Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

OURS: Over- and Under-approximating Reachable Sets for analytic time-invariant differential equations☆

Ruiqi Hu, Zhikun She *

SKLSDE, LMIB and School of Mathematical Sciences, Beihang University, Beijing 100191, China

ARTICLE INFO

Keywords:
Reachable set
Evolution function
Over approximation
Under approximation
Parametric disturbances

ABSTRACT

We present **OURS**, a precision-oriented **MATLAB** tool for computing ***O**ver- and **U**nder- approximations of **R**eachable **S**ets* for analytic time-invariant differential equations. The main theoretical framework behind **OURS** is introduced, including the concept of evolution function, whose zero sub-level sets are used to describe reachable sets, and a series representation of evolution function. Especially, using the partial sums of this series, **OURS** finds over- and under-approximations of evolution function at time-instants: it consecutively estimates each remainder of the corresponding partial sum of the series with interval arithmetics until one remainder satisfies the designated precision, and then builds over- and under-approximations with this remainder. The structure of **OURS** is also presented, such as the forms of inputs and outputs, and technical implementations of the crucial steps inside. Moreover, we compare **OURS** with two other existing methods on some benchmarks. Finally, **OURS** is additionally extended to deal with a class of time-invariant differential systems with disturbances described by uncertain parameters. The performance of **OURS** in dealing with parametric uncertainties is also shown by two examples with comparisons and discussions.

1. Overview of OURS

Safety verification and system falsification [2] are two crucial issues in various realistic problems. However, it is impossible to obtain the exact reachable set in general, especially for non-linear systems. In most cases, it is practical and convenient to find over- or under-approximations of the reachable set instead. Over-approximations of the reachable set can be used to prove that the system avoids the unsafe set; on the other hand, under-approximations of the reachable set can be used to prove that the system reaches the target set [3].

In this paper, we present **OURS**, a **MATLAB** tool for computing both **O**ver- and **U**nder-approximations of **R**eachable **S**ets for *analytic time-invariant differential equations* (ATDEs). Following the theoretical analysis in [4,5], we start with the concept of *evolution function* (EF) of an ATDE and find a series representation of EF w.r.t. time t, named *t-expansion*. Therefore we can get the EF without analytic solutions to the ATDE. **OURS** consecutively conducts *remainder estimations* (RE) for the partial sums of t-expansion with interval arithmetics until one remainder satisfies the designated precision, and then builds over-/under-approximations of EF, leading to the over-/under-approximations of the reachable set. The main features of **OURS** are the following:

1. Equipped with sub-level sets, **OURS** can directly handle non-convex and even non-connected sets (see Fig. 2(c)). Moreover, it can obtain over- and under-approximations of reachable set at any desired instant in the time interval from obtained explicit expressions without repeated computation.
2. Due to the use of interval arithmetics, the current version of **OURS** allows rational functions, and even trigonometric (see Examples 4 and 7) and exponential functions with arbitrary nestings in the input.
3. Different from [4], **OURS** uses the information not only from forward reachable sets but also form backward reachable sets (see Theorem 1 and Algorithm 1), guaranteeing the output over- and under-approximations to be bounded by designated precision (see Theorem 3). Moreover, we particularly illustrate many of the details on implementation in this paper.
4. **OURS** can be extended to deal with a class of time-invariant differential equations with disturbances described by uncertain parameters, and produce over- and under- approximations of both minimal and maximal reachable sets of the uncertain systems, respectively.

☆ This work was supported by the Beijing Natural Science Foundation (Z180005). This is an extended and revised version of our earlier conference paper [1]. Different from [1], we have a new section to explain that **OURS** can be extended to deal with a class of time-invariant differential systems with disturbances described by uncertain parameters, including two examples with comparisons and discussions.

* Corresponding author.
E-mail addresses: by1809102@buaa.edu.cn (R. Hu), zhikun.she@buaa.edu.cn (Z. She).

https://doi.org/10.1016/j.sysarc.2022.102580
Received 5 February 2022; Received in revised form 10 April 2022; Accepted 19 May 2022

1.1. Related work

In recent years, various methods have been discussed in the literature for computing over- and under- approximations of reachable sets, such as the abstraction method [6,7], the Lagrangian method [8,9], the Taylor expansion method [10,11], the support function method [12,13], the constraint solving method [14], the level set method [15,16], etc. Their corresponding tools have also been developed. **CORA** [17], a typical abstraction based tool proposed by M. Althoff, is designed for various kinds of systems with purely continuous dynamics (linear systems, nonlinear systems, differential–algebraic systems, parameter-varying systems, etc.) and supports over-approximative computation of reachable sets. Note that **CORA** is also a Taylor expansion method since **CORA** abstracts the nonlinear system to a Taylor expansion of order k at a point with Lagrange remainder. Different from **CORA**, **OURS** first performs expansion on the evolution function directly instead of the vector field and then estimates the remainder of partial sums arriving at over- and under-approximations. Moreover, **CORA** [18] is also able to extract under-approximations of reachable sets. Lagrangian method [19] studies the continual reachability set and its connection to other backward reachability constructs, further it can be generalized to characterize robust reachable sets or viability kernels [20]. **Flow*** [21,23], a Taylor expansion based tool, first determines a k-order Taylor Model approximation for the flow map starting from the initial set by using Picard iteration and then finds a remainder interval of Picard iteration for generating flowpipes of non-linear hybrid systems. Note that **Flow*** can also handle discrete transitions on Taylor model flowpipe construction based on domain contraction and range over-approximation, allowing to represent non-convex sets [22]. Different from **Flow***, **OURS** uses level sets instead of nonlinear constraints to specify non-convex sets and considers the Taylor expansion on the evolution function instead of the flow map. **SpaceEx** [13], a support function based tool focuses on the verification of safety properties of hybrid systems with piecewise constant bounds on the derivatives, and solves the scalability issues by making use of a wrapping-free algorithm for linear continuous systems. [15] describes reachable sets as the sub-level sets of the viscosity solutions of the time-dependent Hamilton–Jacobi partial differential equations, and then uses an efficient toolbox of level set method [16] to numerically solve them by gridding the state space so that one can tune the accuracy of the results by varying the number of grid points. Note that the results of ToolboxLS are numerical data, and the outputs of **OURS** are forms of algebraic inequalities. The recent method in [24], a constraint solving based tool, also uses the level sets of the solution of Hamilton–Jacobi equations to represent the reachable set, but reduces the computation of approximations to a semi-definite programming problem and then solves this convex optimization by sum-of-squares decomposition tools. Different from the methods in [15,24], **OURS** uses level sets of evolution functions instead of solutions of Hamilton–Jacobi equations to characterize reachable sets.

Moreover, many practical systems are usually additionally affected by uncertainties in the environment. So reachability properties of systems with uncertainties have also been widely discussed in the literature [25–27], where minimal and maximal reachable sets have been introduced. For example, [28] developed a new framework for formulating reachability problems with competing inputs, nonlinear dynamics, and state constraints as optimal control problems and applied it to a two aircraft collision avoidance scenario under target window constraints and in the presence of wind disturbance; [29] proposed a convex programming based method to address a problem of inner-approximating backward reachable sets of state-constrained polynomial systems subject to time-varying uncertainties; [30] proposed an algorithm, which is based on the conservative representation of the nonlinear dynamics by a differential inclusion consisting of a linear term and the Minkowsky sum of two convex sets, to over-approximate the reachable set of nonlinear systems with bounded, time-varying parameters, and uncertain initial conditions.

2. Main theoretical features

OURS deals with analytic time-invariant differential equations (ATDEs) of form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}),\ \mathbf{x} \in \mathbb{D}, \tag{1}$$

where $\mathbf{x}$ is an n-dimensional vector, $\mathbb{D} \subseteq \mathbb{R}^n$ is a domain and $\mathbf{f}(\mathbf{x}) : \mathbb{D} \to \mathbb{R}^n$ is an analytic real function. We define $\phi(\mathbf{x}_0, t)$ as the solution to (1), where $\mathbf{x}_0$ is the initial state. Moreover, we use $(T^-_{\mathbf{x}_0}, T^+_{\mathbf{x}_0})$ to represent the maximal time interval of $\phi(\mathbf{x}_0, t)$, and define $\mathcal{R}^- \equiv \{(\mathbf{x}, t) \mid \mathbf{x} \in \mathbb{D} \wedge t \in (T^-_{\mathbf{x}}, T^+_{\mathbf{x}})\}$.

2.1. EF based description of reachable set

We define *reachable set* starting from the initial set of states as follows.

Definition 1. For given initial set of states $\mathbf{X}_0$ and time instant t, the *reachable set* $Reach^t_{\mathbf{f},\mathbf{X}_0}$ of the ATDE (1) from $\mathbf{X}_0$ at instant t is defined as $Reach^t_{\mathbf{f},\mathbf{X}_0} = \{\phi(\mathbf{x}_0, t) \in \mathbb{R}^n \mid \mathbf{x}_0 \in \mathbf{X}_0\}$. And the *reachable set* within the time interval $[T_1, T_2]$ is defined as $\mathbf{Reach}^{[T_1,T_2]}_{\mathbf{f},\mathbf{X}_0} = \bigcup_{t\in[T_1,T_2]} Reach^t_{\mathbf{f},\mathbf{X}_0}$. For simplicity, we additionally denote $\mathbf{Reach}^{[0,T]}_{\mathbf{f},\mathbf{X}_0}$ as $\mathbf{Reach}^{T}_{\mathbf{f},\mathbf{X}_0}$.

Throughout this paper, if the initial set $\mathbf{X}_0 \subset \mathbb{D}$ can be described by the zero sub-level set $\mu(g)$ of a function $g(\mathbf{x})$, i.e., $X_0 = \mu(g) := \{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \le 0\}$, we use $Reach^t_{\mathbf{f},g}$ ($\mathbf{Reach}^T_{\mathbf{f},g}$) as an alias of $Reach^t_{\mathbf{f},\mathbf{X}_0}$ ($\mathbf{Reach}^T_{\mathbf{f},\mathbf{X}_0}$). Then, we will define evolution function as follows, which will be used for representing the reachable set $Reach^t_{\mathbf{f},\mathbf{X}_0}$ in Proposition 1.

Definition 2. For an analytic function $g : \mathbb{R}^n \to \mathbb{R}$, the *evolution function* of the ATDE (1) with $g(\mathbf{x})$ is defined as $Evo_{\mathbf{f},g}(\mathbf{x}, t) = g(\phi(\mathbf{x}, -t)), \forall(\mathbf{x}, t) \in \mathcal{R}^-$.

