time horizon $[0, t_K]$ is divided into K time intervals $\tau_s = [t_s, t_{s+1}]$, with the individual time step sizes $\Delta t_s = t_{s+1} - t_s > 0$ summing up to t_K. The complete reachable set is obtained by the union $\mathcal{R}([0, t_K]) = \bigcup_{s=0}^{K-1} \mathcal{R}(\tau_s)$. For notational simplicity, we introduce an equivalent notation for the first terms in (2),

$$w_i = f_i(z^*), \; C_{ij} = \frac{\partial f_i(\hat{z})}{\partial \hat{z}_j} \bigg|_{\hat{z}=z^*}, \; D_{ijk} = \frac{\partial^2 f_i(\hat{z})}{\partial \hat{z}_j \partial \hat{z}_k} \bigg|_{\hat{z}=z^*}, \; \ldots \tag{4}$$

where we split the first-order approximation $C = [A \; B]$ into a state matrix $A \in \mathbb{R}^{n \times n}$ and an input matrix $B \in \mathbb{R}^{n \times m}$ for subsequent use. Let us summarize the reachability analysis in Alg. 1 encompassing the core reachable set computation featured in hybridization and on-the-fly methods, such as the ones in [3, 6, 8, 20, 39].

At the start of each step s (Line 4), the operation `taylor` evaluates the Taylor terms of the nonlinear dynamics (4) at the linearization point z^*. Next, we abstract the nonlinear system by a differential inclusion

$$\dot{x}(t) \in \underbrace{Ax(t) + Bu(t) + w}_{f_{\mathrm{lin}}(t)} \oplus \Psi \,, \tag{5}$$

using the linearized vector field f_{lin} and an uncertainty set Ψ enclosing all higher-order terms including the Lagrange remainder. This allows us to apply the superposition principle for linear systems and separate the computation of the next reachable set $\mathcal{R}(t_{s+1})$ into two parts: First, the reachable set $\mathcal{R}_{\mathrm{lin}}$ of the linearized dynamics (Lines 4-5). Second, the set of abstraction errors $\mathcal{R}_{\mathrm{abs}}$ based on the abstraction error Ψ (Lines 6-11).

Algorithm 1 Reachability analysis of nonlinear systems using state-space abstraction.

```
Input: nonlinear function $f(z)$, initial set $\mathcal{R}(t_0) = \mathcal{X}^0$,
       input set $\mathcal{U}$, time horizon $t_K$
Output: $\mathcal{R}([0, t_K])$
1:  $s = 0, t_s = 0$
2:  while $t_s < t_K$ do
3:      $z^*(t_s) \leftarrow \texttt{center}(\mathcal{R}(t_s))$
4:      $w, A, B, D, \ldots \leftarrow \texttt{taylor}(f(z), z^*(t_s))$
5:      $\mathcal{R}_{\text{lin}}(t_{s+1}), \mathcal{R}_{\text{lin}}(\tau_{s+1}) \leftarrow \texttt{linReach}(\mathcal{R}(t_s), w, A, B)$
6:      $\overline{\Psi} = \mathbf{0}$
7:      do
8:          $\overline{\Psi} \leftarrow \texttt{enlarge}(\overline{\Psi})$
9:          $\Psi \leftarrow \texttt{abstrErr}(\mathcal{R}_{\text{lin}}(\tau_{s+1}), \overline{\Psi})$
10:     while $\Psi \not\subseteq \overline{\Psi}$
11:     $\mathcal{R}_{\text{abs}} \leftarrow \texttt{abstrSol}(\Psi)$
12:     $\mathcal{R}(t_{s+1}) = \mathcal{R}_{\text{lin}}(t_{s+1}) \boxplus \mathcal{R}_{\text{abs}}$
13:     $\mathcal{R}(\tau_{s+1}) = \mathcal{R}_{\text{lin}}(\tau_{s+1}) \boxplus \mathcal{R}_{\text{abs}}$
14:     $\mathcal{R}(t_{s+1}) \leftarrow \texttt{red}(\mathcal{R}(t_{s+1})), \mathcal{R}(\tau_{s+1}) \leftarrow \texttt{red}(\mathcal{R}(\tau_{s+1}))$
15:     $t_{s+1} := t_s + \Delta t_s,\ s := s + 1$
16: end while
17: $\mathcal{R}([0, t_K]) = \bigcup_{j=0}^{K-1} \mathcal{R}(\tau_j)$
```

The reachable set $\mathcal{R}_{\text{lin}}$ based on the linearized dynamics $Ax(t) + Bu(t) + w$ is computed by the operation `linReach` (Line 5) using a reachability algorithm for linear systems, e.g., [2, Sec. 3.2]. For the computation of the abstraction error Ψ, we first resolve the mutual dependency between Ψ and $\mathcal{R}_{\text{lin}}$ by an initial estimation $\overline{\Psi}$ of Ψ (Line 6). We now require $\Psi \subseteq \overline{\Psi}$ (Line 7) which is attained by iteratively enlarging $\overline{\Psi}$ by a constant factor greater than 1 using the operation `enlarge` (Line 8) and computing Ψ using the operation `abstrErr` (Line 9). For a linearization approach (see, e.g., [6]), the entire set Ψ is the abstraction error uncorrelated with the state x. For a polynomialization approach (see, e.g., [3]), all time-constant terms at t_s within Ψ represent a higher-order evaluation of the nonlinear dynamics that is correlated with the state x. After containment is ensured, we evaluate the effect of Ψ by the operation `abstrSol` (Line 11), yielding the set of abstraction errors [6, Sec. VI.]

$$\mathcal{R}_{\text{abs}} = \bigoplus_{k=0}^{\eta_{\text{abs}}} \frac{\Delta t^{k+1}}{(k+1)!} A^k \Psi \oplus \mathbf{E}(\Delta t, \eta_{\text{abs}})\, \Delta t\, \Psi\,, \tag{6}$$

where $\mathbf{E} = \mathcal{O}(\Delta t^{\eta_{\text{abs}}+1})$ tends to 0 as $\Delta t \longrightarrow 0$, see [5, Prop. 2].

Due to the aforementioned superposition, we yield the next reachable set $\mathcal{R}(t_{s+1})$ by the addition of $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ (Lines 12-13). Before the next step, the operator $\texttt{red}(\cdot)$ reduces the set representation size (Line 14), which is necessary for reasons of computational efficiency. This provides us with the next time-point solution

$$\mathcal{R}(t_{s+1}) = \texttt{red}\big(\underbrace{e^{A\Delta t_s}\mathcal{R}(t_s) \oplus \mathcal{P}(\tau_s)}_{\mathcal{R}_{\text{lin}}(t_{s+1})} \boxplus \mathcal{R}_{\text{abs}}(\tau_s)\big)\,, \tag{7}$$

where $\mathcal{R}_{\text{lin}}(t_{s+1})$ is obtained by `linReach` with $e^{A\Delta t_s}\mathcal{R}(t_s)$ as the homogeneous solution and $\mathcal{P}(\tau_s)$ as the particular solution. The operator $\boxplus$ corresponds to the Minkowski sum for a linearization approach and to the exact addition as defined in [33, Prop. 10] for a polynomialization approach. The time-interval solution is

$$\mathcal{R}(\tau_{s+1}) = \underbrace{\texttt{conv}\big(\mathcal{R}(t_s), e^{A\Delta t_s}\mathcal{R}(t_s) \oplus \mathcal{P}(\tau_s)\big) \oplus \mathbf{F}_x\mathcal{R}(t_s)}_{\mathcal{R}_{\text{lin}}(\tau_{s+1})} \boxplus \mathcal{R}_{\text{abs}}(\tau_s)\,, \tag{8}$$

assuming $0 \in \mathcal{U}$, with an extension to arbitrary inputs in [2, Sec. 3.2.2]. The time-interval solution of the linearized dynamics $\mathcal{R}_{\text{lin}}(\tau_s)$ is composed of the convex hull of the reachable sets at the beginning and end of the time interval, which is then enlarged by an error $\mathbf{F}_x\mathcal{R}(t_s)$. Please note that this solution is not re-used in the next step as shown in Alg. 1.

Alg. 1 has two sources for the wrapping effect: the set $\mathcal{R}_{\text{abs}}$, which decreases in size as the time step size decreases, and the effect of the reduction operation, which is diminished when used less often due to larger time step sizes. We will refer to these sources as the *abstraction-induced* and *reduction-induced* wrapping effects, respectively. Only time-point solutions are reused in subsequent steps, therefore (7) constitutes the main formula for which both effects need to be balanced. We attempt to find an optimal compromise as shown in Fig. 1 by tuning the algorithm parameters using the methods introduced in the next section.

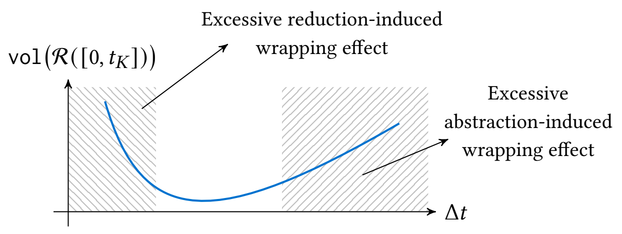


Figure 1: If the time step size Δt is too large or too small, the abstraction-induced or reduction-induced wrapping effect, respectively, are dominating.

3 SELF-PARAMETRIZATION

In this section, we introduce methods to adaptively tune the algorithm parameters used in Alg. 1 as indicated in Fig. 2. Since the described tuning methods are modular, the effect of adapting certain algorithm parameters is encapsulated within the respective modules. Hence, the presented adaptation approach constitutes a general framework, as each tuning module can simply be exchanged, e.g., if different reachable set computations or set representations are chosen. An additional advantage of the modular structure is that we do not have to consider the interplay between certain algorithm parameters, which prevents unforeseen behavior.

Within the tuning modules, a fixed set of parameters ζ is used allowing the automated tuning methods to adapt to each system. This set ζ will be discussed at the end of this section after all tuning methods have been introduced. Furthermore, we will omit the index s for the current step, as all algorithm parameters are adapted each step.

3.1 Propagation Parameters

First, we consider the tuning of the order η of the finite Taylor series of the exponential matrix, affecting the computation of the

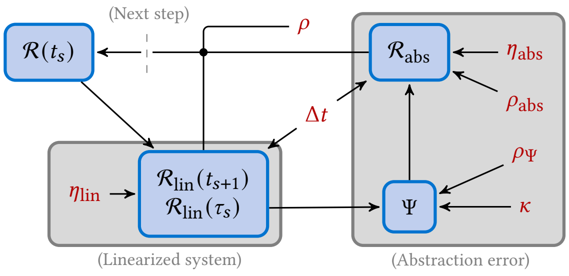


Figure 2: Main workflow for one time step in Alg. 1 and the influence of algorithm parameters (red) on different sets: The time step size Δt affects both the linearized system and the abstraction error, while the abstraction order κ only influences the abstraction error. The propagation parameters η affect the precision of the exponential matrix and the set representation parameters ρ represent the reduction operation, which is applied to various sets within one step.

sets $\mathcal{R}_{\mathrm{abs}}$ in (6) and $\mathcal{R}_{\mathrm{lin}}$ in (7)-(8) as shown graphically in Fig. 2. Since the effect of higher-order terms eventually vanishes, we conservatively determine the specific orders η_{lin} and η_{abs} by truncating the respective sums once the change is negligibly small.

Since the error in $\mathcal{R}_{\mathrm{lin}}(\tau_s)$ is dominated by the term $\mathbf{F}_x\mathcal{R}(t_s)$, which can be computed according to [2, Prop. 3.1] as

$$\mathbf{F}_x = \bigoplus_{\ell=1}^{\eta_{\mathrm{lin}}} \underbrace{\left[(\ell^{\frac{-\ell}{\ell-1}} - \ell^{\frac{-1}{\ell-1}})\Delta t^\ell, 0\right]\frac{A^\ell}{\ell!}}_{:=T^{(\ell)}} \oplus \mathbf{E}, \tag{9}$$

with $\mathbf{E}$ as referenced above, we obtain η_{lin} by setting a threshold $0 < \zeta_{T,\mathrm{lin}} \ll 1$ for the change over successive terms $T^{(\ell)}$:

$$\eta_{\mathrm{lin}} = \min \ell \quad \text{such that} \quad 1 - \frac{\|T^{(\ell-1)}\|_F}{\|T^{(\ell)}\|_F} \leq \zeta_{T,\mathrm{lin}}\,. \tag{10}$$

Similarly, we obtain η_{abs} by truncating the sum in (6) once the relative change in size between two successive truncated sums is sufficiently small. This is achieved by the following criterion, where $0 < \zeta_{T,\mathrm{abs}} \ll 1$:

$$\eta_{\mathrm{abs}} = \min \ell \quad \text{such that} \quad \max_{i\in\{1,\dots,n\}} \frac{d_i\big(\mathrm{box}(\mathcal{R}^{(\ell+1)}_{\mathrm{abs}})\big)}{d_i\big(\mathrm{box}(\mathcal{R}^{(\ell)}_{\mathrm{abs}})\big)} \leq \zeta_{T,\mathrm{abs}}\,, \tag{11}$$

where $\mathcal{R}^{(k)}_{\mathrm{abs}}$ denotes the sum in (6) truncated at order k. The split into two different η for $\mathcal{R}_{\mathrm{lin}}$ and $\mathcal{R}_{\mathrm{abs}}$ is justified by the different values to which η_{lin} and η_{abs} are tuned, as discussed in the numerical examples in Sec. 4. Furthermore, the evaluation of both criteria (10) and (11) can be seamlessly integrated into the computation of the respective terms yielding negligible computational overhead.

3.2 Set Representation

In this work, we restrict the error due to reducing the set representation. For a linearization approach, it suffices to use convex set representations, such as support functions, polytopes, or zonotopes. In this work, we will use zonotopes as they have proven to be a good choice for the linearization approach due to the efficient and exact computation of the two mainly used operations—linear map and Minkowski sum.

Definition 1. *(Zonotopes) [27, Def. 1] Given a center $c \in \mathbb{R}^n$ and $\gamma \in \mathbb{N}$ generator vectors $G = [g^{(1)}, \dots, g^{(\gamma)}]$, we define a zonotope*

$$\mathcal{Z} := \left\{ x \in \mathbb{R}^n \;\middle|\; x = c + \sum_{i=1}^{\gamma} \alpha_i\, g^{(i)}\,, -1 \leq \alpha_i \leq 1 \right\} \tag{12}$$

as well as its order $\rho := \frac{\gamma}{n}$ and the shorthand $\langle c, G\rangle_Z$. □

We now measure the enlargement caused by the reduction of the representation size. Let us first consider the following lemma:

Lemma 3.1. *Let $f : C_1 \times C_2 \to \mathcal{I}$, $g : C_1 \to \mathcal{I}$ be continuous functions, where $C_1, C_2 \subset \mathbb{R}^n$ are two compact sets and $\mathcal{I} \subset \mathbb{R}^n$ is a compact interval. If for each $x \in C_1$, it holds that $\min_{y\in C_2} f(x,y) \leq g(x)$, then*

$$\max_{x\in C_1} \min_{y\in C_2} f(x,y) \leq \max_{x\in C_1} g(x)\,.$$

Proof. Let x^* be a point in C_1 for which the maximum of $\min_{y\in C_2} f(x,y)$ is attained. By assumption, it follows that $\min_{y\in C_2} f(x^*,y) \leq g(x^*)$ and thus

$$\max_{x\in C_1} \min_{y\in C_2} f(x,y) = \min_{y\in C_2} f(x^*,y) \leq g(x^*) \leq \max_{x\in C_1} g(x)\,. \qquad \square$$

The following theorem over-approximates the Hausdorff distance d_H between the original and reduced zonotope based on the commonly used box over-approximation of generators:

Theorem 3.2. *Let $\mathcal{Z} = \langle c, G\rangle_Z \subset \mathbb{R}^n$ be a zonotope and $\mathcal{Z}_B := \mathrm{box}(\mathcal{Z}) = \langle c, G_B\rangle_Z \supseteq \mathcal{Z}$ its box over-approximation. Due to the containment $\mathcal{Z} \subseteq \mathcal{Z}_B$, the Hausdorff distance d_H is given by*

$$d_H(\mathcal{Z}, \mathcal{Z}_B) = \max_{x_B\in\mathcal{Z}_B} \min_{x\in\mathcal{Z}} \|x_B - x\|_2\,. \tag{13}$$

This distance is over-approximated by

$$d_H(\mathcal{Z}, \mathcal{Z}_B) \leq \omega(\mathcal{Z}) := 2\left\|\sum_{k=1}^{\gamma} \widehat{g}^{(k)}\right\|_2 \tag{14}$$

$$\text{with} \quad \widehat{g}^{(k)}_i = \begin{cases} |g^{(k)}_i|, & \text{if } k \neq i^* \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where i^* is the (first) index for which $g^{(k)}_{i^*} = \left\|g^{(k)}\right\|_\infty$.

Proof. Let us express each point $x_B \in \mathcal{Z}_B$ as

$$x_B = M_1|g^{(1)}| + \dots + M_\gamma|g^{(\gamma)}|\,,$$

where each M is a diagonal matrix with entries $m_{ii} \in [-1, 1]$. We can write the difference between any $x_B \in \mathcal{Z}_B$ and $x \in \mathcal{Z}$ as

$$x_B - x = \left(M_1|g^{(1)}| - \alpha_1 g^{(1)}\right) + \dots + \left(M_\gamma|g^{(\gamma)}| - \alpha_\gamma g^{(\gamma)}\right)$$

with $\alpha_k \in [-1, 1]$. We now obtain a bound on $x_B - x$ by choosing a specific α_k for each $g^{(k)}$, namely,

$$\alpha_k = m_{i^*i^*}\,\mathrm{sgn}\left(g^{(k)}_{i^*}\right) \tag{16}$$

with an individual i^* as in Theorem 3.2 for each k. This choice of α_k allows us to eliminate the largest possible entry in $v^{(k)} = M_k|g^{(k)}| - \alpha_k g^{(k)}$, for which we obtain the bound

$$v^{(k)}_i \in \begin{cases} \left[-2|g^{(k)}_i|, 2|g^{(k)}_i|\right], & \text{if } i \neq i^* \\ 0, & \text{otherwise,} \end{cases}$$

which we can rewrite using (15) to $v_i^{(k)} \in [-2\widehat{g}_i^{(k)}, 2\widehat{g}_i^{(k)}]$. Applying (16) to each generator, we obtain the bound

$$x_B - x = v^{(1)} + ... + v^{(\gamma)} \in [-2\widehat{z}, 2\widehat{z}]\,,$$

where $\widehat{z} := \widehat{g}^{(1)} + ... + \widehat{g}^{(\gamma)}$ and consequently,

$$\|x_B - x\|_2 \leq \|2\widehat{z}\|_2 = 2\|\widehat{z}\|_2\,.$$

The above bound holds for any $x_B \in \mathcal{Z}_B$ and therefore, the assumption of Lemma 3.1 is fulfilled. Thus, we obtain (14). □

Using this over-approximation, we now deduce a heuristic which attempts to reduce as many generators as possible while respecting a given threshold for the Hausdorff distance between the original and reduced set: Given a zonotope $\mathcal{Z} = \langle c, G\rangle_Z$, we sort the generators in $G \in \mathbb{R}^{n\times\gamma}$ by the metrics

$$\|g^{(1)}\|_1 - \|g^{(1)}\|_\infty \leq ... \leq \|g^{(\gamma)}\|_1 - \|g^{(\gamma)}\|_\infty \tag{17}$$

originally proposed in [27]. Following this ordering, we pick the first $N \leq \gamma$ generators in (14) until we reach the upper bound

$$\omega_{\max}(\mathcal{Z}) = \zeta_Z \left\|d\big(\mathsf{box}(\mathcal{Z})\big)\right\|_2 \tag{18}$$

where we use a fixed fraction $0 < \zeta_Z \ll 1$ of the diagonal of the box over-approximation of $\mathcal{Z}$. The exact Hausdorff distance between the original and the reduced set is smaller than $\omega_{\max}(\mathcal{Z})$ by Theorem 3.2.

For a polynomialization approach, a non-convex set representation is required since convex sets would almost nullify the benefits of the polynomial abstraction. To obtain tighter results, non-convex set representations also have to be closed under higher-order maps as is the case for Taylor models [15, Sec. II.] or polynomial zonotopes [33, Def. 1]. We choose the latter to exploit their similarities with zonotopes. For polynomial zonotopes we apply the reduction method in [33, Prop. 10], which is based on order reduction for zonotopes so that the bound in (18) can be enforced.

3.3 Abstraction Order

The abstraction order κ in (2) only influences the size of the abstraction error Ψ according to Fig. 2. A larger κ is computationally more demanding as we require to evaluate higher-order maps, but also yields a smaller abstraction error Ψ. This does not necessarily hold for convex set representations since they are not closed under higher-order maps.

Since the linearization approach uses convex set representations, we restrict the admissible values of the abstraction order to $\kappa = \{1, 2\}$ because the highly over-approximative evaluation of cubic or higher-order maps does not justify the additionally required computational effort. The abstraction error is evaluated on the time-interval solution $\mathcal{R}_{\text{lin}}(\tau_s)$, see line 9 in Alg. 1. In each step, the abstraction error Ψ for both $\kappa = \{1, 2\}$ is computed for the time-interval solution $\mathcal{R}_{\text{lin}}(\tau_s)$ using the optimal time step size Δt_*, which will be introduced in Sec. 3.4. The following selection criterion is applied in each step to compute the abstraction order κ for the next step:

$$\kappa \leftarrow \begin{cases} 1, & \text{if } \forall i \in \{1,...,n\} \text{ with } \Psi_i > 0 : \frac{d(\mathsf{box}(\Psi_i(\kappa=2)))}{d(\mathsf{box}(\Psi_i(\kappa=1)))} \geq \zeta_K \in (0,1) \\ 2, & \text{otherwise,} \end{cases} \tag{19}$$

so that we use the more efficient approach for $\kappa = 1$ whenever the loss in accuracy is manageable. The closer ζ_K is to 1, the more conservative the selection becomes, i.e., the more often $\kappa = 2$ will be chosen resulting in both a tighter result as well as longer computation times. For the first step, we use the initial set $\mathcal{R}(t_0) = \mathcal{X}^0$ to compute Ψ and immediately evaluate (19) to compute the first abstraction order κ.

In the polynomialization approach, the non-convex set representation is closed under all higher-order maps [33]. Capturing these nonlinear mappings results in a strong increase in the set representation size. Hence, we restrict the abstraction order in this case to its lowest setting $\kappa = 2$ as higher orders require more reduction increasing the size of the reachable set.

3.4 Time Step Size

The tuning of the time step size Δt is the crucial factor for the success of reachability analysis since it dominates the computation of the two main sets $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ as indicated in Fig. 2. In order to obtain a tight reachable set, we require to tune Δt so that the trade-off between the abstraction-induced and reduction-induced wrapping effects is resolved in a near-optimal way, see Fig. 1. An important prerequisite for tuning Δt is the estimation of the influence of the reduction-induced wrapping by an upper bound in each step as established in Sec. 3.2.

The main idea is to tune the time step size Δt by solving a convex optimization problem which models both wrapping effects. Since the influence of both effects always increases the size of the reachable set, we estimate this size at the end of a finite time horizon Δ computed by different $\Delta t_k = \frac{\Delta}{k}, k \geq 1$ and choose the optimal Δt_* which yields the minimal size.

The reachable set after time Δ is computed by repeatedly applying (7), where we only take the terms contributing to the wrapping effects into account. Furthermore, we explicitly consider $k \in \mathbb{R}$, which requires to consider a last incomplete step of length $q\Delta t_k = (k - \lfloor k \rfloor)\Delta t_k$. Correspondingly, $e^{A\Delta t_k}$ and $\mathcal{R}_{\text{abs}}$ are scaled to $e^{Aq\Delta t_k}$ and $q\mathcal{R}_{\text{abs}}$. This yields

$$\begin{aligned}\tilde{\mathcal{R}}(t+\Delta) = \mathsf{red}\big(e^{Aq\Delta t_k}\mathsf{red}\big(e^{A\Delta t_k} ...\\ \mathsf{red}\big(e^{A\Delta t_k}\mathcal{R}(t) \oplus \mathcal{R}_{\text{abs}}\big) ... \oplus \mathcal{R}_{\text{abs}}\big) \oplus q\mathcal{R}_{\text{abs}}\big)\,.\end{aligned} \tag{20}$$

Estimating $\tilde{\mathcal{R}}(t+\Delta)$. In order to estimate the size of $\tilde{\mathcal{R}}(t+\Delta)$ efficiently, we introduce the following simplifications which allow us to derive a scalar optimization function for Δt_*:

(1) The size of a set $\mathcal{S}$ is measured by its radius

$$r(\mathcal{S}) = \frac{1}{2}\,\|d(\mathsf{box}(\mathcal{S}))\|_2\,.$$

This allows us to replace the respective sets by the scalar variables $r_0 = r\big(\mathcal{R}(t)\big)$ and $r_{\text{abs},k} = r\big(\mathcal{R}_{\text{abs}}(\Delta t_k)\big)$.

(2) The effect of the exponential matrix is captured by its determinant. The scaling over the entire finite horizon can be estimated by $\det(e^{A\Delta}) = e^{\text{tr}(A\Delta)}$ assuming that the matrix A does not change over time. The average scaling factor for each step of length Δt_k is

$$\zeta_P^{\frac{1}{k}} = \left(e^{\text{tr}(A\Delta)}\right)^{\frac{1}{k}} \tag{21}$$

and consequently, the scaling of the last incomplete step is $\zeta_P^{\frac{q}{k}}$.

(3) The enlargement caused by the reduction is measured by multiplying the radius by $(1+2\zeta_Z)$ following (18). The factor for the last step of length $q\Delta t_k$ is scaled to $(1+2\zeta_Z)^q$.

Using these simplifications, we can rewrite (20) in a recursive formula to estimate the set size of $\tilde{\mathcal{R}}(t+j\Delta t_k)$, $1 \leq j \leq \lfloor k \rfloor$ starting with the set size estimate $r_\mathrm{R}(t) = r_0$ at time t:

$$r_\mathrm{R}(t+j\Delta t_k) = (1+2\zeta_Z)\big(\zeta_P^{\frac{1}{k}} r_\mathrm{R}(t+(j-1)\Delta t_k) + r_{\mathrm{abs},k}\big) . \qquad (22)$$

Applying this recursion $\lfloor k \rfloor$ times and including the last step of length $q\Delta t_k$, we obtain an estimate $r_\mathrm{R}(t+\Delta)$ for the size of the reachable set after time Δ:

$$r_\mathrm{R}(t+\Delta) = (1+2\zeta_Z)^q \cdot$$
$$\Big(\zeta_P^{\frac{q}{k}} \underbrace{(1+2\zeta_Z)\big[\zeta_P^{\frac{1}{k}} ...(1+2\zeta_Z)(\zeta_P^{\frac{1}{k}} r_0 + r_{\mathrm{abs},k})... + r_{\mathrm{abs},k}\big]}_{\overset{(22)}{=} r_\mathrm{R}(t+\lfloor k \rfloor \Delta t_k)} + q\, r_{\mathrm{abs},k}\Big) .$$

We first simplify the first $\lfloor k \rfloor$ steps to

$$r_\mathrm{R}(t+\Delta) = (1+2\zeta_Z)^q\Big(\zeta_P^{\frac{q}{k}}\big[r_0(1+2\zeta_Z)^{\lfloor k \rfloor}\zeta_P^{\frac{\lfloor k \rfloor}{k}}$$
$$+ r_{\mathrm{abs},k}\sum_{i=1}^{\lfloor k \rfloor}(1+2\zeta_Z)^i\zeta_P^{\frac{i-1}{k}}\big] + q\, r_{\mathrm{abs},k}\Big),$$

leaving only the last step of length $q\Delta t_k$, which we now include and rearrange to

$$r_\mathrm{R}(t+\Delta) = r_0(1+2\zeta_Z)^k\zeta_P + r_{\mathrm{abs},k}\,\zeta_{\mathrm{P,Z}}(k) \qquad (23)$$
$$\text{with} \quad \zeta_{\mathrm{P,Z}}(k) = \sum_{i=1}^{\lfloor k \rfloor}(1+2\zeta_Z)^{q+i}\,\zeta_P^{\frac{q+i-1}{k}} + q\,(1+2\zeta_Z)^q$$

containing all factors affecting $r_{\mathrm{abs},k}$.

Estimating $r_{abs,k}$. The evaluation of (23) would require us to compute $r_{\mathrm{abs},k}$ for each k. To save computational costs, we approximate $r_{\mathrm{abs},k}$ by multiplying $r_{\mathrm{abs},1}$ obtained by $\Delta t = \Delta$ with a scaling factor which models the behavior of $\mathcal{R}_\mathrm{abs}$ over Δt.

Proposition 1. *Scaling Δt by a factor $\zeta_\delta \in (0,1)$ shrinks the i-th entry of the diameter of $\mathcal{R}_{abs}$, i.e., $d_i\big(\mathsf{box}(\mathcal{R}_{abs})\big)$, by $\zeta_\delta^{\lambda_i}$ with $\lambda_i \in \mathbb{N}$ when Δt goes to 0:*

$$\forall i \in \{1,...,n\} \; : \; \lim_{\Delta t \to 0} \frac{d_i\big(\mathsf{box}(\mathcal{R}_{abs}(\zeta_\delta\Delta t))\big)}{d_i\big(\mathsf{box}(\mathcal{R}_{abs}(\Delta t))\big)} = \zeta_\delta^{\lambda_i} . \qquad (24)$$

Proof. We insert (6) for the computation of $\mathcal{R}_\mathrm{abs}$ in (24) and remove $d(\cdot)$ and $\mathsf{box}(\cdot)$ since these are linear operators up to a constant factor, which cancels out. We factor out Δt to obtain

$$\lim_{\Delta t \to 0} \frac{\Delta t\Big(\bigoplus_{k=0}^{\eta_\mathrm{abs}} \frac{\zeta_\delta^{k+1}\Delta t^k}{(k+1)!}A^k\Psi \oplus \mathbf{E}(\zeta_\delta\Delta t, \eta_\mathrm{abs})\,\zeta_\delta\,\Psi\Big)_i}{\Delta t\Big(\bigoplus_{k=0}^{\eta_\mathrm{abs}} \frac{\Delta t^k}{(k+1)!}A^k\Psi \oplus \mathbf{E}(\Delta t, \eta_\mathrm{abs})\,\Psi\Big)_i} = \frac{\zeta_\delta^{\lambda_i}\Psi_i}{\Psi_i}$$

where $\lambda_i = j+1$, with j being the first non-negative integer such that $(A^j\Psi)_i \neq \{0\}$. □

Two properties follow immediately from (24): First, the maximum possible scaling factor over Δt is given by ζ_δ due to the lower bound $\lambda_i = 1$ attained for $\Delta t \to 0$. Second, the factors ζ_δ^k in the sum and the remainder term yield a superlinear decrease of the ratio in (24) over Δt, i.e., $d_i\big(\mathsf{box}(\mathcal{R}_\mathrm{abs}(\zeta_\delta\Delta))\big) < \zeta_\delta\, d_i\big(\mathsf{box}(\mathcal{R}_\mathrm{abs}(\Delta))\big)$. For later derivations, we define the gain

$$\varphi(\Delta t) = \max_{i\in\{1,...,n\}} \frac{d_i\big(\mathsf{box}(\mathcal{R}_\mathrm{abs}(\zeta_\delta\Delta t))\big)}{d_i\big(\mathsf{box}(\mathcal{R}_\mathrm{abs}(\Delta t))\big)} . \qquad (25)$$

Let us discuss the values of $\varphi(\Delta t)$ for $\lim_{\Delta t\to 0}\varphi(\Delta t)$:

- Linearization approach: For the limit gain, we have $\lim_{\Delta t\to 0}\varphi(\Delta t) = \zeta_\delta$. This follows from the proof of Prop. 1, where we have $\lambda_i = 1$ for all nonlinear equations $\dot{x}_i(t)$ since $(A^0\Psi)_i = \Psi_i \neq \{0\}$.
- Polynomialization approach: The limit gain $\lim_{\Delta t\to 0}\varphi(\Delta t)$ depends on the specifics of A and Ψ, however, it is bounded by $\lim_{\Delta t\to 0}\varphi(\Delta t) \geq \zeta_\delta^\kappa$, as all terms higher than the abstraction order κ are considered as an error and thus the minimum decrease is given by ζ_δ^κ.

