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 size 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

HSCC '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8339-4/21/05…$15.00
https://doi.org/10.1145/3447928.3456643

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 $\mathcal{R}_{\text{ex}}([0, t_K])$ of (1) over the time horizon $t \in [0, t_K]$ is defined as

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

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

$$\dot{x}_i \in \sum_{\nu=0}^{\kappa} \left. \frac{\left((z(t) - z^*)^T \nabla\right)^\nu f_i(\hat{z})}{\nu!} \right|_{\hat{z}=z^*} \oplus \mathcal{L}_i(t) , \tag{2}$$

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),

$$w_i = f_i(z^*),\ C_{ij} = \left. \frac{\partial f_i(\hat{z})}{\partial \hat{z}_j} \right|_{\hat{z}=z^*},\ D_{ijk} = \left. \frac{\partial^2 f_i(\hat{z})}{\partial \hat{z}_j \partial \hat{z}_k} \right|_{\hat{z}=z^*}, \ \ldots \tag{4}$$

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

$$\dot{x}(t) \in \underbrace{Ax(t) + Bu(t) + w}_{f_{\text{lin}}(t)} \oplus \Psi , \tag{5}$$

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 $\mathcal{R}_{\text{lin}}$ of the linearized dynamics (Lines 4-5). Second, the set of abstraction errors $\mathcal{R}_{\text{abs}}$ based on the abstraction error Ψ (Lines 6-11).

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

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: $z^*(t_s) \leftarrow \texttt{center}\big(\mathcal{R}(t_s)\big)$
4: $w, A, B, D, \ldots \leftarrow \texttt{taylor}\big(f(z), z^*(t_s)\big)$
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: $\mathcal{R}(t_{s+1}) \leftarrow \texttt{red}\big(\mathcal{R}(t_{s+1})\big), \mathcal{R}(\tau_{s+1}) \leftarrow \texttt{red}\big(\mathcal{R}(\tau_{s+1})\big)$
15: $t_{s+1} := t_s + \Delta t_s,\ s := s + 1$
16: **end while**
17: $\mathcal{R}([0, t_K]) = \bigcup_{j=0}^{K-1} \mathcal{R}(\tau_j)$

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.]

$$\mathcal{R}_{\text{abs}} = \bigoplus_{k=0}^{\eta_{\text{abs}}} \frac{\Delta t^{k+1}}{(k+1)!} A^k \Psi \oplus \mathrm{E}(\Delta t, \eta_{\text{abs}})\, \Delta t\, \Psi\,, \tag{6}$$

where $\mathrm{E} = \mathcal{O}(\Delta t^{\eta_{\text{abs}}+1})$ tends to 0 as $\Delta t \to 0$, 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.

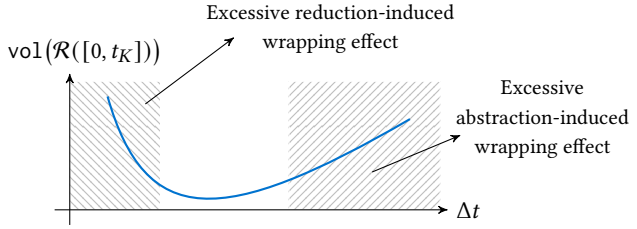


Figure 1: If the time step size Δt is too large or too small, the abstraction-induced or reduction-induced wrapping effect, respectively, are dominating.

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

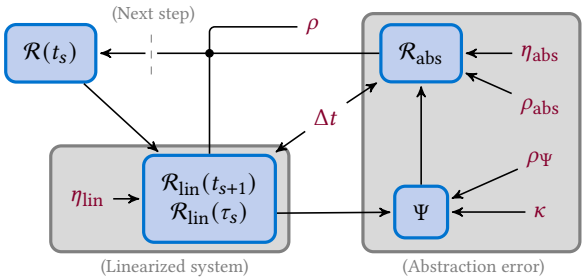


Figure 2: Main workflow for one time step in Alg. 1 and the influence of algorithm parameters (red) on different sets: The time step size Δt affects both the linearized system and the abstraction error, while the abstraction order κ only influences the abstraction error. The propagation parameters η 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 $\mathcal{R}_{\text{abs}}$ in (6) and $\mathcal{R}_{\text{lin}}$ 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 $\mathcal{R}_{\text{lin}}(\tau_s)$ is dominated by the term $\mathbf{F}_x\mathcal{R}(t_s)$, which can be computed according to [2, Prop. 3.1] as