The relationship between $Reach^t_{\mathbf{f},g}$ and $Evo_{\mathbf{f},g}(\mathbf{x}, t)$ can be illustrated as follows.

Proposition 1. *For ATDE (1) and analytic* $g : \mathbb{R}^n \to \mathbb{R}$, $Reach^t_{\mathbf{f},g} = \{\mathbf{x} \in \mathbb{R}^n \mid Evo_{\mathbf{f},g}(\cdot, t) \le 0\}$.

Proposition 1 shows that, we can compute evolution function to represent the reachable set. However, evolution function depends on the solution of (1) which is generally unaccessible. Alternatively, we consider the Taylor expansion of EF w.r.t. time t, named *t-expansion*:

$$Evo_{\mathbf{f},g}(\mathbf{x}, t) = \sum_{i=0}^{+\infty} \frac{\mathcal{M}^i_{\mathbf{f},g}(\mathbf{x})}{i!}(-t)^i, \tag{2}$$

where $\mathcal{M}^i_{\mathbf{f},g}(\mathbf{x})$ is defined inductively as $\mathcal{M}^0_{\mathbf{f},g}(\mathbf{x}) = g(\mathbf{x})$ and $\mathcal{M}^{i+1}_{\mathbf{f},g}(\mathbf{x}) = \frac{\partial \mathcal{M}^i_{\mathbf{f},g}(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{D}$. Moreover, we denote $Evo^N_{\mathbf{f},g}(\mathbf{x}, t)$ as the Nth partial sum of t-expansion, i.e. $Evo^N_{\mathbf{f},g}(\mathbf{x}, t) \equiv \sum_{i=0}^{N} \frac{(-t)^i \mathcal{M}^i_{\mathbf{f},g}(\mathbf{x})}{i!}$.

2.2. RE based approximation of reachable sets

We denote the remainder of the Nth partial sum of $Evo_{\mathbf{f},g}(\mathbf{x}, t)$ as $Rem^N_{\mathbf{f},g}(\mathbf{x}, t) = Evo_{\mathbf{f},g}(\mathbf{x}, t) - Evo^N_{\mathbf{f},g}(\mathbf{x}, t)$. It is proved in [4] that $Rem^N_{\mathbf{f},g}(\mathbf{x}, t)$ can be represented as $Rem^N_{\mathbf{f},g}(\mathbf{x}, t) = -\int_0^t \frac{(r-t)^N}{N!} \mathcal{M}^{N+1}_{\mathbf{f},g}(\phi(\mathbf{x}, -r))dr$. Thus, if we can find upper and lower bounds of $\mathcal{M}^{N+1}_{\mathbf{f},g}(\phi(\mathbf{x}, -r))$, we can estimate $Rem^N_{\mathbf{f},g}(\mathbf{x}, t)$, arriving at over- and under-approximations of evolution function.[1] as shown in Theorem 1.

[1] For two n-dimensional scale functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, $f_1(\mathbf{x})$ is an *over- (or under-) approximation* of $f_2(\mathbf{x})$ over S if $f_1(\mathbf{x}) \le (or \ge) f_2(\mathbf{x}), \forall \mathbf{x} \in S$.

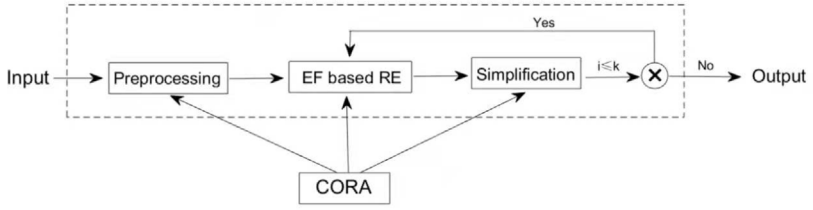
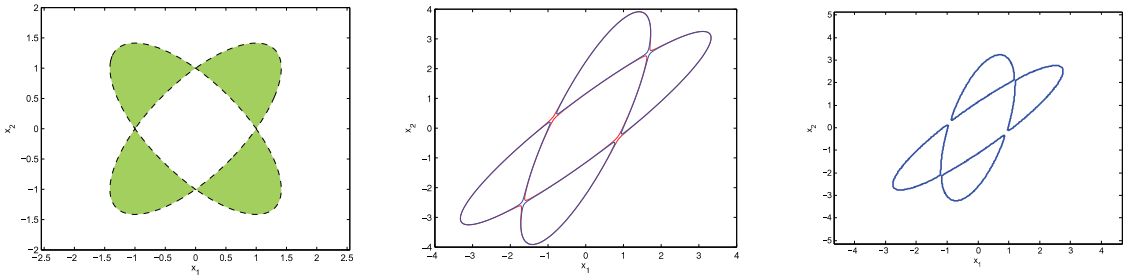


Fig. 1. Structure of **OURS**.



(a) The Initial Set (b) Results at $t = 1$ (c) Under-approximation at $t = 0.8$

Fig. 2. Running example for **OURS**. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Theorem 1. *For given ATDE (1), analytic $g : \mathbb{R}^n \to \mathbb{R}$, degree N and time bound T, assume that S and S' are compact sets of states such that $\mathbb{D} \supseteq S \supseteq \mathbf{Reach}^T_{\mathbf{f},g}$ and $S' \supseteq \mathbf{Reach}^T_{-\mathbf{f},S}$. If we can find constants L_{N+1} and U_{N+1} satisfying that $L_{N+1} \le \mathcal{M}^{N+1}_{\mathbf{f},g}(\mathbf{y}) \le U_{N+1}$, $\forall \mathbf{y} \in S'$, then for all $t \in [0,T]$,*

1. *if N is odd, $Evo^N_{\mathbf{f},g}(\mathbf{x},t) + L_{N+1}\frac{t^{N+1}}{(N+1)!}$ and $Evo^N_{\mathbf{f},g}(\mathbf{x},t) + U_{N+1}\frac{t^{N+1}}{(N+1)!}$ are the over- and under-approximations of $Evo_{\mathbf{f},g}(\mathbf{x},t)$ over S, respectively;*
2. *if N is even, $Evo^N_{\mathbf{f},g}(\mathbf{x},t) - U_{N+1}\frac{t^{N+1}}{(N+1)!}$ and $Evo^N_{\mathbf{f},g}(\mathbf{x},t) - L_{N+1}\frac{t^{N+1}}{(N+1)!}$ are the over- and under-approximations of $Evo_{\mathbf{f},g}(\mathbf{x},t)$ over S, respectively;*
3. *all precisions for above approximations are bounded by $(U_{N+1} - L_{N+1})\frac{t^{N+1}}{(N+1)!}$.*

To find over- and under-approximations of EF, we need to estimate upper and lower bounds of $\mathcal{M}^{N+1}_{\mathbf{f},g}(\phi(\mathbf{x},-r))$. Since $x \in S$ and $-r \in (-t,0)$, we have $\mathbf{y} = \phi(\mathbf{x},-r) \in \mathbf{Reach}^T_{-\mathbf{f},S}$. Considering that it is inconvenient to use $\mathbf{Reach}^T_{-\mathbf{f},S}$ directly as the bound of $\mathbf{y}$ to estimate $\mathcal{M}^{N+1}_{\mathbf{f},g}(\mathbf{y})$, so we use a box S' containing the reachable set as the bound. Then we can iteratively compute upper bound U_{N+1} and lower bound L_{N+1} of each $\mathcal{M}^{N+1}_{\mathbf{f},g}(\mathbf{y})$ for increasing N until $U_{N+1} - L_{N+1} \le \frac{\epsilon\cdot(N+1)!}{T^{N+1}}$, such that the precision for over- and under-approximations of EF are bounded by ϵ, and then generate over- and under-approximations of EF as described in Theorem 1.

Remark 1. There is another method based on quantifier elimination in [4] to estimate over- and under-approximations of EF, that is, with the same precondition as Theorem 1, for given $\varepsilon > 0$, if a function $P(\mathbf{x},t)$ satisfies $P(\mathbf{x},0) = g(\mathbf{x})$ and $-\varepsilon \le (0 \le)\frac{\partial P(\mathbf{x},t)}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}) + \frac{\partial P(\mathbf{x},t)}{\partial t} \le 0 (\le \varepsilon)$ for all $(\mathbf{x},t) \in S' \times [0,T]$, then $P(\mathbf{x},t)$ is an over- (under-) approximation of EF over S, i.e., for all $t \in [0,T]$ and $\mathbf{x} \in S$, $0 \le (-\varepsilon t \le) Evo_{\mathbf{f},g}(\mathbf{x},t) - P(\mathbf{x},t) \le \varepsilon t (\le 0)$. Since this method is based on quantifier elimination [31], it is not suitable for precision-oriented requirement and thus we here only use the method based on RE to design **OURS**.

3. Main implementation features

The structure of **OURS** is shown in Fig. 1. First, **OURS** reads the inputs including the description of ATDE, initial set, time bound, etc., and splits the time bound into segments with equal length. Then, before operating in each time segment, **OURS** computes a series of rough enclosures for reachable sets (*Preprocessing*). For each segment, **OURS** first iteratively computes upper and lower bounds of each remainder of the partial sums for increasing degree until one remainder satisfies the designated precision, and then constructs over- and under-approximations according to Theorem 1 (*EF based RE*); further, **OURS** simplifies the results for output (*Simplification*). **OURS** repeats this procedure for all time-segments and returns piecewise over- and under-approximations of EF, with a list of bounds for reachable sets for the corresponding segments.

3.1. Inputs and outputs

We use the following example as the running example

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -0.2x_1 + x_2 \end{cases},$$

with the initial set $\{(x_1,x_2) \in \mathbb{R}^2 \mid (x_1^2-1)^2 + (x_2^2-1)^2 \le 1\}$ (drawn in Fig. 2(a)) and the time bound $T = 1$, to illustrate the input and output of **OURS**.

The input of **OURS** contains: `varList`, `f`, `g`, `S`, `epsilon`, `T`, `K`, `r`.

- `varList` is the symbolic list of names for all dimensions in the dynamical system. Therefore the length of `varList` is the dimension of the input system variables. In the running example, `varList:=[x1,x2]`.
- `f` is the vector field and `g` determines the initial set $X_0 = \mu(g)$. In the running example, `f:=[x2,-x1/5+x2]`, `g:=(x1^2-1)^2+(x2^2-1)^2-1`. Notice that if `f` or `g` is non-polynomial, so is the output of **OURS**. Current version of **OURS** allows rational functions, and even trigonometric and exponential functions with arbitrary nestings in `f` and `g`.
- `S` is a box containing X_0. In the running example, `S:=[-2,2,-2,2]` is a proper choice of `S`, indicating that $-2 \le$ `x1` ≤ 2 and $-2 \le$ `x2` ≤ 2.
- `epsilon` is the precision requirement for the results. Since **OURS** is a precision-oriented method, we need to designate `epsilon` for over- and under-approximations. In the running example we set `epsilon:=1e-2`.
- `T` is the time bound. In the running example we set `T:=1`, indicating that we estimate $Evo_{\mathbf{f},g}(\mathbf{x},t)$ for $t \in [0,1]$.