Since the superlinearity of (24) extends to (25), the worst-case approximation of the gain φ over Δt is given by linearly interpolating between $\varphi(\Delta t = \Delta) = \varphi_1$ and $\lim_{\Delta t\to 0}\varphi(\Delta t) = \zeta_\delta$:

$$\varphi(\Delta t) \approx \zeta_\delta + \frac{\varphi_1 - \zeta_\delta}{\Delta}\Delta t , \qquad (26)$$

which will be justified in Sec. 4, see Fig. 3. This linear interpolation for φ constitutes the worst-case gain causing us to never underestimate the optimal value of Δt. As a consequence of the interpolation, we only need to compute φ_1 to estimate any $r_{\mathrm{abs},k}$ based on $r_{\mathrm{abs},1}$ and the dependence on φ given by (26). We define $k' \in \mathbb{N}$ as the number of times Δ has been scaled by a fixed $\zeta_\delta \in (0,1)$. Hence, $k = \zeta_\delta^{-k'} \in \mathbb{R}$ is the number of times Δt_k divides into Δ and using

$$\varphi_j = \varphi(\zeta_\delta^{j-1}\Delta) = \zeta_\delta + (\varphi_1 - \zeta_\delta)\zeta_\delta^{j-1} \qquad (27)$$

we obtain an estimate for $r_{\mathrm{abs},k}$:

$$k\, r_{\mathrm{abs},k} = \varphi_1 \cdot ... \cdot \varphi_{k'}\, r_{\mathrm{abs},1} \Longrightarrow r_{\mathrm{abs},k} := \frac{r_{\mathrm{abs},1}}{k}\prod_{j=1}^{k'}\varphi_j . \qquad (28)$$

In the tuning algorithm for Δt, we compute φ_1 given $r_{\mathrm{abs},1}$ and $r_{\mathrm{abs},k}$ by solving the following implicit equation for φ_1:

$$\varphi_1 \cdot \big(\zeta_\delta + (\varphi_1 - \zeta_\delta)\zeta_\delta\big) \cdot ...\big(\zeta_\delta + (\varphi_1 - \zeta_\delta)\zeta_\delta^{k'-1}\big) = k\frac{r_{\mathrm{abs},k}}{r_{\mathrm{abs},1}} . \qquad (29)$$

Optimization function. Inserting (28) in (23) yields

$$r_\mathrm{R}(t+\Delta) = r_0(1+2\zeta_Z)^k\zeta_P + \frac{r_{\mathrm{abs},1}}{k}\,\zeta_{\mathrm{P,Z}}(k)\prod_{j=1}^{k'}\varphi_j , \qquad (30)$$

which we minimize to obtain the optimal time step size

$$\Delta t_* = \Delta\,\zeta_\delta^{k'_*} \qquad (31)$$
$$\text{where} \quad k'_* = \operatorname*{arg\,min}_{k'\in\mathbb{N}} r_\mathrm{R}(t+\Delta) .$$

Since we know that this function has one optimum by construction, we simply increase k' until that optimum is surpassed as the computation time to evaluate (30) is so low that sophisticated algorithms are not required. The obtained value for Δt constitutes an upper bound as it is assumed that the entire margin for the reduction is used each step which is an over-approximation of its true influence. We will show an example evaluation of (30) in Sec. 4.

Finite optimization horizon. Lastly, we require to set the finite horizon Δ over which we evaluate the cost function (30). Since the behavior of $r_{\text{abs},k}$ over shrinking Δt constitutes a key part in the computation of Δt_*, we want it to be captured as precisely as possible. Naturally, the proposed linear interpolation reflects the true behavior more closely if the computed gain φ_1 is sufficiently close to the limit gain at $\Delta t \longrightarrow 0$. Therefore, we determine Δ by

$$\Delta = \min \tau \quad \text{such that} \quad \varphi_1(\tau) \geq \zeta_\Delta \, . \tag{32}$$

Tuning algorithm for Δt. The tuning of the time step size is summarized in Alg. 2. In the initial step $s = 1$, we first decrease an arbitrarily initialized Δt (Line 2) until the condition in (32) is met, yielding Δ with the associated error $\mathcal{R}_{\text{abs}}(\Delta)$ and its scalar correspondence $r_{\text{abs},1}$ as well as φ_1 in the process (Lines 3-7). This allows us to compute the optimal time step size Δt_* for the first step (Line 8).

From the second step onward, the computation of Δt_* is simplified in order to minimize the computational effort: First, we update the finite time horizon Δ (Line 10) to approximate the value in (32). Next, the set $\mathcal{R}_{\text{abs}}(\Delta)$ as well as its scalar correspondence $r_{\text{abs},1}$ are computed (Line 11), which are then used to compute the optimal time step size Δt_* (Line 12). At the end, we update the value of φ_1 for the next step (Line 13). Note that within the propagation using Δt_*, the prediction of the abstraction order κ for the next step as explained in Sec. 3.3, is also made.

Algorithm 2 Tuning of the time step size Δt.

```
Input: $\varphi_1$ and $\Delta$ of previous step (only if $s > 1$), $r_0$, $\zeta_Z$, $\zeta_P$, $\zeta_\delta$, $s$
Output: Optimal time step size $\Delta t_*$, $\varphi_1$
 1: if $s = 1$ then
 2:     Initialize $\Delta t$                                          ▹ Arbitrary initialization
 3:     while $\varphi_1 < \zeta_\Delta$ do
 4:         $\Delta \leftarrow \zeta_\delta \Delta$                                      ▹ Decrease $\Delta t$
 5:         Compute $\mathcal{R}_{\text{abs}}(\Delta), \mathcal{R}_{\text{abs}}(\zeta_\delta \Delta)$   ▹ Note: $\mathcal{R}_{\text{abs}}$ reusable
 6:         Compute $\varphi_1$ by (25)                       ▹ Behavior of $\mathcal{R}_{\text{abs}}$ over $\Delta t$
 7:     end while
 8:     Compute $\Delta t_*$ by (31)                               ▹ Optimal time step size
 9: else
10:     $\Delta \leftarrow \Delta \frac{\zeta_\delta - \zeta_\Delta}{\zeta_\delta - \varphi_1}$                                 ▹ Approximation of (32)
11:     Compute $\mathcal{R}_{\text{abs}}(\Delta)$ and $r_{\text{abs},1}$       ▹ Error using finite horizon
12:     Compute $\Delta t_*$ by (31)                               ▹ Optimal time step size
13:     Compute $\varphi_1$ by (29)                                ▹ Update for next step
14: end if
```

Fixing the global parameters ζ. The global parameters ζ used in the adaptation of all algorithm parameters have been fixed to suitable values in Table 1, thereby allowing the respective tuning methods to adapt the algorithm parameters according to the demands of the current system behavior. These fixed values for each ζ are well-suited to produce tight results for a wide variety of different nonlinear systems as shown in the next section. Thus, there is no more manual tuning effort required for all considered problems. With the further development of the tuning methods, the value of a specific ζ might change, but the general applicability remains.

Table 1: Setting of the parameters ζ.

Approach	$\zeta_{T,\text{lin}}$	$\zeta_{T,\text{abs}}$	ζ_Z	ζ_K	ζ_Δ	ζ_δ
Linearization	0.0005	0.005	0.0005	0.90	0.85	0.90
Polynomialization	0.0005	0.005	0.0002	—	0.80	0.90

4 NUMERICAL EXAMPLES

In this section, we apply the adaptive parameter tuning presented in the previous section to numerous benchmark examples taken from various sources [6, 14, 17, 26]. The adaptation of the algorithm parameters over time is shown and discussed for selected benchmark systems. All computations have been performed in MATLAB on an Intel® Core™ i7-9850 CPU @2.59GHz with 32GB memory.

4.1 Evaluation of the Optimization Function

We first want to offer additional insights concerning the optimization function (30), which balances the wrapping effects introduced in Sec. 2.2. An important part is the approximation of the abstraction-induced wrapping, represented by the variable φ, see (27). In Fig. 3, the values of φ of all systems in this section have been computed over decreasing Δt starting at Δ for the first step at $t = 0$. The generality of the worst-case assumption made in Sec. 3.4 is justified by the dashed lines representing the linear interpolation expressed in (26). All systems converge towards $\varphi = \zeta_\delta$ using the linearization approach, whereas for the polynomialization approach, all systems are still bounded by the worst-case assumption given by the linear interpolation between $\varphi(\Delta) \approx \zeta_\Delta$ and $\lim_{\Delta t \to 0} \varphi(\Delta t) = \zeta_\delta$, despite their individually distinct behaviors over Δt. This follows the analytical derivations made in the introduction of φ in the previous section.

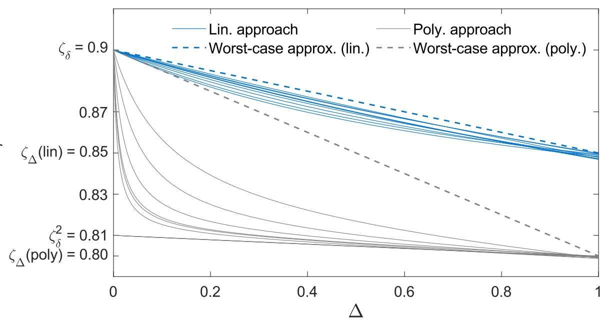


Figure 3: Computation of φ by (25) for all systems in Sec. 4 and both approaches (linearization in blue and polynomialization in gray) over $0 < \Delta t < \Delta$ (normalized to $[0, 1]$), with Δ computed by (32).

Next, we want to show an example evaluation of the optimization function. Therefore, consider the following example:

Example 4.1. Jet Engine [10, (19)]

$$\dot{x} = \begin{pmatrix} -x_2 - 1.5x_1^2 - 0.5x_1^3 - 0.5 \\ 3x_1 - x_2 \end{pmatrix}, \qquad \mathcal{X}^0 = \begin{pmatrix} [0.90, 1.10] \\ [0.90, 1.10] \end{pmatrix}$$

with the time horizon $t_K = 8s$. □

Fig. 4 shows the optimization function $r_R(t+\Delta)$ and its minimum at Δt_* evaluated at the first step of example 4.1. For the finite horizon, the heuristics in (32) yield $\Delta = 0.0226$. We clearly recognize the two wrapping effects as Fig. 4 exhibits a qualitatively similar behavior compared to Fig. 1, which serves as the basis for the adaptation of Δt. Our approach returns an optimal value of $\Delta t_* = 0.0071$ for the first step.

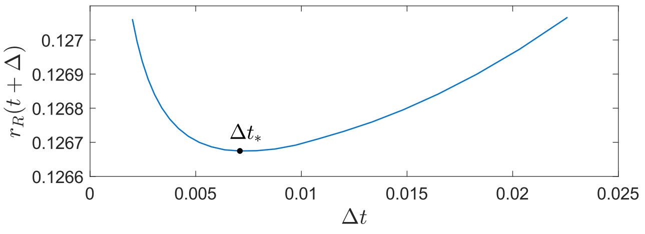


Figure 4: Evaluation of the cost function $r_R(t+\Delta)$ (30) over different values for Δt at $t = 0$ for example 4.1.

4.2 ARCH Benchmarks

We analyzed the production-destruction benchmark (PRDE20) and the Laub-Loomis benchmark (LALO20) from the ARCH competition [26]. This allows us to compare our results with other reachability tools using algorithm parameters tuned by experts. We first consider the PRDE20 benchmark.

Example 4.2. (PRDE20) This benchmark models a biogeochemical reaction, describing an algal bloom transforming nutrients (x_1) into detritus (x_3) using phytoplankton (x_2) [35, Sec. 3]. The dynamics are presented in [26, Sec. 3.1.1], the initial set is $\mathcal{X}^0 = ([9.50, 10.00], 0.01, 0.01)^T$, and the time horizon is $t_K = 100s$. □

Fig. 5 shows the reachable sets computed using the linearization approach. We observe two fairly linear regions interrupted by a sharp turn. Thus, we expect the automated parameter tuning to adapt each algorithm parameter to the demands of the current dynamics.

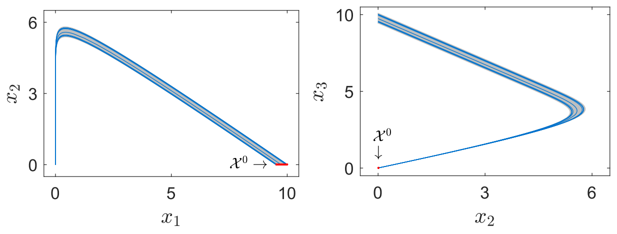


Figure 5: Reachable set $\mathcal{R}([0, t_K])$ of example 4.2 in gray, initial set $\mathcal{X}^0$ in red, simulations in blue.

The algorithm parameters over time are shown in Fig. 6: In the region of the sharp turn ($10.6s < t < 11.6s$), the time step size Δt (Fig. 6a) becomes very small since the optimization function estimates a smaller total error by reducing Δt as the decrease in the abstraction error outweighs the increase in the reduction error. After the sharp turn, Δt greatly increases as the abstraction error becomes small. Each of the truncation orders η_{lin} and η_{abs} (Fig. 6b) reaches their maximum at the sharp turn as the dynamics there require more terms within the Taylor series of the exponential matrix to provide satisfactory accuracy.

The zonotope order ρ (Fig. 6c) increases at the turn because we cannot reduce many generators without inducing large over-approximations. Afterwards, it reaches its minimum where it stays until the end of the time horizon since the set is accurately described using a small number of generators. Over the whole time horizon, the zonotope order does not exceed 20, enabling a very efficient evaluation of the set operations contained within each step. Concerning the abstraction order κ, the adaptive tuning in (19) resulted in $\kappa = 2$ for $t \in [0, 13.45]$ and $\kappa = 1$ for $t \in [13.45, 100]$, which confirms the rather linear system behavior after the sharp turn.

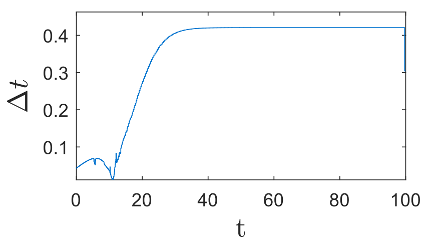
Table 2: Comparing our approach with different reachability tools on ARCH benchmarks using the tightness measurements $\mu_1 = \texttt{vol}(\texttt{box}(\mathcal{R}(t_K)))$ and $\mu_2 = l_4$, where $l = d(\texttt{box}(\mathcal{R}(t_K)))$ as in [26].

Tool (Language)	**PRDE20**		**LALO20**	
	Time	μ_1	Time	μ_2
Lin. Approach (Matlab)	7.0s	8.0e−21	8.9s	0.045
Poly. Approach (Matlab)	6.5s	1.0e−19	19s	0.025
Ariadne (C++)	8.6s	1.7e−13	664s	0.058
CORA (Matlab)	16s	1.2e−21	7.6s	0.04
DynIbex (C++)	12s	3.9e−17	27s	0.40
Flow* (C++)	4.1s	8.0e−21	2.3s	0.06
Isabelle/HOL (SML)	11s	3.3e−20	13s	0.48
JuliaReach (Julia)	1.5s	3.3e−20	1.5s	0.017

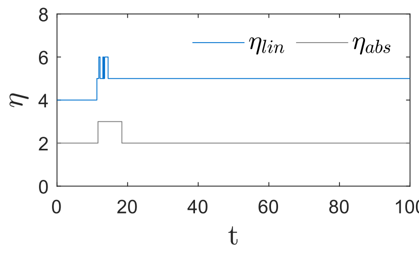
Table 2 shows the computation time and the tightness measurement by volume of the final set for both approaches using adaptive parameter tuning and other reachability tools. Due to the large ratio of the largest to the smallest time step size and consequently the saving of many time steps, both approaches yield similar computation times compared to the expert-tuned tools, many of which are written in languages such as C++ or Julia, which are faster than MATLAB. The tightness of the reachable sets computed for each approach using adaptive parameter tuning is among the top results obtained by expert tuning, thereby demonstrating the high accuracy of the presented tuning methods. Next, we consider the LALO20 benchmark.

Example 4.3. (LALO20) This benchmark system models changes in enzymatic activities [38, (1-7)], whose dynamics are given in [26, Sec. 3.3.1]. For the initial set, we enlarge the point $x(0) = (1.2, 1.05, 1.5, 2.4, 1, 0.1, 0.45)^T$ by the uncertainty $W = 0.05$ to obtain $\mathcal{X}^0 = [x(0) - W, x(0) + W]$. The time horizon is $t_K = 20s$. □

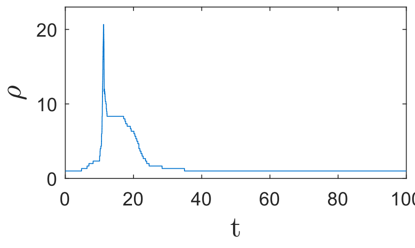
Using the polynomialization approach for the analysis, the time step size Δt and the polynomial zonotope order ρ are plotted over time in Fig. 7: Due to the size of the first few reachable sets, Δt is smallest in the beginning and then gradually increases. The



(a) Time step size Δt.



(b) Truncation orders η_{lin}, η_{abs}.



(c) Zonotope order ρ.

Figure 6: Algorithm parameters of example 4.2 over time.

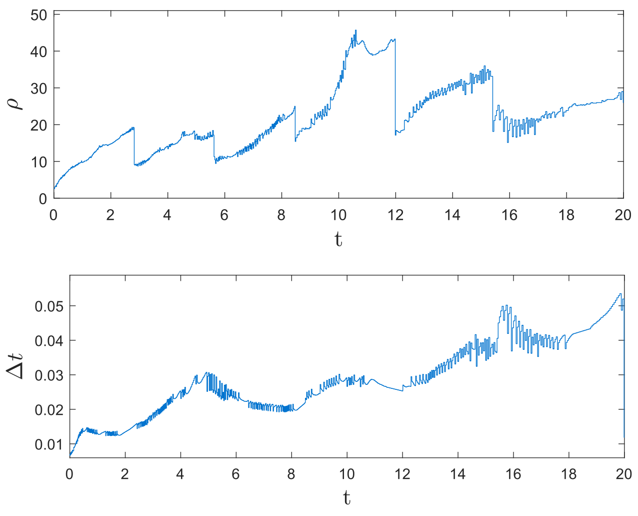


Figure 7: Time step size Δt and polynomial zonotope order ρ of example 4.3 over time.

sudden drops in ρ occur due to the restructuring of all independent generators into the dependent generator matrix [33, Prop. 17]. Other than that, the order ρ remains below 40 at all times, keeping the set operations efficient without compromising the tightness of the reachable sets. The orders η do not change a lot over the whole time horizon, with $\eta_{\text{lin}} = \{4, 5\}$ and $\eta_{\text{abs}} = 2$ remaining constant throughout. The abstraction order is fixed at $\kappa = 2$ as stated in Sec. 3.3.

Comparing the results obtained by our adaptive parameter tuning with other reachability tools, we see an average performance in terms of the computation time. This is due to the high system dimension for which the evaluation of the abstraction error Ψ and the reduction of the set representation size are computationally demanding. However, both approaches return tighter results, except for JuliaReach.

4.3 Quantitative Performance Analysis

While we discussed some benchmarks in detail in the previous section, we now analyze the performance of our tuning approach on many different benchmarks. For all systems, we provide the longest edge of the box over-approximation of the final set, that is,

$$l_{\max} = \max_{i\in\{1,\ldots,n\}} d_i\big(\text{box}\big(\mathcal{R}(t_K)\big)\big) \tag{33}$$

as well as a tightness measure proposed in [18, Sec. VI.]

$$\gamma_{\min} = \min_{i\in\{1,\ldots,n\}} \frac{d_i\big(\text{box}\big(\mathcal{R}_{\text{sim}}(t_K)\big)\big)}{d_i\big(\text{box}\big(\mathcal{R}(t_K)\big)\big)}\,, \tag{34}$$

where $\mathcal{R}_{\text{sim}}(t_K)$ denotes the set of states at t_K of 1000 simulation runs. The closer $\gamma_{\min}$ is to 1, the tighter is the reachable set.

Table 3 shows the results for all investigated systems ordered by dimension and analyzed by both approaches using adaptively tuned algorithm parameters. Due to space constraints, we cannot go into details. Therefore, we want to discuss general tendencies and explain unexpected results.

By construction, the linearization approach is limited to systems with only mild nonlinearties. Hence, it failed to produce tight results for the van der Pol oscillator and the Roessler attractor as indicated by the small values for $\gamma_{\min}$. The insufficiency of the linearization approach in the case of the van der Pol oscillator has already been discussed in [33, Sec. 4]. Comparing the tightness across the two approaches, either of the two measures $l_{\max}$ and $\gamma_{\min}$ reveals that the polynomialization approach generally yields tighter enclosures. However, the tightness of the reachable sets computed with the linearization approach is probably already satisfactory in cases where $\gamma_{\min} > 0.7$. Concerning the scalability of the tuning methods, the tightness of the resulting reachable sets shows by high values for $\gamma_{\min}$ and the similar computation times compared to those of lower-dimensional systems.

On average, the ratio between the largest and smallest time step is about 1-2 orders of magnitude. By exploiting this potential, we drastically reduce the number of steps in the analysis and increase the tightness of the resulting reachable sets. This is especially valuable for the polynomialization approach where the reduction is a lot more over-approximative due to using non-convex sets.

With respect to the set representation, we see that the zonotope order ρ is smaller using the linearization approach as Theorem 3.2 exploits the potential of reducing many generators. The polynomialization approach uses higher polynomial zonotope orders since the generators cannot be substantially reduced without inducing large over-approximations. It should also be noted that the values given for $\rho_{\max}$ represent the highest orders attained during the analysis which may only last for a few steps as discussed earlier and shown in Fig. 6c. This also explains the fairly small computation times for the corresponding systems.

Table 3: Evaluation of nonlinear benchmark systems: n: system dimension, t_K: time horizon, $\mathcal{X}^0$: initial set, $[\Delta t_{\mathbf{min}}, \Delta t_{\mathbf{max}}]$: range of time step sizes, $\rho_{\mathbf{max}}$: max. zonotope order, $l_{\mathbf{max}}$ and $\gamma_{\mathbf{min}}$: measurements by (33) and (34).

Benchmark	n	t_K	$\mathcal{X}^0$	**Linearization Approach**					**Polynomialization Approach**				
				Time	$[\Delta t_{\text{min}}, \Delta t_{\text{max}}]$	ρ_{max}	l_{max}	γ_{min}	Time	$[\Delta t_{\text{min}}, \Delta t_{\text{max}}]$	ρ_{max}	l_{max}	γ_{min}
Jet Engine [10, (19)]	2	8	$[0.90, 1.10]^n$	1.4s	$[0.007, 0.124]$	16	0.062	0.5094	6.5s	$[0.003, 0.062]$	25.5	0.0441	0.7125
van der Pol [6, Sec. VII]	2	6.74	$[1.30, 1.50]$ $[2.35, 2.45]$	3.7s	$[0.004, 0.034]$	16	1.87	0.1915	14.1s	$[0.002, 0.017]$	60	0.6070	0.5705
Brusselator [14, Ex. 3.4.1]	2	5	$[0.90, 1.00]$ $[0.00, 0.10]$	1.3s	$[0.019, 0.056]$	43	0.082	0.7367	4.8s	$[0.005, 0.029]$	121.5	0.066	0.9343
Roessler [47, (2)]	3	6	$[-0.20, 0.20]$ $[-8.60, -8.20]$ $[-0.20, 0.20]$	1.6s	$[0.006, 0.058]$	10.33	4.28	0.1745	6.7s	$[0.0007, 0.0469]$	20.33	2.65	0.5591
Lorenz [41, (25-27)]	3	2	$[14.90, 15.10]$ $[14.90, 15.10]$ $[34.90, 35.10]$	2.1s	$[0.0004, 0.004]$	9	0.275	0.8095	5.9s	$[0.0005, 0.0079]$	36.67	0.243	0.9279
Spring-Pendulum [14, Ex. 3.3.12]	4	1	$[1.1, 1.3]$ $[0.4, 0.6]$ $[0.0, 0.1]$ $[0.0, 0.1]$	2.3s	$[0.006, 0.024]$	12.5	0.518	0.6543	5.9s	$[0.003, 0.012]$	25.75	0.432	0.7759
Lotka-Volterra [49, (1)]	5	5	$[0.90, 1.00]^n$	0.6s	$[0.010, 0.117]$	10.4	0.082	0.8809	2.5s	$[0.005, 0.097]$	106.6	0.074	0.9756
Biological Model [32]	7	2	$[0.99, 1.01]^n$	1.9s	$[0.004, 0.019]$	45.43	0.117	0.7099	7.7s	$[0.002, 0.011]$	182.29	0.096	0.9240
Genetic Model [50, (1)]	9	0.1	see [17, Sec. V.]	0.5s	$[0.0007, 0.0024]$	5.78	5.55	0.7996	1.5s	$[0.0005, 0.0014]$	28	5.32	0.9302

4.4 Discussion

The presented methods for adaptive parameter tuning allow us to fully automatically obtain reachable sets without tuning any algorithm parameters. The computation of the reachable sets is executed in a single run. Thus, the computation times in Table 2 and Table 3 are truly the time required to analyze the system as opposed to manual tuning typically requiring many runs.

For systems with only mild nonlinearities, the linearization approach quickly produces good results while for severe nonlinearities, the reduction of the set representation within the polynomialization approach is the limiting factor. Our adaptive parameter tuning would greatly benefit from improvements in order reduction techniques, which can simply replace exisiting ones due to the modularity of our framework. The optimization function used to determine a near-optimal time step size balances the two main causes for unsatisfactory results by minimizing their joint influence. Thus, in case the reachability analysis fails to produce usable results, we can estimate that the cause for this is not the tuning of the parameters, but rather the insufficiency of either the reachability algorithm itself, or the handling of the set representation size.

5 CONCLUSION

In this paper, we presented the first adaptive tuning algorithm for all algorithm parameters of state-space abstracted reachability analysis of nonlinear systems. The modular construction treats each parameter separately and therefore maximizes the transparency and robustness of the adaptation as well as enables the applicability to other similar reachability algorithms or set representations. The numerical examples show the fast and reliable computation of tight reachable sets without the need to set any of the internally required algorithm parameters. This greatly facilitates the usage of reachability analysis for practitioners.

ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial supports from the research training group CONVEY funded by the German Research Foundation under grant GRK 2428 and the project justIT-SELF funded by the European Research Council (ERC) under grant agreement No 817629.

REFERENCES

[1] G. Alefeld and G. Mayer. 2000. Interval Analysis: Theory and Applications. *Computational and Applied Mathematics* 121 (2000), 421–464.

[2] M. Althoff. 2010. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars.* Dissertation. Technische Universität München.

[3] M. Althoff. 2013. Reachability Analysis of Nonlinear Systems using Conservative Polynomialization and Non-Convex Sets. In *Proc. of the 16th International Conference on Hybrid Systems: Computation and Control.* ACM, 173–182.

[4] M. Althoff. 2015. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems.* 120–151.

[5] M. Althoff, C. Le Guernic, and B. H. Krogh. 2011. Reachable Set Computation for Uncertain Time-Varying Linear Systems. In *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control.* ACM, 93–102.

[6] M. Althoff, O. Stursberg, and M. Buss. 2008. Reachability Analysis of Nonlinear Systems with Uncertain Parameters using Conservative Linearization. In *Proc. of the 47th IEEE Conference on Decision and Control.* 4042–4048.

[7] E. Asarin, T. Dang, and A. Girard. 2003. Reachability analysis of nonlinear systems using conservative approximation. In *Hybrid Systems: Computation and Control, 6th International Workshop.* Springer, 20–35.

[8] E. Asarin and et al. 2007. Hybridization Methods for the Analysis of Nonlinear Systems. *Acta Informatica* 43 (2007), 451–476.

[9] U. M. Ascher and et al. 1994. *Numerical solution of boundary value problems for ordinary differential equations.* SIAM.

[10] E. Aylward, P. Parrilo, and J-J. Slotine. 2008. Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming. *Automatica* 44, 8 (2008), 2163–2170.

[11] L. Benvenuti and et al. 2014. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *International Journal of Robust and Nonlinear Control* 24 (2014), 699–724.

[12] M. Berz and G. Hoffstätter. 1998. Computation and Application of Taylor Polynomials with Interval Remainder Bounds. *Reliable Computing* 4 (1998), 83–97.

[13] S. Bogomolov and et al. 2019. JuliaReach: a toolbox for set-based reachability. In *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control.* ACM, 39–44.

[14] X. Chen. 2015. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models.* Dissertation. RWTH Aachen University.

[15] X. Chen and et al. 2012. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. In *Proc. of the 33rd Real-Time Systems Symposium.* IEEE.

[16] X. Chen and et al. 2013. Flow*: An Analyzer for Non-Linear Hybrid Systems. In *Proc. of Computer-Aided Verification (LNCS 8044).* Springer, 258–263.

[17] X. Chen and S. Sankaranarayanan. 2016. Decomposed reachability analysis for nonlinear systems. In *Proc. of the 37th Real-Time Systems Symposium.* IEEE, 13–24.

[18] X. Chen, S. Sankaranarayanan, and E. Ábrahám. 2014. Under-approximate flowpipes for non-linear continuous systems. In *Formal Methods in Computer-Aided Design (FMCAD).* IEEE, 59–66.

[19] A. Chutinan and B. H. Krogh. 2003. Computational Techniques for Hybrid System Verification. *IEEE Trans. Automat. Control* 48, 1 (2003), 64–75.

[20] T. Dang and et al. 2010. Accurate Hybridization of Nonlinear Systems. In *Proc. of the 13th International Conference on Hybrid Systems: Computation and Control.* ACM, 11–19.

[21] T. Dang, C. Le Guernic, and O. Maler. 2009. Computing reachable states for non-linear biological models. In *International Conference on Computational Methods in Systems Biology.* Springer, 126–141.

[22] J. Alexandre dit Sandretto and A. Chapoutot. 2016. DynBEX: a Differential Constraint Library for Studying Dynamical Systems. *hal.archives-ouvertes.fr: hal-01297273.*

[23] P.S. Duggirala and et al. 2015. C2E2: A verification tool for stateflow models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems.* Springer, 68–82.

[24] P.S. Duggirala and M. Viswanathan. 2016. Parsimonious, Simulation Based Verification of Linear Systems. In *Proc. of the 28th International Conference on Computer Aided Verification.* Springer, 477–494.

[25] G. Frehse and et al. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Proc. of the 23rd International Conference on Computer Aided Verification (LNCS 6806).* Springer, 379–395.

[26] L. Geretti and et al. 2020. ARCH-COMP20 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics. In *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems.* 49–75.

[27] A. Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control, 8th International Workshop (LNCS 3414).* Springer, 291–305.

[28] Z. Han and B.H. Krogh. 2006. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *Proc. of the American Control Conference.* IEEE, 1505–1510.

[29] J. Hoefkens and et al. 2001. *Scientific Computing, Validated Numerics, Interval Methods.* Springer, Chapter Verified High-Order Integration of DAEs and Higher-Order ODEs, 281–292.

[30] F. Immler. 2015. Tool Presentation: Isabelle/HOL for Reachability Analysis of Continuous Systems. In *Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems.* 180–187.

[31] M. Kerbl. 1991. Stepsize strategies for inclusion algorithms for ODE's. *Computer Arithmetic, Scientific Computation, and Mathematical Modelling, IMACS Annals on Computing and Appl. Math* 12 (1991), 437–452.

[32] E. Klipp and et al. 2005. *Systems biology in practice: concepts, implementation and application.* John Wiley & Sons.

[33] N. Kochdumper and M. Althoff. 2020. Sparse Polynomial Zonotopes: A Novel Set Representation for Reachability Analysis. *IEEE Early Access, Transactions of Automatic Control* (2020).

[34] H. Kong and et al. 2017. Safety verification of nonlinear hybrid systems based on invariant clusters. In *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control.* ACM, 163–172.

[35] S. Kopecz and A. Meister. 2018. On order conditions for modified Patankar–Runge–Kutta schemes. *Applied Numerical Mathematics* 123 (2018), 159–179.

[36] G. Lafferriere and et al. 2001. Symbolic Reachability Computation for Families of Linear Vector Fields. *Symbolic Computation* 32 (2001), 231–253.

[37] L. Lapidus and J. H. Seinfeld. 1971. *Numerical solution of ordinary differential equations.* Academic press.

[38] M. Laub and W. Loomis. 1998. A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium. *Molecular biology of the cell* 9, 12 (1998), 3521–3532.

[39] D. Li, S. Bak, and S. Bogomolov. 2020. Reachability Analysis of Nonlinear Systems Using Hybridization and Dynamics Scaling. In *International Conference on Formal Modeling and Analysis of Timed Systems.* Springer, 265–282.

[40] J. Liu and et al. 2011. Computing semi-algebraic invariants for polynomial dynamical systems. In *Proc. of the 9th International Conference on Embedded Software.* ACM, 97–106.

[41] E. Lorenz. 1963. Deterministic nonperiodic flow. *Journal of the atmospheric sciences* 20, 2 (1963), 130–141.

[42] K. Makino and M. Berz. 2009. Rigorous Integration of Flows and ODEs using Taylor Models. In *Proc. of Symbolic-Numeric Computation.* 79–84.

[43] N. Matringe and et al. 2010. Generating invariants for non-linear hybrid systems by linear algebraic methods. In *International Static Analysis Symposium.* Springer, 373–389.

[44] I.M. Mitchell and et al. 2005. A Time-Dependent Hamilton–Jacobi Formulation of Reachable Sets for Continuous Dynamic Games. *IEEE Trans. Automat. Control* 50 (2005), 947–957.

[45] N.S. Nedialkov. 2000. *Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation.* Dissertation. University of Toronto.

[46] P. Prabhakar and M. Viswanathan. 2011. A Dynamic Algorithm for Approximate Flow Computations. In *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control.* ACM, 133–142.

[47] O. Rössler. 1976. An equation for continuous chaos. *Physics Letters A* 57, 5 (1976), 397–398.

[48] W. Rufeger and E. Adams. 1993. A step size control for Lohner's enclosure algorithm for ordinary differential equations with initial conditions. In *Mathematics in Science and Engineering.* Vol. 189. Elsevier, 283–299.

[49] J. Vano and et al. 2006. Chaos in low-dimensional Lotka–Volterra models of competition. *Nonlinearity* 19, 10 (2006), 2391.

[50] J.M.G. Vilar and et al. 2002. Mechanisms of noise-resistance in genetic oscillators. *Proc. of the National Academy of Sciences* 99, 9 (2002), 5988–5992.

[51] M. Wetzlinger, N. Kochdumper, and M. Althoff. 2020. Adaptive Parameter Tuning for Reachability Analysis of Linear Systems. In *Proc. of the 59th Conference on Decision and Control.* IEEE, 5145–5152.

A.6 Adaptive Reachability Algorithms for Nonlinear Systems Using Abstraction Error Analysis

Summary An essential step toward the automated verification of the task in Problem 1 for nonlinear systems is the analysis of the approximation error. This information is critical for refining the tightness of the computed reachable sets and, thus, for successful verification.

In this work, we thoroughly analyze the influence of the time step size on the local abstraction error for a reachability algorithm based on on-the-fly abstraction, as used in Appendix A.5. To this end, we introduce the *gain order*, which compares abstraction errors for different time step sizes. We show how to compute this gain order and that for most nonlinear systems, the gain in the abstraction error is linear in the limit for a time step size converging to zero. Consequently, the chosen reachability algorithm cannot avoid an explosion of the reachable set size for general nonlinear systems if the uncertainties in the model parameters and the time horizon are large enough.