$$\mathbf{F}_x = \bigoplus_{\ell=1}^{\eta_{\text{lin}}} \underbrace{\left[(\ell^{\frac{-\ell}{\ell-1}} - \ell^{\frac{-1}{\ell-1}})\Delta t^\ell, 0\right]\frac{A^\ell}{\ell!}}_{:=T^{(\ell)}} \oplus \mathbf{E}, \tag{9}$$

with $\mathbf{E}$ as referenced above, we obtain η_{lin} by setting a threshold $0 < \zeta_{T,\text{lin}} \ll 1$ for the change over successive terms $T^{(\ell)}$:

$$\eta_{\text{lin}} = \min \ell \quad \text{such that} \quad 1 - \frac{\|T^{(\ell-1)}\|_F}{\|T^{(\ell)}\|_F} \leq \zeta_{T,\text{lin}}\,. \tag{10}$$

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 $0 < \zeta_{T,\text{abs}} \ll 1$:

$$\eta_{\text{abs}} = \min \ell \quad \text{such that} \quad \max_{i\in\{1,\dots,n\}} \frac{d_i\left(\text{box}\left(\mathcal{R}_{\text{abs}}^{(\ell+1)}\right)\right)}{d_i\left(\text{box}\left(\mathcal{R}_{\text{abs}}^{(\ell)}\right)\right)} \leq \zeta_{T,\text{abs}}\,, \tag{11}$$

where $\mathcal{R}_{\text{abs}}^{(k)}$ denotes the sum in (6) truncated at order k. The split into two different η for $\mathcal{R}_{\text{lin}}$ and $\mathcal{R}_{\text{abs}}$ 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

$$\max_{x\in C_1} \min_{y\in C_2} f(x,y) = \min_{y\in C_2} f(x^*,y) \leq g(x^*) \leq \max_{x\in C_1} g(x)\,.$$ □

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 $\mathcal{Z}_B := \text{box}(\mathcal{Z}) = \langle c, G_B\rangle_Z \supseteq \mathcal{Z}$ 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

$$d_H(\mathcal{Z}, \mathcal{Z}_B) \leq \omega(\mathcal{Z}) := 2\left\|\sum_{k=1}^{\gamma} \widehat{g}^{(k)}\right\|_2 \tag{14}$$

$$\text{with} \quad \widehat{g}_i^{(k)} = \begin{cases} |g_i^{(k)}|, & \text{if } k \neq i^* \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where i^* is the (first) index for which $g_{i^*}^{(k)} = \left\|g^{(k)}\right\|_\infty$.

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,

$$\alpha_k = m_{i^*i^*}\,\text{sgn}\left(g_{i^*}^{(k)}\right) \tag{16}$$

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

$$v_i^{(k)} \in \begin{cases} \left[-2|g_i^{(k)}|, 2|g_i^{(k)}|\right], & \text{if } i \neq i^* \\ 0, & \text{otherwise,} \end{cases}$$

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

$$x_B - x = v^{(1)} + \ldots + v^{(\gamma)} \in [-2\widehat{z}, 2\widehat{z}]\,,$$

where $\widehat{z} := \widehat{g}^{(1)} + \ldots + \widehat{g}^{(\gamma)}$ 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

$$\|g^{(1)}\|_1 - \|g^{(1)}\|_\infty \leq \ldots \leq \|g^{(\gamma)}\|_1 - \|g^{(\gamma)}\|_\infty \tag{17}$$

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:

$$\kappa \leftarrow \begin{cases} 1, & \text{if } \forall i \in \{1, \ldots, n\} \text{ with } \Psi_i > 0 : \frac{d(\mathsf{box}(\Psi_i(\kappa=2)))}{d(\mathsf{box}(\Psi_i(\kappa=1)))} \geq \zeta_K \in (0, 1) \\ 2, & \text{otherwise,} \end{cases} \tag{19}$$