- K is a natural number to split time bound. Since **OURS** uses time-splitting technique, K is required to separate the given time interval. In the running example, we set K:=10, so that each time-segment is $[0, 0.1]$.
- r is a number within $(0, 1]$. It represents the negotiation rate of precision allocation: **OURS** will allocate a part of precision, r * epsilon, to *EF based RE* and the rest, $(1 - r) *$ epsilon, to *Simplification*. In the running example and examples in Section 4, we set r:=0.2.

The output of **OURS** contains: Over, Under, Bound.

- Over and Under are two lists of functions representing over- and under-approximations of EF over Bound respectively. They can be treated as piecewise functions $Over(\mathbf{x}, t)$ and $Under(\mathbf{x}, t)$ such that for the ith seg-ment, $Over^i(\mathbf{x}, t)$ and $Under^i(\mathbf{x}, t)$ are stored in Over(i) and Under(i) (i.e. $Over^i(\mathbf{x}, t)$ = Over(i) and $Under^i(\mathbf{x}, t)$ = Under(i)) for $\mathbf{x} \in$ Bound(i). Thus, $Over(\mathbf{x}, t) = Over^j(\mathbf{x}, t - \frac{j-1}{\mathrm{K}}\mathrm{T})$, $Under(\mathbf{x}, t) = Under^j(\mathbf{x}, t - \frac{j-1}{\mathrm{K}}\mathrm{T})$ for all $\mathbf{x} \in$ Bound(i) and $t \in [\frac{j-1}{\mathrm{K}}\mathrm{T}, \frac{j}{\mathrm{K}}\mathrm{T}], 1 \le j \le \mathrm{K}$.
- Bound is the list of enclosures for reachable sets in each time-segment.

Results of the running example at $t = 1$ are depicted in Fig. 2(b) using ezplot in **MATLAB**. The red/blue lines represent the boundaries of the over-/under-approximation of reachable set at $t = 1$ respectively. It is worthy to note that **OURS** can deal with non-convex even disconnected sets. For example, in the running example, the under-approximation of EF at $t = 0.8$ is 0.81682*x1^4-1.8506*x1^3*x2+1.5878*x1^2*x2^2-1.8307*x1^2 -0.56579*x1*x2^3 +1.8725*x1*x2+0.11316*x2^4-0.91999*x2^2+1.0039. The corresponding zero sub-level set in Bound(8) is non-connected, as shown in Fig. 2(c).

3.2. Technical implementation

Based on the structure of **OURS** in the previous subsection, we here present the pseudo-code of **OURS** in Algorithm 1. **OURS** first defines time length ΔT := T/K and precision requirement $\Delta\epsilon$:= epsilon/K for time-segment, and consecutively calls procedures *Preprocessing* and *EF based RE* for computing over- and under-approximations of EF. Therein, for each iteration, we denote the initial function as $Init(\mathbf{x})$ and the Nth partial sum of EF as $Tr(\mathbf{x}, t)$. Further, **OURS** calls procedures *Simplification* to simplify $Tr(\mathbf{x}, t)$ and then obtain both over- and under-approximations of evolution function with less terms, as shown in Lines 8–9 in Algorithm 1. Clearly, the performances of these three procedures are crucial to the efficacy of **OURS**. In this subsection we explain how *Preprocessing*, *EF based RE* and *Simplification* work when calling **OURS**.

3.2.1. Preprocessing

In *Preprocessing*, **OURS** computes a rough enclosure of the reachable set for each segment as follows, which is necessary for using Theorem 1. First, starting from the input S, **OURS** successively calls **CORA**[2] with Taylor terms = 3 to get an enclosure Bound for the reachable set in each time segment based on the previous enclosure. **CORA** can compute the over-approximation of reachable set within given time-interval and return the resulting boxes, which is really suitable for our purpose. Then, **OURS** uses **CORA** for each time segment to calculate the over-approximation of the backward reachable set of previous over-approximation, denoted as Interval. Here Bound and Interval satisfy that: $\mathbf{Reach}_{\mathbf{f},g}^{[(i-1)\Delta T, i\Delta T]} \subseteq$ Bound(i) for all $1 \le i \le$ K, Bound(K) = Interval(K), and $\mathbf{Reach}_{-\mathbf{f},\mathrm{Interval}(K)}^{[(K-i)\Delta T, K-i+1\Delta T]} \subseteq$ Interval($i-1$) for all $1 \le i \le$ K. Note that for the ith segment, the initial set is $\mathbf{X}_{i-1} = Reach_{\mathbf{f},g}^{(i-1)\Delta T}$, $\mathbf{Reach}_{\mathbf{f},\mathbf{X}_{i-1}}^{\Delta T} \subseteq$ Bound(i) and $\mathbf{Reach}_{-\mathbf{f},\mathrm{Bound}(i)}^{\Delta T} \subseteq$ Interval($i-1$); moreover, Bound(i)/Interval($i-1$) will be the enclosure S/S' used for Theorem 1.

[2] The version we use is **CORA**−2020. We can also replace **CORA** with some other tools that can calculate a rough enclosure of the reachable set, such as **VNODE-LP** [32] and **RealPaver** [33] etc..

Algorithm 1 Tool **OURS**

Input: varList, f, g, S, epsilon, T, K, r;
Output: Over, Under, Bound.

1: ΔT := T/K, $\Delta\epsilon$:= epsilon/K;
2: Initialize $Init(\mathbf{x}) \Leftarrow$ g, $Tail_o \Leftarrow 0$, $Tail_u \Leftarrow 0$, $i \Leftarrow 1$;
3: Call *Preprocessing* (i.e, Algorithm 2) for Bound and Interval;
4: **while** $i \le$ K **do**
5: $S' \Leftarrow$ Interval($i-1$), $S \Leftarrow$ Interval(i) and $i \Leftarrow i+1$;
6: Call *EF based RE* (i.e., Algorithm 3) for $Tr(\mathbf{x}, t)$, $Tail_o'(t)$, $Tail_u'(t)$;
7: Call *Simplification* (i.e., Algorithm 4) for $Tr(\mathbf{x}, t)$;
8: $\mathrm{Over}^i \Leftarrow Tr(\mathbf{x}, t) + Tail_o - \frac{(1-\mathrm{r})\Delta\epsilon}{2} + Tail_o'(t)$;
9: $\mathrm{Under}^i \Leftarrow Tr(\mathbf{x}, t) + Tail_u + \frac{(1-\mathrm{r})\Delta\epsilon}{2} + Tail_u'(t)$;
10: $Tail_o \Leftarrow Tail_o - \frac{(1-\mathrm{r})\Delta\epsilon}{2} + Tail_o'(\Delta T), Tail_u \Leftarrow Tail_u + \frac{(1-\mathrm{r})\Delta\epsilon}{2} + Tail_u'(\Delta T)$;
11: $Init(\mathbf{x}) \Leftarrow Tr(\mathbf{x}, \Delta T)$;
12: $\mathrm{Over} := \mathrm{Over}^j(\mathbf{x}, t - \frac{j-1}{\mathrm{K}}\mathrm{T}), \forall t \in [\frac{j-1}{\mathrm{K}}\mathrm{T}, \frac{j}{\mathrm{K}}\mathrm{T}], 1 \le j \le \mathrm{K}$;
13: $\mathrm{Under} := \mathrm{Under}^j(\mathbf{x}, t - \frac{j-1}{\mathrm{K}}\mathrm{T}), \forall t \in [\frac{j-1}{\mathrm{K}}\mathrm{T}, \frac{j}{\mathrm{K}}\mathrm{T}], 1 \le j \le \mathrm{K}$;

Algorithm 2 *Preprocessing* in **OURS**

Input: f, S, ΔT, K;
Output: Bound, Interval.

1: $i \Leftarrow 1$, $S \Leftarrow$ S;
2: **while** $i \le$ K **do**
3: Call **CORA**(f, S, ΔT) to update S, and Bound(i) $\Leftarrow S$;
4: $i \Leftarrow i+1$;
5: $i \Leftarrow$ K, Interval(K) $\Leftarrow$ Bound(K);
6: **while** $i \ge 1$ **do**
7: $i \Leftarrow i-1$;
8: Call **CORA**(−f, S, ΔT) to update S, and Interval(i) $\Leftarrow S$;

3.2.2. EF based RE

In *EF based RE*, **OURS** computes the over- and under-approximations of evolution function satisfying the designated precision in S, according to Theorem 1. Specifically, for given f, $Init(\mathbf{x})$, S', ΔT and designated precision r$\Delta\epsilon$, **OURS** iteratively increases the degree N and estimate the lower bound L and upper bound U of $\mathcal{M}_{\mathbf{f},g}^{N+1}(\mathbf{x})$ over S' with interval arithmetics in **CORA** until $U - L \le \frac{\mathrm{r}\Delta\epsilon\cdot(N+1)!}{T^{N+1}}$. Then **OURS** returns the current partial sum of t-expansion $Tr(\mathbf{x}, t) = Evo_{\mathbf{f},Init}^{N}(\mathbf{x}, t)$, and the remainders of over- and under-approximations $(Tail_o'(t), Tail_u'(t))$ of EF in S, obtained based on Theorem 1:

- $Tail_o'(t) = L\frac{t^{N+1}}{(N+1)!}$, $Tail_u'(t) = U\frac{t^{N+1}}{(N+1)!}$, if N is odd.
- $Tail_o'(t) = -U\frac{t^{N+1}}{(N+1)!}$, $Tail_u'(t) = -L\frac{t^{N+1}}{(N+1)!}$, if N is even.

Theorem 1 further shows us that $|Tail_o'(t) - Tail_u'(t)|$, the error of the current step, is not greater than r$\Delta\epsilon$.