Apart from the main contribution, we propose several bounds on the Hausdorff distance between a zonotope and its order-reduced counterpart, contributing to the automated tuning of the zonotope order. Furthermore, we extend our automated reachability algorithm from Appendix A.5 to other system classes. The numerical evaluation indicates the broad applicability of the proposed adaptive tuning methods as tight results can also be obtained for nonlinear discrete-time systems and nonlinear differential-algebraic systems.

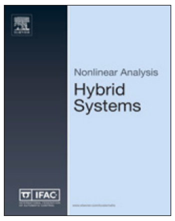
Author contributions M.W. laid out how to investigate the abstraction error, implemented the extended code for other system classes, conducted the numerical evaluation, and wrote most of the manuscript. A.K. investigated the dependency of the set of abstraction errors on the time step size and its relation to the global and local abstraction errors. A.LeP. conducted investigations relevant to the zonotope order reduction. M.A. provided substantial improvements for presenting the main contribution among general feedback for improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

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



Adaptive reachability algorithms for nonlinear systems using abstraction error analysis

Mark Wetzlinger [*], Adrian Kulmburg, Alexis Le Penven, Matthias Althoff

Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching b. München, Germany

A R T I C L E   I N F O

Article history:
Received 6 October 2021
Received in revised form 17 May 2022
Accepted 22 July 2022
Available online xxxx

Keywords:
Convergence order
Gain order
Parameter tuning
Zonotope order reduction
Hausdorff distance
reachability analysis
Nonlinear systems
Differential-algebraic equations

A B S T R A C T

In many reachability algorithms for nonlinear ordinary differential equations (ODEs), the tightness of the computed reachable sets mainly depends on abstraction errors and the choice of the set representation. One has to mitigate the resulting wrapping effects by suitable tuning of internally-used algorithm parameters since there exists no wrapping-free algorithm to date. In this work, we investigate the fundamentals governing the abstraction error in reachability algorithms – which we also refer to as set-based solvers – and its dependence on the time step size, leading to the introduction of a *gain order*. This order is related to measures for local and global abstraction errors and thus relates the well-known concept of convergence order from classical ODE solvers to set-based solvers. Furthermore, the simplification of the set representation is tackled by limiting the Hausdorff distance between the original and reduced sets; we demonstrate this for zonotopes. Both these theoretical advancements are incorporated in a modular adaptive parameter tuning algorithm suited for multiple classes of nonlinear ODEs whose efficiency is demonstrated on a wide range of benchmarks.

1. Introduction

Reachability analysis is a formal verification method for mixed discrete/continuous systems under the influence of uncertainty in the initial states, inputs, and parameters, offering provable guarantees with respect to unsafe states. The computation of exact reachable sets is only possible for a limited number of system classes [1], thus most algorithms compute over-approximative reachable sets in order to retain formal correctness. Despite the design of algorithms for a multitude of system classes and the exploitation of block structures or subspace behavior—all of which enhance the applicability of reachability analysis—the performance of these algorithms still depends heavily on the correct tuning of algorithm parameters. Poor tuning may result in large over-approximations and can ultimately lead to spurious counterexamples, where a given safety property cannot be verified even though a tighter over-approximation would be able to do so. Therefore, the automated tuning of algorithm parameters constitutes a crucial next step in the advancement of reachability algorithms, enabling non-experts and practitioners to apply reachability analysis in their respective fields. In this article, we address this demand by proposing an automated parameter tuning approach for state-space-abstracted reachability analysis of nonlinear ordinary differential equations and nonlinear differential-algebraic equations. An integral part is the tuning of the time step size, for which we thoroughly investigate its effect on the error caused by abstracting the system dynamics within the reachability analysis—an error that is well-studied for classical ODE solvers but has not yet been examined in detail for reachability analysis.

[*] Corresponding author.
E-mail addresses: m.wetzlinger@tum.de (M. Wetzlinger), adrian.kulmburg@tum.de (A. Kulmburg), alexis.le-penven@polytechnique.edu (A. Le Penven), althoff@tum.de (M. Althoff).

https://doi.org/10.1016/j.nahs.2022.101252
1751-570X/© 2022 Elsevier Ltd. All rights reserved.

Related work. There exist several approaches to compute reachable sets of nonlinear systems: By definition, any invariant set containing the initial set is also a reachable set, thus approaches for invariant generation [2–4] can be used for reachability analysis although the result may be unnecessarily conservative. One can also obtain reachable sets by solving a reformulation of the original problem to a Hamilton–Jacobi equation [5,6], whose solution scales exponentially with the system dimension. Another option is to obtain reachable sets from validated simulations with the help of annotations, i.e., additional information about the analyzed system [7], which also scales exponentially as one has to cover the initial set by enough initial states. Current state-of-the-art approaches for reachability analysis either abstract the solution space or the state space: For solution space abstraction, the Picard iteration is lifted to set-based analysis by representing reachable sets using Taylor models [8–10]. In state space abstraction, the system dynamics are abstracted by a Taylor series and its corresponding Lagrange remainder to obtain a differential inclusion, ranging from hybridization approaches [11–14] to on-the-fly linearizations [15–17] or polynomial abstractions [18]. The algorithmic differences between this group and approaches based on solution space abstraction are extensive and impede a joint parameter tuning framework for both groups. Hence, we will restrict ourselves to approaches based on state space abstraction in this work. Nonetheless, some of the presented ideas may be beneficial in the pursuit of similar automated tuning methods for any of the abovementioned methods. Many of the presented approaches are implemented in specialized reachability tools, namely Ariadne [19], C2E2 [20], CORA [21], DynIBEX [22], Flow* [23], Isabelle/HOL [24], and JuliaReach [25].

Reachability algorithms require a suitable set representation balancing an accurate representation of the reachable sets with computational efficiency of the set operations. The main trade-off occurs between the closedness of operations and the growth of the set representation size: As an example, the set representation size of ellipsoids is fixed, but they are not closed under the Minkowski sum, whereas for zonotopes an exact result can be obtained at the cost of increasing the representation size. This creates the need for so-called order reduction methods [26, Sec. 3.4] [27, Sec. 3.2]; comparisons of these methods are presented in [28,29]. Essentially, two types of approaches exist: The majority of methods computes an over-approximation that is as tight as possible for a desired (lower) user-specified order. The alternative approach in [30, Thm. 3.2] reduces the order as much as possible for a given bound of the induced over-approximation.

Many reachability algorithms depend on algorithm parameters that can be tuned to achieve tighter approximations, at the cost of a longer runtime. Some algorithms return tighter results than others for the same runtime, which offers one way of comparison and allows for a categorization of classical ODE solvers using the concept of consistency order [31,32]. For set-based reachability analysis, an estimation for the accuracy and convergence of certain algorithms has been performed in [33].

An open research question is how to automatically tune algorithm parameters for reachability analysis. This problem has been extensively studied for classical ODE solvers, which led to a wide variety of different methods and thorough investigations of stability and convergence [34]. Great effort has also been devoted to the automated tuning of the time step size [35,36], resulting in powerful solvers, which are ubiquitously applied in research and industry alike. As these solvers only compute approximations to the exact solution, a next step was to enclose a single trajectory, for which guaranteed integration methods provide several automated time step size control strategies [37,38].

In reachability analysis, automated techniques are scarce due to the presence of uncertainty in the initial state, input, and model parameters. This severely complicates matters as the tuning of algorithm parameters does not only comprise the time step size, but also effects related to a number of other algorithm parameters, such as the set representation as well as any emerging interdependency. For linear systems, there are approaches approximating the actual flow within a user-defined error bound [39], or adapting the time step size in each step in order to satisfy a linearly increasing, user-defined error bound [40]. More comprehensive approaches [41,42] adapt all algorithm parameters using over-approximation measures related to the Hausdorff distance to enable users to tune the desired accuracy. For nonlinear systems, adaptive methods have been explored in [43], where the time step size is tuned within a user-defined range. Another method [44] proposes iterative recomputations of the reachable set from scratch, while refining the parameter values in discrete steps in between runs. The work in [30] presents the first fully automatic reachability algorithm for nonlinear systems, which not only optimizes the time step size, but also other algorithm parameters such as the representation size.

Contributions. Our work is based on the adaptive parameter tuning approach in [30], which is significantly enhanced in the following three ways:

1. We extend the zonotope order reduction method introduced in [30] by two additional bounds for the Hausdorff distance between a zonotope and its box over-approximation.
2. While the analysis of the abstraction error in [30] was restricted to parameter tuning, we now dive a lot deeper into this topic: In order to lift for the first time convergence orders of classical solvers to reachability algorithms, we rigorously introduce a novel concept called *gain order*, which offers a similar yet more accurate description of the influence of the time step size on local and global abstraction errors.
3. Our tuning methods are no longer restricted to only nonlinear continuous-time systems as in [30], but can now also be applied to systems with algebraic constraints, parametric uncertainties, and in discrete time.

This article is structured as follows: A summary of reachability analysis for multiple nonlinear system classes is presented in Section 2. In Section 3, several bounds on the Hausdorff distance between a zonotope and its reduced counterpart are proven. Next in Section 4, we thoroughly analyze the abstraction error of the reachability algorithm leading to the introduction of the *gain order*, which translates the concept of convergence order known from numerical ODE theory to set-based solvers. Based on these theoretic novelties, the tuning modules that constitute our adaptive parameter tuning approach are described in Section 5. Finally, the evaluation of numerical examples in Section 6 demonstrates the practical usability of our tuning methods for a variety of nonlinear system classes, followed by concluding remarks in Section 7.

2. Preliminaries

In this section, we recall reachability analysis for nonlinear systems based on abstractions in the state space. This outline is particularly important for the investigation of the abstraction error in Section 4 as well as for the adaptive parameter tuning in Section 5.

2.1. Notation

We denote vectors by lower-case letters and matrices by upper-case letters. An all-zero vector or matrix of proper dimension is represented by $\mathbf{0}$. For a vector $v \in \mathbb{R}^n$, $|v| \in \mathbb{R}^n$ is the element-wise computed absolute value and v_i returns the ith entry of v. Analogously, m_{ij} refers to the entry in the ith row and jth column of a matrix $M \in \mathbb{R}^{n\times p}$. We denote the concatenation of two matrices M_1 and M_2 by $[M_1\ M_2]$. The operation $\operatorname{diag}(v)$ returns a matrix with v on its diagonal and otherwise zeros. The identity matrix of proper dimension is denoted by I. Moreover, $\|M\|_\infty$ refers to the matrix norm of M induced by the infinity norm. Calligraphic upper-case letters represent sets: We write $\mathcal{B} = [a,b] \subset \mathbb{R}^n$, where $\forall i \in \{1,\dots,n\} : a_i \leq b_i$, to denote an n-dimensional axis-aligned box. Its diameter is defined by $d\big(\mathcal{B}\big) := b - a \in \mathbb{R}^n$ and the absolute value by $\texttt{abs}(\mathcal{B}) := [-c, c] \subset \mathbb{R}^n$, where $c = \max\{|a|,|b|\}$ is evaluated element-wise [45, eq. (10)]. Furthermore, we abbreviate the Cartesian product of identical lower and upper limits for n consecutive dimensions by $[a,b]^n$. We use upper-case boldface letters to represent interval matrices $\mathbf{I} = [P, Q] \in \mathbb{R}^{m\times n}$, where $\forall i \in \{1,\dots,m\}, \forall j \in \{1,\dots,n\} : p_{ij} \leq q_{ij}$. For operations on sets, we use $\oplus$ for the Minkowski sum, $\ominus$ for the Minkowski difference with $\mathcal{S}_1 \ominus \mathcal{S}_2 := \{s | s + \mathcal{S}_2 \subseteq \mathcal{S}_1\}$, and introduce the operator $\boxplus$ representing either the Minkowski sum or the exact addition as defined in [46, Prop. 10]; additionally, we define the operators $\texttt{center}(\mathcal{S})$, $\texttt{box}(\mathcal{S})$, and $\texttt{vol}(\mathcal{S})$, which respectively return the geometric center, the tightest axis-aligned box over-approximation, and the volume of a set $\mathcal{S} \subset \mathbb{R}^n$. The radius of a set is defined as $\texttt{rad}(\mathcal{S}) := 0.5\,\big\|d\big(\texttt{box}(\mathcal{S})\big)\big\|_2$. Additionally, $\mathcal{S}_i = e_i^\top \mathcal{S}$, with e_i being the ith basis vector, denotes the projection of $\mathcal{S}$ onto the ith axis. We also write $\texttt{conv}(\mathcal{S}_1, \mathcal{S}_2)$ for the convex hull of two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$. The floor operator $\lfloor k \rfloor$ returns the next smaller integer number k, the sign function is denoted by $\operatorname{sgn}(\cdot)$, and the Frobenius norm by $\|\cdot\|_F$. The set $\mathbb{N}_0$ denotes the natural numbers including 0.

2.2. Reachability analysis of nonlinear systems using abstractions in the state space

The presented techniques for automated parameter tuning are applied to several classes of nonlinear systems, the most general of which are semi-explicit index-1 differential–algebraic (DA) equations [47], which can be formulated as

$$\begin{aligned}\dot{x}(t) &= f(x(t), y(t), u(t)) \\ 0 &= g(x(t), y(t), u(t))\ ,\end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a sufficiently smooth nonlinear function, $x(t) \in \mathbb{R}^n$ is the state vector, $y(t) \in \mathbb{R}^{n_a}$ is the vector of algebraic variables, and $u(t) \in \mathbb{R}^m$ is the input vector. Omitting the algebraic equation and algebraic variables yields a standard nonlinear ordinary differential equation. Note that these also encompass parametric systems: Each constant parameter can be defined as an additional state with the dynamics $\dot{x}_i(t) = 0$ and each time-varying parameter as an additional uncertain input. Moreover, we will consider discrete-time systems $x_{k+1} = f(x_k, u_k)$.

Let us introduce $\xi_{\mathrm{ex}}(t; x(0), y(0), u(\cdot))$ as the solution of (1) at time t for the initial values $x(0)$ and $y(0)$. Then, the exact reachable set $\mathcal{R}_{\mathrm{ex}}\big([0, t_{\mathrm{end}}]\big)$ of (1) over the time horizon $t \in [0, t_{\mathrm{end}}]$ is given by

$$\mathcal{R}_{\mathrm{ex}}\big([0,t_{\mathrm{end}}]\big) = \Big\{\xi_{\mathrm{ex}}(t; x(0), y(0), u(\cdot)) \,\Big|\, x(0) \in \mathcal{X}^0,\ y(0) \in \mathcal{Y}^0,\ t \in [0, t_{\mathrm{end}}], \forall \theta \in [0,t] : u(\theta) \in \mathcal{U}\Big\}\ , \tag{2}$$

with the initial sets $\mathcal{X}^0 \subset \mathbb{R}^n$, $\mathcal{Y}^0 \subset \mathbb{R}^{n_a}$ and the input set $\mathcal{U} \subset \mathbb{R}^m$. In this work we use abstractions in the state space, where both $f(\cdot)$ and $g(\cdot)$ are abstracted by a Taylor series of order κ at an expansion point z^* [18, eq. (2)] [47, eq. (8)], so that

$$\begin{aligned}\dot{x}_i(t) &\in \sum_{\nu=0}^{\kappa} \frac{\big((z(t)-z^*)^\top \nabla\big)^\nu f_i(\hat{z})}{\nu!}\bigg|_{\hat{z}=z^*} \oplus \mathcal{L}_i^{(x)}(t)\ , \\ 0 &\in \sum_{\nu=0}^{\kappa} \frac{\big((z(t)-z^*)^\top \nabla\big)^\nu g_i(\hat{z})}{\nu!}\bigg|_{\hat{z}=z^*} \oplus \mathcal{L}_i^{(y)}(t)\ ,\end{aligned} \tag{3}$$

using the extended vector $z = [x^\top y^\top u^\top]^\top \in \mathbb{R}^{n+n_a+m}$ and the Nabla operator $\nabla = \sum_{i=1}^{n+n_a+m} e_i \frac{\partial}{\partial z_i}$. The Lagrange remainders $\mathcal{L}_i^{(x)}$ and $\mathcal{L}_i^{(y)}$ are defined by [18, eq. (2)] [47, eq. (8)]

$$\begin{aligned}\mathcal{L}_i^{(x)}(t) &:= \left\{ \frac{\left((z(t)-z^*)^\top \nabla\right)^{\kappa+1} f_i(\hat{z})}{(\kappa+1)!} \middle| \hat{z} = z^* + \alpha(z(t)-z^*), \alpha \in [0,1] \right\}, \\ \mathcal{L}_i^{(y)}(t) &:= \left\{ \frac{\left((z(t)-z^*)^\top \nabla\right)^{\kappa+1} g_i(\hat{z})}{(\kappa+1)!} \middle| \hat{z} = z^* + \alpha(z(t)-z^*), \alpha \in [0,1] \right\},\end{aligned} \tag{4}$$

and evaluated using range-bounding techniques such as interval arithmetic [48].

The time horizon $[0, t_{\text{end}}]$ is divided into time intervals $\tau_k = [t_k, t_{k+1}]$ with the individual time step sizes $\Delta t_k = t_{k+1} - t_k > 0$ summing up to t_{end}. The reachable set for the entire time horizon is obtained by unifying the sequence of time-interval reachable sets $\mathcal{R}(\tau_k)$. For notational simplicity, we introduce an equivalent notation for the first terms in (3),

$$\begin{aligned} w_i^{(x)} &= f_i(z^*),\ C_{ij}^{(x)} = \left.\frac{\partial f_i(\hat{z})}{\partial \hat{z}_j}\right|_{\hat{z}=z^*},\ D_{ijk}^{(x)} = \left.\frac{\partial^2 f_i(\hat{z})}{\partial \hat{z}_j \partial \hat{z}_k}\right|_{\hat{z}=z^*}, \dots \\ w_i^{(y)} &= g_i(z^*),\ C_{ij}^{(y)} = \left.\frac{\partial g_i(\hat{z})}{\partial \hat{z}_j}\right|_{\hat{z}=z^*},\ D_{ijk}^{(y)} = \left.\frac{\partial^2 g_i(\hat{z})}{\partial \hat{z}_j \partial \hat{z}_k}\right|_{\hat{z}=z^*}, \dots . \end{aligned} \tag{5}$$

Alg. 1 summarizes the reachability algorithm for nonlinear DA systems featured in on-the-fly methods, such as the ones in [17,18], which extends to hybridization approaches [12,14,16] with minor adaptations. After its presentation, we will discuss the simplifications that can be made for the other system classes mentioned at the beginning of this overview.

Algorithm 1 Reachability analysis of continuous-time nonlinear systems using abstractions in the state space.

Input: nonlinear function $f(z)$, time horizon t_{end}, initial set $\mathcal{R}(t_0) = \mathcal{X}^0$, input set $\mathcal{U}$; *only DA*: algebraic equation $g(z)$, initial algebraic set $\mathcal{R}^y(t_0) = \mathcal{Y}^0$
Output: reachable set $\mathcal{R}([0, t_{\text{end}}])$

```
1:  $k \leftarrow 0, t_k \leftarrow 0$
2:  while $t_k < t_{\text{end}}$ do
3:      $z^*(t_k) \leftarrow \texttt{linPoint}(\mathcal{R}(t_k), f, \mathcal{R}^y(t_k), g)$
4:      $w^{(x)}, w^{(y)}, C^{(x)}, C^{(y)}, \dots \leftarrow \texttt{taylor}\big(f(z), z^*(t_k)\big)$
5:      $w, A, B \leftarrow \texttt{linSys}(w^{(x)}, w^{(y)}, C^{(x)}, C^{(y)})$
6:      $\mathcal{R}_{\text{lin}}(t_{k+1}), \mathcal{R}_{\text{lin}}(\tau_{k+1}) \leftarrow \texttt{linReach}(\mathcal{R}(t_k), w, A, B)$
7:      $\overline{\Psi} \leftarrow \mathbf{0}$
8:      do
9:          $\overline{\Psi} \leftarrow \texttt{enlarge}(\overline{\Psi})$
10:         $\Psi, \mathcal{R}^y(t_{k+1}) \leftarrow \texttt{abstrErr}(\mathcal{R}_{\text{lin}}(\tau_{k+1}), \overline{\Psi}, \kappa)$
11:     while $\Psi \not\subseteq \overline{\Psi}$
12:     $\mathcal{R}_{\text{abs}} \leftarrow \texttt{abstrSol}(\Psi)$
13:     $\mathcal{R}(t_{k+1}) \leftarrow \mathcal{R}_{\text{lin}}(t_{k+1}) \boxplus \mathcal{R}_{\text{abs}}$
14:     $\mathcal{R}(\tau_{k+1}) \leftarrow \mathcal{R}_{\text{lin}}(\tau_{k+1}) \boxplus \mathcal{R}_{\text{abs}}$
15:     $\mathcal{R}(t_{k+1}) \leftarrow \texttt{red}(\mathcal{R}(t_{k+1})), \mathcal{R}(\tau_{k+1}) \leftarrow \texttt{red}(\mathcal{R}(\tau_{k+1}))$
16:     $t_{k+1} \leftarrow t_k + \Delta t_k,\ k \leftarrow k + 1$
17: end while
18: $\mathcal{R}([0, t_{\text{end}}]) \leftarrow \bigcup_{j=0}^{k-1} \mathcal{R}(\tau_j)$
```

At the start of each step k, the operation `taylor` evaluates both Taylor series at the linearization point z^* (Line 4) returned by the operation `linPoint` (Line 3):

$$x^* = \texttt{center}(\mathcal{R}(t_k)) + \frac{1}{2}\Delta t_k f\big(\texttt{center}(\mathcal{R}(t_k))\big), \quad u^* = \texttt{center}(\mathcal{U}), \quad \text{and} \quad y^* \leftarrow 0 = g(x^*, y^*, u^*), \tag{6}$$

where $y^* \in \mathcal{R}^y(t_k)$ is obtained by solving the algebraic equation using a Newton–Raphson algorithm [47, Sec. IV-A]. Next, we abstract the nonlinear system by a differential inclusion

$$\dot{x}(t) \in \underbrace{Ax(t) + Bu(t) + w}_{f_{\text{lin}}(t)} \boxplus \Psi, \tag{7}$$

using the linearized vector field f_{lin} and an uncertainty set Ψ enclosing all higher-order terms including the Lagrange remainder. The constant offset w, the state matrix $A \in \mathbb{R}^{n\times n}$, and the input matrix $B \in \mathbb{R}^{n\times m}$ are returned by the operation `linSys` [47, Sec. IV]:

$$w = w^{(x)} - \tilde{Y}\tilde{Z}^{-1}w^{(y)},\ A = \tilde{A} - \tilde{Y}\tilde{Z}^{-1}\tilde{V},\ B = \tilde{B} - \tilde{Y}\tilde{Z}^{-1}\tilde{W},$$
$$\text{using } C^{(x)} = [\tilde{A}\ \tilde{Y}\ \tilde{B}],\ C^{(y)} = [\tilde{V}\ \tilde{Z}\ \tilde{W}],$$

where $\tilde{A} \in \mathbb{R}^{n\times n}$, $\tilde{Y} \in \mathbb{R}^{n\times n_a}$, $\tilde{B} \in \mathbb{R}^{n\times m}$, $\tilde{V} \in \mathbb{R}^{n_a\times n}$, $\tilde{Z} \in \mathbb{R}^{n_a\times n_a}$, $\tilde{W} \in \mathbb{R}^{n_a\times m}$. The reformulation in (7) allows us to apply the superposition principle for linear systems and separate the computation of the next reachable set $\mathcal{R}(t_{k+1})$ into two parts: First, the reachable set $\mathcal{R}_{\text{lin}}$ of the linearized dynamics (Lines 4–6); second, the set of abstraction errors $\mathcal{R}_{\text{abs}}$ based on the abstraction error Ψ [47, eq. (15)] (Lines 7–12). Let us briefly highlight an important difference between two groups of algorithms: Linearization algorithms [17] may also use polynomial abstractions ($\kappa \geq 2$) in (3), but the evaluation of all higher-order terms until the Lagrange remainder loses the state correlation with respect to the linear dynamics, which results in an essentially linear approximation of $f(x)$ despite the polynomial abstraction. In contrast, polynomialization algorithms [18] retain the state correlation in the evaluation of all time-constant, higher-order ($\kappa \geq 2$) terms in (3) allowing for a truly polynomial abstraction. For simplicity, we will restrict the remainder of this overview to linearization algorithms; the extension to polynomialization algorithms can be found in [18].

The operation `linReach` (Line 6) returns the reachable set $\mathcal{R}_{\text{lin}}$ using a reachability algorithm for the linearized dynamics $Ax(t)+Bu(t)+w$, e.g., [49, Sec. 3.2]. The computation of the abstraction error Ψ requires to resolve the mutual dependency between Ψ and $\mathcal{R}_{\text{lin}}$. To this end, we initially estimate Ψ to be $\overline{\Psi}$ (Line 7) and use the operation `enlarge` to enlarge this set by a constant factor greater than 1 (Line 9) until the containment $\Psi \subseteq \overline{\Psi}$ (Line 8) is ensured. Within this loop, the set Ψ is iteratively computed using the operation `abstrErr` (Line 10), which depends on the correlation of the state x with the abstraction error [17, Prop. 1] [18, Sec. 4.1]. The algebraic reachable set $\mathcal{R}^y(t_{k+1})$ is obtained as a byproduct [47, Prop. 2]. Next, the operation `abstrSol` (Line 12) computes the set of abstraction errors based on Ψ by [17, Sec. VI.]

$$\mathcal{R}_{\text{abs}} = \bigoplus_{p=0}^{\eta_{\text{abs}}} \frac{\Delta t^{p+1}}{(p+1)!} A^p \Psi \oplus \mathbf{E}(\Delta t, \eta_{\text{abs}})\, \Delta t\, \Psi, \tag{8}$$

with the remainder of the exponential matrix [50, Prop. 2]

$$\mathbf{E}(\Delta t, \eta) = [-E(\Delta t, \eta), E(\Delta t, \eta)], \quad E(\Delta t, \eta) = e^{|A|\Delta t} - \sum_{i=0}^{\eta} \frac{\left(|A|\Delta t\right)^i}{i!}, \tag{9}$$

where $\mathbf{E} = \mathcal{O}(\Delta t^{\eta+1})$ for $\Delta t \to 0$.

By exploiting the superposition principle in (7), the next reachable set $\mathcal{R}(t_{k+1})$ is obtained by the addition of $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ (Lines 13–14). For reasons of computational efficiency, we require to reduce the set representation size using the operation $\mathtt{red}(\cdot)$ at the end of each step (Line 15). Then, we obtain the next time-point solution as

$$\mathcal{R}(t_{k+1}) = \mathtt{red}(\underbrace{e^{A\Delta t_k}\mathcal{R}(t_k) \oplus \mathcal{P}(\tau_k)}_{=:\,\mathcal{R}_{\text{lin}}(t_{k+1})} \boxplus \mathcal{R}_{\text{abs}}). \tag{10}$$

For later use, we expand the set $\mathcal{R}_{\text{lin}}$ obtained from the operation `linReach` into its homogeneous and particular solutions, $e^{A\Delta t_k}\mathcal{R}(t_k)$ and $\mathcal{P}(\tau_k)$, respectively. For the time-interval solution, we compute the reachable set of the linearized dynamics $\mathcal{R}_{\text{lin}}(\tau_k)$ using the convex hull of sets at the beginning and end of the time interval and enlarge the result by an error set $\mathbf{F}_x\mathcal{R}(t_k)$ with [49, Prop. 3.1]

$$\mathbf{F}_x = \bigoplus_{p=2}^{\eta_{\text{lin}}} \underbrace{\left[(p^{\frac{-p}{p-1}} - p^{\frac{-1}{p-1}})\Delta t^p, 0\right]\frac{A^p}{p!}}_{:=\mathbf{T}^{(p)}} \oplus \mathbf{E}(\Delta t, \eta_{\text{lin}}), \tag{11}$$

which ultimately yields [17, Sec. IV.-A]

$$\mathcal{R}(\tau_{k+1}) = \underbrace{\mathtt{conv}(\mathcal{R}(t_k), e^{A\Delta t_k}\mathcal{R}(t_k) \oplus \mathcal{P}(\tau_k)) \oplus \mathbf{F}_x\mathcal{R}(t_k)}_{=:\,\mathcal{R}_{\text{lin}}(\tau_k)} \boxplus \mathcal{R}_{\text{abs}}(\tau_k), \tag{12}$$

assuming $0 \in \mathcal{U}$, with an extension to arbitrary inputs in [49, Sec. 3.2.2].

Finally, let us briefly highlight the algorithmic simplifications for the other aforementioned system classes: The changes required to accommodate standard ordinary differential equations follow directly by omitting each occurrence of the algebraic equation. For discrete-time systems, we replace (7) by the following difference inclusion

$$x_{k+1} \in \underbrace{Ax_k + Bu_k + w}_{f_{\text{lin}}(x_k,u_k)} \boxplus \Psi. \tag{13}$$

The operation linReach (Line 6) therefore becomes the straightforward evaluation of $f_{\text{lin}}(x_k, u_k)$. Next, the process of estimation and containment check (Lines 7–11) is shortened to a single evaluation of the operation abstrErr on the start set $\mathcal{R}(t_k)$. The next time-point solution (Line 13) is computed by the addition of $\mathcal{R}_{\text{lin}}$ and Ψ following directly from (13). Naturally, there is neither a computation of $\mathcal{R}_{\text{abs}}$ (Line 12) nor of a time-interval solution (Line 14).

3. Hausdorff reduction

Reachable sets are often represented by zonotopes because they are closed under linear maps and Minkowski sums, and these operations can be computed efficiently. Over subsequent steps of Alg. 1, the representation size of zonotopes grows, necessitating *order reduction*. In this section, we will provide several bounds so that the representation size of a zonotope can be reduced while satisfying any desired over-approximation error. In comparison with our previous work [30], we derive two more error bounds and also a novel generator selection criterion for the order reduction. Additionally, we provide insights into the performance of each bound for different classes of zonotopes.

Let us first define zonotopes and the order reduction operation:

Definition 1 (*Zonotopes [26] Def. 1*)**.** Given a center $c \in \mathbb{R}^n$ and $\gamma \in \mathbb{N}$ generator vectors $G = [g^{(1)} \dots g^{(\gamma)}]$, a zonotope is defined as

$$\mathcal{Z} := \left\{ x \in \mathbb{R}^n \;\middle|\; x = c + \sum_{i=1}^{\gamma} \alpha_i g^{(i)}, -1 \leq \alpha_i \leq 1 \right\}.$$

We also define its order by $\rho := \frac{\gamma}{n}$ and the shorthand $\langle c, G \rangle_Z$. □

For later use, let us also introduce the operations $\texttt{center}(\mathcal{Z}) = c$ and $\texttt{box}(\mathcal{Z}) = \langle c, \text{diag}(\sum_{i=1}^{\gamma} |g^{(i)}|)\rangle_Z$ returning a box over-approximation of $\mathcal{Z}$. For order reduction, we select the method introduced in [26, Sec. 3.4], which is comprised of the following steps:

1. Split the given zonotope $\mathcal{Z}_{\text{full}}$ into two parts $\mathcal{Z}_{\text{full}} = \mathcal{Z}' \oplus \mathcal{Z}$.
2. Enclose $\mathcal{Z}$ by a tight box $\mathcal{Z}_{\text{box}} = \texttt{box}(\mathcal{Z}) \supseteq \mathcal{Z}$.
3. Compose the reduced zonotope as $\mathcal{Z}_{\text{red}} = \mathcal{Z}' \oplus \mathcal{Z}_{\text{box}} \supseteq \mathcal{Z}_{\text{full}}$.