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

$$\begin{aligned}\tilde{\mathcal{R}}(t+\Delta) = \mathsf{red}\big(e^{Aq\Delta t_k}\mathsf{red}\big(e^{A\Delta t_k}\ldots \\ \mathsf{red}\big(e^{A\Delta t_k}\mathcal{R}(t) \oplus \mathcal{R}_{\text{abs}}\big)\ldots \oplus \mathcal{R}_{\text{abs}}\big) \oplus q\mathcal{R}_{\text{abs}}\big)\,.\end{aligned} \tag{20}$$

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

$$r\big(\mathcal{S}\big) = \frac{1}{2}\,\|d\big(\mathsf{box}\big(\mathcal{S}\big)\big)\|_2\,.$$

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:

$$r_\mathrm{R}(t+j\Delta t_k) = (1+2\zeta_Z)\big(\zeta_P^{\frac{1}{k}} r_\mathrm{R}(t+(j-1)\Delta t_k) + r_{\mathrm{abs},k}\big)\,. \quad (22)$$

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

$$r_\mathrm{R}(t+\Delta) = r_0 (1+2\zeta_Z)^k \zeta_P + r_{\mathrm{abs},k}\, \zeta_{\mathrm{P,Z}}(k) \quad (23)$$

$$\text{with} \quad \zeta_{\mathrm{P,Z}}(k) = \sum_{i=1}^{\lfloor k \rfloor} (1+2\zeta_Z)^{q+i} \zeta_P^{\frac{q+i-1}{k}} + q\,(1+2\zeta_Z)^q$$

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., $d_i\big(\mathsf{box}\big(\mathcal{R}_{abs}\big)\big)$, by $\zeta_\delta^{\lambda_i}$ with $\lambda_i \in \mathbb{N}$ when Δt goes to 0:*

$$\forall i \in \{1,...,n\} : \quad \lim_{\Delta t \to 0} \frac{d_i\big(\mathsf{box}\big(\mathcal{R}_{abs}(\zeta_\delta \Delta t)\big)\big)}{d_i\big(\mathsf{box}\big(\mathcal{R}_{abs}(\Delta t)\big)\big)} = \zeta_\delta^{\lambda_i}\,. \quad (24)$$

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., $d_i\big(\mathsf{box}\big(\mathcal{R}_\mathrm{abs}(\zeta_\delta \Delta t)\big)\big) < \zeta_\delta\, d_i\big(\mathsf{box}\big(\mathcal{R}_\mathrm{abs}(\Delta t)\big)\big)$. For later derivations, we define the gain

$$\varphi(\Delta t) = \max_{i \in \{1,...,n\}} \frac{d_i\big(\mathsf{box}\big(\mathcal{R}_\mathrm{abs}(\zeta_\delta \Delta t)\big)\big)}{d_i\big(\mathsf{box}\big(\mathcal{R}_\mathrm{abs}(\Delta t)\big)\big)}\,. \quad (25)$$

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 $\lim_{\Delta t \to 0} \varphi(\Delta t) = \zeta_\delta$:

$$\varphi(\Delta t) \approx \zeta_\delta + \frac{\varphi_1 - \zeta_\delta}{\Delta} \Delta t\,, \quad (26)$$

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

$$\varphi_j = \varphi(\zeta_\delta^{j-1} \Delta) = \zeta_\delta + (\varphi_1 - \zeta_\delta) \zeta_\delta^{j-1} \quad (27)$$

we obtain an estimate for $r_{\mathrm{abs},k}$:

$$k\, r_{\mathrm{abs},k} = \varphi_1 \cdot ... \cdot \varphi_{k'}\, r_{\mathrm{abs},1} \Longrightarrow r_{\mathrm{abs},k} := \frac{r_{\mathrm{abs},1}}{k} \prod_{j=1}^{k'} \varphi_j\,. \quad (28)$$

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:

$$\varphi_1 \cdot \big(\zeta_\delta + (\varphi_1 - \zeta_\delta)\zeta_\delta\big) \cdot ... \big(\zeta_\delta + (\varphi_1 - \zeta_\delta)\zeta_\delta^{k'-1}\big) = k \frac{r_{\mathrm{abs},k}}{r_{\mathrm{abs},1}}\,. \quad (29)$$