Moreover, for each time-segment, the corresponding initial functions $g(\mathbf{x})$ for the over-approximation and the under-approximation are $Init(\mathbf{x}) + Tail_o$ and $Init(\mathbf{x}) + Tail_u$, respectively. Observing that $Tail_o$ and $Tail_u$ are constants, it is easy to see that even though these two initial functions $Init(\mathbf{x}) + Tail_o$ and $Init(\mathbf{x}) + Tail_u$ are different, for all $i \ge 0$, $\mathcal{M}_{\mathbf{f},g}^{i+1}(\mathbf{x}) = \frac{\partial\mathcal{M}_{\mathbf{f},g}^{i}(\mathbf{x})}{\partial\mathbf{x}} \cdot \mathbf{f}(\mathbf{x}) = \frac{\partial\mathcal{M}_{\mathbf{f},Init}^{i}(\mathbf{x})}{\partial\mathbf{x}} \cdot \mathbf{f}(\mathbf{x}) = \mathcal{M}_{\mathbf{f},Init}^{i+1}(\mathbf{x})$ holds for these two different initial functions, indicating that $Tail_o$ and $Tail_u$ do not influence the satisfaction of $U - L \le \frac{\mathrm{r}\Delta\epsilon\cdot(N+1)!}{T^{N+1}}$ in *EF based RE* and only the $Init(\mathbf{x})$ that appears in both $Init(\mathbf{x}) + Tail_o$ and $Init(\mathbf{x}) + Tail_u$

Algorithm 3 *EF based RE* in **OURS**

Input: $\mathtt{f}$, $Init(\mathbf{x})$, $S^{'}$, ΔT, $\mathtt{r}\Delta\epsilon$;
Output: $Tr(\mathbf{x},t), Tail_o^{'}(t), Tail_u^{'}(t)$.

1: Initialize $Tr(\mathbf{x},t) \Leftarrow Init(\mathbf{x})$, $\mathbf{f}(\mathbf{x}) \Leftarrow \mathtt{f}$, $M(\mathbf{x}) \Leftarrow \frac{\partial Init(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x})$, and $N \Leftarrow 0$;
2: Call **CORA** for L and U of $M(\mathbf{x})$ in $S^{'}$;
3: **while not** $U - L \leq \frac{\mathtt{r}\Delta\epsilon\cdot(N+1)!}{T^{N+1}}$ **do**
4: $\quad Tr(\mathbf{x},t) \Leftarrow Tr(\mathbf{x},t) + \frac{(-t)^{N+1}}{(N+1)!} M(\mathbf{x})$;
5: $\quad M(\mathbf{x}) \Leftarrow \frac{\partial M(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x})$, $N \Leftarrow N+1$;
6: $\quad$ Call **CORA** for L and U of $M(\mathbf{x})$ in $S^{'}$;
7: Compute the remainder $Tail_o^{'}(t)$ and $Tail_u^{'}(t)$ according to Theorem 1;

Algorithm 4 *Simplification* in **OURS**

Input: $Tr(\mathbf{x},t), S, \Delta T, \frac{(1-\mathtt{r})\Delta\epsilon}{2}$
Output: Updated $Tr(\mathbf{x},t)$.

1: List all terms in $Tr(\mathbf{x},t)$, denoted as P;
2: Compute an upper bound L of absolute value of each element in P in S by **CORA**;
3: $\bar{P} := sortrow([P,L],2)$, $M \Leftarrow 1$;
4: **while not** sum $(\bar{P}(1:M,2)) \geq \frac{(1-\mathtt{r})\Delta\epsilon}{2}$ **do**
5: $\quad M \Leftarrow M+1$;
6: $Tr(\mathbf{x},t) \Leftarrow sum(\bar{P}(M:end,1))$;

matters. Consequently, in each iteration, we can always generate both over- and under-approximations of EF in a single sweep.

3.2.3. *Simplification*

We noticed that during time-splitting, if we directly use $Tr(\mathbf{x},t)$ obtained from *EF based RE* to initialize the successive iteration, as shown in Line 11 in Algorithm 1, the scale of $Init(\mathbf{x})$ will grow rapidly when i increases. So as a remedy, we introduce procedure *Simplification* into **OURS**. In *Simplification*, **OURS** lists all terms in the result $Tr(\mathbf{x},t)$ of *EF based RE* as a list P, and then calls **CORA** to compute an upper bound of absolute value of each element in P in S. After sorting P by the upper bound in ascending order, denoted as $\bar{P}$, **OURS** gathers the terms in $\bar{P}$ until the sum of the corresponding upper bound of the collected terms exceeds $\frac{(1-\mathtt{r})\Delta\epsilon}{2}$ and removes the previously collected terms. Then **OURS** update $Tr(\mathbf{x},t)$ with the remaining terms, and adds $-\frac{(1-\mathtt{r})\Delta\epsilon}{2}$ / $\frac{(1-\mathtt{r})\Delta\epsilon}{2}$ to the over/under approximation respectively in the output.

3.3. Correctness analysis of *OURS*

Now we prove that $\mathtt{Over}(\mathbf{x},t)$ and $\mathtt{Under}(\mathbf{x},t)$ satisfy the set inclusion relation and designated precision requirement. Specifically, we prove that, the errors of over- and under-approximations of EF are bounded in each segment (see Theorem 2), and correctness is maintained after iteration (see Theorem 3).

Theorem 2. *In the ith segment,* $\mathtt{Over}^i$ *and* $\mathtt{Under}^i$ *satisfy that: for all* $(\mathbf{x},t) \in \mathtt{Interval}(i) \times [0,\Delta T]$, $0 \leq \mathtt{Under}^i(\mathbf{x},t) - \mathtt{Over}^i(\mathbf{x},t) \leq i\Delta\epsilon$.

Proof. Based on Algorithm 3, with the terminating condition of the **while** loop and Theorem 1, the output of *EF based RE* satisfies $|Tail_o'(t) - Tail_u'(t)| < \mathtt{r}\Delta\epsilon$, $\forall(\mathbf{x},t) \in \mathtt{Interval}(i) \times [0,\Delta T]$. Moreover, according to the configuration of $Tail_o$ and $Tail_u$, shown in Line 10 in Algorithm 1, $|Tail_o - Tail_u|$ increases no greater than $\Delta\epsilon$ for each segment. Thus, according to the configuration of $\mathtt{Over}^i$ and $\mathtt{Under}^i$, shown in Lines 8–9 in Algorithm 1, we complete the proof. □

Table 1
Data of results for Examples.

Ex.	RT (s)	$deg_{\mathbf{x},t}$	deg_t	NT	At t = 1	
					$deg_{\mathbf{x}}$	NT
Ex. 1	18/29/58	11/14/–	8/14/–	56/680/–	4/14/–	9/120/–
Ex. 2	53/456/3	15/20/–	6/20/–	410/1771/–	10/20/–	95/231/–
Ex. 3	74/7/5	20/10/–	5/10/–	370/286/–	18/10/–	121/33/–
Ex. 4	6/–/2	–/–/–	2/–/–	5/–/–	–/–/–	4/–/–
Ex. 5	41/–/2	9/–/–	4/–/–	174/–/–	7/–/–	70/–/–
Ex. 6	26/267/43	7/6/–	4/6/–	111/462/–	6/6/–	42/210/–
Ex. 7	821/–/–	–/–/–	8/–/–	4198/–/–	–/–/–	915/–/–

Table 2
Data of results for different parameters.

Ex.	T/ϵ	RT (s)	$deg_{\mathbf{x},t}$	deg_t	NT	At t = T	
						$deg_{\mathbf{x}}$	NT
Ex. 1	$10/10^{-32}$	72	51	49	352	4	9
Ex. 2	$1/10^{-8}$	286	30	9	1710	27	248
Ex. 2	$1/10^{-16}$	998	53	16	8731	46	725
Ex. 3	$1/10^{-8}$	490	65	12	4276	54	671
Ex. 4	$5/10^{-8}$	1156	–	7	1690	–	355
Ex. 4	$10/10^{-4}$	660	–	5	1310	–	387
Ex. 5	$2/10^{-4}$	846	17	5	1625	15	505
Ex. 5	$2/10^{-8}$	5692	27	8	10 211	23	1975
Ex. 6	$1/10^{-8}$	313	14	7	1171	11	242
Ex. 6	$3/10^{-4}$	2215	30	9	8277	25	1240
Ex. 7	$2/10^{-2}$	124 327	–	10	93 485	–	13 792
Ex. 7	$1/10^{-4}$	4179	–	10	18 127	–	2714

Remark 2. According to Theorem 1 and Lines 8–9 in Algorithm 1, we can derive that $0 \leq Evo_{\mathtt{f},\mathtt{Over}^{i-1}(\mathbf{x},\Delta T)}(\mathbf{x},t) - \mathtt{Over}^i(\mathbf{x},t) \leq \Delta\epsilon$ and $0 \leq \mathtt{Under}^i(\mathbf{x},t) - Evo_{\mathtt{f},\mathtt{Under}^{i-1}(\mathbf{x},\Delta T)}(\mathbf{x},t) \leq \Delta\epsilon$ for all $(\mathbf{x},t) \in \mathtt{Interval}(i) \times [0,\Delta T]$, where $\mathtt{Over}^{i-1}/\mathtt{Under}^{i-1}(\mathbf{x},\Delta T) = Init(\mathbf{x}) + Tail_o/Tail_u$.

Theorem 3. *The piecewise functions* $\mathtt{Over}(\mathbf{x},t)$ *and* $\mathtt{Under}(\mathbf{x},t)$ *returned by* ***OURS*** *satisfy that: for all* $(\mathbf{x},t) \in \mathtt{Bound(K)} \times [0,T]$, $-\epsilon \leq \mathtt{Over}(\mathbf{x},t) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq 0$ *and* $0 \leq \mathtt{Under}(\mathbf{x},t) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq \epsilon$.

Proof. We inductively prove this theorem for $\mathtt{Interval}(i)$ since $\mathtt{Bound(K)} \subseteq \mathtt{Interval}(i)$. For simplicity, denote $\mathtt{Interval}(i) \times [(i-1)\Delta T, i\Delta T]$ as $Range(i)$.

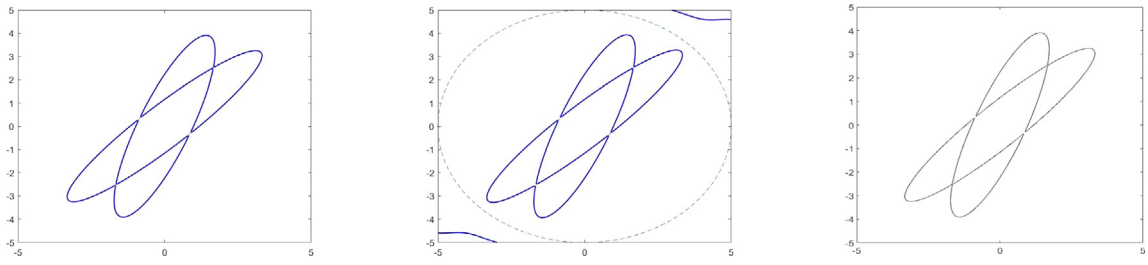
In the first segment, due to Remark 2, $-\Delta\epsilon \leq \mathtt{Over}^1(\mathbf{x},t) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq 0$ and $0 \leq \mathtt{Under}^1(\mathbf{x},t) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq \Delta\epsilon$, $\forall(\mathbf{x},t) \in Range(1)$.