To quantitatively measure the error induced by the order reduction, we use the Hausdorff distance:

Definition 2 (*Hausdorff Distance*)**.** For two compact sets $\mathcal{V}, \mathcal{W} \subset \mathbb{R}^n$, the Hausdorff distance is defined as

$$d_H(\mathcal{V}, \mathcal{W}) := \max\left\{ d_H^{(1)}(\mathcal{V}, \mathcal{W}), d_H^{(2)}(\mathcal{V}, \mathcal{W}) \right\},$$

where

$$d_H^{(1)}(\mathcal{V}, \mathcal{W}) := \max_{v \in \mathcal{V}} \min_{w \in \mathcal{W}} \|v - w\|_2 \qquad \text{and} \qquad d_H^{(2)}(\mathcal{V}, \mathcal{W}) := \max_{w \in \mathcal{W}} \min_{v \in \mathcal{V}} \|w - v\|_2. \quad \square$$

For subsequent derivations, let us introduce a short lemma:

Lemma 1. *For the compact sets $\mathcal{S}, \mathcal{V}, \mathcal{W} \subset \mathbb{R}^n$, the following inequality holds:*

$$d_H(\mathcal{S} \oplus \mathcal{V}, \mathcal{S} \oplus \mathcal{W}) \leq d_H(\mathcal{V}, \mathcal{W}).$$

Proof. We only prove the inequality for $d_H^{(1)}$:

$$\begin{aligned} d_H^{(1)}(\mathcal{S} \oplus \mathcal{V}, \mathcal{S} \oplus \mathcal{W}) &= \max_{\substack{v \in \mathcal{V} \\ s^{(1)} \in \mathcal{S}}} \min_{\substack{w \in \mathcal{W} \\ s^{(2)} \in \mathcal{S}}} \|v + s^{(1)} - w - s^{(2)}\|_2 \\ &\leq \max_{\substack{v \in \mathcal{V} \\ s^{(1)} \in \mathcal{S}}} \min_{w \in \mathcal{W}} \|v + s^{(1)} - w - s^{(1)}\|_2 \\ &= \max_{v \in \mathcal{V}} \min_{w \in \mathcal{W}} \|v - w\|_2 \\ &= d_H(\mathcal{V}, \mathcal{W}). \end{aligned}$$

The reasoning is analogous for $d_H^{(2)}$. □

In the context of zonotope order reduction, Lemma 1 implies

$$d_H(\mathcal{Z}_{\text{full}}, \mathcal{Z}_{\text{red}}) = d_H(\mathcal{Z}' \oplus \mathcal{Z}, \mathcal{Z}' \oplus \mathcal{Z}_{\text{box}}) \leq d_H(\mathcal{Z}, \mathcal{Z}_{\text{box}}),$$

which lets us restrict our attention to the computation of the Hausdorff distance between the partial zonotope $\mathcal{Z}$ and its box over-approximation $\mathcal{Z}_{\text{box}}$ without loss of generality. Computing the exact Hausdorff distance between two zonotopes is NP-hard in general, since the Hausdorff distance between a point (represented by a zonotope without generators) and an arbitrary zonotope can be reformulated as a *longest vector sum* problem [51] (this equivalence is easier to see using the arguments from [52, p. 268]); in fact, the work in [51] even shows that this problem is APX-hard. Hence, in practice, we are limited to finding bounds on the Hausdorff distance:

Theorem 1. *Let $\mathcal{Z} = \langle c, G\rangle_Z \subset \mathbb{R}^n$ be a zonotope and $\mathcal{Z}_{box} := \mathtt{box}(\mathcal{Z}) = \langle c, G_{box}\rangle_Z \supseteq \mathcal{Z}$ its box over-approximation. Due to the containment $\mathcal{Z} \subseteq \mathcal{Z}_{box}$, the Hausdorff distance d_H is given by*

$$d_H(\mathcal{Z}, \mathcal{Z}_{box}) = \max_{x_{box}\in\mathcal{Z}_{box}} \min_{x\in\mathcal{Z}} \|x_{box} - x\|_2 \ . \tag{14}$$

This distance is over-approximated by the following three bounds:

$$\omega_{\max}(\mathcal{Z}) := 2 \left\| \sum_{p=1}^{\gamma} \widehat{g}^{(p)} \right\|_2 , \tag{15}$$

$$\omega_{rad}(\mathcal{Z}) := \sum_{p=1}^{\gamma} \left\| g^{(p)} \right\|_2 , \tag{16}$$

$$\omega_{cut}(\mathcal{Z}) := \sqrt{2} \sum_{p=1}^{\gamma} \left\| g^{(p)} \right\|_2 \sqrt{1 - \frac{\sum_{i=1}^{n} \left(g_i^{(p)}\right)^4}{\left\| g^{(p)} \right\|_2^4}}, \tag{17}$$

with

$$\widehat{g}_i^{(p)} = \begin{cases} \left|g_i^{(p)}\right|, & \text{if } i \neq i^*, \\ 0, & \text{otherwise,} \end{cases} \tag{18}$$

where i^* is the (first) index for which $g_{i^*}^{(p)} = \left\| g^{(p)} \right\|_\infty$.

Proof. The proof can be found in Appendix A. □

The accuracy of the bounds ω_{max}, ω_{rad}, and ω_{cut} depends on the reduced zonotopes: While ω_{rad} performs better on average for random zonotopes, in practice ω_{max} performs better for reachability analysis. This originates from a bias of zonotopes in the reachability algorithm towards axis-aligned generators $g^{(p)}$ resulting in orthogonal $\widehat{g}^{(p)}$. This bias can be explained by the fact that on several occasions in the reachability algorithm, interval boxes are added to the current solution. Reducing the order of zonotopes by using box over-approximations further adds to this bias. The bound ω_{cut} performs slightly worse than ω_{max} in those biased cases. This can be seen by analyzing the effect of adding another generator g^* to the zonotope. Two effects can influence the performance of either bound: On the one hand, if g^* is diagonal (i.e., if the components of g^* have the same length, as opposed to g^* having only one non-zero component), ω_{max} will grow larger than ω_{cut} and thus perform worse. On the other hand, if $\widehat{g}^*$ is orthogonal to the other vectors $\widehat{g}^{(p)}$ then ω_{max} performs better than ω_{cut}.

In other words, ω_{max} performs better if generators are added that are axis-aligned and orthogonal up to one component, which is the case for the generators of an interval box. Nevertheless, the overall performance of ω_{cut} is similar to ω_{max} in low dimensions and significantly better in higher dimensions. Since the computation (17) of ω_{cut} contains the formula (16) for ω_{rad}, we combine both bounds to

$$\omega_{\text{rad,cut}}(\mathcal{Z}) := \sum_{p=1}^{\gamma} \left\| g^{(p)} \right\|_2 \min \left\{ 1, \sqrt{2} \sqrt{1 - \frac{\sum_{i=1}^{n} \left(g_i^{(p)}\right)^4}{\left\| g^{(p)} \right\|_2^4}} \right\} . \tag{19}$$

In Fig. 1, we see that this new bound generally yields better results than ω_{max} as it combines the advantages of ω_{cut} and ω_{rad}. Note that the joint bound $\omega_{\text{rad,cut}}$ performs better than either of the bounds ω_{rad} and ω_{cut} as the minimum in (19) is taken generator-wise. A potential combination of all three bounds is dismissed due to the resulting computational overhead.

Let us now provide a heuristic for generator selection, where we aim to reduce as many generators as possible while respecting a given threshold for the Hausdorff distance between the original and reduced set: Given a zonotope $\mathcal{Z} = \langle c, G\rangle_Z$, we sort the generators in $G \in \mathbb{R}^{n\times\gamma}$ using a cost function $\varrho(g)$. For ω_{max}, this cost function is

$$\varrho_{\max}(g) := \|g\|_1 - \|g\|_\infty , \tag{20}$$

originally proposed in [26]. For $\omega_{\text{rad,cut}}$, we exploit that the contribution of each generator to $\omega_{\text{rad,cut}}$ can be separated, yielding the cost function

$$\varrho_{\text{rad,cut}}(g) := \|g\|_2 \min \left\{ 1, \sqrt{2} \sqrt{1 - \frac{\sum_{i=1}^{n} (g_i)^4}{\|g\|_2^4}} \right\} . \tag{21}$$

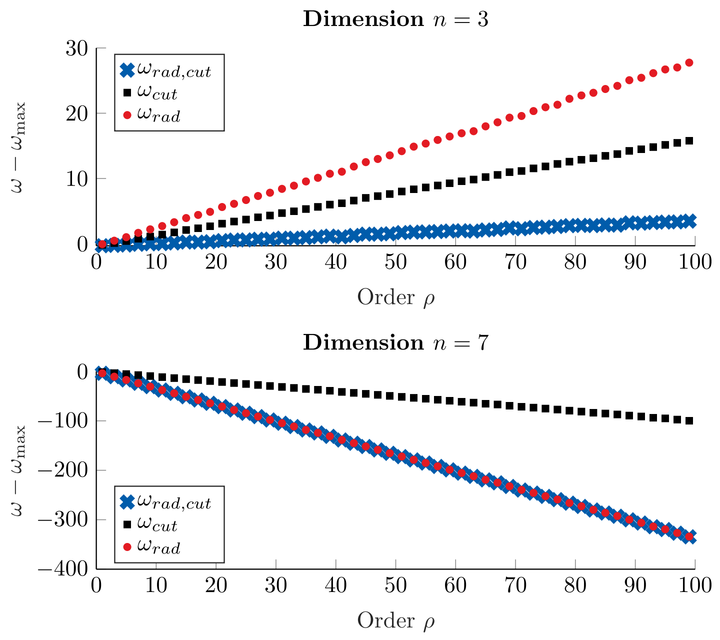


Fig. 1. Comparison of ω_{rad}, ω_{cut}, and $\omega_{\text{rad,cut}}$ to $\omega_{\max}$ (i.e., $\omega_{\max}$ coincides with the x-axis), by computing each bound for 1000 zonotopes centered at the origin with generator matrices that have random entries between -1 and $+1$, and taking the mean. Negative values of $\omega - \omega_{\max}$ mean that the bound performs better than $\omega_{\max}$. One can see that for low dimensions, the combination of ω_{rad} and ω_{cut} can lead to better results, which are comparable to those of $\omega_{\max}$. For high dimensions, ω_{rad} is typically much more precise than ω_{cut} or $\omega_{\max}$.

Note that the cost function $\varrho_{\text{rad}} = \|\cdot\|_2$ is exactly the cost function described in [27,53] to sort generators. However, since this is already computed when evaluating (21), the analysis of the computational complexity and the accuracy of ϱ_{rad} is similar to that of $\varrho_{\text{rad,cut}}$. We will utilize the presented novel bounds for zonotope order reduction in our adaptive parameter tuning approach in Section 5.3.

4. Gain order of set-based ODE solvers

We first recall some basics about the numerical approximation of ODE solutions using classical solvers and discuss why an immediate transferability of these concepts to reachability algorithms is inadequate. Afterwards, we will outline our solution, which will be presented in the remainder of this section.

In classical ODE theory [31,34], the quality of a numerical approximation of an ODE

$$\dot{x} = f(x, t) \tag{22}$$

is typically measured by the concept of *convergence order*:

Definition 3 (*Convergence Order of Classical ODE Solvers*). Let $\xi_{\text{ex}}(\Delta t; t_0, x_0)$ be the exact solution of (22) at time $t_0 + \Delta t$, with initial condition $x(t_0) = x_0$, and let $\widehat{\xi}(\Delta t; t_0, x_0)$ be an approximation of the exact solution at time $t_0 + \Delta t$. The *convergence order* (in short: *order*) of the solver is the number $q \in \mathbb{N}_0$, for which the following inequality holds:

$$\varepsilon_{\Delta t,\text{loc}} := |\xi_{\text{ex}}(\Delta t; t_0, x_0) - \widehat{\xi}(\Delta t; t_0, x_0)| \leq c\Delta t^{q+1}, \quad \forall t_0, x_0, \tag{23}$$

where c is a constant that neither depends on t_0, x_0, nor Δt, and $\varepsilon_{\Delta t,\text{loc}}$ is called the *local truncation error*. □

The error (23) is a *local* measure as it only measures the error committed in a single time step. To estimate the accumulated error over the entire time interval $[0, t_{\text{end}}]$, let us denote by $\widehat{\xi}_N$ the approximation after N steps using the time step size $\Delta t = t_{\text{end}}/N$. Then, the *global* error ε_N between the exact point $\xi_{\text{ex}}(t_{\text{end}}; t_0, x_0)$ and the approximation $\widehat{\xi}_N$ is bounded by [32, p. 318, Theorem 12.2]

$$\varepsilon_N := |\xi_{\text{ex}}(t_{\text{end}}; t_0, x_0) - \widehat{\xi}_N| \leq \widehat{c}\Delta t^q, \tag{24}$$

where $\widehat{c}$ is a constant that depends on t_{end}, but not on Δt. Thus, the order q provides an estimate on how small the time step size has to be in order to achieve good approximations—low-order methods, such as the explicit Euler method [31, Chapter 2] with order $q = 1$, will typically need much smaller time step sizes than high-order methods like the third-order variant of Heun's method [31, Chapter 9.5] with order $q = 3$.

For set-based methods used in reachability analysis, this error estimation cannot be generalized directly for several reasons:

- **Difference between sets:** Expressions such as $|\xi_{\text{ex}}(\Delta t; t_0, x_0) - \widehat{\xi}(\Delta t; t_0, x_0)|$ in (23) are difficult to evaluate if $\xi_{\text{ex}}(\Delta t; t_0, x_0)$ and $\widehat{\xi}(\Delta t; t_0, x_0)$ are replaced by their set-based analogues, i.e., the reachable set and some approximation that outputs a set.
- **Wrapping effects:** The magnitude of some approximation errors, primarily due to wrapping effects, such as the iterative order reduction operations discussed in Section 3, is independent of the time step size and thus cannot be reduced by choosing a smaller time step size. Even worse, reducing the time step size leads to a higher number of iterations and might result in an overall larger error, which does not happen for classical ODEs.

To address these problems, we propose a novel concept for the convergence order of set-based solvers. Instead of focusing on a local truncation error $\varepsilon_{\Delta t,\text{loc}}$, we will define a *gain function* $\varphi(t;\delta)$ that measures the overall error for varying time step sizes, and for which holds that

$$\varepsilon_N \le c\,\Delta t^{-1}\varphi(t_{\text{end}}; \frac{1}{N}), \tag{25}$$

where ε_N is an appropriate measure for the global error of a set-based solver and c is a constant that neither depends on Δt nor N.

For conciseness, our subsequent analysis will only be applied to systems of ordinary differential equations (22), where we will assume a smooth right-hand side. Under certain simple assumptions, we will show that $\varphi(t_{\text{end}}; \frac{1}{N}) \le 1/N^{q+1}$ for some $q \in \mathbb{N}_0$ that may depend on the right hand side f of (22), leading to the result

$$\varepsilon_N \le \widehat{c}\,\Delta t^{q}, \tag{26}$$

where $\widehat{c}$ is a constant that depends on t_{end}, but not on Δt nor N. As a consequence, the gain function $\varphi(t;\delta)$ allows us to mimic the result from (24) that one can get via classical ODE theory, without having to analyze the precise behavior of the local truncation error for each point in the initial set.

The next subsection Section 4.1 defines local and global abstraction errors of set-based solvers, followed by an investigation of the local error over varying time step sizes Δt using a formal definition of the aforementioned gain function in Section 4.2. Finally, we extend the obtained results from local to global errors in Section 4.3. Most of the proofs are to be found in Appendix B to enhance the fluidity of our presentation.

4.1. Errors of set-based ODE solvers

In order to define the notion of order for a given set-based solver, we need an estimation of the local error that such a solver produces (similarly to Definition 3). We focus our attention on the set of abstraction errors $\mathcal{R}_{\text{abs}}$ (8), which by definition collects the approximation errors induced by $\mathcal{R}_{\text{lin}}$ in (10). This is comparable to the work in [33]. More generally, we propose the following measure for the local and global error:

Definition 4 (*Errors of Set-based Solvers*)**.** Let $\mathcal{R}(\Delta t; t_0, \mathcal{X}^0)$ be a set-based approximation for (22), with initial value $x(t_0) \in \mathcal{X}^0$ and time step size Δt. Using the abstraction error as defined in (8), we have that

$$\mathcal{R}(\Delta t; t_0, \mathcal{X}^0) = \widehat{\mathcal{R}}(\Delta t; t_0, \mathcal{X}^0) \oplus \mathcal{R}_{\text{abs}}(\Delta t; t_0, \mathcal{X}^0), \tag{27}$$

where $\widehat{\mathcal{R}}(\Delta t; t_0, \mathcal{X}^0)$ is the approximation of the solution to the ODE (22). Let $\mathcal{R}(t_k)$ be the output after $k \in \{0, 1, \ldots, N\}$ iterations of $\mathcal{R}$ and let $\widehat{\mathcal{R}}(t_k)$ be the output after iteratively using $\widehat{\mathcal{R}}$ instead of $\mathcal{R}$. Then, the *local abstraction error* at time t_k is defined as

$$\varepsilon_{k,\text{loc}} := \left\| d\Big(\text{box}(\mathcal{R}(\Delta t; t_k, \mathcal{R}(t_k)) \ominus \widehat{\mathcal{R}}(\Delta t; t_k, \mathcal{R}(t_k)))\Big) \right\|_\infty = \left\| d\Big(\text{box}(\mathcal{R}_{\text{abs}}(\Delta t; t_k, \mathcal{R}(t_k)))\Big) \right\|_\infty, \tag{28}$$

whereas

$$\varepsilon_N := \left\| d\Big(\text{box}(\mathcal{R}(t_{\text{end}}) \ominus \widehat{\mathcal{R}}(t_{\text{end}}))\Big) \right\|_\infty. \tag{29}$$

is the *global accumulated abstraction error* over N steps. □

Similarly to classical ODE theory, the global error can be linked to the local error in the following manner:

Lemma 2 (*Local and Global Abstraction Error*)**.** *For solvers of the form described in (10), the global abstraction error after N steps can be bounded by the maximal local abstraction error as*

$$\varepsilon_N \le \frac{c}{\Delta t} \max_{1 \le \ell \le N} \varepsilon_{\ell,loc} \tag{30}$$

for some constant c that may depend on t_{end}, but not on Δt nor N.

Proof. The proof can be found in Appendix B. □

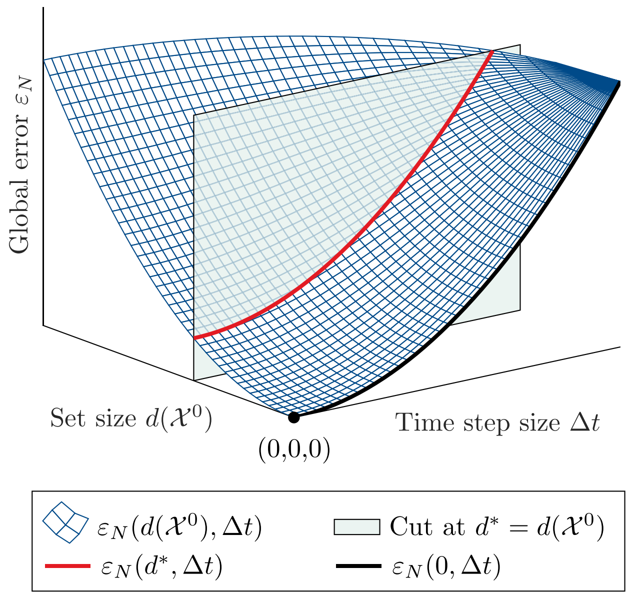


Fig. 2. Schematic evaluation of the global abstraction error ε_N as a function of space (set size $d(\mathcal{X}^0)$) and time (time step size Δt); two projections for $d(\mathcal{X}^0) = 0$ (classical solvers) and $d(\mathcal{X}^0) = d^* > 0$ (set-based solvers), resulting in the black and red curve, respectively, show a different limit value in the limit $\Delta t \to 0$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. The gain function

Now that we know how to quantify the error of a set-based solver, we investigate the behavior of the error for varying time step sizes Δt. For simplicity, we will drop the arguments k, t_k, t_0, and $\mathcal{X}^0$ for the rest of this section to focus on the effect of Δt on the local abstraction error. Moreover, for the sake of simplicity, instead of working with the local abstraction error as defined in (28), we will use the more accurate formulation

$$\varepsilon_{\mathrm{loc}} = \left\| d\big(\mathtt{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(\Delta t))\big) \right\|_{\infty}, \tag{31}$$

where $\mathcal{R}^{\infty}_{\mathrm{abs}}(\Delta t)$ is the *non-truncated* set of abstraction errors replacing (8), i.e.,

$$\mathcal{R}^{\infty}_{\mathrm{abs}}(\Delta t) = \bigoplus_{p=0}^{\infty} \frac{\Delta t^{p+1}}{(p+1)!} A^p(\Delta t) \Psi(\Delta t). \tag{32}$$

While the local error (31) explicitly depends on the time step size Δt, Ψ also depends on the initial set $\mathcal{X}^0$. The Taylor expansion of (31) entails that there exist some non-negative integers q_s and q_t such that

$$\varepsilon_{\mathrm{loc}} = \Delta t \left[\mathcal{O}(\Delta t^{q_t}) + \mathcal{O}(d(\mathcal{X}^0)^{q_s}) \right] \tag{33}$$

for small enough Δt and $d(\mathcal{X}^0)$, where $d(\mathcal{X}^0)$ is the diameter of $\mathcal{X}^0$. This is similar to classical ODE theory [31, Chapter 9]. The error $\varepsilon_{\mathrm{loc}}$ can thus be decreased either by reducing the time step size, or by splitting the initial set as in [33]. As Lemma 2 shows, the global error ε_N is bounded by a term proportional to $\varepsilon_{\mathrm{loc}}$. Its dependency on the set size and the time step size is illustrated in Fig. 2: For classical solvers, we have $d(\mathcal{X}^0) = 0$ yielding the black curve which converges to $\varepsilon_N = 0$ for $\Delta t \to 0$. In contrast, for set-based solvers we obtain a behavior like the red curve as we evaluate $\varepsilon_N(d(\mathcal{X}^0), \Delta t)$ on a projection (indicated by the gray plane) at $d(\mathcal{X}^0) = d^* > 0$. Crucially, this results in a value $\varepsilon_N > 0$ for $\Delta t \to 0$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The estimate (33) can be made independent of the right hand side f of the ODE, up to a multiplicative constant depending on the smoothness of f (see for example [31, Chapter 9.4]). Thus, it would allow one to separately define a *time* order q_t and a *space* order q_s. However, these concepts of order are difficult to measure locally as one would need to measure the combined error as a function of several variables. In contrast, classical ODE theory only considers the local error with respect to time.

Therefore, a different approach is more enticing: Instead of analyzing the precise behavior of $\varepsilon_{\mathrm{loc}}$ as a function of time and space (as in [33]), we consider its overall expansion over time by defining the following *gain function*:

Definition 5 (*Gain Function*). The *gain* $\varphi(h;\delta)$ over the time span h with relative increments $\delta \in [0,1]$ is the function

$$\varphi(h;\delta) = \max_{i\in\{1,\dots,n\}} \varphi_i(h;\delta) \quad \text{with} \quad \varphi_i(h;\delta) = \frac{d_i\big(\mathtt{box}(\mathcal{R}^{\infty}_{\text{abs}}(\delta h))\big)}{d_i\big(\mathtt{box}(\mathcal{R}^{\infty}_{\text{abs}}(h))\big)}\,, \tag{34}$$

where $\mathcal{R}^{\infty}_{\text{abs}}(h)$ is computed by (32). □

If $\Psi(h)$ in (32) is represented by a zonotope, the gain φ can be simplified using the following Proposition:

Proposition 1 (*Diameter of Set of Abstraction Errors*). *Suppose $\Psi(h)$ is given as the zonotope $\langle c(h), G(h)\rangle_Z$. Then we obtain*

$$d_i\big(\mathtt{box}(\mathcal{R}^{\infty}_{abs}(h))\big) = 2\sum_{p=0}^{\infty} \frac{h^{p+1}}{(p+1)!} \left\| A^p(h)G(h)\right\|_{1,i}\,, \tag{35}$$

where for a matrix M, $\|M\|_{1,i}$ denotes the 1-norm of the ith row of M.

Proof. The proof can be found in Appendix B. □

Following the result of Proposition 1, we can analyze the behavior of φ by examining the behavior of the coefficients $\|A^p(h)G(h)\|_{1,i}$, for which we now present a more concrete characterization:

Lemma 3 (*Expansion of Coefficients*). *Assume that the right-hand side f of (22) is smooth. Then, $A(h)$ and $G(h)$ are smooth, and for all $p \in \mathbb{N}_0$ and $i = \{1,\dots,n\}$, the coefficient $\|A^p(h)G(h)\|_{1,i}$ may either be written as*

$$\|A^p(h)G(h)\|_{1,i} = h^{q_i^{(p)}}\|a_i^{(p)} + b_i^{(p)}(h)\|_1, \tag{36}$$

for some $q_i^{(p)} \in \mathbb{N}_0$, $a_i^{(p)} \in \mathbb{R}\setminus\{0\}$, $b_i^{(p)}(h) = \mathcal{O}(h)$, or as

$$\|A^p(h)G(h)\|_{1,i} \equiv 0, \tag{37}$$

in which case we use the convention that $q_i^{(p)} = \infty$, as well as $a_i^{(p)} = 0$ and $b_i^{(p)}(h) = 0$. Furthermore, for all $p \in \mathbb{N}_0$ and $i = \{1,\dots,n\}$, the function

$$Q_i^{(p)}(h) = \|a_i^{(p)} + b_i^{(p)}(h)\|_1 \tag{38}$$

is non-negative and piecewise smooth.

Proof. The proof can be found in Appendix B. □

We can now use this knowledge about the numerator and denominator defining $\varphi(h;\delta)$ in order to investigate its behavior for $h \to 0$, through which a certain notion of order will arise naturally:

Theorem 2 (*Limit Gain*). *Suppose the right hand side f of (22) is smooth, and for each $p \in \mathbb{N}_0$ and $i = \{1,\dots,n\}$ let $a_i^{(p)}$ and $q_i^{(p)}$ be defined as in Lemma 3. Then for the gain in the ith dimension $\varphi_i(h;\delta)$ there holds*

$$\lim_{h\to 0} \varphi_i(h;\delta) = \delta^{q_i+1}, \tag{39}$$

where

$$q_i = \max\left\{ j \in \mathbb{N}_0 \;\middle|\; \sum_{p+q_i^{(p)}=j} \|a_i^{(p)}\|_1 \neq 0 \right\}. \tag{40}$$

Note that the condition on j in (40) is met for a given j if and only if either there does not exist a tuple $(p, q_i^{(p)})$ such that $p + q_i^{(p)} = j$, or if for any such tuple there holds $a_i^{(p)} = 0$, i.e., $[A^p(h)G(h)]_i \equiv 0$, according to the convention in Lemma 3.

Proof. Using (35), we deduce by (36) that

$$d_i\big(\mathtt{box}(\mathcal{R}^{\infty}_{\text{abs}}(h))\big) = 2\sum_{p=0}^{\infty}\frac{h^{p+1}}{(p+1)!}h^{q_i^{(p)}}\|a_i^{(p)} + b_i^{(p)}(h)\|_1 = 2\sum_{j=0}^{\infty} h^{j+1} \sum_{p+q_i^{(p)}=j} \frac{\|a_i^{(p)} + b_i^{(p)}(h)\|_1}{(p+1)!}.$$

Inserting this into (34) yields

$$\varphi_i(h;\delta) = \frac{\sum_{j=0}^{\infty} h^{j+1}\delta^{j+1}\sum_{p+q_i^{(p)}=j}\frac{\|a_i^{(p)}+b_i^{(p)}(h\delta)\|_1}{(p+1)!}}{\sum_{j=0}^{\infty} h^{j+1}\sum_{p+q_i^{(p)}=j}\frac{\|a_i^{(p)}+b_i^{(p)}(h)\|_1}{(p+1)!}} \xrightarrow[h\to 0]{} \delta^{q_i+1},$$

where while passing to the limit we used the fact that $b_i^{(p)} = \mathcal{O}(h)$ together with the assumption that

$$\sum_{p+q_i^{(p)}=q_i} \frac{\|a_i^{(p)}\|_1}{(p+1)!} \neq 0,$$

which is equivalent to (40). □

The quantities q_i have the unique property that they can describe the overall behavior of the error for different time step sizes. Consequently, they constitute the basis of our concept of order:

Definition 6 (*Gain Order of Set-based Solvers*)**.** Let $\mathcal{R}$ be an approximation of (22), and let q_i be defined as in (40). Then $q := \min_i q_i$ is called the *gain order* of the method, and q_i is called the gain order of the ith dimension. Note that $\mathcal{R}$ yields zero abstraction error if and only if $q = \infty$. This is because from (35) it follows that the abstraction error is zero if and only if $\|A^p(h)G(h)\|_{1,i} \equiv 0$ for all p and i, which is equivalent to $q = \infty$. □

We aim to show that, under certain simple assumptions, φ is monotonically decreasing in h. This will be crucial in practice because it shows that even if the global abstraction error does not decrease by $\mathcal{O}(\Delta t^p)$ for some $p \geq 1$ (where Δt is the time step size), it does indeed decrease notably for smaller time step sizes.

Proposition 2 (*Derivative of Gain*)**.** *Let $q_i^{(p)}$ and $Q_i^{(p)}$ be defined as in Lemma 3, and let q_i be defined as in (40). Assume that all $Q_i^{(p)}$ are differentiable at $t = 0$. Then, for $\delta \in [0, 1]$, there holds*

$$\lim_{h\to 0} \frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} \leq 0. \tag{41}$$

Furthermore, for fixed h, if $\forall i \in \{1, \dots, n\}$ and $p \in \mathbb{N}_0$ the value of $Q_i^{(p)}(t)$ is constant over $t \in [0, h]$, then

$$\varphi_i(t;\delta) \leq \delta^{q_i+1}, \quad \forall t \in [0, h],\ i \in \{1, \dots, n\}. \tag{42}$$

Proof. The proof can be found in Appendix B. □

As we have seen, the gain function provides an alternative way of estimating the dependency of the local abstraction error with respect to the time step size, and this estimation can describe non-integer orders by means of the function φ.

4.3. Global abstraction error via gain order

After estimating the local error using the gain function, we now want to extend these results to an estimation of the global abstraction error. To do so, we select h to be a fixed finite time horizon that we will use as a unit of measurement, and through which we will comparatively observe the effect of reducing the time step size $\Delta t < h$. For some fixed $\delta \in [0, 1]$, by (41) we can choose h to be small enough such that $\varphi(t;\delta)$ is decreasing on $t \in [0, 2h]$. By (39), the limit of $\varphi(t;\delta)$ for $t \to 0$ is δ^{q+1}, therefore on $[0, 2h]$ there must hold that $\varphi(t;\delta) \leq \delta^{q+1}$. For $t = h$, this yields the relation

$$\varphi(h;\delta) \leq \delta^{q+1}. \tag{43}$$

The latter inequality holds even for relatively large h in practice, as we shall discuss later on.

Since the gain φ provides an estimate for the variation of the local abstraction error, φ depends on the step k just like $\varepsilon_{k,\mathrm{loc}}$ (28) does. Let φ_k denote the gain function corresponding to step k. From (30) it follows that the global error ε_N after N steps can be estimated by

$$\begin{aligned}
\varepsilon_N &\overset{(30)}{\leq} c\,\Delta t^{-1} \max_{1\leq k\leq N} \max_i d_i\big(\mathrm{box}(\mathcal{R}^\infty_{\mathrm{abs}}(\Delta t; t_k, \mathcal{R}(t_k)))\big)\\
&\overset{\text{Definition 5}}{\leq} c\,\Delta t^{-1} \max_{1\leq k\leq N} \varphi_k(h; \tfrac{1}{N}) \max_i d_i\big(\mathrm{box}(\mathcal{R}^\infty_{\mathrm{abs}}(h; t_k, \mathcal{R}(t_k)))\big)\\
&\leq c\,\Delta t^{-1} \max_{1\leq k\leq N} \varphi_k(h; \tfrac{1}{N}) \max_{1\leq k\leq N} \max_i d_i\big(\mathrm{box}(\mathcal{R}^\infty_{\mathrm{abs}}(h; t_k, \mathcal{R}(t_k)))\big)\\
&\overset{(43)}{\leq} c\,\Delta t^{-1} \frac{1}{N^{q+1}} \max_{1\leq k\leq N} \max_i d_i\big(\mathrm{box}(\mathcal{R}^\infty_{\mathrm{abs}}(h; t_k, \mathcal{R}(t_k)))\big).
\end{aligned}$$

From [18, (11)] it obviously follows that $\mathcal{R}^\infty_{\mathrm{abs}}(h + t_k; 0, \mathcal{X}^0) \subseteq \mathcal{R}^\infty_{\mathrm{abs}}(2h; 0, \mathcal{X}^0)$, since $t_k \leq h$. Therefore, we can write

$$\varepsilon_N \leq c\,\Delta t^{-1} \frac{1}{N^{q+1}} \mathcal{R}^\infty_{\mathrm{abs}}(2h; 0, \mathcal{X}^0),$$

which eventually yields a bound

$$\varepsilon_N \leq c' \Delta t^{-1} \frac{1}{N^{q+1}}.$$

Since $N = h/\Delta t$, we conclude that

$$\varepsilon_N \leq \widehat{c}(h) \Delta t^q,$$

where $\widehat{c}(h)$ is a constant that depends on h, but not on Δt nor N. Consequently, our definition of the order of a solver yields similar results to classical ODE theory, except that $\varphi(h; \frac{1}{N})$ can give even more precise information, e.g., non-integer orders, about the local improvement one would get by decreasing the time step size.

In practice, we will primarily use $\varphi(h; \delta)$ to determine an approximation of $\varepsilon_{k,\text{loc}}$ for small variations of Δt. As we saw earlier, (41) implies that $\varphi(h; \delta) \leq \delta^{q+1}$ if h is small enough, a restriction that can be loosened for larger h using Proposition 2 under the assumption that $A^k(h)G(h)$ in Proposition 1 does not vary much with respect to h, as this would imply that the $Q_i^{(p)}$ are constant. This assumption holds as long as $\Psi(h)$ does not vary much, which is true in practice as long as the computation of $\Psi(h)$ converges.

In the next section, we will make use of the gain function φ (see Definition 5), its limit value (see Theorem 2), and its derivative (see Proposition 2 and the discussion above) in order to construct an optimization function for the crucial tuning of the time step size.

5. Adaptive parameter tuning methods

In this section, we propose individual tuning methods to adaptively tune each algorithm parameter used in Alg. 1. Fig. 3 provides an overview of the reachable set computation within one time step, where arrows indicate the effect of algorithm parameters on sets. All described tuning methods are modular, that is, the algorithm parameters are adapted independently of one another. The final composition of all tuning modules in an adaptive tuning algorithm yields a general framework for state-space-abstracted reachability algorithms such as Alg. 1. Each tuning module can simply be exchanged for a different one, e.g., if different reachable set computations or set representations are chosen.

We will generally omit the index k for the current step as all algorithm parameters are adapted in each step. The remainder of this section describes the individual tuning methods for the propagation parameters η (Section 5.1), the abstraction order κ (Section 5.2), the set representation size ρ (Section 5.3) – all of which are tuned by threshold conditions determining sufficient accuracy – and the time step size Δt (Section 5.4) obtained by solving an optimization problem that minimizes the unavoidable over-approximation over a finite time horizon. Finally, we introduce the automated tuning algorithm as an enhancement to Alg. 1 in Section 5.5.

5.1. Propagation parameters

As schematically indicated in Fig. 3, the computation of the sets $\mathcal{R}_{\text{abs}}$ in (8) and $\mathcal{R}_{\text{lin}}$ in (10) and (12) requires us to tune the order η of the finite Taylor series of the exponential matrix. The main idea is to exploit that the contribution of higher-order terms eventually vanishes. As a consequence, we truncate the respective Minkowski sums once the contribution of the additional term has become small enough to determine the orders η_{lin} and η_{abs}.