Optimization function. Inserting (28) in (23) yields

$$r_\mathrm{R}(t+\Delta) = r_0 (1+2\zeta_Z)^k \zeta_P + \frac{r_{\mathrm{abs},1}}{k} \zeta_{\mathrm{P,Z}}(k) \prod_{j=1}^{k'} \varphi_j\,, \quad (30)$$

which we minimize to obtain the optimal time step size

$$\Delta t_* = \Delta\, \zeta_\delta^{k'_*} \quad (31)$$

$$\text{where} \quad k'_* = \operatorname*{arg\,min}_{k' \in \mathbb{N}} r_\mathrm{R}(t+\Delta)\,.$$

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 $\Delta t \to 0$. 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: φ_1 and Δ of previous step (only if $s > 1$), r_0, ζ_Z, ζ_P, ζ_δ, s
Output: Optimal time step size Δt_*, φ_1

1: **if** $s = 1$ **then**
2: Initialize Δt ▷ Arbitrary initialization
3: **while** $\varphi_1 < \zeta_\Delta$ **do**
4: $\Delta \leftarrow \zeta_\delta \Delta$ ▷ Decrease Δt
5: Compute $\mathcal{R}_{\text{abs}}(\Delta), \mathcal{R}_{\text{abs}}(\zeta_\delta \Delta)$ ▷ Note: $\mathcal{R}_{\text{abs}}$ reusable
6: Compute φ_1 by (25) ▷ Behavior of $\mathcal{R}_{\text{abs}}$ over Δt
7: **end while**
8: Compute Δt_* 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 Δt_* by (31) ▷ Optimal time step size
13: Compute φ_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.

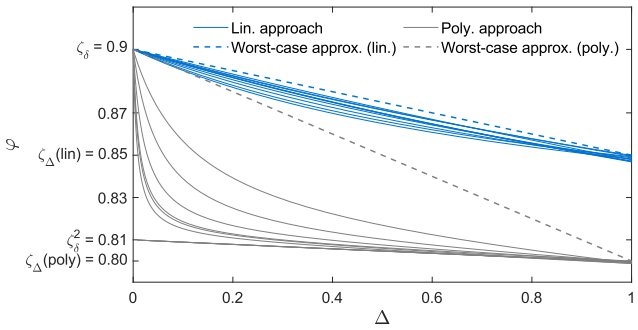


Figure 3: Computation of φ by (25) for all systems in Sec. 4 and both approaches (linearization in blue and polynomialization in gray) over $0 < \Delta t < \Delta$ (normalized to $[0, 1]$), with Δ computed by (32).

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.

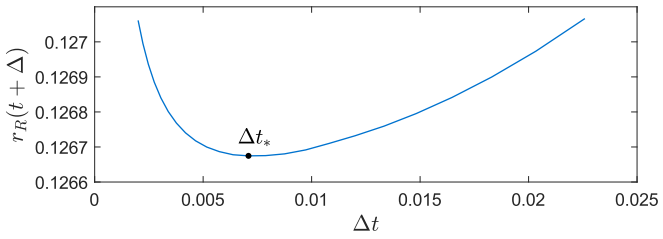


Figure 4: Evaluation of the cost function $r_R(t+\Delta)$ (30) over different values for Δt at $t = 0$ for example 4.1.

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.

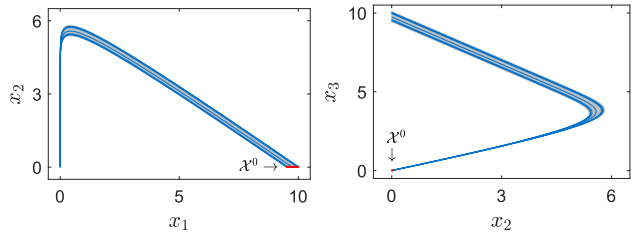


Figure 5: Reachable set $\mathcal{R}([0, t_K])$ of example 4.2 in gray, initial set $\mathcal{X}^0$ in red, simulations in blue.

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 overapproximations. 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.