Assume that in the ith segment, we have $-i\Delta\epsilon \leq \mathtt{Over}^i(\mathbf{x},t-(i-1)\Delta T) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq 0$ and $0 \leq \mathtt{Under}^i(\mathbf{x},t-(i-1)\Delta T) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq i\Delta\epsilon$, $\forall(\mathbf{x},t) \in Range(i)$. From Algorithm 2, $\mathtt{Interval}(i)$ is an enclosure of the backward reachable set and thus $\mathtt{Interval}(i+1) \subseteq \mathtt{Interval}(i) \subseteq \mathbf{Reach}^{\Delta T}_{\mathbf{f},\mathtt{Interval}(i)}$. Hence, due to the assumption, $Evo_{\mathtt{f},\mathtt{Over}^i(\mathbf{x},\Delta T)}(\mathbf{x},t-i\Delta T) - Evo_{\mathtt{f},g}(\mathbf{x},t) \leq 0$ and $0 \leq Evo_{\mathtt{f},\mathtt{Under}^i(\mathbf{x},\Delta T)}(\mathbf{x},t-i\Delta T) - Evo_{\mathtt{f},g}(\mathbf{x},t), \forall(\mathbf{x},t) \in Range(i+1)$.

Let us consider the $(i+1)$th segment. Due to Remark 2, for all $(\mathbf{x},t) \in Range(i+1)$, $\mathtt{Over}^{i+1}(\mathbf{x},t-i\Delta T) - Evo_{\mathtt{f},\mathtt{Over}^i(\mathbf{x},\Delta T)}(\mathbf{x},t-i\Delta T) \leq 0$ and $0 \leq \mathtt{Under}^{i+1}(\mathbf{x},t-i\Delta T) - Evo_{\mathtt{f},\mathtt{Under}^i(\mathbf{x},\Delta T)}(\mathbf{x},t-i\Delta T)$. Thus, combining with the result obtained in the above paragraph, we have that $\mathtt{Over}^{i+1}(\mathbf{x},t-i\Delta T) \leq Evo_{\mathbf{f},g}(\mathbf{x},t) \leq \mathtt{Under}^{i+1}(\mathbf{x},t-i\Delta T)$, $\forall(\mathbf{x},t) \in Range(i+1)$. Further, from Theorem 2, we have that $-(i+1)\Delta\epsilon \leq \mathtt{Over}^{i+1}(\mathbf{x},t) - \mathtt{Under}^{i+1}(\mathbf{x},t) \leq 0$, $\forall(\mathbf{x},t) \in Range(i+1)$. Thus, we can infer that for all $(\mathbf{x},t) \in Range(i+1)$, $-(i+1)\Delta\epsilon \leq \mathtt{Over}^{i+1}(\mathbf{x},t-i\Delta T) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq 0$ and $0 \leq \mathtt{Under}^{i+1}(\mathbf{x},t-i\Delta T) - Evo_{\mathbf{f},g}(\mathbf{x},t) \leq (i+1)\Delta\epsilon$, which completes the proof. □

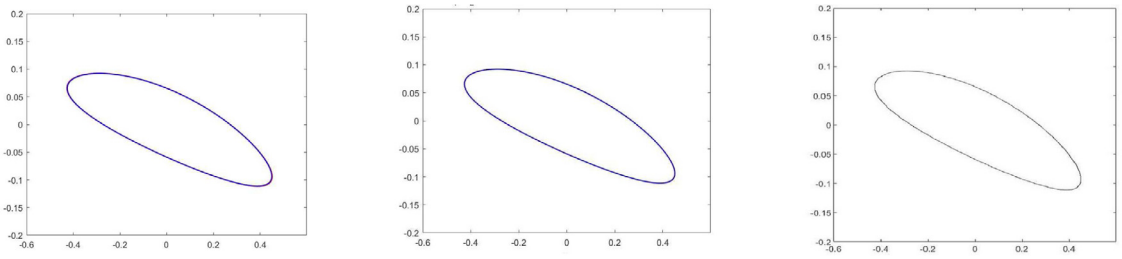
4. Comparisons with examples

In this section, we will show the performance of **OURS** by comparing it with the method in [24] and the **ToolboxLS** from [34] with seven examples. Especially, Example 1 deals with a disconnected set,



(a) **OURS** (b) Method in [22] (c) **ToolboxLS**

Fig. 3. Results of Example 1 at $t = 1$.



(a) **OURS** (b) Method in [22] (c) **ToolboxLS**

Fig. 4. Results of Example 2 at $t = 1$.

Example 4 deals with a 2-dimensional non-polynomial system, and Example 7 deals with a 8-dimensional non-polynomial systems. Some data of the results of these seven examples are listed in Table 1, where we use '-' to represent all unavailable data, for example, we cannot determine the highest degree of variables in examples 4 and 7 since the outputs contain trigonometric functions. Moreover, we also changed the time interval and designated precision for **OURS** and more data are listed in Table 2 with discussions. Note that we have been authorized by the authors of [24] to run their source code. Moreover, all examples were performed on a Laptop with 1.8 GHz Intel Core i7 (4 cores) and 8 Gb of RAM.

Example 1. We consider the running example with $\epsilon = 10^{-8}$. Additionally, we set degree = 14 for the method in [24] which makes the precision be 1.05×10^{-8}, and g.dx = 2/100 and accuracy = 'medium' for **ToolboxLS**. The results at $t = 1$ are shown in Fig. 3. Note that the dashed curve in Fig. 3(b) is the boundary of an appropriate compact set $\mathcal{Y} \subset \mathbb{R}^n$ (see Definition 3 in [24] for details).

Example 2. We consider the non-linear system from [4,24]

$$\begin{cases} \dot{x_1} = x_1 - 2x_2 \\ \dot{x_2} = x_1x_2 + 0.5x_2^2 \end{cases},$$

with the initial set $\{\mathbf{x} \mid x_1^2 + x_2^2 \le 0.01\}$. We want to compute the approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. We set $\epsilon = 10^{-4}$ for **OURS**, degree = 20 for the method in [24] which makes the precision be 1.31×10^{-4}, and g.dx = 1/100 and accuracy = 'medium' for **ToolboxLS**. The results at $t = 1$ are shown in Fig. 4.

Example 3. Consider the classical Van der Pol circuit system:

$$\begin{cases} \dot{x_1} = \dfrac{1}{C}x_2 \\ \dot{x_2} = \dfrac{1}{L}\cdot(-x_1 + \mu x_2 - x_2^3) \end{cases}.$$

We assume that $C = \mu = L = 1$, the initial set is $\{\mathbf{x} \mid (x_1 - 0.1)^2 + (x_2 - 0.1)^2 \le 0.01\}$. We want to compute the approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. We set $\epsilon = 10^{-4}$ for **OURS**, degree = 10 for the method in [24] which makes the precision be 2.03×10^{-4}, and g.dx = 1/100 and accuracy = 'medium' for **ToolboxLS**. The results at $t = 1$ are shown in Fig. 5.

Example 4. Consider the non-polynomial system from [4]:

$$\begin{cases} \dot{x_1} = 0.1\sin(x_2) \\ \dot{x_2} = 0.1x_2 - 0.02\sin(x_1)^2 \end{cases},$$

with the initial set $\{\mathbf{x} \mid x_1^2 + x_2^2 \le 0.01\}$. We want to compute the approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. We set $\epsilon = 10^{-4}$ for **OURS**, and g.dx = 1/100 and accuracy = 'medium' for **ToolboxLS**. The results at $t = 1$ are shown in Fig. 6. Note that the method in [24] cannot handle this example directly since it is a non-polynomial system.

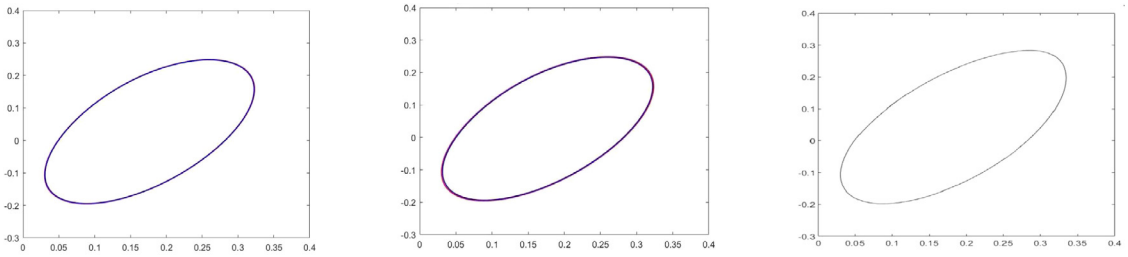
Example 5. We consider the 3-dimensional Lotka–Volterra system from [24]:

$$\begin{cases} \dot{x_1} = -x_1x_2 + x_1x_3 \\ \dot{x_2} = -x_2x_3 + x_2x_1 \\ \dot{x_3} = -x_3x_1 + x_3x_2 \end{cases},$$

with the initial set $\{\mathbf{x} \mid x_1^2 + x_2^2 + x_3^2 \le 0.01\}$. We want to compute the approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. We set $\epsilon = 10^{-4}$ for **OURS** and g.dx = 1/100 and accuracy = 'medium' for **ToolboxLS**. The result of **OURS** at $t = 1$ on the plane with $x_2 = 0$ is shown in Fig. 7(a) and the result of **ToolboxLS** at $t = 1$ is shown in Fig. 7(b). But the method in [24] cannot achieve the precision $\epsilon = 10^{-4}$.