The over-approximation error in $\mathcal{R}_{\text{lin}}(\tau_k)$ is dominated by the error term $\mathbf{F}_x\mathcal{R}(t_k)$ [42, eq. (21)]. Thus, we tune η_{lin} using a fixed threshold $0 < \zeta_{T,\text{lin}} \ll 1$ related to the influence of additional terms $\mathbf{T}^{(p)}$ in (11):

$$\eta_{\text{lin}} = \min_{\substack{p\in\mathbb{N} \\ p\geq 2}} p \qquad \text{such that} \qquad 1 - \frac{\left\|\mathbf{T}^{(p-1)}\right\|_F}{\left\|\mathbf{T}^{(p)}\right\|_F} \leq \zeta_{T,\text{lin}}. \tag{44}$$

Using the same idea of comparing successive orders, we determine the order η_{abs} by truncating the sum in (8) according to the criterion

$$\eta_{\text{abs}} = \min_{p\in\mathbb{N}_0} p \qquad \text{such that} \qquad \max_{i\in\{1,\dots,n\}} \frac{d_i\left(\text{box}(\mathcal{R}_{\text{abs}}^{(p+1)})\right)}{d_i\left(\text{box}(\mathcal{R}_{\text{abs}}^{(p)})\right)} \leq \zeta_{T,\text{abs}}, \tag{45}$$

with $\mathcal{R}_{\text{abs}}^{(p)}$ denoting the sum in (8) truncated at order p and $0 < \zeta_{T,\text{abs}} \ll 1$.

As both criteria (44) and (45) can be evaluated during the iterative computation of the respective sets, the tuning itself yields negligible computational overhead. Note that there are no propagation parameters in the reachable set computation of discrete-time systems.

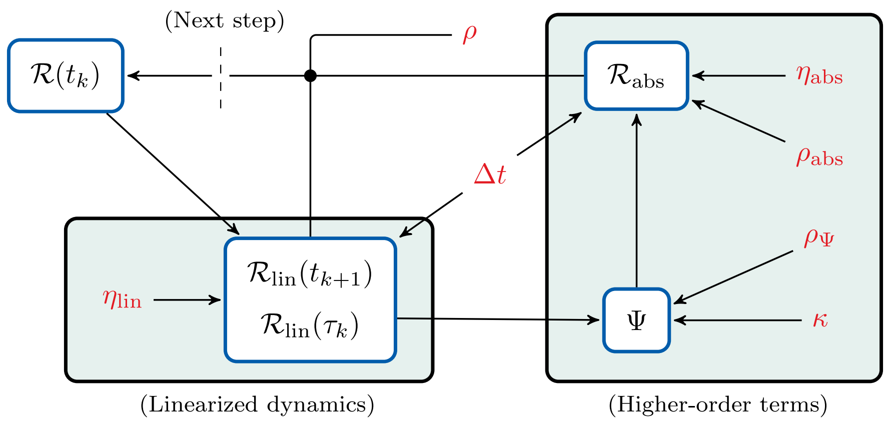


Fig. 3. Main workflow for one time step in Alg. 1 and the influence of algorithm parameters on different sets (an arrow $A \rightarrow B$ means that A is used to compute B): The propagation parameters η affect the precision of the exponential matrix and the set representation parameters ρ represent the reduction operation, which is applied to various sets within one step.

5.2. Abstraction order

According to Fig. 3, the abstraction order κ in (3) directly influences the abstraction error Ψ and subsequently the set of abstraction errors $\mathcal{R}_{\text{abs}}$. In general, larger values of κ are computationally more demanding due to the evaluation of higher-order maps, but also decrease the size of the set of abstraction errors $\mathcal{R}_{\text{abs}}$.

For a linearization approach, we restrict the admissible values of the abstraction order to $\kappa = \{1, 2\}$, as the evaluation of cubic or higher-order maps is highly over-approximative using zonotopes. In constrast, non-convex set representations are closed under higher-order maps but significantly increase the set representation size which cannot be handled by current reduction methods. Hence, the abstraction order for the polynomialization approach is fixed to $\kappa = 2$.

We propose a two-step selection criterion to limit the computational overhead:

1. The set of abstraction errors $\mathcal{R}_{\text{abs}}$ of the current step k is compared to the last comparison, denoted by k'. To establish a level playing field, we estimate the size of $\mathcal{R}_{\text{abs}}(\Delta t_k)$ for the time step size $\Delta t_{k'}$ using the gain φ^* based on (34). The condition
$$\left| \frac{\varphi^* \, \texttt{rad}(\mathcal{R}_{\text{abs}}(\Delta t_k))}{\texttt{rad}(\mathcal{R}_{\text{abs}}(\Delta t_{k'}))} \right| > 1 - \zeta_K, \quad \zeta_K \in (0, 1), \tag{46}$$
decides whether we progress to the second step below. Informally, we compare the sizes of the set of abstraction errors and only if the size difference is large enough, the value for κ is re-tuned.
2. If the condition (46) is fulfilled, we also compute $\mathcal{R}_{\text{abs}}(\Delta t_k)$ for the other value of κ and decide the value of κ for the next step according to the following condition:
$$\kappa \leftarrow \begin{cases} 1, & \text{if } \forall i \in \{1, \dots, n\} \text{ with } d_i(\mathcal{R}_{\text{abs}}) > 0 : \frac{d_i(\texttt{box}(\mathcal{R}_{\text{abs}}(\kappa=2)))}{d_i(\texttt{box}(\mathcal{R}_{\text{abs}}(\kappa=1)))} \geq \zeta_K \\ 2, & \text{otherwise.} \end{cases} \tag{47}$$

The closer ζ_K is to 1, the more conservative the selection becomes, i.e., the more often $\kappa = 2$ will be chosen resulting in both a tighter result as well as longer computation times. For the first step, we use the initial set $\mathcal{R}(t_0) = \mathcal{X}^0$ to compute Ψ and immediately evaluate (47) to compute the first abstraction order κ. For discrete-time systems, we exploit two properties to greatly simplify the computation: First, the abstraction error Ψ is used directly in the reachable set computation replacing $\mathcal{R}_{\text{abs}}$ as seen in (13). Second, we do not need to compensate for different time step sizes in subsequent steps so that we always have $\varphi^* = 1$ in (46).

5.3. Set representation

A reduction of the representation size can only avoid large over-approximations if the reduction error is restricted by an upper bound. We utilize the two bounds ω_{max} (15) and $\omega_{\text{rad,cut}}$ (19) for the Hausdorff distance between the original zonotope and its reduced counterpart from Section 3. For either bound, we sort the generators of $\mathcal{Z}$ in ascending order with respect to its respective cost function ϱ_{max} (20) or $\varrho_{\text{rad,cut}}$ (21). Then, we select the first $\gamma^* \leq \gamma$ generators, until we reach the upper bound

$$\sum_{p=1}^{\gamma^*} \varrho\big(g^{(p)}\big) \leq \zeta_Z \left\| d\big(\texttt{box}(\mathcal{Z})\big) \right\|_2, \tag{48}$$

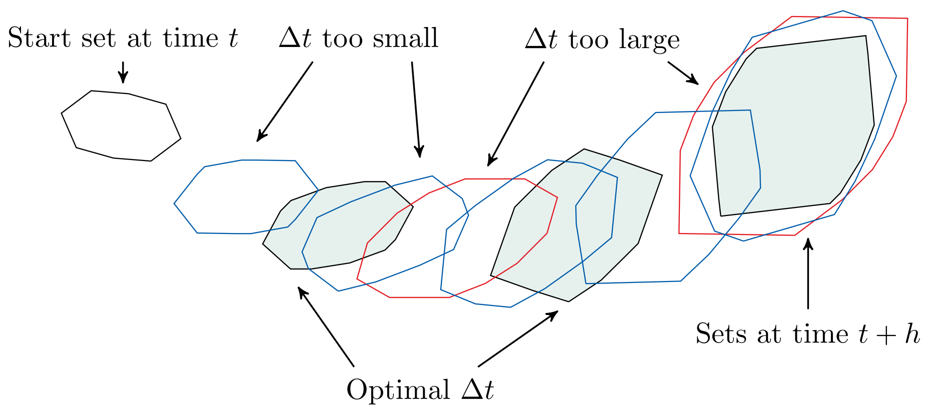


Fig. 4. The optimal value Δt_* (gray sets) for the set propagation over a time horizon $[t, t+h]$ is obtained by balancing the wrapping effects: If Δt is too large (red sets), the set of abstraction errors $\mathcal{R}_{\text{abs}}$ is too large; if Δt is too small (blue sets), the reduction operation excessively increases the set size. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where we use a fixed fraction $0 < \zeta_Z \ll 1$ of the diagonal of the box over-approximation of the original zonotope $\mathcal{Z}$. The exact Hausdorff distance between the original and the reduced set is smaller than the left-hand side in (48) by Theorem 1.

Our polynomialization approach is best used with a non-convex set representation, where we choose polynomial zonotopes allowing us to exploit their similarities to zonotopes which we extensively covered in Section 3. The reduction method for polynomial zonotopes described in [46, Prop. 10] is based on zonotope order reduction, so that we can reuse the bound (48) to decrease the representation size.

5.4. Time step size

Alg. 1 contains two main sources for over-approximation, both of which are related to the time step size Δt: First, we have the set of abstraction errors $\mathcal{R}_{\text{abs}}$ whose behavior over Δt has been thoroughly investigated in Section 4. By *decreasing* the time step size, we reduce the size of $\mathcal{R}_{\text{abs}}$ and thus alleviate the wrapping effect originating from the iterative addition of $\mathcal{R}_{\text{abs}}$. Second, the reduction operation induces another wrapping effect whose effect is diminished by *increasing* time step sizes. Section 5.3 describes the error induced by a single reduction operation, which we now have to consider over multiple steps.

In order to obtain a tight reachable set, we require to tune Δt so that the trade-off between both wrapping effects is optimized as shown in Fig. 4: The start set is propagated over a finite time horizon h using different candidate time step sizes $\Delta t^{(\lambda)} = \frac{h}{\lambda}$, $\lambda \geq 1$. The optimal time step size Δt_* (using 3 steps) balances both wrapping effects so that neither the set of abstraction errors $\mathcal{R}_{\text{abs}}$ is too large (using 2 steps) nor is the reduction operation applied too often (using 5 steps)—both of which yield a larger set at time $t+h$.

In order to efficiently solve the optimization problem in each step, we make some design choices without which the comparison of different time step sizes $\Delta t^{(\lambda)}$ would become infeasible in practice:

(a) We assume the system matrix A, i.e., the Jacobian matrix of $f(x)$, to be constant over $[t, t+h]$ and use $A = A(t)$.
(b) We will neglect the particular solution $\mathcal{P}(\tau_k)$ in the propagation formula for the reachable set (10).
(c) For each $\Delta t^{(\lambda)}$, we assume the set of abstraction errors $\mathcal{R}_{\text{abs}}(\Delta t^{(\lambda)})$ to be constant over $[t, t+h]$ and use $\mathcal{R}_{\text{abs}} = \mathcal{R}_{\text{abs}}(\Delta t^{(\lambda)})$ obtained at time t.
(d) We linearly interpolate the gain φ (34) between $\varphi(\Delta t = h) = \varphi^{(1)}$ and the limit gain computed in Theorem 2 $\lim_{\Delta t \to 0} \varphi(\Delta t) = \delta^{q+1}$ to obtain

$$\varphi(\Delta t) \approx \zeta_\delta + \frac{\varphi^{(1)} - \zeta_\delta}{h} \Delta t, \tag{49}$$

where we replace δ by the fixed value $\zeta_\delta \in (0, 1)$, potentially underestimating the actual gain in order to prevent the time step size from decreasing too much. We will use this interpolation for all $\Delta t^{(\lambda)}$.

Note that we will explicitly consider $\lambda \in \mathbb{R}$ to facilitate any candidate time step size $\Delta t^{(\lambda)} \in (0, h]$, which requires to take a last incomplete step of length $b\Delta t^{(\lambda)} = (\lambda - \lfloor \lambda \rfloor)\Delta t^{(\lambda)}$ into account in order to compare the resulting sets at the same point in time as shown in Fig. 4. In addition to the design choices (a)–(d), we simplify the set-based evaluation to scalar values in three ways:

1. The sizes of the start set $\mathcal{R}(t)$ and the set of abstraction errors $\mathcal{R}_{\text{abs}}(\Delta t^{(\lambda)})$ are approximated by their respective radii $r_0 = \texttt{rad}(\mathcal{R}(t))$ and $r_{\text{abs}}^{(\lambda)} = \texttt{rad}(\mathcal{R}_{\text{abs}}(\Delta t^{(\lambda)}))$.

2. The effect of the exponential matrix is captured by its determinant, which over the entire finite time horizon can be estimated by $\det(e^{Ah}) = e^{\mathrm{tr}(Ah)}$, leading to a scaling factor of
$$\zeta_A^{\frac{1}{\lambda}} = \left(e^{\mathrm{tr}(Ah)}\right)^{\frac{1}{\lambda}} \tag{50}$$
for each partial step of length $\Delta t^{(\lambda)}$. For the scaling of the last incomplete step, we have $\zeta_A^{\frac{b}{\lambda}}$.
3. The enlargement caused by the zonotope order reduction is measured by multiplying the radius with $(1 + 2\zeta_Z)$ following (48). The factor for the last incompete step is $(1 + 2\zeta_Z)^b$.

As only time-point solutions are reused in subsequent steps, we repeatedly apply (10) to compute the reachable set after time h, omitting the particular solution as stated in design choice (b):
$$\tilde{\mathcal{R}}(t+h) = \mathtt{red}(e^{Ab\Delta t^{(\lambda)}}\mathtt{red}(e^{A\Delta t^{(\lambda)}}...\mathtt{red}(e^{A\Delta t^{(\lambda)}}\mathcal{R}(t) \boxplus \mathcal{R}_{\mathrm{abs}})... \boxplus \mathcal{R}_{\mathrm{abs}}) \boxplus b\mathcal{R}_{\mathrm{abs}}), \tag{51}$$
where $e^{A\Delta t^{(\lambda)}}$ and $\mathcal{R}_{\mathrm{abs}}$ are scaled to $e^{Ab\Delta t^{(\lambda)}}$ and $b\mathcal{R}_{\mathrm{abs}}$ in the last incomplete step. Based on the aforementioned simplifications, we now rewrite the set-based formula (51) to a scalar estimate for the set size of the reachable set using the recursive formula
$$r_R(t + j\Delta t^{(\lambda)}) = (1 + 2\zeta_Z)\big(\zeta_A^{\frac{1}{\lambda}} r_R(t + (j-1)\Delta t^{(\lambda)}) + r_{\mathrm{abs}}^{(\lambda)}\big), \tag{52}$$
which starts with the set size estimate at time t given by $r_R(t) := r_0$. To obtain an estimate after time h, we apply the recursion $\lfloor\lambda\rfloor$ times and then include the last incomplete step:
$$r_R(t+h) = (1+2\zeta_Z)^b \left(\zeta_A^{\frac{b}{\lambda}} \underbrace{(1+2\zeta_Z)(\zeta_A^{\frac{1}{\lambda}}...(1+2\zeta_Z)(\zeta_A^{\frac{1}{\lambda}} r_0 + r_{\mathrm{abs}}^{(\lambda)})... + r_{\mathrm{abs}}^{(\lambda)})}_{\overset{(52)}{=} r_R(t+\lfloor\lambda\rfloor\Delta t^{(\lambda)})} + b\, r_{\mathrm{abs}}^{(\lambda)}\right).$$
Summarizing the first $\lfloor\lambda\rfloor$ steps yields
$$r_R(t+h) = (1+2\zeta_Z)^b \left(\zeta_A^{\frac{b}{\lambda}} \left(r_0(1+2\zeta_Z)^{\lfloor\lambda\rfloor}\zeta_A^{\frac{\lfloor\lambda\rfloor}{\lambda}} + r_{\mathrm{abs}}^{(\lambda)} \sum_{j=1}^{\lfloor\lambda\rfloor}(1+2\zeta_Z)^j \zeta_A^{\frac{j-1}{\lambda}}\right) + b\, r_{\mathrm{abs}}^{(\lambda)}\right),$$
after which we include the last incomplete step and rearrange to
$$r_R(t+h) = r_0(1+2\zeta_Z)^{\lambda}\zeta_A + r_{\mathrm{abs}}^{(\lambda)}\zeta_{A,Z}(\lambda), \quad \zeta_{A,Z}(\lambda) = \sum_{j=1}^{\lfloor\lambda\rfloor}(1+2\zeta_Z)^{b+j}\zeta_A^{\frac{b+j-1}{\lambda}} + b(1+2\zeta_Z)^b. \tag{53}$$

Since the evaluation of (53) would require us to compute $r_{\mathrm{abs}}^{(\lambda)}$ for each λ (which is obviously undesirable in practice due to the large computational overhead), we approximate $r_{\mathrm{abs}}^{(\lambda)}$ utilizing design choice (d). Let us first define $\lambda' \in \mathbb{N}$ as the number of times h has been scaled by a fixed ζ_δ. Hence, $\lambda = \zeta_\delta^{-\lambda'} \in \mathbb{R}$ is the number of times $\Delta t^{(\lambda)}$ divides into h and using
$$\varphi^{(j)} = \varphi(\zeta_\delta^{j-1}h) = \zeta_\delta + (\varphi^{(1)} - \zeta_\delta)\zeta_\delta^{j-1}, \tag{54}$$
we obtain an estimate for $r_{\mathrm{abs}}^{(\lambda)}$ based only on $r_{\mathrm{abs}}^{(1)}$ and $\varphi^{(1)}$:
$$\lambda\, r_{\mathrm{abs}}^{(\lambda)} = \varphi^{(1)} \cdot ... \cdot \varphi^{(\lambda')} r_{\mathrm{abs}}^{(1)} \quad \Rightarrow \quad r_{\mathrm{abs}}^{(\lambda)} = \frac{r_{\mathrm{abs}}^{(1)}}{\lambda}\prod_{j=1}^{\lambda'}\varphi^{(j)}. \tag{55}$$
One can also compute $\varphi^{(1)}$ given $r_{\mathrm{abs}}^{(1)}$ and $r_{\mathrm{abs}}^{(\lambda)}$ by solving the following implicit equation for $\varphi^{(1)}$ based on combining (54)–(55), which will be used later on in the tuning algorithm for Δt:
$$\varphi^{(1)} \cdot \left(\zeta_\delta + (\varphi^{(1)} - \zeta_\delta)\zeta_\delta\right) \cdot ... \cdot \left(\zeta_\delta + (\varphi^{(1)} - \zeta_\delta)\zeta_\delta^{\lambda'-1}\right) = \lambda \frac{r_{\mathrm{abs}}^{(\lambda)}}{r_{\mathrm{abs}}^{(1)}}. \tag{56}$$
Inserting (55) in (53) yields the cost function
$$r_R(t+h) = r_0(1+2\zeta_Z)^{\lambda}\zeta_A + \frac{r_{\mathrm{abs}}^{(1)}}{\lambda}\zeta_{A,Z}(\lambda)\prod_{j=1}^{\lambda'}\varphi^{(j)}, \tag{57}$$
which we minimize to obtain the optimal time step size
$$\Delta t_* = h\,\zeta_\delta^{\lambda'_*} \quad \text{where} \quad \lambda'_* = \underset{\lambda' \in \mathbb{N}}{\arg\min}\; r_R(t+h). \tag{58}$$

Algorithm 2 Adaptively-tuned reachable set computation for one step $k > 1$

Input: nonlinear function $f(z)$, algebraic equation $g(z)$, start set $\mathcal{R}(t_k)$, algebraic start set $\mathcal{R}^y(t_k)$, input set $\mathcal{U}$, abstraction order κ_k, gain of last step $\varphi^{(1)}_{k-1}$, finite time horizon of last step h_{k-1}, $r^{(\lambda*)}_{\text{abs},k'}$ of step k' (last evaluation of (47)), set of global parameters ζ
Output: $\mathcal{R}(t_{k+1})$, $\mathcal{R}(\tau_{k+1})$, $\mathcal{R}^y(t_{k+1})$, κ_{k+1}, h_k, $\varphi^{(1)}_k$

```
1: $h_k \leftarrow h_{k-1} \frac{\zeta_\delta - \zeta_h}{\zeta_\delta - \varphi^{(1)}_{k-1}}$, $\Delta t_k \leftarrow h_k$
2: for $a = 1 : 2$ do
3:     $z^*(t_k) \leftarrow \texttt{linPoint}(\mathcal{R}(t_k), f, \mathcal{R}^y(t_k), g)$
4:     $w^{(x)}, w^{(y)}, C^{(x)}, C^{(y)} \leftarrow \texttt{taylor}\big(f(z), z^*(t_k), \kappa_k\big)$
5:     $w, A, B \leftarrow \texttt{linSys}(w^{(x)}, w^{(y)}, C^{(x)}, C^{(y)})$
6:     $\mathcal{R}_{\text{lin}}(t_{k+1}), \mathcal{R}_{\text{lin}}(\tau_{k+1}) \leftarrow \texttt{linReachAdaptive}(\mathcal{R}(t_k), w, A, B)$
7:     $\Psi, \mathcal{R}^y(t_{k+1}) \leftarrow \texttt{abstrErrLoop}(\mathcal{R}_{\text{lin}}(\tau_{k+1}), \overline{\Psi}, \kappa_k)$
8:     if $a = 1$ then
9:         $\mathcal{R}_{\text{abs}}(h), r^{(1)}_{\text{abs},k} \leftarrow \texttt{abstrSolAdaptive}(\Psi)$
10:        $\Delta t_* \leftarrow \texttt{optDeltat}(h_k, \varphi^{(1)}_k, r^{(1)}_{\text{abs},k})$, $\Delta t_k \leftarrow \Delta t_*$
11:    else
12:        $\mathcal{R}_{\text{abs}}(\Delta t_*), r^{(\lambda*)}_{\text{abs},k} \leftarrow \texttt{abstrSolAdaptive}(\Psi)$
13:        $\kappa_{k+1} \leftarrow \texttt{tuneAbstrOrder}(r^{(\lambda*)}_{\text{abs},k}, r^{(\lambda*)}_{\text{abs},k'})$
14:    end if
15: end for
16: $\varphi^{(1)}_k \leftarrow \texttt{estimateGain}(r^{(1)}_{\text{abs},k}, r^{(\lambda*)}_{\text{abs},k})$
17: $\mathcal{R}(t_{k+1}) = \mathcal{R}_{\text{lin}}(t_{k+1}) \boxplus \mathcal{R}_{\text{abs}}(\Delta t_*)$, $\mathcal{R}(\tau_{k+1}) = \mathcal{R}_{\text{lin}}(\tau_{k+1}) \boxplus \mathcal{R}_{\text{abs}}(\Delta t_*)$
18: $\mathcal{R}(t_{k+1}) \leftarrow \texttt{redAdaptive}(\mathcal{R}(t_{k+1}))$, $\mathcal{R}(\tau_{k+1}) \leftarrow \texttt{redAdaptive}(\mathcal{R}(\tau_{k+1}))$
```

For the evaluation, we simply increase λ' until the objective value $r_R(t+h)$ increases again as such a simple scalar formula does not require more sophisticated algorithms.

As a final step, we have to determine the finite time horizon h for the evaluation of the cost function (57). The key element in the derivation of the optimization problem is the approximative evaluation (55) of $r^{(p)}_{\text{abs}}$ based on the gain φ in (34). The proposed linear interpolation (49) reflects the actual progression of φ over Δt more accurately the closer the used gain $\varphi^{(1)}$ is to the limit gain $\lim_{\Delta t \to 0} \varphi(\Delta t) = \zeta_\delta^{q+1}$, see Theorem 2, which depends on the order q. Using a threshold value $\zeta_h(q)$, we determine h by

$$h = \min \tau \quad \text{such that} \quad \varphi^{(1)}(\tau) \geq \zeta_h(q). \tag{59}$$

Additionally, we restrict $\zeta_h(q)$ to be smaller than ζ_δ^{q+1} since this value is reached from below for small values of Δt as discussed in Section 4.3.

5.5. Automated parameter tuning algorithm

Let us now present Alg. 2, which enhances the reachable set computation shown in Alg. 1 by the adaptive parameter tuning methods introduced in this section. In order to reduce the computational effort, we utilize available information from previous steps for the adaptation of the time step size Δt and the abstraction order κ.

First, we update the finite time horizon h (Line 1) by the value in (59). Using the finite time horizon as the time step size Δt_k, we follow the procedure for the computation of the sets $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ known from Alg. 1, where the operations `linReachAdaptive` and `abstrSolAdaptive` contain the automated tuning of the propagation parameters η_{lin} and η_{abs} as described in Section 5.1. For conciseness, we comprise lines 7–11 from Alg. 1 by the operation `abstrErrLoop`. At the end of this computation, the operation `optDeltat` computes the optimal time step size Δt_*, using the just computed value $r^{(1)}_{\text{abs}} = \texttt{rad}(\mathcal{R}_{\text{abs}}(h))$ and the gain φ_1 from the last step (Line 10). In the second iteration, we compute the sets $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ using $\Delta t_k = \Delta t_*$ and tune the abstraction order κ (Line 13) by the operation `tuneAbstrOrder`, comprising the method from Section 5.2. Additionally, we approximate the gain φ_1 (Line 16) by implicitly solving (56), denoted by the operator `estimateGain`, taking the estimates $r^{(1)}_{\text{abs}}$ and $r^{(\lambda*)}_{\text{abs}}$ for the finite time horizon and the optimal time step size, respectively, as input arguments. At the end of the step, the reachable sets $\mathcal{R}(t_{k+1})$ and $\mathcal{R}(\tau_{k+1})$ are computed and subsequently reduced by the operation `redAdaptive`, according to the method described in Section 5.3.

For the time step size, we first decrease an arbitrarily initialized Δt until the condition in (59) is met, yielding h with the associated error $\mathcal{R}_{\text{abs}}(h)$ and its scalar correspondence $r^{(1)}_{\text{abs}}$ as well as $\varphi^{(1)}$ in the process. Then, the operation `optDeltat` returns the optimal time step size Δt_*, after which the remainder of the step is executed as shown in Alg. 2.

Table 1
Setting of the global parameters ζ.

Approach	$\zeta_{T,\text{lin}}$	$\zeta_{T,\text{abs}}$	ζ_Z	ζ_K	$\zeta_h(q=0)$	$\zeta_h(q=1)$	ζ_δ
Linearization	0.0005	0.005	0.0005	0.90	0.85	0.76	0.90
Polynomialization	0.0005	0.005	0.0001	–	0.80	0.80	0.90

Finally, let us briefly discuss the set of global parameters ζ introduced in the respective tuning methods of the algorithm parameters in this section: Table 1 shows the values to which all global parameters ζ have been fixed. The first three $\zeta_{T,\text{lin}}$, $\zeta_{T,\text{abs}}$, and ζ_Z constitute threshold values representing sufficient accuracy of the tuned set operation. The value of ζ_K is a similarity measure for the comparison of two different abstraction orders. The final two values ζ_h (depending on the order q) and ζ_δ allow us to determine the finite time horizon and candidate time step sizes for the optimization function tuning the time step size. Further development of the proposed tuning methods may change the value of a specific ζ, however, the current setting is justified by the tight reachable sets obtained for a wide variety of different nonlinear systems as shown in the next section.

6. Numerical examples

In this section, we evaluate the adaptive parameter tuning approach presented in the previous section on all system classes introduced in Section 2. We first analyze two selected benchmark systems from the ARCH competition [54,55] allowing us to compare our approach to expert-tuned state-of-the-art reachability tools. Then, a wide variety of different benchmark systems taken from various sources [17,43,56,57] is used to provide a general overview of the performance. The adaptive parameter tuning approach was implemented in MATLAB R2022a and evaluated on an Intel® Core™ i7-9850 CPU @2.59 GHz with 32 GB memory. The following evaluation is based on [30], but considers more configurations of the ARCH benchmarks and extends the additional benchmark systems by including differential–algebraic and discrete-time systems.

6.1. ARCH benchmarks

In the ARCH competition, reachability tools compete with one another in solving benchmark systems, where the computation time and an accuracy measure are used for evaluation. Due to the lack of automated parameter tuning, each tool has to be tuned manually for each system. We select two benchmarks, namely the production-destruction benchmark (PRDE20) and the Laub–Loomis benchmark (LALO20), to assess the quality of our results in comparison with state-of-the-art tools. Let us first introduce the PRDE20 benchmark.

Example 1 (*PRDE20*)**.** This benchmark models a bio-geochemical reaction, describing an algal bloom transforming nutrients (x_1) into detritus (x_3) using phytoplankton (x_2) [58, Sec. 3]. The dynamics presented in [54, Sec. 3.1.1] also contain parametric uncertainty. Based on the initial state $x(0) = (9.98, 0.01, 0.01)^\top$ and the parameter $a = 0.3$, there are three configurations of this benchmark:

1. (Case I) Only uncertainty in the first initial state: $x_1(0) \in [9.50, 10.00]$.
2. (Case P) Only the parameter is uncertain: $a \in [0.296, 0.304]$.
3. (Case I&P) Uncertainty in the first initial state and the parameter: $x_1(0) \in [9.80, 10.00]$, $a \in [0.298, 0.302]$.

The time horizon is $t_{\text{end}} = 100\text{s}$. □

Due to the small size of the initial set $\mathcal{X}^0$, a linearization approach already yields tight reachable sets in all three cases. Fig. 5 shows the reachable sets for case I, which serves as an illustrative example for the tuning of the algorithm parameters. The projections show a sharp turn (in the time interval $t \in [10.6, 11.6]$) imposing strongly nonlinear behavior which is both preceded and succeeded by rather calm dynamics.

Fig. 6 shows how the adaptive parameter tuning reacts to the change in the degree of nonlinearity: The left graph plots the time step size Δt over time, which reaches its minimum value $\Delta t \approx 0.012$ during the sharp turn. There, the optimization function reduces Δt, thus decreases the abstraction error in order to optimize the estimated over-approximation error at that time. Afterwards, the value gradually increases towards its maximum value $\Delta t \approx 0.4$, which exploits that the dynamics are better approximated in rather linear regions, yielding small abstraction errors even for relatively large time step sizes. The right graph plots the zonotope order ρ over time whose behavior can be explained in a similar way: At the sharp turn, more generators have to be kept in order to avoid inducing large over-approximations, yielding a maximum zonotope order of $\rho = 20$. As a consequence of the calmer dynamics after the sharp turn, the complexity of the shape decreases as we observe in Fig. 5: The sets after the turn are much more straightened compared to the "bent" sets at the time of the turn. This reduces the number of generators required for an accurate representation of the set and thus lowers the zonotope order.

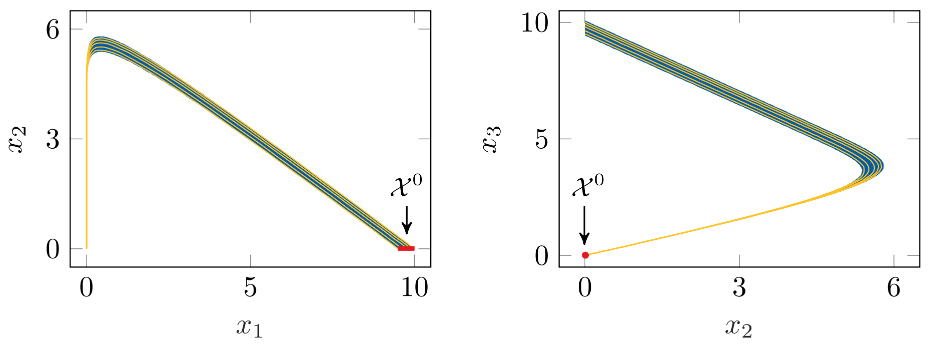


Fig. 5. Projections of the reachable set $\mathcal{R}([0, t_{\text{end}}])$ of Example 1, case I. Initial set in red, reachable sets in blue, single simulation runs in yellow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

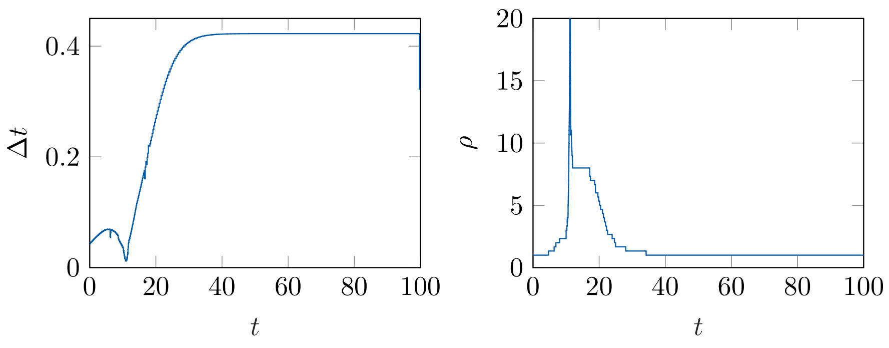


Fig. 6. Time step size Δt and zonotope order ρ of Example 1, case I, over time.

Table 2
ARCH benchmark PRDE20: Comparison of our approach with state-of-the-art reachability tools in terms of computation time and the tightness measurement $\mu = \mathtt{vol}(\mathtt{box}(\mathcal{R}(t_{end})))$ as in [54].