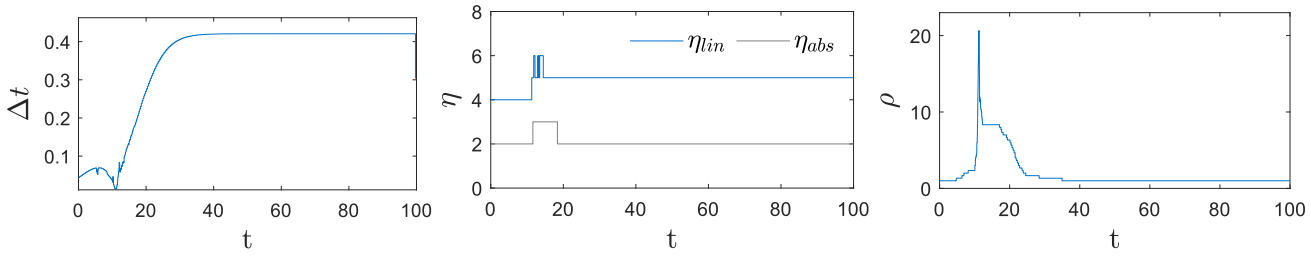
Table 2: Comparing our approach with different reachability tools on ARCH benchmarks using the tightness measurements $\mu_1 = \texttt{vol}(\texttt{box}(\mathcal{R}(t_K)))$ and $\mu_2 = l_4$, where $l = d(\texttt{box}(\mathcal{R}(t_K)))$ as in [26].

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 $\eta_{\text{lin}}, \eta_{\text{abs}}$.** **(c) Zonotope order ρ.**

Figure 6: Algorithm parameters of example 4.2 over time.

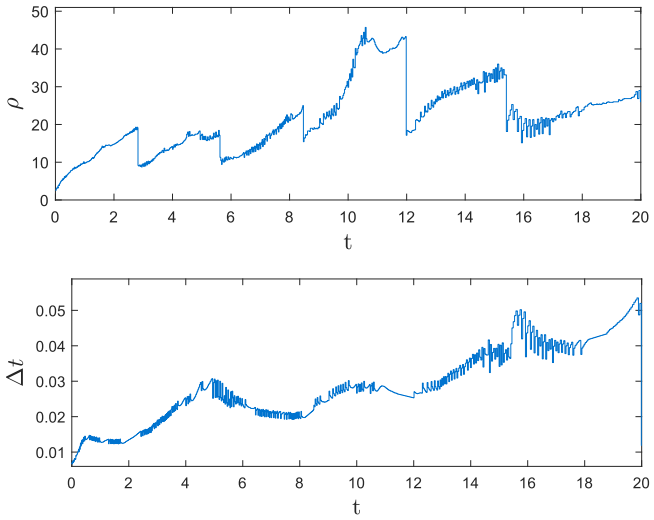


Figure 7: Time step size Δt and polynomial zonotope order ρ of example 4.3 over time.

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,

$$l_{\max} = \max_{i \in \{1,\dots,n\}} d_i\big(\text{box}\big(\mathcal{R}(t_K)\big)\big) \tag{33}$$

as well as a tightness measure proposed in [18, Sec. VI.]

$$\gamma_{\min} = \min_{i \in \{1,\dots,n\}} \frac{d_i\big(\text{box}\big(\mathcal{R}_{\text{sim}}(t_K)\big)\big)}{d_i\big(\text{box}\big(\mathcal{R}(t_K)\big)\big)}, \tag{34}$$

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 nonlinearities. 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, $[\Delta t_{\min}, \Delta t_{\max}]$: range of time step sizes, $\rho_{\max}$: max. zonotope order, $l_{\max}$ and $\gamma_{\min}$: measurements by (33) and (34).

Benchmark	n	t_K	$\mathcal{X}^0$	Linearization Approach					Polynomialization Approach				
				Time	$[\Delta t_{\min}, \Delta t_{\max}]$	$\rho_{\max}$	$l_{\max}$	$\gamma_{\min}$	Time	$[\Delta t_{\min}, \Delta t_{\max}]$	$\rho_{\max}$	$l_{\max}$	$\gamma_{\min}$
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 nonlinear 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.