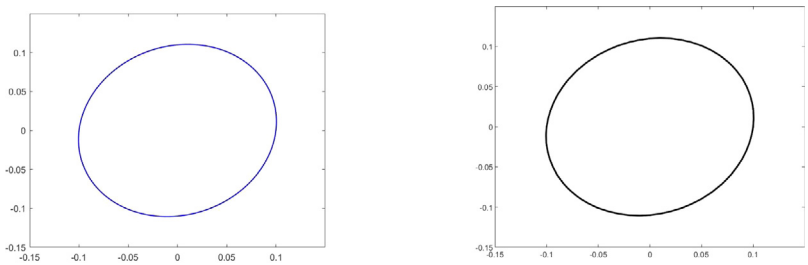
Example 6. We consider a 4-dimensional system from [4,35] which is the dynamics of an enzymatic reaction:

$$\begin{cases} \dot{x_1} = -k_f x_1x_2 + (k_b + k_m)x_3 \\ \dot{x_2} = -k_f x_1x_2 + k_b x_3 \\ \dot{x_3} = k_f x_1x_2 - (k_b + k_m)x_3 \\ \dot{x_4} = k_m x_3 \end{cases}.$$



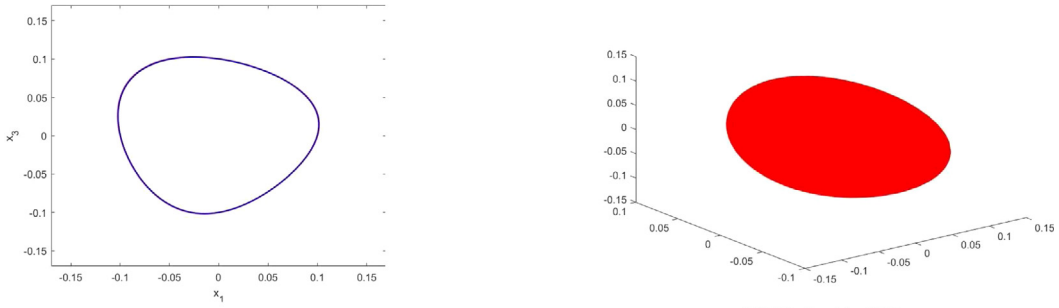
(a) **OURS** (b) Method in [22] (c) **ToolboxLS**

Fig. 5. Results of Example 3 at $t = 1$.



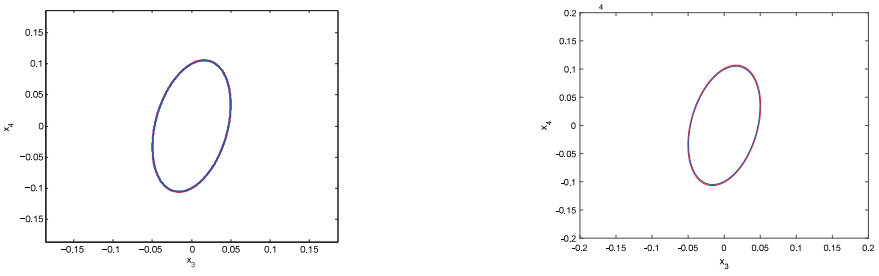
(a) **OURS**. (b) **ToolboxLS**.

Fig. 6. Results of Example 4 at $t = 1$.



(a) **OURS** on the plane with $x_2 = 0$. (b) Method in [22].

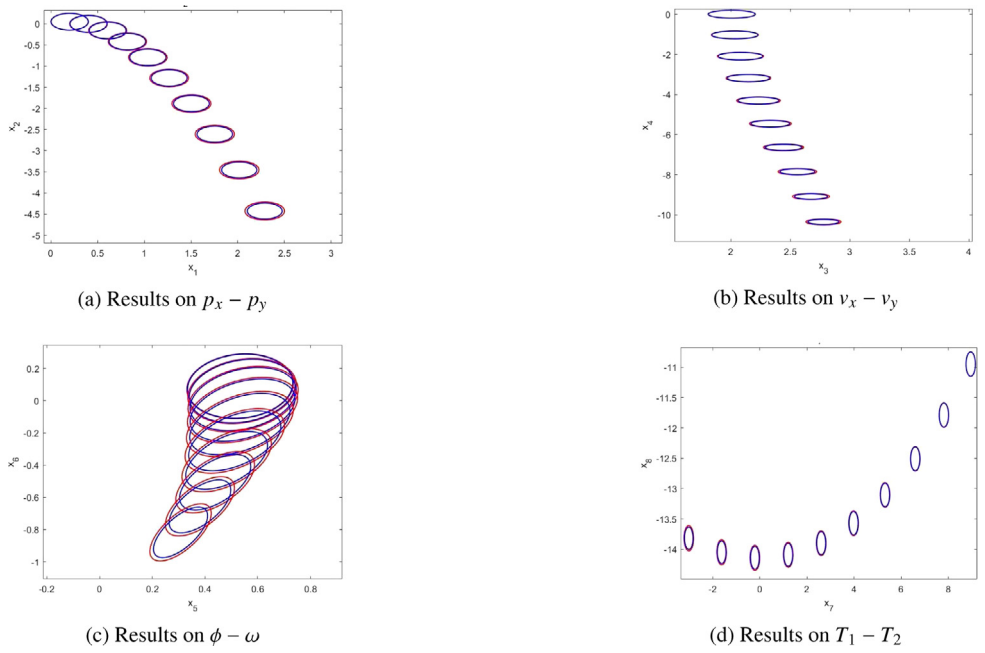
Fig. 7. Results of Example 5 at $t = 1$.



(a) **OURS**. (b) Method in [22].

Fig. 8. Results of Example 6 at $t = 1$ on the plane with $x_1 = x_2 = 0$.

We assume that $k_m = k_f = 0.5$, $k_b = 0.1$, the initial set is $\{\mathbf{x} \mid x_1^2 + x_2^2 + x_3^2 + x_4^2 \leq 0.01\}$. We want to compute the approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. We set $\epsilon = 10^{-2}$ for **OURS**, degree = 6 for the method in [24] which makes the precision be 2.2×10^{-2}, and g.dx = 1/100 and accuracy = 'medium' for **ToolboxLS**. The results of **OURS** and the method in [24] at $t = 1$ on the plane with $x_1 = x_2 = 0$ are shown



(a) Results on $p_x - p_y$

(b) Results on $v_x - v_y$

(c) Results on $\phi - \omega$

(d) Results on $T_1 - T_2$

Fig. 9. Results of Example 7 at time instants $t = 0.1, 0.2, \ldots, 1$.

in Fig. 8. Note that it is difficult to obtain the result of **ToolboxLS** at $t = 1$ on the same plane since the corresponding result has only discrete data.

Example 7. We consider an 8-dimensional non-polynomial system which is modified from [36] with certain inputs T_1 and T_2:

$$\begin{cases} \dot{p_x} = v_x \\ \dot{p_y} = v_y \\ \dot{v_x} = -\frac{1}{m} C_D^v v_x - \frac{T_1}{m} \sin\phi - \frac{T_2}{m} \sin\phi \\ \dot{v_y} = -\frac{1}{m}(mg + C_D^v v_y) + \frac{T_1}{m} \cos\phi + \frac{T_2}{m} \cos\phi \\ \dot{\phi} = \omega \\ \dot{\omega} = -\frac{1}{I_{yy}} C_D^\phi \omega + \frac{l}{I_{yy}} T_1 + \frac{l}{I_{yy}} T_2 \\ \dot{T_1} = \alpha T_2 \\ \dot{T_2} = -\alpha T_1 \end{cases},$$

where p_x, p_y, ϕ represent the horizontal, vertical, rotational positions of the quadrotor, v_x, v_y, ω represent the corresponding velocities, respectively, and T_1, T_2 are input thrusts exerted on either end of the quadrotor. For the coefficients in the system, we choose $C_D^v = 0.1$ for translational drag, $m = 5$ for the vehicle's mass, $g = 9.8$ for gravity, $C_D^\phi = 0.1$ for rotational drag, $I_{yy} = 10$ for the moment of inertia, $l = 0.5$ and $\alpha = 1$. The initial set $\mu(g)$ is represented by an 8-dimensional ball with radius 0.2, centering at $(0, 0, 2, 1, \frac{\pi}{6}, 0.1, 10, -10)$. We compute the over- and under-approximations of $Reach^t_{\mathbf{f},g}$ with $t \in [0, 1]$. Note that the method in [24] cannot operate directly; and because of the high dimension and large computation interval, **ToolboxLS** is not easy to deal with this example either.

Fig. 9(a) shows the projected results at time instants $t = 0.1, 0.2, \ldots, 1$ onto the corresponding $p_x - p_y$ planes defined by letting $(v_x, v_y, \phi, \omega, T_1, T_2) = (\Phi_2(t), \Phi_4(t), \Phi_5(t), \Phi_6(t), \Phi_7(t), \Phi_8(t))$, where $\Phi(t)$ is the simulated trajectory starting from the center $\mathbf{x}_0 = (0, 0, 2, 1, \frac{\pi}{6}, 0.1, 10, -10)^T$. Similarly, Figs. 9(b), 9(c), and 9(d) show the projected results at time instants $t = 0.1, 0.2, \ldots, 1$ onto the corresponding $v_x - v_y$, $\phi - \omega$, and $T_1 - T_2$ planes, respectively.

In Table 1, we list certain data of the obtained results for **OURS**/the method in [24]/**ToolboxLS**, i.e. the running times(RT), the maximal number of terms(NT) in outputs, the highest degree of all variables $deg_{\mathbf{x},t}$ and the degree of time variable deg_t. Moreover, after assigning $t = T$ for the outputs, we also list the highest degree of variables $deg_{\mathbf{x}}$ and the number of terms in the corresponding results. Considering that the results obtained by **ToolboxLS** are numerical data, it is meaningless to list them except the running times of **ToolboxLS**. Additionally, Examples 4 and 7 are non-polynomial and thus the method in [24] cannot directly handle them; and for Example 5, the method in [24] cannot achieve the given precision $\epsilon = 10^{-4}$.

Different from **ToolboxLS**, the outputs obtained by **OURS** and the method in [24] are forms of algebraic inequalities, which can be directly used to obtain the results at any time instant within the time interval and are more convenient for the next call. Moreover, for higher-dimensional systems (see Example 6), the results of **OURS** and the method in [24] can be more easily used to obtain the results at any cross section by directly operating on the obtained algebraic inequalities with variable assignments (see Fig. 8), while **ToolboxLS** has only discrete data. Additionally, according to Table 1, **ToolboxLS** requires more time for higher-dimensional systems (see Examples 6 and 7); and **OURS** has better time performance than the method in [24] except Exampe 3 since Example 3 is very sensitive to the size of the initial set and the length of the time horizon; moreover, compared with the method in [24], **OURS** can directly handle non-polynomial systems (see Examples 4 and 7).