Tool (Language)	Case I		Case P		Case I&P	
	Time	μ	Time	μ	Time	μ
Alg. 2 (Matlab)	10.8s	7.8e−21	12.4s	**3.5e−24**	18.6s	**8.6e−24**
Ariadne (C++)	8.6s	1.7e−13	79s	2.3e−17	39s	1.0e−13
CORA (Matlab)	16s	**1.2e−21**	28s	2.2e−23	28s	2.0e−23
DynIbex (C++)	12s	3.9e−17	13s	4.8e−17	26s	1.2e−17
Flow* (C++)	4.1s	8.0e−21	9s	1.4e−22	5.2s	4.8e−21
Isabelle/HOL (SML)	11s	3.3e−20	12s	7.3e−21	26s	2.6e−20
JuliaReach (Julia)	**1.5s**	3.3e−20	**3.9s**	6.5e−21	**3.0s**	1.0e−20

The propagation parameters η_{lin} and η_{abs} do not change much over time as we have $\eta_{\text{lin}} \in \{4, 5, 6\}$ and $\eta_{\text{abs}} \in \{2, 3\}$, where the respective maxima are reached at the sharp turn. The tuning of the abstraction order κ results in $\kappa = 2$ at the beginning until $t \approx 17.3$ and $\kappa = 1$ for the remainder of the time horizon, thereby confirming the rather linear dynamics after the sharp turn.

Table 2 allows us to compare the results obtained by our adaptive parameter tuning approach with state-of-the-art reachability tools for all three cases specified in Example 1. The obtained accuracy ranks among the best, even topping the chart in cases P and I&P. The computation time is average in all cases, partly caused by the speed discrepancy in programming languages as C++ and Julia are known to operate faster than MATLAB. The comparison with CORA in particular shows competitiveness since our computation is faster due to the large ratio of the largest to the smallest time step size saving many time steps. Next, we consider the Laub–Loomis benchmark.

Example 2 (*LALO20*)**.** The dynamics of this benchmark [55, Sec. 3.3.1] represent changes in enzymatic activities introduced in [59, (1-7)]. The initial set is given by $\mathcal{X}^0 = [x(0) - W, x(0) + W]$, where $x(0) = (1.2, 1.05, 1.5, 2.4, 1, 0.1, 0.45)^\top$ is enlarged by either of the uncertainties $W \in \{0.01, 0.05, 0.1\}$, representing configurations of increasing difficulty. The time horizon is $t_{\text{end}} = 20s$. □

While a linearization approach still suffices for small ($W = 0.01$) and moderate ($W = 0.05$) sizes of the initial set $\mathcal{X}^0$, the largest size ($W = 0.1$) can only be solved using a polynomialization approach. For conciseness, we only plot the

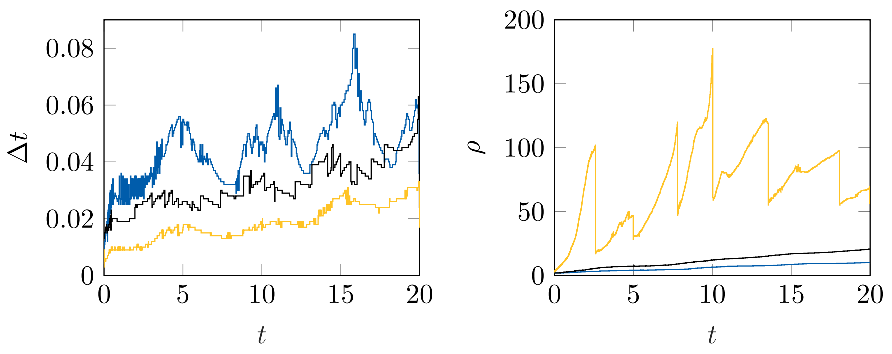


Fig. 7. Time step size Δt and zonotope order ρ of Example 2 over time: Different cases $W = \{0.01, 0.05, 0.1\}$ in blue, black, and yellow, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
ARCH benchmark LALO20: Comparison of our approach with state-of-the-art reachability tools in terms of computation time and the tightness measurement $\mu = l_4$, where $l = d(\mathrm{box}(\mathcal{R}(t_{end})))$ as in [55].

Tool (Language)	$W = 0.01$		$W = 0.05$		$W = 0.1$	
	Time	μ	Time	μ	Time	μ
Alg. 2 (Matlab)	3.9s	**0.004**	7.8s	0.049	91s	0.068
Ariadne (C++)	5.7s	0.01	11s	0.031	31s	0.071
CORA (Matlab)	1.9s	0.005	8.4s	0.035	38s	0.116
DynIbex (C++)	10s	0.01	27s	0.40	1851s	2.07
JuliaReach (Julia)	**1.1s**	**0.004**	**1.5s**	**0.017**	**1.4s**	**0.033**
Kaa (Python)	238s	22	253s	23	257s	49

time step size Δt and the zonotope order ρ over time for all three cases ($W \in \{0.01, 0.05, 0.1\}$) in Fig. 7: All curves for Δt increase similarly over time in multiple waves, where the lowermost curve is obtained for $W = 0.1$, the middle curve for $W = 0.05$, and the uppermost curve for $W = 0.01$, showing that a smaller set size allows larger time step sizes and vice versa. The curve of the zonotope order ρ for the case $W = 0.1$ (uppermost curve) differs from the ones for $W = 0.05$ (middle curve) and $W = 0.1$ (lowermost curve) because the polynomialization approach uses non-convex sets. The vertical drops of ρ are caused by the restructuring operation [46, Prop. 17], where all independent generators are first reduced and then converted to dependent generators for reasons of computational accuracy in subsequent steps. Using a linearization approach, the curves for ρ reach their maximum at the end of the time horizon at values of 10 and 20 for $W = 0.01$ and $W = 0.05$, respectively, which keeps the set operations efficient without compromising the tightness of the reachable sets.

The evaluation of the LALO20 benchmark in both computation time and accuracy is shown in Table 3, offering a similar picture as for the PRDE20 benchmark: Again, our computation times are only average across all tools, mainly due to the costly evaluation of the abstraction error Ψ as well as the computationally demanding reduction of the set representation size for the system dimension $n = 7$. In contrast, the accuracy is better than most others, being co-leader for the smallest size $W = 0.01$ and second for the largest size $W = 0.1$. This demonstrates the competitiveness of our adaptive parameter tuning for both linearization and polynomialization approaches.

6.2. Further benchmarks

After the detailed discussion of the ARCH benchmarks, we now analyze the performance on a broader range of benchmarks, also including differential–algebraic (DA) and discrete-time systems. Table 4 provides some information about the benchmarks, such as the system dimension n and algebraic dimension n_a as well as the time horizon t_{end} and the initial set $\mathcal{X}^0$. The benchmarks range from standard models like the van-der-Pol oscillator over chaotic systems, such as the Roessler attractor and Lorenz attractor, to higher-dimensional biologically and mechanically inspired models. Both differential–algebraic models are power systems, namely a 3-bus system and a single machine infinite bus (SMIB) system, where the algebraic equations originate from the network constraints. The SMIB system has different dynamics for the standard operation and fault scenario caused by a loss in the network connection occurring at $t \in [0.01, 0.02]$. Finally, we discretized a six-dimensional water tank benchmark whose dynamics are based on Torricelli's Law using a time step size of $\Delta t = 0.05$ s.

The tightness of the reachable sets is quantified using two different metrics: First, we provide the longest edge of the box over-approximation of the final set $\mathcal{R}(t_{\mathrm{end}})$, namely,

$$d_{\max} = \max_{i \in \{1,\dots,n\}} d_i\big(\mathrm{box}(\mathcal{R}(t_{\mathrm{end}}))\big). \tag{60}$$

Table 4
List of considered benchmarks: n: system dimension, n_a: algebraic dimension, t_{end}: time horizon, $\mathcal{X}^0$: initial set.

Benchmark	n	n_a	t_{end}	$\mathcal{X}^0$
Jet Engine [60, (19)]	2	–	8	$[0.9, 1.1]^n$
van der Pol [17, Sec. VII]	2	–	6.74	$\big([1.30, 1.50][2.35, 2.45]\big)^\top$
Brusselator [43, Ex. 3.4.1]	2	–	5	$\big([0.9, 1.0][0.0, 0.1]\big)^\top$
Roessler [61, (2)]	3	–	6	$\big([-0.2, 0.2][-8.6, -8.2][-0.2, 0.2]\big)^\top$
Lorenz [62, (25-27)]	3	–	2	$\big([14.9, 15.1][14.9, 15.1][34.9, 35.1]\big)^\top$
Spring-Pendulum [43, Ex. 3.3.12]	4	-	1	$\big([1.1, 1.3][0.4, 0.6][0.0, 0.1][0.0, 0.1]\big)^\top$
Lotka–Volterra [63, (1)]	5	–	5	$[0.90, 1.00]^n$
Biological Model [64]	7	–	2	$[0.99, 1.01]^n$
Genetic Model [65, (1)]	9	–	0.1	see [56, Sec. V.]
3-Bus [66, Sec. 4]	2	6	5	$\big([379.90, 380.10][0.69, 0.71]\big)^\top$
SMIB [57, Sec. 2.5.1.2]	2	4	0.23	$\big([0.65075, 0.66675][0.008, 0.008]\big)^\top$
Tank-DT [17, Sec. VII]	6	-	100	$\big([1.8, 2.2][3.8, 4.2][3.8, 4.2][1.8, 2.2][9.8, 10.2][3.8, 4.2]\big)^\top$

Table 5
Evaluation of nonlinear benchmark systems using an adaptively tuned linearization and polynomialization approaches: $[\Delta t_{\min}, \Delta t_{\max}]$: range of time step sizes, $\rho_{\max}$: max. zonotope order, $d_{\max}$ and $\gamma_{\min}$: measurements by (60) and (61).

Benchmark	Linearization Approach					Polynomialization Approach				
	Time	$[\Delta t_{\min}, \Delta t_{\max}]$	$\rho_{\max}$	$d_{\max}$	$\gamma_{\min}$	Time	$[\Delta t_{\min}, \Delta t_{\max}]$	$\rho_{\max}$	$d_{\max}$	$\gamma_{\min}$
Jet Engine	2.5s	$[0.007, 0.117]$	13.5	0.0562	0.5594	11.8s	$[0.002, 0.049]$	26	0.0395	0.7955
van der Pol	5.5s	$[0.002, 0.051]$	16.5	1.79	0.2004	26s	$[0.0005, 0.0127]$	47.5	0.5234	0.6621
Brusselator	2.0s	$[0.011, 0.062]$	80.5	0.075	0.7901	9.5s	$[0.004, 0.021]$	219.5	0.065	0.9469
Roessler	2.7s	$[0.006, 0.047]$	10.33	4.34	0.1725	13.7s	$[0.0022, 0.0328]$	20.33	2.47	0.6623
Lorenz	4.1s	$[0.0004, 0.0098]$	10	0.268	0.8337	10.8s	$[0.0004, 0.0067]$	46	0.237	0.9562
Spring-Pendulum	4.5s	$[0.006, 0.022]$	12.75	0.522	0.6484	10.7s	$[0.002, 0.009]$	42.75	0.424	0.7884
Lotka–Volterra	1.0s	$[0.010, 0.107]$	12.2	0.083	0.8722	4.1s	$[0.003, 0.078]$	195.2	0.074	0.9794
Biological Model	1.8s	$[0.004, 0.019]$	44.71	0.117	0.7115	10.7s	$[0.001, 0.008]$	182.14	0.094	0.9163
Genetic Model	0.7s	$[0.0005, 0.0023]$	6	5.55	0.7939	2.7s	$[0.0002, 0.0010]$	37.11	5.30	0.9509
3-Bus	4.3s	$[0.015, 0.054]$	13	2.28	0.6791	–	–	–	–	–
SMIB	36s	$[0.00005, 0.00200]$	6.5	0.0004	0.3059	–	–	–	–	–
Tank-DT	13s	fixed to 0.05	28.67	0.6246	0.7517	47s	fixed to 0.05	35.5	0.6044	0.7689

Second, we use the ratio of under-approximation to over-approximation proposed in [67, Sec. VI.]

$$\gamma_{\min} = \min_{i \in \{1,\dots,n\}} \frac{d_i\big(\mathtt{box}(\mathcal{R}_{\text{sim}}(t_{\text{end}}))\big)}{d_i\big(\mathtt{box}(\mathcal{R}(t_{\text{end}}))\big)}, \tag{61}$$

where $\mathcal{R}_{\text{sim}}(t_{\text{end}})$ denotes the set of states at t_{end} of 1000 simulation runs. The tightness increases for $\gamma_{\min} \to 1$ as the under-/over-approximation approach one another. Space constraints prevent a detailed discussion of every result, which is why we will discuss general tendencies as well as unexpected results.

Table 5 shows the results for all systems from Table 4 using a linearization and a polynomialization approach with adaptively tuned algorithm parameters. The linearization approach is by construction limited to systems with only mild nonlinearities, leading to low values of $\gamma_{\min}$ for the Roessler attractor and the van-der-Pol oscillator (similar results for the latter have already been discussed in [46, Sec. 4]). The tightness is still satisfactory in most cases, especially where $\gamma_{\min} > 0.7$. Moreover, the computation times and tightness measures are similar over increasing system dimension, showing the scalability of our proposed tuning methods. In contrast, the polynomialization approach yields both higher computation times and improved accuracy according to the tightness measures $d_{\max}$ and $\gamma_{\min}$. Both approaches exploit the range of different time step sizes of 1–2 orders of magnitude on average. The total number of steps in the analysis is drastically reduced, thus significantly speeding up the computation compared to fixed time step sizes.

For the linearization approach, the maximum zonotope order $\rho_{\max}$ is often rather low (between 10 and 20); for other cases, it should be noted that the highest order may only last for a few steps as shown in Fig. 6 and therefore does not pose major problems to the efficiency of the computations. For the polynomialization approach, higher zonotope orders are reached because the reduction of polynomial zonotopes is too over-approximative to allow for substantial reductions. This also entails an increase in computation time since the set operations then become more costly for larger set representation sizes, as well as an increase in accuracy, where we note that five systems reach a value of $\gamma_{\min} > 0.9$. This leads to the conclusion that the reduction of the set representation within the polynomialization approach is its limiting factor for the success of the algorithm. Our adaptive parameter tuning would greatly benefit, especially for polynomial zonotopes, from improvements in order reduction techniques similar to our considerations presented in Section 3 for (linear) zonotopes, as updated methods can simply replace existing ones due to the modularity of our tuning framework.

Finally, we discuss the results for the DA systems, for which there does not yet exist a polynomialization algorithm. The evaluation of both DA systems does not show major differences to standard nonlinear systems, except for the high

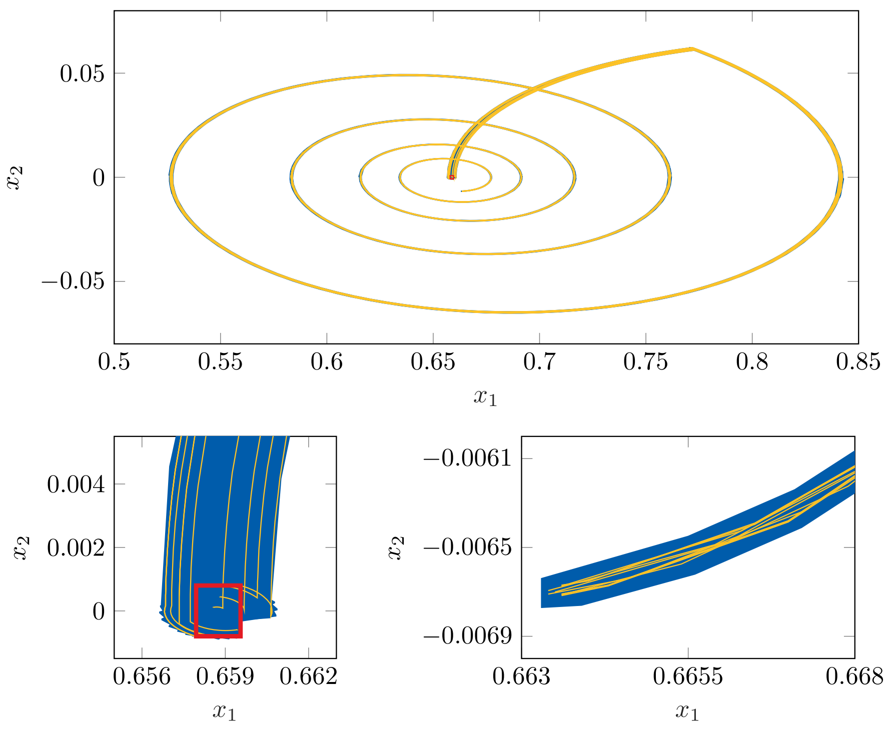


Fig. 8. Reachable sets of the SMIB system, see Table 4, over the full time horizon (top), at the start (bottom left), and at the end (bottom right). The divergence from the initial set (red) is caused by the fault scenario, after the return to the normal operation mode the system behavior stabilizes. The reachable sets (blue) are largely covered by single simulation runs (yellow). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

computation time and low tightness measure $\gamma_{\min}$ for the SMIB system. Fig. 8 shows the reachable sets, putting the low value for $\gamma_{\min}$ in perspective, as the enclosure of the simulated trajectories is still tight. In the discrete-time example, the time step size is fixed by definition, yielding a total number of 2000 steps. The suitability of the remaining tuning methods is shown by the fairly low value for $\rho_{\max}$ and high value for $\gamma_{\min}$.

In summary, our evaluation shows that the presented methods for adaptive parameter tuning allow us to obtain tight reachable sets in a broad variety of different systems. Due to the fully automatic tuning, the computation of the reachable sets is executed in a single run as opposed to the many trial-and-error runs in manual tuning.

7. Conclusion

We presented the first fully-automatic reachability algorithm for nonlinear systems. To this end, the fundamentals of the two main wrapping effects in reachability analysis of nonlinear systems have been thoroughly investigated: First, we presented an exhaustive derivation of various bounds for the Hausdorff distance between a zonotope and its box over-approximation, with associated generator selection strategies for the order reduction of zonotopes. Second, a rigorous examination of the set of abstraction errors accounting for higher-order nonlinearities led to the introduction of a *gain order*, which describes the effect of the time step size on the local and global abstraction error in the analysis. These theoretical insights were then utilized in our adaptive parameter tuning algorithm, most notably for the derivation of an optimization function to tune the time step size. The evaluation on multiple nonlinear system classes showed competitiveness with state-of-the-art reachability tools, as well as an efficient computation of tight reachable sets in a variety of further benchmark problems. Our approach requires no longer expert knowledge about the reachability algorithm, which greatly simplifies the usage of reachability analysis in a broad variety of possible applications.

CRediT authorship contribution statement

Mark Wetzlinger: Writing – original draft, Writing – review & editing, Software, Validation, Investigation, Visualization, Conceptualization. **Adrian Kulmburg:** Writing – original draft, Writing – review & editing, Investigation, Visualization, Conceptualization. **Alexis Le Penven:** Investigation, Resources. **Matthias Althoff:** Supervision, Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

The authors gratefully acknowledge partial financial supports from the research training group ConVeY funded by the German Research Foundation under grant GRK 2428 and the project justITSELF funded by the European Research Council (ERC) under grant agreement No. 817629.

Appendix A. Proof of Theorem 1

Let us express each point $x_{\text{box}} \in \mathcal{Z}_{\text{box}}$ as

$$x_{\text{box}} = M^{(1)}|g^{(1)}| + \cdots + M^{(\gamma)}|g^{(\gamma)}|\,,$$

where each $M^{(p)}$ is a diagonal matrix with diagonal entries $\mu_i^{(p)} \in [-1, 1]$ where $i \in \{1, \ldots, n\}$. The function $\|x_{\text{box}} - x\|_2$ is convex w.r.t. $\mu_i^{(p)}$ allowing us to restrict our attention to the cases where $\mu_i^{(p)} \in \{-1, 1\}$, since by the Bauer maximum principle (see [68]), the maximum of a convex function over the polytope $[-1, 1]^{\gamma n}$ is always reached at one of its vertices, i.e., some element of $\{-1, 1\}^{\gamma n}$. Let us write the difference between any $x_{\text{box}} \in \mathcal{Z}_{\text{box}}$ and $x \in \mathcal{Z}$ as

$$x_{\text{box}} - x = \left(M^{(1)}|g^{(1)}| - \alpha_1 g^{(1)}\right) + \cdots + \left(M^{(\gamma)}|g^{(\gamma)}| - \alpha_\gamma g^{(\gamma)}\right), \tag{A.1}$$

where $\forall p \in \{1, \ldots, \gamma\} : \alpha_p \in [-1, 1]$. Note that the minimum of $\|x_{\text{box}} - x\|_2$ w.r.t. α_p does not have a closed-form formula in general [69, Sec. 9]. However, one can obtain a bound on $x_{\text{box}} - x$ by choosing a specific α_p for each $g^{(p)}$. The bounds $\omega_{\max}$(15), ω_{rad} (16), and ω_{cut} (17) are obtained by different choices for α_p and subsequently derived in detail:

Bound $\omega_{\max}$: Let us start with the following choice for α_p:

$$\alpha_p = \mu_{i^*}^{(p)} \operatorname{sgn}\left(g_{i^*}^{(p)}\right), \tag{A.2}$$

with an individual i^* for each p as in Theorem 1. Consequently, we can eliminate the largest possible entry in

$$v^{(p)} = M^{(p)}|g^{(p)}| - \alpha_p g^{(p)}, \tag{A.3}$$

for which we obtain the bound

$$v_i^{(p)} \in \begin{cases} \left[-2|g_i^{(p)}|, 2|g_i^{(p)}|\right], & \text{if } i \neq i^* \\ 0, & \text{otherwise,} \end{cases}$$

which we can rewrite to $v_i^{(p)} \in [-2\widehat{g}_i^{(p)}, 2\widehat{g}_i^{(p)}]$ using (18). Applying (A.2) to each generator, we obtain the bounds

$$x_{\text{box}} - x = v^{(1)} + \cdots + v^{(\gamma)} \in [-2\widehat{z}, 2\widehat{z}]\,,$$

where $\widehat{z} = \widehat{g}^{(1)} + \cdots + \widehat{g}^{(\gamma)}$ and ultimately,

$$\|x_{\text{box}} - x\|_2 \leq \|2\widehat{z}\|_2 = 2\,\|\widehat{z}\|_2\,.$$

The above bound holds for any $x_{\text{box}} \in \mathcal{Z}_{\text{box}}$, which fulfills the assumption of [30, Lemma 3.1] and thus proves that $d_H(\mathcal{Z}, \mathcal{Z}_{\text{box}}) \leq \omega_{\max}$.

Bound ω_{rad}: Another way to choose α_p is to find an optimal minimum of $\|v^{(p)}\|_2$ defined in (A.3) w.r.t. α_p and then use the inequality

$$\|x_{\text{box}} - x\|_2 \leq \sum_{p=0}^{\gamma} \left\|v^{(p)}\right\|_2.$$

Since the exact minimum of $\|v^{(p)}\|_2$ equal to the minimum of $\|v^{(p)}\|_2^2$, we insert (A.3) into the squared expression, differentiate w.r.t α_p, and solve for α_p, yielding the minimizer

$$\alpha_p^* = \frac{\sum_{i=1}^n \mu_i^{(p)} \operatorname{sgn}(g_i^{(p)}) \left(g_i^{(p)}\right)^2}{\left\|g^{(p)}\right\|_2^2}.$$

By inserting the expression for α_p^* back into Eq. (A.3), one obtains

$$\min_{x \in \mathcal{Z}} \left\|v^{(p)}\right\|_2 = \left\|g^{(p)}\right\|_2 \sqrt{1 - \left(\frac{\sum_{i=1}^n \mu_i^{(p)} \operatorname{sgn}(g_i^{(p)}) \left(g_i^{(p)}\right)^2}{\left\|g^{(p)}\right\|_2^2}\right)^2}.$$

Since we maximize this expression w.r.t. $\mu_i^{(p)} \in \{-1, 1\}$ and $v^{(l)}$ does not depend on any $\mu_i^{(p)}$ if $p \neq l$, we can replace expressions such as $\mu_i^{(p)} \operatorname{sgn}(g_i^{(p)})$ by $\widehat{\mu}_i^{(p)} \in \{-1, 1\}$, which yields

$$\max_{x_{\text{box}} \in \mathcal{Z}_{\text{box}}} \min_{x \in \mathcal{Z}} \|x_{\text{box}} - x\|_2 \leq \max_{\widehat{\mu}_i^{(l)} \in \{-1,1\}^{\gamma}} \sum_{p=0}^{\gamma} \left\|g^{(p)}\right\|_2 \sqrt{1 - \left(\frac{\sum_{i=1}^n \widehat{\mu}_i^{(p)} \left(g_i^{(p)}\right)^2}{\left\|g^{(p)}\right\|_2^2}\right)^2} .$$

Each summand depends on exactly one $\mu_i^{(p)}$, thus the sum and the maximum commute, yielding

$$\max_{x_{\text{box}} \in \mathcal{Z}_{\text{box}}} \min_{x \in \mathcal{Z}} \|x_{\text{box}} - x\|_2 \leq \sum_{p=0}^{\gamma} \left\|g^{(p)}\right\|_2 \sqrt{1 - \min_{\mu \in \{-1,1\}^n} \left(\frac{\sum_{i=1}^n \mu_i \left(g_i^{(p)}\right)^2}{\left\|g^{(p)}\right\|_2^2}\right)^2} . \tag{A.4}$$

To get a new bound on the Hausdorff distance, we can therefore restrict ourselves to finding a lower-approximation of

$$C_p := \min_{\mu \in \{-1,1\}^n} \left[\sum_{i=1}^n \mu_i \left(g_i^{(p)}\right)^2\right]^2 = \min_{\mu \in \{-1,1\}^n} \sum_{i=1}^n \sum_{j=1}^n \mu_i \mu_j \left(g_i^{(p)}\right)^2 \left(g_j^{(p)}\right)^2 . \tag{A.5}$$

Using the trivial estimate $C_p \geq 0$, we can simplify (A.4) to

$$\max_{x_{\text{box}} \in \mathcal{Z}_{\text{box}}} \min_{x \in \mathcal{Z}} \|x_{\text{box}} - x\|_2 \leq \sum_{p=1}^{\gamma} \left\|g^{(p)}\right\|_2 ,$$

which proves that ω_{rad} is a valid over-approximation of the Hausdorff distance.

Bound ω_{cut}: For our final bound, we reformulate (A.5) to

$$C_p = \min_{\mu \in \{-1,1\}^n} \sum_{i=1}^n \left(g_i^{(p)}\right)^4 + \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n \mu_i \mu_j \left(g_i^{(p)}\right)^2 \left(g_j^{(p)}\right)^2 .$$

Since $\mu_i \in \{-1, 1\}$, and thus $\forall i, j = \{1, \dots, n\} : \mu_i \mu_j \geq -1$, we deduce that

$$C_p \geq 2 \sum_{i=1}^n \left(g_i^{(p)}\right)^4 - \left\|g^{(p)}\right\|_2^4 ,$$

which yields

$$\max_{x_{\text{box}} \in \mathcal{Z}_{\text{box}}} \min_{x \in \mathcal{Z}} \|x_{\text{box}} - x\|_2 \leq \sum_{p=1}^{\gamma} \left\|g^{(p)}\right\|_2 \sqrt{2 - 2\frac{\sum_{i=1}^n \left(g_i^{(p)}\right)^4}{\left\|g^{(p)}\right\|_2^4}},$$

proving that ω_{cut} is also an over-approximation of the Hausdorff distance. □

Appendix B. Proofs for Section 4

Proof of Lemma 2. (inspired by [32, p. 318, Theorem 12.2], with a few adaptations due to the set-based nature of the computations)

For ease of notation, we drop the Δt-dependency of $A(\Delta t)$ and write A instead. Furthermore, for simplicity, we will ignore all order reduction operations. In that case, adding an arbitrary zonotope $\mathcal{Z}$ centered at 0 to the initial set $\mathcal{X}^0$ does not influence the expansion point z^* (see (6)), thus the set of particular solutions $\mathcal{P}([t_0, t_0 + \Delta t])$ is unaffected (see also [49, (3.5)]). Since $\mathcal{R}_{\text{abs}}(\Delta t; t_k, \mathcal{R}(t_k))$ is either a zonotope centered at the origin, or can w.l.o.g. be over-approximated by one, by Definition 4 we can write

$$\forall k \in \{0, 1, \dots, N\} : \quad \mathcal{R}(t_{k+1}) = e^{A\Delta t}\mathcal{R}(t_k) \oplus \mathcal{P}(\tau_k) \boxplus \mathcal{R}_{\text{abs}}(\Delta t; t_k, \mathcal{R}(t_k)), \tag{B.1}$$

$$\widehat{\mathcal{R}}(t_{k+1}) = e^{A\Delta t}\widehat{\mathcal{R}}(t_k) \oplus \mathcal{P}(\tau_k). \tag{B.2}$$

Crucially, both (B.1) and (B.2) share the same term $\mathcal{P}(\tau_k)$. Therefore, the global abstraction error ε_{k+1} after $k + 1$ steps defined in Definition 4 may be written as

$$\begin{aligned}
\varepsilon_{k+1} = \; & \left\| d\big(\mathrm{box}(\mathcal{R}(t_{k+1}) \ominus \widehat{\mathcal{R}}(t_{k+1}))\big) \right\|_\infty \\
\overset{\text{(B.1) and (B.2)}}{=} \; & \left\| d\big(\mathrm{box}(e^{A\Delta t}\mathcal{R}(t_k) \oplus \mathcal{P}(\tau_k) \boxplus \mathcal{R}_{\mathrm{abs}}(\Delta t; t_k, \mathcal{R}(t_k)) \ominus e^{A\Delta t}\widehat{\mathcal{R}}(t_k) \ominus \mathcal{P}(\tau_k))\big) \right\|_\infty \\
\overset{\mathcal{P}(\tau_k)\ominus\mathcal{P}(\tau_k)=\{0\}}{=} \; & \left\| d\big(\mathrm{box}(e^{A\Delta t}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)) \boxplus \mathcal{R}_{\mathrm{abs}}(\Delta t; t_k, \mathcal{R}(t_k), \Delta t))\big) \right\|_\infty \\
\overset{\text{Triangle ineq.}}{\leq} \; & \left\| d\big(\mathrm{box}(e^{A\Delta t}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)))\big) \right\|_\infty + \left\| d\big(\mathrm{box}(\mathcal{R}_{\mathrm{abs}}(\Delta t; t_k, \mathcal{R}(t_k)))\big) \right\|_\infty .
\end{aligned}$$

With a few simple calculations one can show that

$$\begin{aligned}
\left\| d\big(\mathrm{box}(e^{A\Delta t}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)))\big) \right\|_\infty \overset{\substack{\text{Taylor approx.}\\ \text{of } e^{A\Delta t}}}{=} \; & \left\| d\big(\mathrm{box}((I + \sum_{i=1}^{\infty} \frac{A^i \Delta t^i}{i!})(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)))\big) \right\|_\infty \\
\overset{\text{Triangle ineq.}}{\leq} \; & \left\| d\big(\mathrm{box}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k))\big) \right\|_\infty + \left\| d\big(\mathrm{box}(\sum_{i=1}^{\infty} \frac{A^i \Delta t^i}{i!}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)))\big) \right\|_\infty \\
\overset{\substack{\text{Definition 4 and}\\ i\to i-1}}{=} \; & \varepsilon_k + \Delta t \left\| d\big(\mathrm{box}(\sum_{i=0}^{\infty} \frac{A^{i+1} \Delta t^i}{(i+1)!}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k)))\big) \right\|_\infty \\
\leq \; & \varepsilon_k + \Delta t \left\| \sum_{i=0}^{\infty} \frac{A^{i+1} \Delta t^i}{(i+1)!} \right\|_\infty \left\| d\big(\mathrm{box}(\mathcal{R}(t_k) \ominus \widehat{\mathcal{R}}(t_k))\big) \right\|_\infty \\
\overset{\text{Definition 4}}{=} \; & \varepsilon_k + \Delta t \left\| \sum_{i=0}^{\infty} \frac{A^{i+1} \Delta t^i}{(i+1)!} \right\|_\infty \varepsilon_k .
\end{aligned}$$

In order to make this bound independent of Δt, we can use the fact that $\| \sum_{i=0}^{\infty} \frac{A^{i+1}\Delta t^i}{(i+1)!} \|_\infty$ is continuous w.r.t. Δt, and is thus Lipschitz continuous over the bounded domain $\Delta t \in [0, t_{\mathrm{end}}]$ with a Lipschitz constant $\widehat{L} \geq 0$ that may depend on t_{end} but not Δt. Combining this again with $\Delta t \leq t_{\mathrm{end}}$, we obtain a bound

$$\left\| \sum_{i=0}^{\infty} \frac{A^{i+1} \Delta t^i}{(i+1)!} \right\|_\infty \leq \widehat{L}\Delta t \leq \widehat{L} t_{\mathrm{end}} =: L,$$

where L is now a constant that is independent of Δt. This implies the following iterative bound on ε_{k+1}:

$$\varepsilon_{k+1} \leq (1 + L\Delta t)\varepsilon_k + \varepsilon_{k,\mathrm{loc}}.$$

From this point onwards, we can use the same argument as in [32, p. 318, Theorem 12.2] to show that

$$\varepsilon_k \leq \frac{\max_{1\leq \ell \leq N} \varepsilon_{\ell,\mathrm{loc}}}{\Delta t} \frac{1}{L}(e^{Lt_{\mathrm{end}}} - 1).$$

The coefficient $\frac{1}{L}(e^{Lt_{\mathrm{end}}} - 1)$ may depend on t_{end}, but not N nor Δt, which yields (30). □

Proof of Proposition 1. Let h be arbitrary, but fixed. Let $\mathcal{S} \subset \mathbb{R}^n$ be some bounded, centrally symmetric set with center c. Then we have

$$d_i\big(\mathrm{box}(\mathcal{S})\big) = \max_{s\in\mathcal{S}} s_i - \min_{s\in\mathcal{S}} s_i = 2 \max_{s\in\mathcal{S}} |s_i - c_i|. \tag{B.3}$$

After inserting the definition $\Psi(h) = \langle c(h), G(h)\rangle_Z$ in (32) and extracting the center $\widehat{c} := \sum_{p=0}^{\infty} \frac{h^{p+1}}{(p+1)!} A^p(h)c(h)$, we can apply (B.3) to (32) to obtain

$$d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h))\big) = 2 \max_{\substack{\beta^{(p)}\in[-1,1]^m\\ p\in\mathbb{N}_0}} \left| \sum_{p=0}^{\infty} \frac{h^{p+1}}{(p+1)!} A^p(h)G(h)\beta^{(p)} \right|_i .$$

Since the maximization term is convex w.r.t. the $\beta^{(p)}$, we can change the domain of $\beta^{(p)}$ from $[-1, 1]^m$ to $\{-1, 1\}^m$ by the Bauer maximum principle (see [68]). Each summand depends on exactly one $\beta^{(p)}$, thus we obtain

$$d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h))\big) = 2 \sum_{p=0}^{\infty} \max_{\beta\in\{-1,1\}^m} \left| \frac{h^{p+1}}{(p+1)!} A^p(h)G(h)\beta \right|_i = 2 \sum_{p=0}^{\infty} \frac{h^{p+1}}{(p+1)!} \|A^p(h)G(h)\|_{1,i},$$

using the fact that the maximum of $|M\beta|_i$ for a matrix M and $\beta \in \{-1, 1\}^m$ can easily be seen to be $\|M\|_{1,i}$. □

Proof of Lemma 3. The fact that $A(h)$ and $G(h)$ are both smooth follows for example from [18, p. 4] since f is smooth. From this, it follows from the Taylor expansion of $[A(h)G(h)]_i$ that for any arbitrarily large N, there exists an expansion of the form

$$[A^p(h)G(h)]_i = \sum_{j=0}^{N} c_j h^j + \epsilon(h), \tag{B.4}$$

where $c_j \in \mathbb{R}$ for all $j \in \{1, \dots, N\}$ and $\epsilon(h) = \mathcal{O}(h^{N+1})$. Let j be the smallest index such that $c_j \neq 0$, and define $q_i^{(p)}$ to be this index. If for all $N \in \mathbb{N}_0$ such an index does not exist, it trivially follows that $[A^p(h)G(h)]_i \equiv 0$, and in that case we can set $q_i^{(p)} = \infty$. If $q_i^{(p)} < \infty$, we may rewrite (B.4) for all $N > q_i^{(p)}$ as

$$[A^p(h)G(h)]_i = c_{q_i^{(p)}} h^{q_i^{(p)}} + \sum_{j=q_i^{(p)}+1}^{N} c_j h^j + \epsilon(h).$$

Clearly, since $\epsilon(h) = \mathcal{O}(h^{N+1})$ this function may be written as $\epsilon(h) = h^{q_i^{(p)}} \widehat{\epsilon}(h)$ where $\widehat{\epsilon}(h) = \mathcal{O}(h^{N+1-q_i^{(p)}})$, and more specifically $\widehat{\epsilon}(h) = \mathcal{O}(h)$. Therefore, by defining $a_i^{(p)} = c_{q_i^{(p)}}$ and $b_i^{(p)}(h) = \sum_{j=q_i^{(p)}+1}^{N} c_j h^{j-q_i^{(p)}} + \widehat{\epsilon}(h)$, we obtain the form (36).

Finally, the function $Q_i^{(p)}(h)$ is easily seen to be non-negative since $\|\cdot\|_1$ is non-negative, and piecewise smooth since $\|\cdot\|_1$ is smooth almost everywhere, and $b_i^{(p)}(h)$ is smooth since $A(h)$ and $G(h)$ are smooth. □

Proof of Proposition 2. We begin by proving (41). Differentiating φ_i in (34) using Lemma 3 yields

$$\frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} = \frac{\mathrm{d}}{\mathrm{d}h} \frac{d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(\delta h))\big)}{d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h))\big)} = \frac{\delta \frac{\mathrm{d}}{\mathrm{d}h'}\Big[d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h'))\big)\Big]\Big|_{h'=\delta h} - \varphi_i(h;\delta)\frac{\mathrm{d}}{\mathrm{d}h'}\Big[d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h'))\big)\Big]\Big|_{h'=h}}{d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h))\big)}.$$

By using the expansion of $d_i\big(\mathrm{box}(\mathcal{R}^{\infty}_{\mathrm{abs}}(h))\big)$ as in the proof of Theorem 2 together with the fact that, by definition of q_i, the coefficients

$$\sum_{p+q_i^{(p)}=j} \frac{\|a_i^{(p)} + b_i^{(p)}(h)\|_1}{(p+1)!}$$

are zero for all $j < q_i$, we conclude that

$$\frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} = \frac{\sum_{j=q_i}^{\infty} h^j \sum_{p+q_i^{(p)}=j} \frac{1}{(p+1)!}\left[\delta^{j+1} - \varphi_i(h;\delta)\right]\left[(j+1)Q_i^{(p)}(\delta h) + h\dot{Q}_i^{(p)}(\delta h)\right]}{\sum_{j=q_i}^{\infty} h^{j+1} \sum_{p+q_i^{(p)}=j} \frac{1}{(p+1)!} Q_i^{(p)}(h)}. \tag{B.5}$$

We then expand the numerator for $j = q_i$, $j = q_i + 1$, and $j > q_i + 1$, and the denominator for $j = q_i$ and $j > q_i$:

$$\begin{aligned}\frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} = &\frac{h^{q_i}\left(\delta^{q_i+1} - \varphi_i(h;\delta)\right)\sum_{p+q_i^{(p)}=q_i} \frac{(q_i+1)Q_i^{(p)}(\delta h) + h\dot{Q}_i^{(p)}(\delta h)}{(p+1)!}}{h^{q_i+1}\sum_{p+q_i^{(p)}=q_i}\frac{Q_i^{(p)}(h)}{(p+1)!} + \mathcal{O}(h^{q_i+2})} \\ &+ \frac{h^{q_i+1}\left(\delta^{q_i+2} - \varphi_i(h;\delta)\right)\sum_{p+q_i^{(p)}=q_i+1} \frac{(q_i+2)Q_i^{(p)}(\delta h) + h\dot{Q}_i^{(p)}(\delta h)}{(p+1)!}}{h^{q_i+1}\sum_{p+q_i^{(p)}=q_i}\frac{Q_i^{(p)}(h)}{(p+1)!} + \mathcal{O}(h^{q_i+2})} \\ &+ \frac{\mathcal{O}(h^{q_i+3})}{h^{q_i+1}\sum_{p+q_i^{(p)}=q_i}\frac{Q_i^{(p)}(h)}{(p+1)!} + \mathcal{O}(h^{q_i+2})}.\end{aligned}$$

Taking advantage of the fact that $\forall i, p : Q_i^{(p)}(0) > 0$, passing to the limit $h \to 0$ yields

$$\lim_{h\to 0}\frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} = (q_i+1)\lim_{h\to 0}\frac{\delta^{q_i+1} - \varphi_i(h;\delta)}{h} + \delta^{q_i+1}(\delta-1)(q_i+2)\frac{\sum_{p+q_i^{(p)}=q_i+1}\frac{Q_i^{(p)}(0)}{(p+1)!}}{\sum_{p+q_i^{(p)}=q_i}\frac{Q_i^{(p)}(0)}{(p+1)!}},$$

where we used the fact that $\varphi_i(h;\delta) \to \delta^{q_i+1}$ for $h \to 0$, as shown in Theorem 2. Using the same tools as in the proof of Theorem 2, one can easily show that

$$\lim_{h\to 0} \frac{\delta^{q_i+1} - \varphi_i(h;\delta)}{h} = \delta^{q_i+1}(1-\delta)\frac{\sum_{p+q_i^{(p)}=q_i+1} \frac{Q_i^{(p)}(0)}{(p+1)!}}{\sum_{p+q_i^{(p)}=q_i} \frac{Q_i^{(p)}(0)}{(p+1)!}},$$

which implies that

$$\lim_{h\to 0} \frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} = -(1-\delta)\delta^{q_i+1}\frac{\sum_{p+q_i^{(p)}=q_i+1} \frac{Q_i^{(p)}(0)}{(p+1)!}}{\sum_{p+q_i^{(p)}=q_i} \frac{Q_i^{(p)}(0)}{(p+1)!}} \leq 0.$$

This shows (41).

Now, we prove the second statement of Proposition 2, i.e., the inequality (42). This requires an intermediate step: if the $Q_i^{(p)}$ are constant (i.e., $\dot{Q}_i^{(p)} = 0$), the following implication holds

$$\varphi_i(h;\delta) \geq \delta^{q_i+1} \quad \Rightarrow \quad \frac{\mathrm{d}\varphi_i(h;\delta)}{\mathrm{d}h} \leq 0. \tag{B.6}$$

Indeed, if $\delta^{q_i+1} \leq \varphi_i(h;\delta)$, it follows that $\delta^{q_i+1+\ell} \leq \varphi_i(h;\delta)$ for any $\ell \geq 0$, since $\delta \leq 1$. Additionally, $Q_i^{(p)} \geq 0$ by definition. Using these facts together with (B.5) and $\dot{Q}_i^{(p)} = 0$ yields the implication (B.6). We can now show the inequality (42), by using a proof by contradiction:

Assume, for the sake of contradiction, that there exists a time $t \in [0,h]$ such that $\varphi(t;\delta) > \delta^{q+1}$, and let $\mathcal{T}$ be the set of all those elements. Let $t_- := \inf_{t\in\mathcal{T}} t$ be the highest lower bound of $\mathcal{T}$. Since φ is continuous in t, the set $\mathcal{T}$ is open, and we can find some $t_+ \in \mathcal{T}$ such that $(t_-, t_+) \subseteq \mathcal{T}$. Since φ is strictly decreasing for any element of $\mathcal{T}$, it is also strictly decreasing over the interval (t_-, t_+). If $\varphi(t_-,\delta) \leq \delta^{q+1}$, since φ is decreasing we conclude that $\delta^{q+1} \leq \varphi(t_+;\delta)$, which contradicts our assumption on $t_+ \in \mathcal{T}$. Therefore, there must hold $\varphi(t_-,\delta) > \delta^{q+1}$. We can thus find an intermediary value $\delta^{q+1} < \varphi_* < \varphi(t_-,\delta)$.

On the other hand, as we have seen in Theorem 2, $\varphi(t;\delta) \to \delta^{q+1}$ and $\frac{d}{dt}\varphi(t;\delta) \leq 0$ for $t \to 0$, so that there always exists a small enough $t' \geq 0$ such that $\varphi(t';\delta) \leq \delta^{q+1}$ and $t' \leq t_-$. By the intermediary value theorem, there exists t_* such that $t' \leq t_* \leq t_-$ and $\varphi_* = \varphi(t_*,\delta)$. By assumption, $\varphi_* < \varphi(t_-,\delta)$, hence $t_* \neq t_-$, and since $\varphi(t_*,\delta) > \delta^{q+1}$ we also have $t_* \in \mathcal{T}$. However, this contradicts the definition of t_-, as it should be a lower bound of $\mathcal{T}$. We thus get a contradiction, proving that $\varphi(t;\delta) \leq \delta^{q+1}$ must hold for all $t \in [0,h]$. □

References

[1] T. Gan, et al., Reachability analysis for solvable dynamical systems, IEEE Trans. Automat. Control 63 (7) (2018) 2003–2018, http://dx.doi.org/10.1109/TAC.2017.2763785.

[2] J. Liu, et al., Computing semi-algebraic invariants for polynomial dynamical systems, in: Proc. of the 9th ACM International Conference on Embedded Software, 2011, pp. 97–106, http://dx.doi.org/10.1145/2038642.2038659.

[3] K. Ghorbal, A. Platzer, Characterizing algebraic invariants by differential radical invariants, in: Proc. of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2014, pp. 279–294, http://dx.doi.org/10.1007/978-3-642-54862-8_19.

[4] M. Boreale, Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial ODEs, Sci. Comput. Programm. 193 (2020) http://dx.doi.org/10.1016/j.scico.2020.102441.

[5] I. Mitchell, et al., A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games, IEEE Trans. Automat. Control 50 (7) (2005) 947–957, http://dx.doi.org/10.1109/TAC.2005.851439.

[6] S. Bansal, M. Chen, S. Herbert, C. Tomlin, Hamilton-Jacobi reachability: A brief overview and recent advances, in: Proc. of the 56th Annual Conference on Decision and Control, IEEE, 2017, pp. 2242–2253, http://dx.doi.org/10.1109/CDC.2017.8263977.

[7] P. Duggirala, S. Mitra, M. Viswanathan, Verification of annotated models from executions, in: Proc. of the International Conference on Embedded Software, IEEE, 2013, http://dx.doi.org/10.1109/EMSOFT.2013.6658604.

[8] J. Hoefkens, et al., Scientific computing, validated numerics, interval methods, in: W. Krämer, J.W. von Gudenberg (Eds.), Springer, 2001 pp. 281–292, http://dx.doi.org/10.1007/978-1-4757-6484-0_23, Ch. Verified high-order integration of DAEs and higher-order ODEs.

[9] K. Makino, M. Berz, Rigorous integration of flows and ODEs using Taylor models, in: Proc. of Symbolic-Numeric Computation, ACM, 2009 pp. 79–84, http://dx.doi.org/10.1145/1577190.1577206.

[10] X. Chen, et al., Taylor model flowpipe construction for non-linear hybrid systems, in: Proc. of the 33rd Real-Time Systems Symposium, IEEE, 2012, http://dx.doi.org/10.1109/RTSS.2012.70.

[11] E. Asarin, T. Dang, A. Girard, Reachability analysis of nonlinear systems using conservative approximation, in: 6th International Workshop on Hybrid Systems: Computation and Control, Springer, 2003, pp. 20–35, http://dx.doi.org/10.1007/3-540-36580-X_5.

[12] E. Asarin, et al., Hybridization methods for the analysis of nonlinear systems, Acta Inform. 43 (2007) 451–476, http://dx.doi.org/10.1007/s00236-006-0035-7.

[13] Z. Han, B. Krogh, Reachability analysis of nonlinear systems using trajectory piecewise linearized models, in: Proc. of the American Control Conference, IEEE, 2006, pp. 1505–1510, http://dx.doi.org/10.1109/ACC.2006.1656431.

[14] D. Li, S. Bak, S. Bogomolov, Reachability analysis of nonlinear systems using hybridization and dynamics scaling, in: International Conference on Formal Modeling and Analysis of Timed Systems, in: LNCS 12288, Springer, 2020, pp. 265–282, http://dx.doi.org/10.1007/978-3-030-57628-8_16.

[15] T. Dang, C. Le Guernic, O. Maler, Computing reachable states for nonlinear biological models, in: International Conference on Computational Methods in Systems Biology, Springer, 2009, pp. 126–141, http://dx.doi.org/10.1007/978-3-642-03845-7_9.

[16] T. Dang, et al., Accurate hybridization of nonlinear systems, in: Proc. of the 13th ACM International Conference on Hybrid Systems: Computation and Control, 2010, pp. 11–19, http://dx.doi.org/10.1145/1755952.1755956.
[17] M. Althoff, O. Stursberg, M. Buss, Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization, in: Proc. of the 47th IEEE Conference on Decision and Control, 2008, pp. 4042–4048, http://dx.doi.org/10.1109/CDC.2008.4738704.
[18] M. Althoff, Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets, in: Proc. of the 16th ACM International Conference on Hybrid Systems: Computation and Control, 2013, pp. 173–182, http://dx.doi.org/10.1145/2461328.2461358.
[19] L. Benvenuti, et al., Assume-guarantee verification of nonlinear hybrid systems with ARIADNE, Internat. J. Robust Nonlinear Control 24 (4) (2014) 699–724, http://dx.doi.org/10.1002/rnc.2914.
[20] P. Duggirala, et al., C2E2: A verification tool for stateflow models, in: Proc. of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2015, pp. 68–82, http://dx.doi.org/10.1007/978-3-662-46681-0_5.
[21] M. Althoff, An introduction to CORA 2015, in: Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems, 2015 pp. 120–151, http://dx.doi.org/10.29007/zbkv.
[22] J. Alexandre dit Sandretto, A. Chapoutot, DynIBEX: a differential constraint library for studying dynamical systems, 2016, Hal.Archives-Ouvertes.Fr: Hal-01297273.
[23] X. Chen, et al., Flow*: An analyzer for non-linear hybrid systems, in: Proc. of the 25th International Conference Computer-Aided Verification, in: LNCS 8044, Springer, 2013, pp. 258–263, http://dx.doi.org/10.1007/978-3-642-39799-8_18.
[24] F. Immler, Tool presentation: Isabelle/HOL for reachability analysis of continuous systems, in: Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems, 2015, pp. 180–187, http://dx.doi.org/10.29007/b3wr.
[25] S. Bogomolov, et al., JuliaReach: a toolbox for set-based reachability, in: Proc. of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, 2019, pp. 39–44, http://dx.doi.org/10.1145/3302504.3311804.
[26] A. Girard, Reachability of uncertain linear systems using zonotopes, in: 8th International Workshop on Hybrid Systems: Computation and Control, Springer, 2005, pp. 291–305, http://dx.doi.org/10.1007/978-3-540-31954-2_19.
[27] C. Combastel, A state bounding observer based on zonotopes, in: European Control Conference, ECC, IEEE, 2003, pp. 2589–2594, http://dx.doi.org/10.23919/ECC.2003.7085991.
[28] A.-K. Kopetzki, B. Schürmann, M. Althoff, Methods for order reduction of zonotopes, in: Proc. of the 56th IEEE Conference on Decision and Control, 2017, pp. 5626–5633, http://dx.doi.org/10.1109/CDC.2017.8264508.
[29] X. Yang, J.K. Scott, A comparison of zonotope order reduction techniques, Automatica 95 (2016) 378–384, http://dx.doi.org/10.1016/j.automatica.2018.06.006.
[30] M. Wetzlinger, A. Kulmburg, M. Althoff, Adaptive parameter tuning for reachability analysis of nonlinear systems, in: Proc. of the 24th ACM International Conference on Hybrid Systems: Computation and Control, 2021, http://dx.doi.org/10.1145/3447928.3456643.
[31] D. Griffiths, D. Higham, Numerical Methods for Ordinary Differential Equations: Initial Value Problems, Springer, 2010, http://dx.doi.org/10.1016/C2013-0-10643-5.
[32] E. Süli, D.F. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, 2003, http://dx.doi.org/10.1017/CBO9780511801181.
[33] M. Rungger, M. Zamani, Accurate reachability analysis of uncertain nonlinear systems, in: Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control, 2018, http://dx.doi.org/10.1145/3178126.3178127.
[34] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, John Wiley & Sons, 2016, http://dx.doi.org/10.1002/9781119121534.
[35] L. Lapidus, J.H. Seinfeld, Numerical Solution of Ordinary Differential Equations, Academic Press, 1971.
[36] U.M. Ascher, et al., Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM, 1994, http://dx.doi.org/10.1137/1.9781611971231.
[37] W. Rufeger, E. Adams, A step size control for Lohner's enclosure algorithm for ordinary differential equations with initial conditions, in: Mathematics in Science and Engineering, Vol. 189, Elsevier, 1993, pp. 283–299, http://dx.doi.org/10.1016/S0076-5392(08)62849-0.
[38] N. Nedialkov, Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation (Dissertation), University of Toronto, 2000.
[39] P. Prabhakar, M. Viswanathan, A dynamic algorithm for approximate flow computations, in: Proc. of the 14th ACM International Conference on Hybrid Systems: Computation and Control, 2011, pp. 133–142, http://dx.doi.org/10.1145/1967701.1967722.
[40] G. Frehse, et al., SpaceEx: scalable verification of hybrid systems, in: Proc. of the 23rd International Conference on Computer Aided Verification, in: LNCS 6806, Springer, 2011, pp. 379–395, http://dx.doi.org/10.1007/978-3-642-22110-1_30.
[41] G. Frehse, R. Kateja, C. Le Guernic, Flowpipe approximation and clustering in space-time, in: Proc. of the 16th ACM International Conference on Hybrid Systems: Computation and Control, 2013, pp. 203–212, http://dx.doi.org/10.1145/2461328.2461361.
[42] M. Wetzlinger, N. Kochdumper, M. Althoff, Adaptive parameter tuning for reachability analysis of linear systems, in: Proc. of the 59th IEEE Conference on Decision and Control, 2020, pp. 5145–5152, http://dx.doi.org/10.1109/CDC42340.2020.9304431.
[43] X. Chen, Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models (Dissertation), RWTH Aachen University, 2015.
[44] S. Bak, et al., High-level hybrid systems analysis with Hypy, in: Proc. of the Workshop on Applied Verification of Continuous and Hybrid Systems, 2016, pp. 80–90, http://dx.doi.org/10.29007/4f3d.
[45] G. Alefeld, G. Mayer, Interval analysis: Theory and applications, Comput. Appl. Math. 121 (1–2) (2000) 421–464, http://dx.doi.org/10.1016/S0377-0427(00)00342-3.
[46] N. Kochdumper, M. Althoff, Sparse polynomial zonotopes: a novel set representation for reachability analysis, IEEE Trans. Automat. Control 66 (2) (2021) 4043–4058, http://dx.doi.org/10.1109/TAC.2020.3024348.
[47] M. Althoff, B.H. Krogh, Reachability analysis of nonlinear differential-algebraic systems, IEEE Trans. Automat. Control 59 (2) (2014) 371–383, http://dx.doi.org/10.1109/TAC.2013.2285751.
[48] M. Berz, G. Hoffstätter, Computation and application of Taylor polynomials with interval remainder bounds, Reliab. Comput. 4 (1998) 83–97, http://dx.doi.org/10.1023/A:1009958918582.
[49] M. Althoff, Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars (Dissertation), Technische Universität München, 2010.
[50] M. Althoff, C. Le Guernic, B.H. Krogh, Reachable set computation for uncertain time-varying linear systems, in: Proc. of the 14th ACM International Conference on Hybrid Systems: Computation and Control, 2011, pp. 93–102, http://dx.doi.org/10.1145/1967701.1967717.
[51] V.V. Shenmaier, Complexity and approximation of finding the longest vector sum, Comput. Math. Math. Phys. 58 (6) (2018) 850–857, http://dx.doi.org/10.1134/S0965542518060131.
[52] A.E. Baburin, A.V. Pyatkin, Polynomial algorithms for solving the vector sum problem, J. Appl. Ind. Math. 1 (3) (2007) 268–272, http://dx.doi.org/10.1134/S1990478907030027.
[53] T. Alamo, J. Bravo, E. Camacho, Guaranteed state estimation by zonotopes, Automatica 41 (2005) 1035–1043, http://dx.doi.org/10.1016/j.automatica.2004.12.008.
[54] L. Geretti, et al., ARCH-COMP20 category report: continuous and hybrid systems with nonlinear dynamics, in: ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems, EasyChair, 2020, pp. 49–75, http://dx.doi.org/10.29007/zkf6.

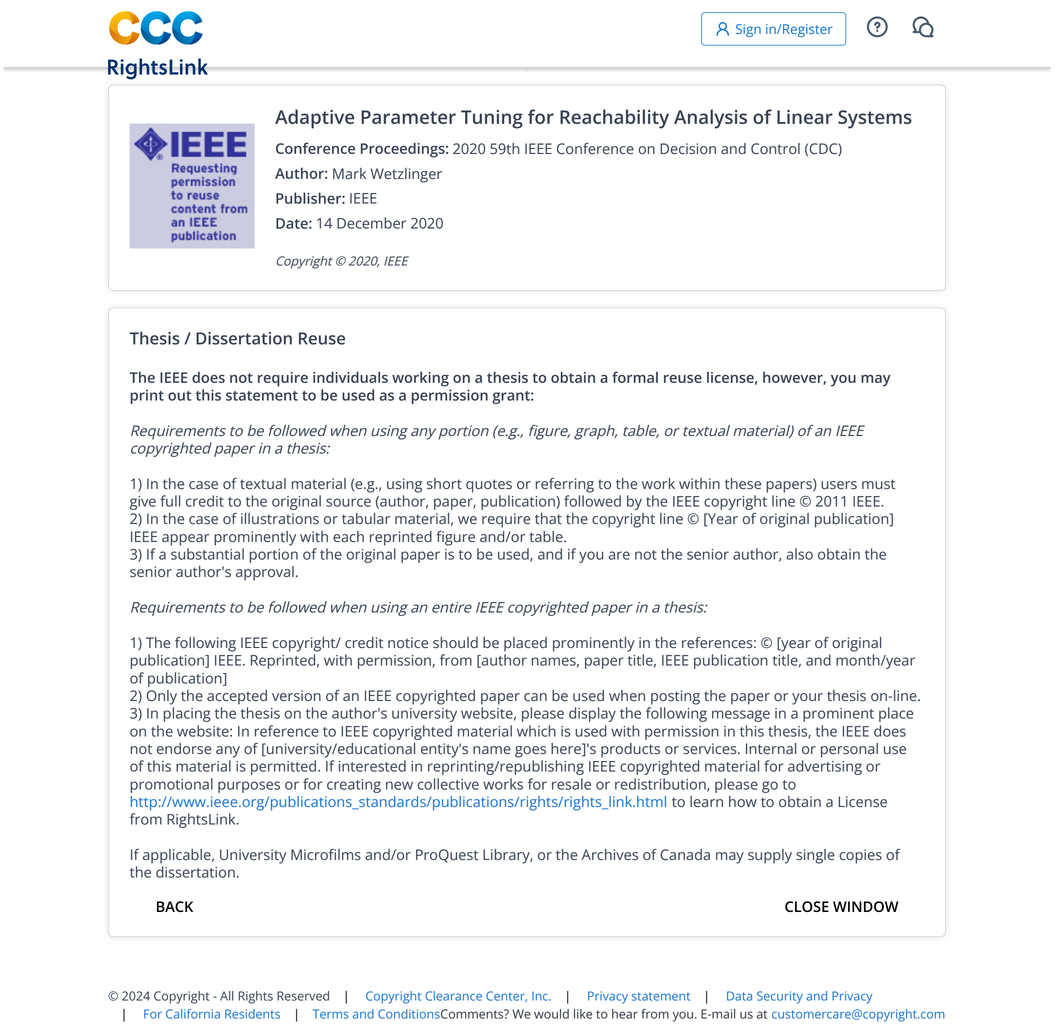
[55] L. Geretti, et al., ARCH-COMP21 category report: continuous and hybrid systems with nonlinear dynamics, in: G. Frehse, M. Althoff (Eds.), Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems, 2021, pp. 32–54, http://dx.doi.org/10.29007/2jw8.
[56] X. Chen, S. Sankaranarayanan, Decomposed reachability analysis for nonlinear systems, in: Proc. of the 37th Real-Time Systems Symposium, IEEE, 2016, pp. 13–24, http://dx.doi.org/10.1109/RTSS.2016.011.
[57] A. El-Guindy, Control and Stability of Power Systems using Reachability Analysis (Dissertation), Technische Universität München, 2017.
[58] S. Kopecz, A. Meister, On order conditions for modified Patankar–Runge–Kutta schemes, Appl. Numer. Math. 123 (2018) 159–179, http://dx.doi.org/10.1016/j.apnum.2017.09.004.
[59] M. Laub, W. Loomis, A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium, Mol. Biol. Cell 9 (12) (1998) 3521–3532, http://dx.doi.org/10.1091/mbc.9.12.3521.
[60] E. Aylward, P. Parrilo, J.-J. Slotine, Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming, Automatica 44 (8) (2008) 2163–2170, http://dx.doi.org/10.1016/j.automatica.2007.12.012.
[61] O. Rössler, An equation for continuous chaos, Phys. Lett. A 57 (5) (1976) 397–398, http://dx.doi.org/10.1016/0375-9601(76)90101-8.
[62] E. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141, http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.
[63] J. Vano, et al., Chaos in low-dimensional Lotka–Volterra models of competition, Nonlinearity 19 (10) (2006) 2391, http://dx.doi.org/10.1088/0951-7715/19/10/006.
[64] E. Klipp, et al., Systems Biology in Practice: Concepts, Implementation and Application, John Wiley & Sons, 2005, http://dx.doi.org/10.1002/3527603603.
[65] J. Vilar, et al., Mechanisms of noise-resistance in genetic oscillators, Proc. Natl. Acad. Sci. 99 (9) (2002) 5988–5992, http://dx.doi.org/10.1073/pnas.092133899.
[66] Y. Chen, A. Domínguez-García, Assessing the impact of wind variability on power system small-signal reachability, in: Proc. of the 44th Hawaii International Conference on System Sciences, IEEE, 2011, http://dx.doi.org/10.1109/HICSS.2011.77.
[67] X. Chen, S. Sankaranarayanan, E. Ábrahám, Under-approximate flowpipes for non-linear continuous systems, in: Formal Methods in Computer-Aided Design, FMCAD, IEEE, 2014, pp. 59–66, http://dx.doi.org/10.1109/FMCAD.2014.6987596.
[68] H. Bauer, Minimalstellen von Funktionen und Extremalpunkte, Archiv der Mathematik 9 (1958) 389–393, http://dx.doi.org/10.1007/BF01898615.
[69] G. Golub, M. Saunders, Linear least squares and quadratic programming, Integer Nonlinear Programm. (1969).

B Licenses

This chapter contains all explicit licenses for the publications reprinted in Appendix A, as required by the TUM Graduate School.

License for Appendix A.1

Rightslink® by Copyright Clearance Center https://s100.copyright.com/AppDispatchServlet#formTop



Sign in/Register

Adaptive Parameter Tuning for Reachability Analysis of Linear Systems

Conference Proceedings: 2020 59th IEEE Conference on Decision and Control (CDC)

Author: Mark Wetzlinger

Publisher: IEEE

Date: 14 December 2020

Copyright © 2020, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

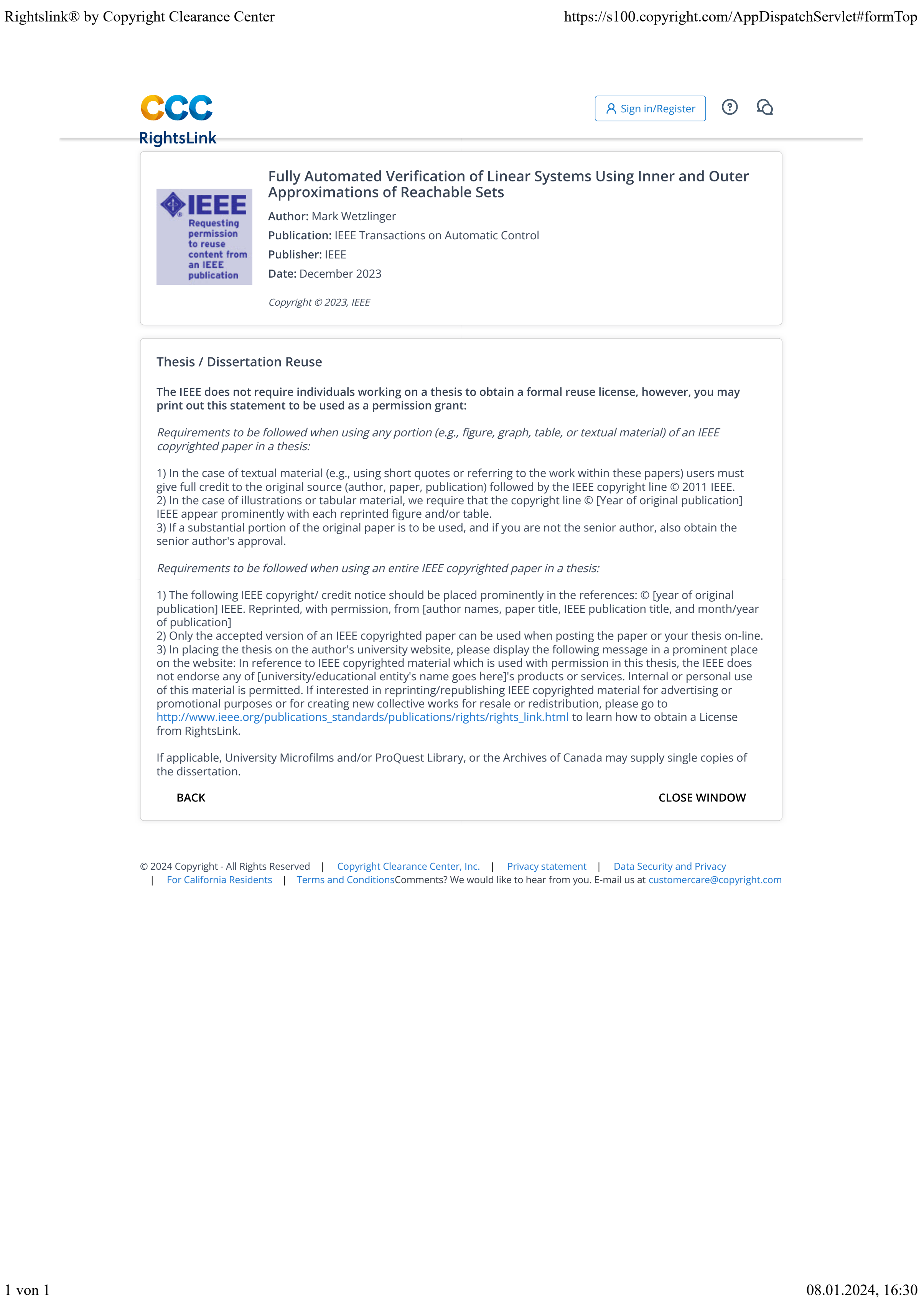
1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

1 von 1 08.01.2024, 16:31

License for Appendix A.2



Rightslink® by Copyright Clearance Center https://s100.copyright.com/AppDispatchServlet#formTop

RightsLink
Sign in/Register
Requesting permission to reuse content from an IEEE publication

Fully Automated Verification of Linear Systems Using Inner and Outer Approximations of Reachable Sets

Author: Mark Wetzlinger

Publication: IEEE Transactions on Automatic Control

Publisher: IEEE

Date: December 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

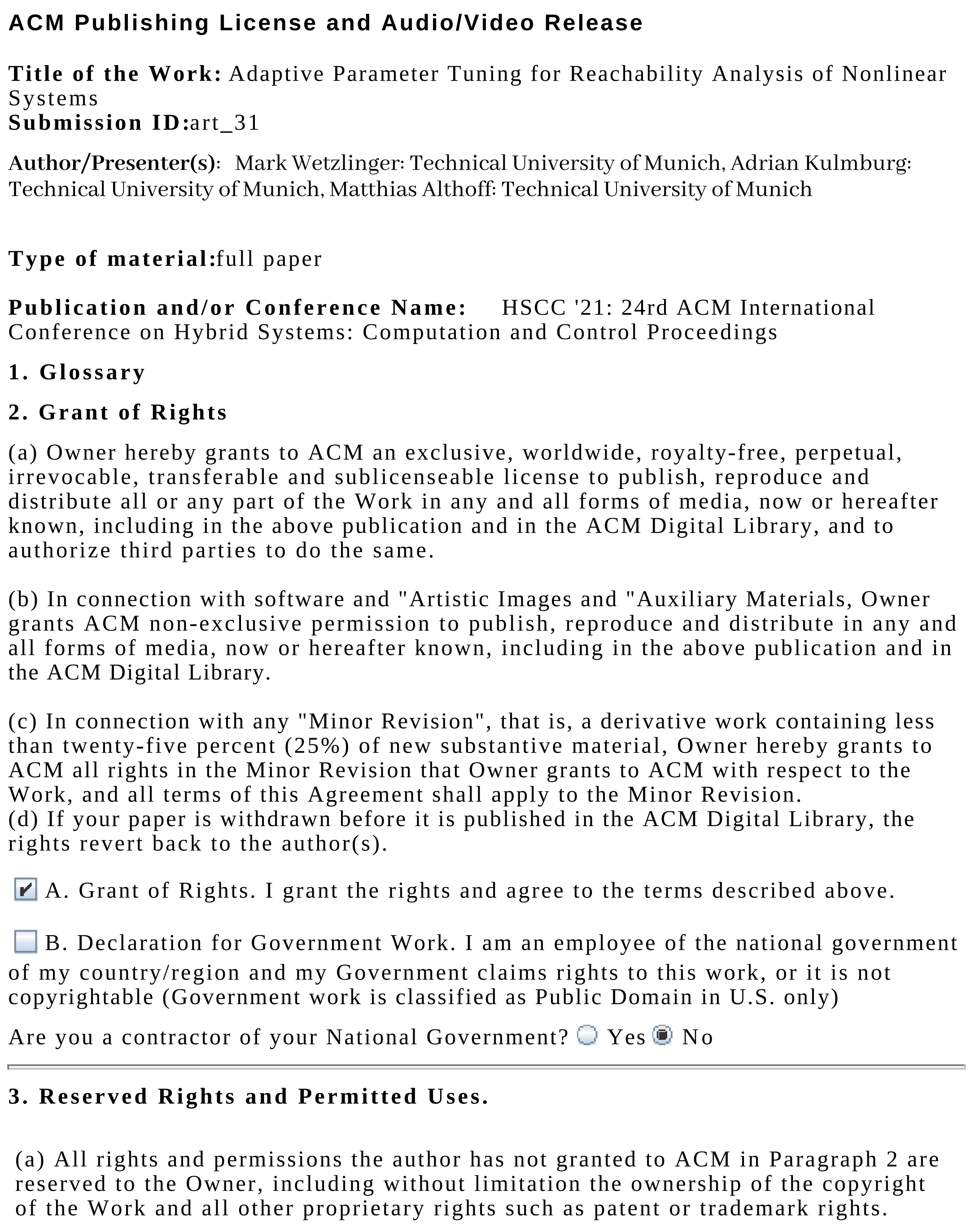
Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

1 von 1 08.01.2024, 16:30

License for Appendix A.3



ACM Publishing License and Audio/Video Release

Title of the Work: Adaptive Parameter Tuning for Reachability Analysis of Nonlinear Systems
Submission ID:art_31

Author/Presenter(s): Mark Wetzlinger: Technical University of Munich, Adrian Kulmburg: Technical University of Munich, Matthias Althoff: Technical University of Munich

Type of material:full paper

Publication and/or Conference Name: HSCC '21: 24rd ACM International Conference on Hybrid Systems: Computation and Control Proceedings

1. Glossary

2. Grant of Rights

(a) Owner hereby grants to ACM an exclusive, worldwide, royalty-free, perpetual, irrevocable, transferable and sublicenseable license to publish, reproduce and distribute all or any part of the Work in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library, and to authorize third parties to do the same.