We have also increased the time interval or changed the designated precision for these examples, and listed the corresponding data of results in Table 2. Since **OURS** need to increase time segmentation to deal with longer time intervals and increase the degree of t-expansion to achieve the higher precision requirement, these will definitely lead to an increase of running time and number of terms in the output of **OURS**. Note that the current version of **CORA** has an error of 'reachable set explosion' when handling high-order systems with long time interval, which affects the scalability of **OURS**. We have reported this problem to the authors and it may be addressed in the future version.

In general, for the low degree and trigonometric system, **OURS** shows good performance. The output of **OURS** has low degree and few terms, such that **OURS** can carry out more expansion to deal with higher dimensional systems, achieve higher precision requirements and extend to a longer time. Moreover, in future version, we will consider

to realize automatic segmentation of input time bound, and to design a better simplification program to further deal with the problem of excessive exponential growth caused by Taylor expansion.

5. Extension to systems with disturbances

Obviously, due to the use of time-splitting technique, **OURS** can deal with a class of switched systems, whose switching rules are time-dependent rather than state-dependent. In this section, we will additionally extend **OURS** to deal with systems with disturbances described by uncertain parameters, i.e., systems of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),\ \mathbf{x} \in \mathbb{D}, \mathbf{u} \in \prod_{i=1}^{m} [a_i, b_i], \tag{3}$$

where $\mathbf{x}$ is an n-dimensional vector, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is an analytic real function, $\mathbb{D} \subseteq \mathbb{R}^n$ is a domain, and $\mathbf{u}$ is an m-dimensional bounded disturbance parameter defined in $\mathbb{U} = \prod_{i=1}^{m} [a_i, b_i] \subseteq \mathbb{R}^m$. Letting $\mathbf{X}_0 \subseteq \mathbb{D}$ be an initial set, the *solution* of system (3) with given disturbance parameter $\mathbf{u}$ starting from the initial state $\mathbf{x}_0 \in \mathbf{X}_0$ is denoted as $\phi_{\mathbf{u}}(\mathbf{x}_0, t)$. Moreover, we use $\mathbb{T}_{\mathbf{x}_0,\mathbf{u}} = [T^-_{\mathbf{x}_0,\mathbf{u}}, T^+_{\mathbf{x}_0,\mathbf{u}}]$ to represent the maximal time interval of all $\phi_{\mathbf{u}}(\mathbf{x}_0, t)$.

For system (3) with given disturbance parameter $\mathbf{u}$, similar with Definition 1, we define the *reachable set* from the initial set $\mathbf{X}_0 = \mu(g)$ at instant $t \in \mathbb{T}_{\mathbf{x}_0,\mathbf{u}}$ as $Reach^{\mathbf{u},t}_{\mathbf{f},g} = \{\mathbf{x} \in \mathbb{D} \mid \exists \mathbf{x}_0 \in \mu(g), \mathbf{x} = \phi_{\mathbf{u}}(\mathbf{x}_0, t)\}$, and the *reachable set* within the time interval $[0, T] \subseteq \mathbb{T}_{\mathbf{x}_0,\mathbf{u}}$ is defined as $\mathbf{Reach}^{\mathbf{u},T}_{\mathbf{f},g} = \bigcup_{t\in[0,T]} Reach^{\mathbf{u},t}_{\mathbf{f},g}$.

Since $\mathbf{u}$ is a disturbance parameter which cannot be specified, following [25], we further define the maximal and minimal reachable sets, which can be used to verify the reach-avoid problem, as follows.

Definition 3. For given system (3), the maximal reachable set $Reach^{max\ t}_{\mathbf{f},g}$ and the minimal reachable set $Reach^{min\ t}_{\mathbf{f},g}$ from the initial set $\mu(g)$ at instant $t \in \mathbb{T}_{\mathbf{x}_0,\mathbf{u}}$ is defined as

$$Reach^{max\ t}_{\mathbf{f},g} = \bigcup_{\mathbf{u}\in\mathbb{U}} Reach^{\mathbf{u},t}_{\mathbf{f},g};\ Reach^{min\ t}_{\mathbf{f},g} = \bigcap_{\mathbf{u}\in\mathbb{U}} Reach^{\mathbf{u},t}_{\mathbf{f},g}.$$

Moreover, the maximal and minimal reachable sets from $\mu(g)$ within time interval $[0, T] \subseteq \mathbb{T}_{\mathbf{x}_0,\mathbf{u}}$ are respectively defined as:

$$\mathbf{Reach}^{max\ T}_{\mathbf{f},g} = \bigcup_{t\in[0,t]} Reach^{max\ t}_{\mathbf{f},g};\ \ \mathbf{Reach}^{min\ T}_{\mathbf{f},g} = \bigcup_{t\in[0,T]} Reach^{min\ t}_{\mathbf{f},g}.$$

Clearly, the computation of $Reach^{max\ t}_{\mathbf{f},g}$ and $Reach^{min\ t}_{\mathbf{f},g}$ can be converted to finding $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ for all $\mathbf{u} \in \mathbb{U}$. However, since $\mathbf{u}$ is a disturbance, it is inconvenient to directly compute $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ for obtaining $Reach^{max\ t}_{\mathbf{f},g}$ and $Reach^{min\ t}_{\mathbf{f},g}$. So, we consider the following $(m+n)$-dimensional ATDE:

$$\dot{\mathbf{y}} = F(\mathbf{y}),\ \mathbf{y} \in \mathbb{D}_y, \tag{4}$$

where $y = \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} \in \mathbb{R}^{n+m}$, $F(\mathbf{y}) = \begin{bmatrix} f(\mathbf{x},\mathbf{u})\\ \mathbf{0}\end{bmatrix}$, and $\mathbb{D}_y = \mathbb{D} \times \mathbb{U}$.

Based on Definition 2, after extending the domain for defining g from $\mathbb{R}^n$ to $\mathbb{R}^{n+m}$, the evolution function of system (4) with initial set $\mathbf{Y}_0 = \{\mathbf{y} = (\mathbf{x}, \mathbf{u}) : g(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}) \le 0\}$ (i.e., $\mathbf{Y}_0 = \mu(g)$) is $Evo_{\mathbf{F},g}(\mathbf{y}, t) = g(\phi(\mathbf{y}, -t))$. Especially, due to the form of F, i.e., $\dot{\mathbf{u}} \equiv \mathbf{0}$, for any initial state $\mathbf{y}_0 = \begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_0\end{bmatrix} \in \mathbf{Y}_0$, $\mathbf{y} = \phi(\mathbf{y}_0, t) = \begin{bmatrix}\phi_{\mathbf{u}_0}(\mathbf{x}_0, t)\\ \mathbf{u}_0\end{bmatrix}$, and thus we have $Evo_{\mathbf{F},g}(\phi(\mathbf{y}_0, t), t) = g(\phi_{\mathbf{u}_0}(\mathbf{x}, -t), \mathbf{u}_0) = g(\mathbf{x}_0)$. So, we immediately have that:

$$Reach^{\mathbf{u}_0,t}_{\mathbf{f},g} = \mu(g(\phi_{\mathbf{u}_0}(\mathbf{x}, -t))) = \{\mathbf{x} \mid Evo_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t) \le 0\}.$$

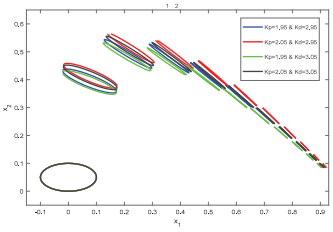
Thus, we naturally consider computing $Evo_{\mathbf{F},g}(\phi(\mathbf{y}_0, t), t)$ for $Reach^{\mathbf{u}_0,t}_{\mathbf{f},g}$. According to formula (2), for all $\mathbf{y} \in \mathbb{D}_y$, we have $Evo_{\mathbf{F},g}(\mathbf{y}, t) = \sum_{i=0}^{+\infty} \frac{\mathcal{M}^i_{\mathbf{F},g}(\mathbf{y})}{i!}(-t)^i$, where $\mathcal{M}^0_{\mathbf{F},g}(\mathbf{y}) = g(\mathbf{y})$ and $\mathcal{M}^{i+1}_{\mathbf{F},g}(\mathbf{y}) = \frac{\partial \mathcal{M}^i_{\mathbf{F},g}(\mathbf{y})}{\partial \mathbf{y}} \cdot \mathbf{F}(\mathbf{y})$. Similar to Section 2, we also denote $Evo^N_{\mathbf{F},g}(\mathbf{y}, t) = \sum_{i=0}^{N} \frac{\mathcal{M}^i_{\mathbf{F},g}(\mathbf{y})}{i!}(-t)^i$ as the Nth partial sum of $Evo_{\mathbf{F},g}(\mathbf{y}, t)$ and $Rem^N_{\mathbf{F},g}(\mathbf{y}, t)$ as the remainder for Nth partial sum of $Evo^N_{\mathbf{F},g}(\mathbf{y}, t)$. Following the theoretical analysis in Section 2.2, we can easily obtain that $Rem^N_{\mathbf{F},g}(\mathbf{y}, t) = -\int_0^t \frac{(r-t)^N}{N!} \mathcal{M}^{N+1}_{\mathbf{F},g}(\phi(\mathbf{y}, -r))dr$. Moreover, to obtain over- and under- approximations of $Evo_{\mathbf{F},g}(\mathbf{y}, t)$, we similarly should first specify a set S containing the reachable set $\mathbf{Reach}^T_{\mathbf{F},g}$. Since for system (4), $\phi(\mathbf{y}_0, t) = \begin{bmatrix}\phi_{\mathbf{u}_0}(\mathbf{x}_0, t)\\ \mathbf{u}_0\end{bmatrix}$ with $\mathbf{u}_0 \in \mathbb{U}$, if a compact set S_x is a bound of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$, then $S = S_x \times \mathbb{U}$ will be a bound of $\mathbf{Reach}^T_{\mathbf{F},g}$. Thus, according to Theorem 1, for two compact sets S_x and S' satisfying that $\mathbf{Reach}^{max\ T}_{\mathbf{f},g} \subseteq S_x$, $\mathbb{D}_y \supseteq S = S_x \times \mathbb{U} \supseteq \mathbf{Reach}^T_{\mathbf{F},g}$ and $S' \supseteq \mathbf{Reach}^T_{-\mathbf{F},S}$, if we can find constants L_{N+1} and U_{N+1} satisfying that $L_{N+1} \le \mathcal{M}^{N+1}_{\mathbf{F},g}(\mathbf{y}) \le U_{N+1},\ \forall \mathbf{y} \in S'$, for all $\mathbf{u}_0 \in \mathbb{U}$ and $t \in [0, T]$, then we have that:

1. if N is odd, $Evo^N_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t) + L_{N+1}\frac{t^{N+1}}{(N+1)!}$/ $Evo^N_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t) + U_{N+1}\frac{t^{N+1}}{(N+1)!}$ is an over-/ under-approximation of $Evo_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t)$ over S_x, respectively ;
2. if N is even, $Evo^N_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t) - U_{N+1}\frac{t^{N+1}}{(N+1)!}$/ $Evo^N_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t) - L_{N+1}\frac{t^{N+1}}{(N+1)!}$ is an over-/ under-approximation of $Evo_{\mathbf{F},g}((\mathbf{x}, \mathbf{u}_0), t)$ over S_x, respectively;
3. all precisions for above approximations are bounded by $(U_{N+1} - L_{N+1})\frac{t^{N+1}}{(N+1)!}$.