(b) In connection with software and "Artistic Images and "Auxiliary Materials, Owner grants ACM non-exclusive permission to publish, reproduce and distribute in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library.

(c) In connection with any "Minor Revision", that is, a derivative work containing less than twenty-five percent (25%) of new substantive material, Owner hereby grants to ACM all rights in the Minor Revision that Owner grants to ACM with respect to the Work, and all terms of this Agreement shall apply to the Minor Revision.
(d) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s).

[✓] A. Grant of Rights. I grant the rights and agree to the terms described above.

☐ B. Declaration for Government Work. I am an employee of the national government of my country/region and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

Are you a contractor of your National Government? Yes [✓ No]

---

3. Reserved Rights and Permitted Uses.

(a) All rights and permissions the author has not granted to ACM in Paragraph 2 are reserved to the Owner, including without limitation the ownership of the copyright of the Work and all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM in Paragraph 2(a), Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work.

When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page.

The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

Please put the following LaTeX commands in the preamble of your document - i.e., before \begin{document}:

```
\copyrightyear{2021}
\acmYear{2021}
\setcopyright{acmlicensed}\acmConference[HSCC '21]{24th ACM International
Conference on Hybrid Systems: Computation and Control}{May 19--21,
2021}{Nashville, TN, USA}
\acmBooktitle{24th ACM International Conference on Hybrid Systems:
Computation and Control (HSCC '21), May 19--21, 2021, Nashville, TN, USA}
\acmPrice{15.00}
\acmDOI{10.1145/3447928.3456643}
\acmISBN{978-1-4503-8339-4/21/05}
```

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

HSCC '21, May 19–21, 2021, Nashville, TN, USA
© 2021 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-8339-4/21/05...$15.00
https://doi.org/10.1145/3447928.3456643

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing.

---

4. ACM Citation and Digital Object Identifier.

(a) In connection with any use by the Owner of the Definitive Version, Owner shall include the ACM citation and ACM Digital Object Identifier (DOI).
(b) In connection with any use by the Owner of the Submitted Version (if accepted) or the Accepted Version or a Minor Revision, Owner shall use best efforts to display the ACM citation, along with a statement substantially similar to the following:

> "© [Owner] [Year]. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in {Source Publication}, https://doi.org/10.1145/{number}."

5. Audio/Video Recording

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? [✓ Yes] No

6. Auxiliary Material

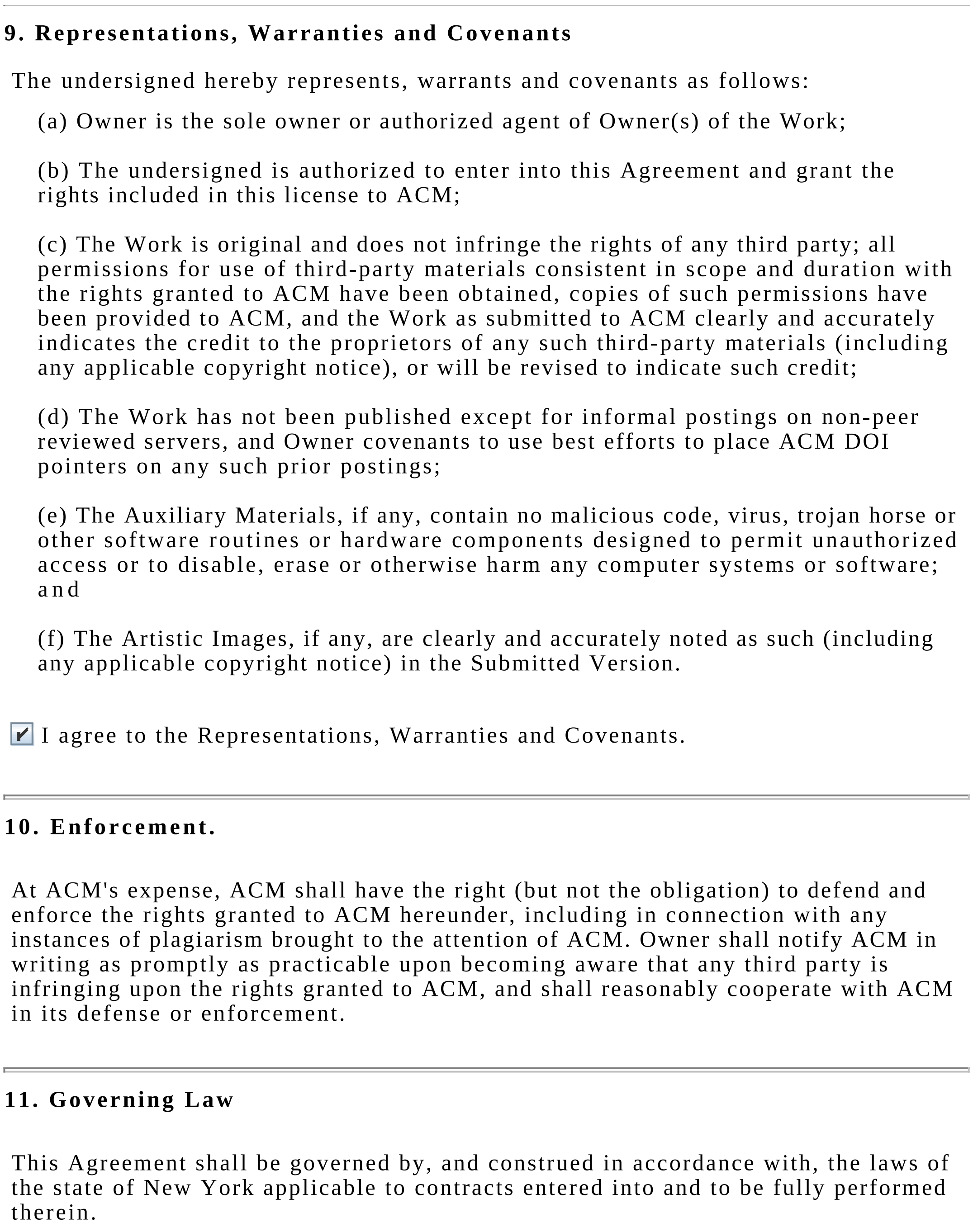
Do you have any Auxiliary Materials? Yes [✓ No]

7. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

[✓ We/I have not used third-party material.]
We/I have used third-party materials and have necessary permissions.

8. Artistic Images
If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part IV and be sure to include a notice of copyright with each such image in the paper.
[✓ We/I do not have any artistic images.]
We/I have any artistic images.



9. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

✓ I agree to the Representations, Warranties and Covenants.

10. Enforcement.

At ACM's expense, ACM shall have the right (but not the obligation) to defend and enforce the rights granted to ACM hereunder, including in connection with any instances of plagiarism brought to the attention of ACM. Owner shall notify ACM in writing as promptly as practicable upon becoming aware that any third party is infringing upon the rights granted to ACM, and shall reasonably cooperate with ACM in its defense or enforcement.

11. Governing Law

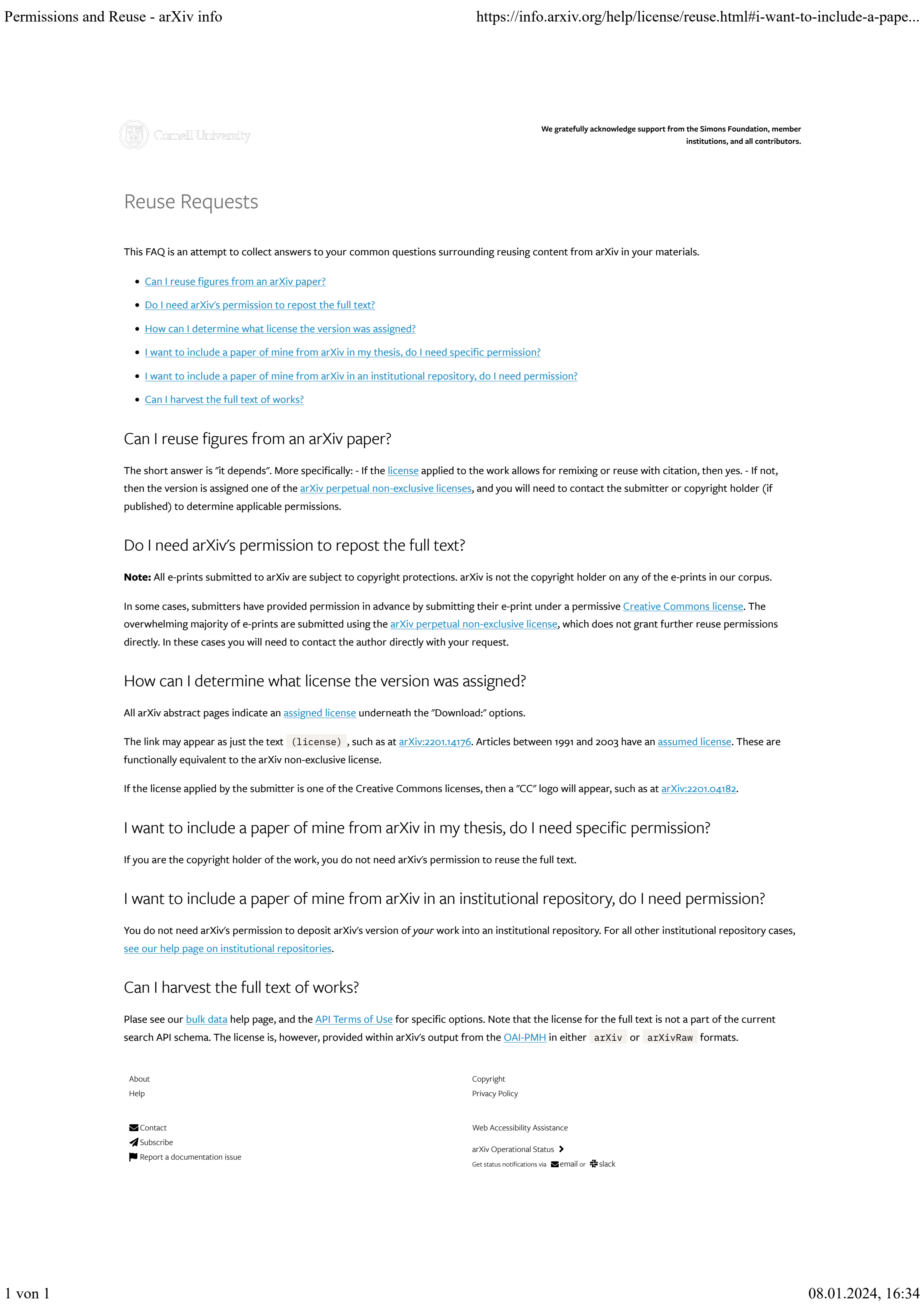
This Agreement shall be governed by, and construed in accordance with, the laws of the state of New York applicable to contracts entered into and to be fully performed therein.

Funding Agents

1. Deutsche Forschungsgemeinschaft award number(s):GRK 2428

2. European Research Council award number(s):817629

License for Appendix A.4



Permissions and Reuse - arXiv info https://info.arxiv.org/help/license/reuse.html#i-want-to-include-a-pape...

Cornell University

We gratefully acknowledge support from the Simons Foundation, member institutions, and all contributors.

Reuse Requests

This FAQ is an attempt to collect answers to your common questions surrounding reusing content from arXiv in your materials.

- Can I reuse figures from an arXiv paper?
- Do I need arXiv's permission to repost the full text?
- How can I determine what license the version was assigned?
- I want to include a paper of mine from arXiv in my thesis, do I need specific permission?
- I want to include a paper of mine from arXiv in an institutional repository, do I need permission?
- Can I harvest the full text of works?

Can I reuse figures from an arXiv paper?

The short answer is "it depends". More specifically: - If the license applied to the work allows for remixing or reuse with citation, then yes. - If not, then the version is assigned one of the arXiv perpetual non-exclusive licenses, and you will need to contact the submitter or copyright holder (if published) to determine applicable permissions.

Do I need arXiv's permission to repost the full text?

Note: All e-prints submitted to arXiv are subject to copyright protections. arXiv is not the copyright holder on any of the e-prints in our corpus.

In some cases, submitters have provided permission in advance by submitting their e-print under a permissive Creative Commons license. The overwhelming majority of e-prints are submitted using the arXiv perpetual non-exclusive license, which does not grant further reuse permissions directly. In these cases you will need to contact the author directly with your request.

How can I determine what license the version was assigned?

All arXiv abstract pages indicate an assigned license underneath the "Download:" options.

The link may appear as just the text `(license)`, such as at arXiv:2201.14176. Articles between 1991 and 2003 have an assumed license. These are functionally equivalent to the arXiv non-exclusive license.

If the license applied by the submitter is one of the Creative Commons licenses, then a "CC" logo will appear, such as at arXiv:2201.04182.

I want to include a paper of mine from arXiv in my thesis, do I need specific permission?

If you are the copyright holder of the work, you do not need arXiv's permission to reuse the full text.

I want to include a paper of mine from arXiv in an institutional repository, do I need permission?

You do not need arXiv's permission to deposit arXiv's version of *your* work into an institutional repository. For all other institutional repository cases, see our help page on institutional repositories.

Can I harvest the full text of works?

Plase see our bulk data help page, and the API Terms of Use for specific options. Note that the license for the full text is not a part of the current search API schema. The license is, however, provided within arXiv's output from the OAI-PMH in either `arXiv` or `arXivRaw` formats.

About
Help

Contact
Subscribe
Report a documentation issue

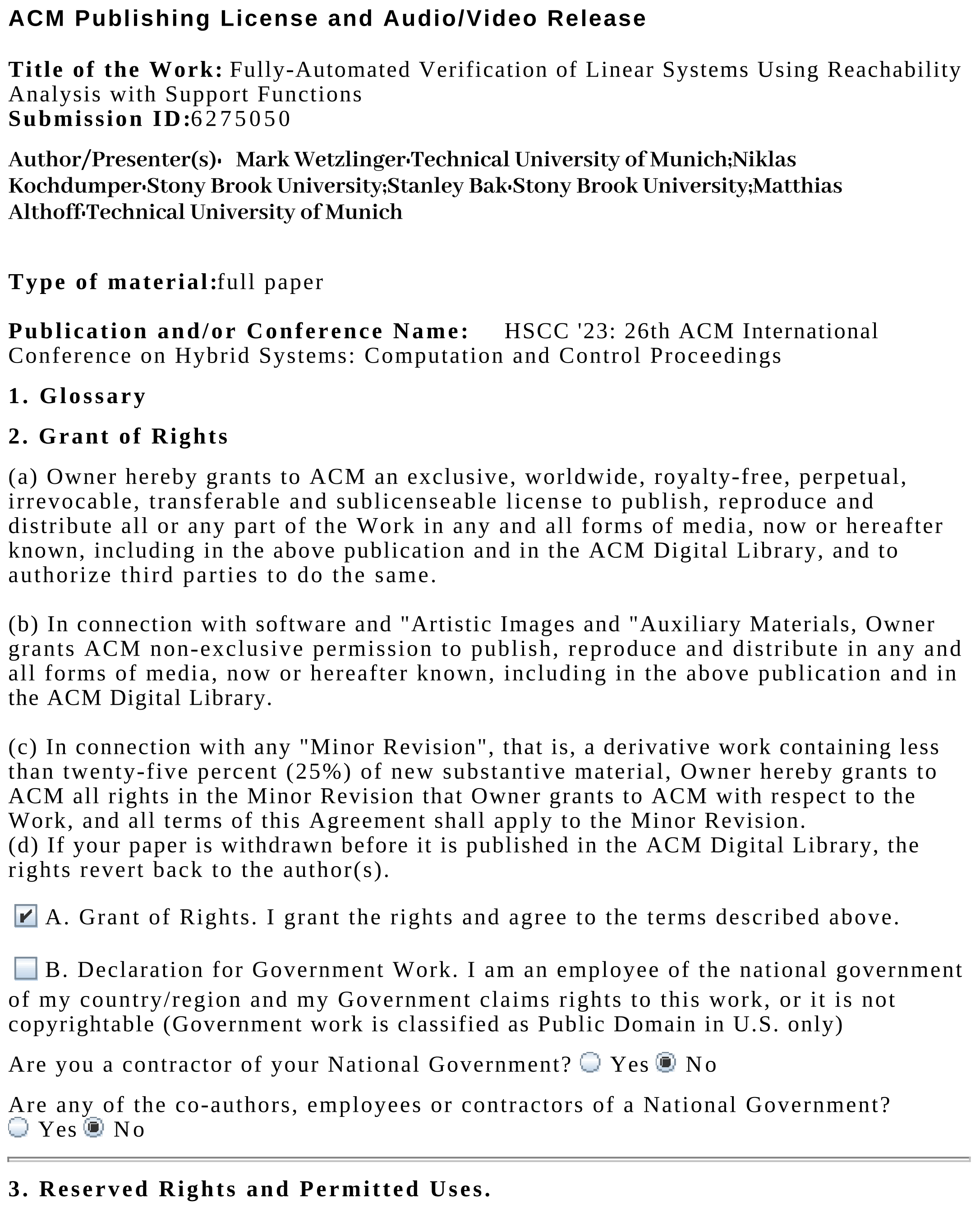
Copyright
Privacy Policy

Web Accessibility Assistance

arXiv Operational Status
Get status notifications via email or slack

1 von 1 08.01.2024, 16:34

License for Appendix A.5



ACM Publishing License and Audio/Video Release

Title of the Work: Fully-Automated Verification of Linear Systems Using Reachability Analysis with Support Functions
Submission ID:6275050

Author/Presenter(s)· Mark Wetzlinger·Technical University of Munich;Niklas Kochdumper·Stony Brook University;Stanley Bak·Stony Brook University;Matthias Althoff·Technical University of Munich

Type of material:full paper

Publication and/or Conference Name: HSCC '23: 26th ACM International Conference on Hybrid Systems: Computation and Control Proceedings

evaluation.)

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

Please put the following LaTeX commands in the preamble of your document - i.e., before \begin{document}:

```
\copyrightyear{2023}
\acmYear{2023}
\setcopyright{acmlicensed}\acmConference[HSCC '23]{Proceedings of the 26th
ACM International Conference on Hybrid Systems: Computation and
Control}{May 9--12, 2023}{San Antonio, TX, USA}
\acmBooktitle{Proceedings of the 26th ACM International Conference on Hybrid
Systems: Computation and Control (HSCC '23), May 9--12, 2023, San Antonio,
TX, USA}
\acmPrice{15.00}
\acmDOI{10.1145/3575870.3587121}
\acmISBN{979-8-4007-0033-0/23/05}
```

NOTE: For authors using the ACM Microsoft Word Master Article Template and Publication Workflow, The ACM Publishing System (TAPS) will add the rights statement to your papers for you. Please check with your conference contact for information regarding submitting your source file(s) for processing.

If you are using the ACM Interim Microsoft Word template, or still using or older versions of the ACM SIGCHI template, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing.

---

4. ACM Citation and Digital Object Identifier.

(a) In connection with any use by the Owner of the Definitive Version, Owner shall include the ACM citation and ACM Digital Object Identifier (DOI).
(b) In connection with any use by the Owner of the Submitted Version (if accepted) or the Accepted Version or a Minor Revision, Owner shall use best efforts to display the ACM citation, along with a statement substantially similar to the following:

> "© [Owner] [Year]. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in {Source Publication}, https://doi.org/10.1145/{number}."

5. Livestreaming and Distribution

You are giving a presentation at the annual conference. This section of the rights form gives you the opportunity to grant or deny ACM the ability to make this presentation more widely seen, through (a) livestreaming of the presentation during the conference and/or (b) distributing the presentation after the conference in the ACM Digital Library, the "Conference Presentations" USB, and media outlets such as Vimeo and YouTube. It also provides you the opportunity to grant or deny our use of the presentation in promotional and marketing efforts after the conference.

Not all conference presentations are livestreamed; you will be notified in advance of the possibility of your presentation being livestreamed.

The permissions granted and/or denied here apply to all presentations of this material at the conference, including (but not limited to) the primary presentation and any program-specific "fast forward" presentations.

ACM's policy on the use of third-party material applies to your presentation as well as the documentation of your work; if you are using others' material in your presentation, including audio, you must identify that material on the ACM rights form and in the presentation where it is used, and secure permission to use the material where necessary.

Livestreaming.
I grant permission to ACM to livestream my presentation during the conference (a "livestream" is a synchronous distribution of the presentation to the public, separate from the presentation distributed to conference registrants).
[✓ Yes]
No

Post-Conference Distribution.
I grant permission to ACM to distribute the recording of my presentation after the conference as listed above.
[✓ Yes]
No

6. Auxiliary Material

Do you have any Auxiliary Materials? Yes [✓ No]

7. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

[✓ We/I have not used third-party material.]
We/I have used third-party materials and have necessary permissions.

8. Artistic Images
If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part IV and be sure to include a notice of copyright with each such image in the paper.
[✓ We/I do not have any artistic images.]
We/I have any artistic images.

---

9. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

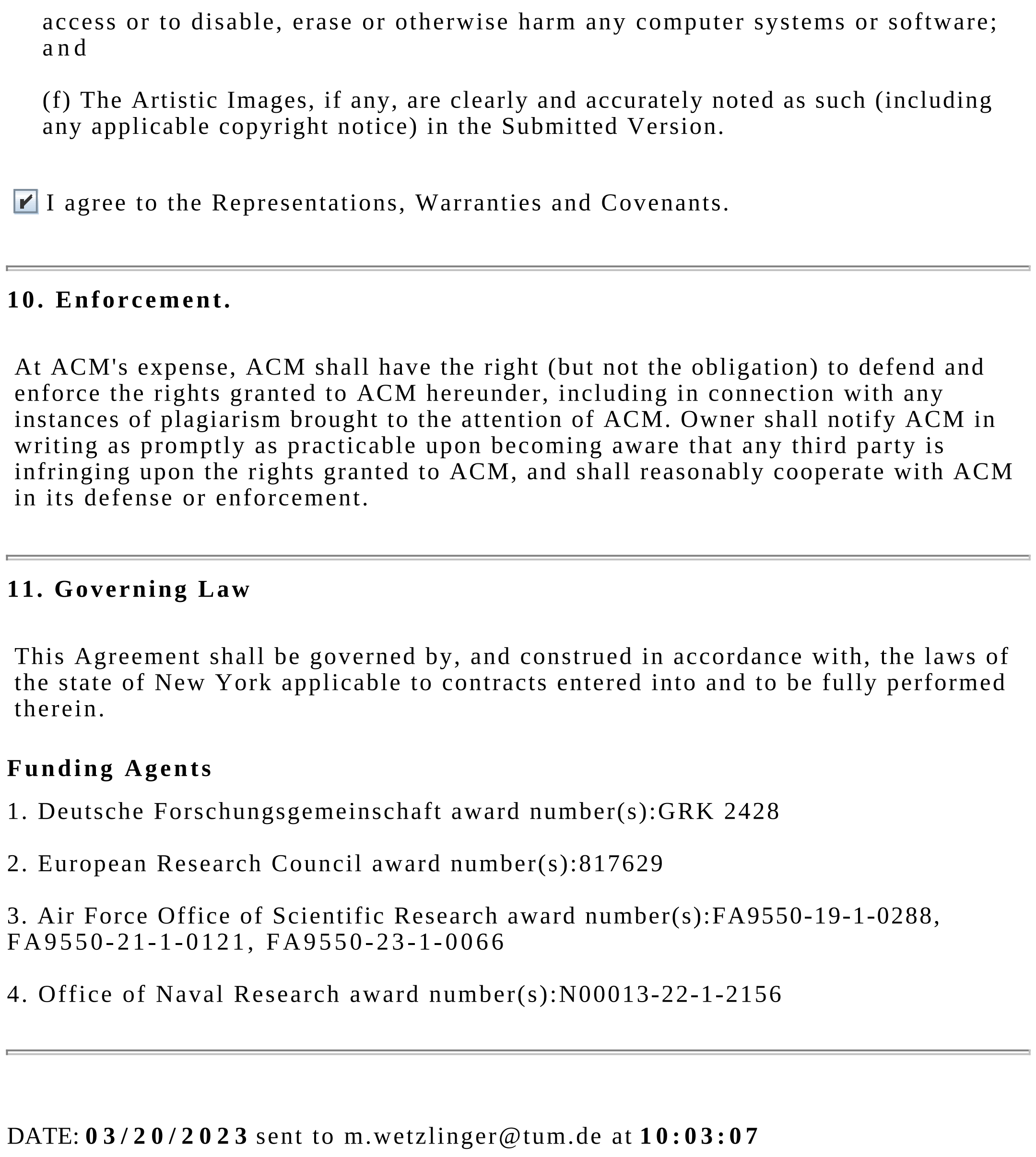
(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized



access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

✓ I agree to the Representations, Warranties and Covenants.

10. Enforcement.

At ACM's expense, ACM shall have the right (but not the obligation) to defend and enforce the rights granted to ACM hereunder, including in connection with any instances of plagiarism brought to the attention of ACM. Owner shall notify ACM in writing as promptly as practicable upon becoming aware that any third party is infringing upon the rights granted to ACM, and shall reasonably cooperate with ACM in its defense or enforcement.

11. Governing Law

This Agreement shall be governed by, and construed in accordance with, the laws of the state of New York applicable to contracts entered into and to be fully performed therein.

Funding Agents

1. Deutsche Forschungsgemeinschaft award number(s):GRK 2428

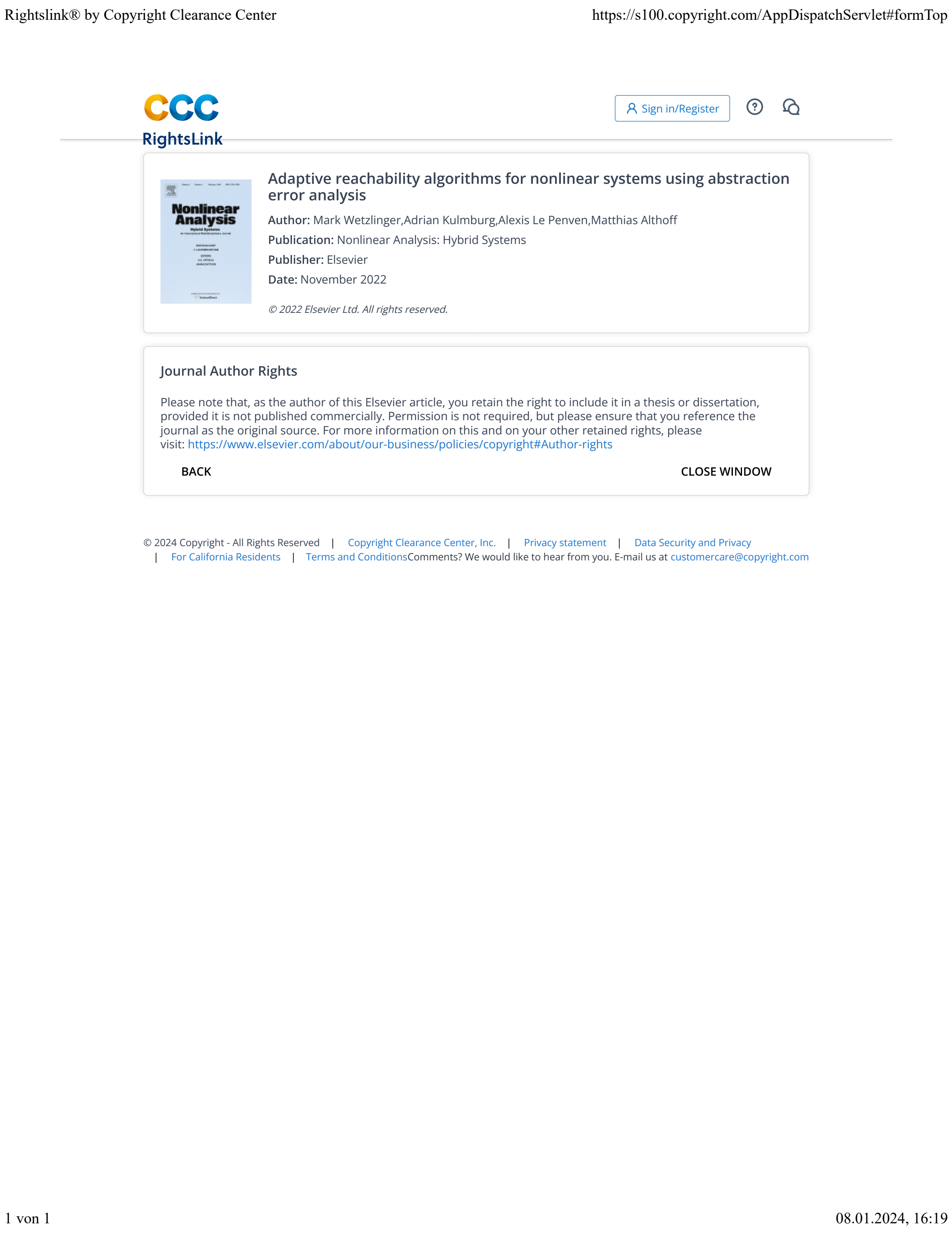
2. European Research Council award number(s):817629

3. Air Force Office of Scientific Research award number(s):FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066

4. Office of Naval Research award number(s):N00013-22-1-2156

DATE: **03/20/2023** sent to m.wetzlinger@tum.de at **10:03:07**

License for Appendix A.6



Rightslink® by Copyright Clearance Center https://s100.copyright.com/AppDispatchServlet#formTop

Adaptive reachability algorithms for nonlinear systems using abstraction error analysis

Author: Mark Wetzlinger,Adrian Kulmburg,Alexis Le Penven,Matthias Althoff

Publication: Nonlinear Analysis: Hybrid Systems

Publisher: Elsevier

Date: November 2022

© 2022 Elsevier Ltd. All rights reserved.

Journal Author Rights

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

1 von 1 08.01.2024, 16:19