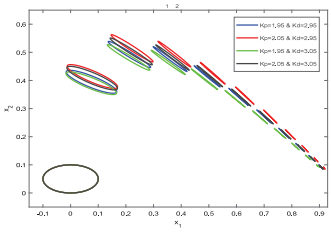
Clearly, according to Section 3.2, **OURS** can easily deal with system (4) and obtain the output $Over(\mathbf{y}, t)$, $Under(\mathbf{y}, t)$, and $\mathtt{Bound}$. Specifically, since $\dot{\mathbf{u}} = \mathbf{0}$, if we set the input $\mathtt{S} = \mathtt{S}_x \times \mathbb{U}$, where $\mathtt{S}_x$ is a box containing $\mu(g)$, then Algorithm 2 will output the $\mathtt{Bound}$, which has the form of $\mathtt{Bound}(i) = \mathtt{Bound}_x(i) \times \mathbb{U}$, where $\mathtt{Bound}_x(i)$ satisfies that $\mathtt{Bound}_x(i) \supseteq \mathbf{Reach}^{max\ [(i-1)\Delta T, i\Delta T]}_{f,g}$; moreover, since for each segment, $Over^i(\mathbf{y}, t)$ and $Under^i(\mathbf{y}, t)$ are over- and under- approximations of $Evo_{\mathbf{F},g_y}(\mathbf{y}, t)$ over $\mathtt{Bound}(i)$, then for specifically given $\mathbf{u}_0 \in \mathbb{U}$, $Over^i(\mathbf{x}, \mathbf{u}_0, t)$ and $Under^i(\mathbf{x}, \mathbf{u}_0, t)$ are naturally over- and under-approximations of $Evo_{\mathbf{F},g_u}((\mathbf{x}, \mathbf{u}_0), t)$ over $\mathtt{Bound}_x(i)$ with precisions all bounded by ϵ, respectively. As a result, according to Definition 3, **OURS** can produce the over-approximation of maximal reachable set, the under-approximation of maximal reachable set, the over-approximation of minimal reachable set, and the under-approximation of minimal reachable set of system (3) over $\mathtt{Bound}_x$ as $\bigcup_{\mathbf{u}_0\in\mathbb{U}} \mu(Over(\mathbf{x}, \mathbf{u}_0, t))$, $\bigcup_{\mathbf{u}_0\in\mathbb{U}} \mu(Under(\mathbf{x}, \mathbf{u}_0, t))$, $\bigcap_{\mathbf{u}_0\in\mathbb{U}} \mu(Over(\mathbf{x}, \mathbf{u}_0, t))$, and $\bigcap_{\mathbf{u}_0\in\mathbb{U}} \mu(Under(\mathbf{x}, \mathbf{u}_0, t))$, respectively.

Next, the performance of **OURS** in dealing with parametric uncertainties is shown by two examples with comparisons and discussions. For these two examples, we set $\epsilon = 10^{-2}$ as our precision, and illustrate the obtained over- and under-approximations of reachable sets for specific parameters with figures, from which the information of over- and under- approximations of maximal and minimal reachable sets can be intuitively imaged. Moreover, for these two examples, we compare our method with **CORA** and **Flow*** on the precision for over-approximations of maximal reachable sets; especially, for Example 8, which has also been used in [37], we additionally compare our method with the method in [37] for both over- and under- approximations of maximal and minimal reachable sets. The results, given by **CORA**, **Flow***, and the method in [37], are also listed. Therein, we set options.timeStep = 0.05, options.zonotopeOrder = 10 and options.taylorTerms = 5 for **CORA**, adaptive orders $\{min4, max16\}$, remainder estimation 1e−4 and fixed steps 0.001 for **Flow*** and sampling-time = 0.02, order = 3 for the method in [37]. Note that, we do not compare our method with **CORA** and **Flow*** for under-approximations of minimal reachable sets since it is difficult to adapt **CORA**-2021 and **Flow***-2.1.0 for our systems; and we do not compare our method with the method in [37] for Example 8 since it is also hard to adapt the method in [37] for it.

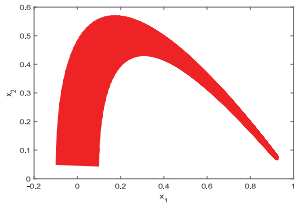
Example 8. Consider a PD-controller [37], controlling position x and velocity v of a car by adjusting its acceleration depending on the current



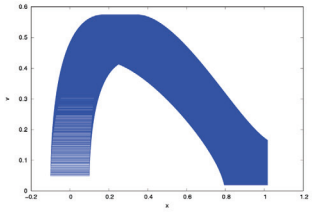
(a) Over approximations of $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ for specific $\mathbf{u}$ by **OURS**



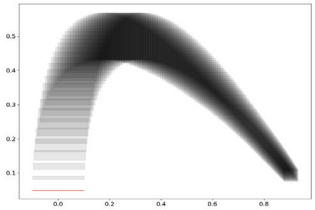
(b) Under approximations of $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ for specific $\mathbf{u}$ by **OURS**



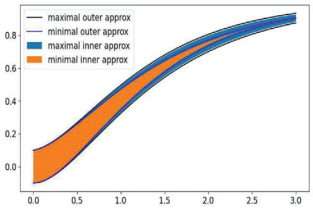
(c) Over-approx. of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$ by **CORA**



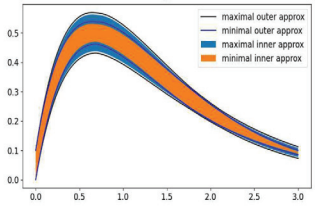
(d) Over-approx. of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$ by **Flow***



(e) Over-approx. of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$ by [35]

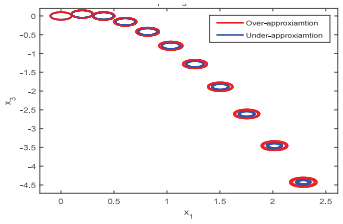


(f) Evolutions of $x(t)$ w.r.t t by [35]

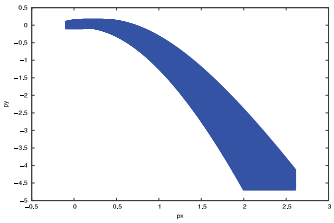


(g) Evolutions of $v(t)$ w.r.t t by [35]

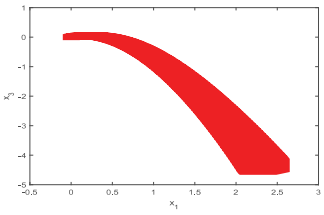
Fig. 10. Results of Example 8. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Approx. of $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ for specific $\mathbf{u}$ by **OURS**



(b) Over-approx. of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$ by **Flow***



(c) Over-approx. of $\mathbf{Reach}^{max\ T}_{\mathbf{f},g}$ by **CORA**

Fig. 11. Results of Example 9.

distance to a reference position p_r:

$$\begin{bmatrix}\dot{x}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} v \\ -K_p(x-p_r) - K_d v\end{bmatrix}.$$

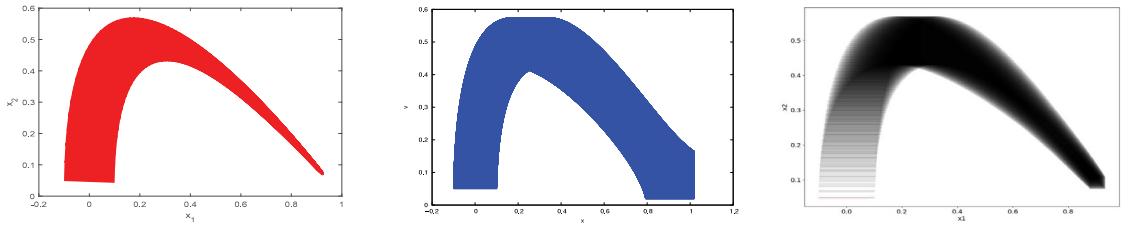
We set the initial set as $\mathbf{X}_0 = \{(x,v) \mid x^2 + (2v-0.1)^2 - 0.01 \le 0\}$ and $p_r = 1$, and assume that $\mathbf{u} = (K_p, K_d) \in [1.95, 2.05] \times [2.95, 3.05]$. We want to compute the over- and under-approximations of reachable sets with $t \in [0,3]$. The obtained over- and under-approximation results of reachable sets corresponding to $\mathbf{u} = (2 \pm 0.05, 3 \pm 0.05)$ at $t = 0, 0.3, \ldots, 3$ are shown in Fig. 10(a) and 10(b) respectively. Note that, in Fig. 10(a), the intersection of over-approximations of $Reach^{\mathbf{u},t}_{\mathbf{f},g}$ at $t = 0.6$ with $\mathbf{u} = (2.05, 2.95)$ (i.e.,red line) and $\mathbf{u} = (1.95, 3.05)$ (i.e., green line) is empty, and thus the minimal reachable set is an empty set.

We also list the results for over-approximations of maximal reachable sets with the initial set $\{(x,v) : x \in [-0.1, 0.1], v = 0.05\}$ obtained by **CORA**-2021, **Flow***-2.1.0, and the method in [37] in Figs. 10(c), 10(d), and 10(e), respectively. From Figs. 10(a), 10(b), 10(c), 10(d), and 10(e), we can easily see that our method is slightly better than **CORA**, **CORA** is distinguishingly better than the method in [37], and the method in [37] is distinguishingly better than **Flow***. Note that we also use $[-0.1, 0.1] \times [0, 0.1]$ as initial for the method in [37] here and show the results of over- and under- approximations of maximal and minimal reachable sets in Figs. 10(f) and 10(g), which show that the results of our method and the method in [37] are comparable.

Example 9. We modify Example 7 with uncertain disturbances D_x, D_y and D_ϕ:

$$\begin{bmatrix}\dot{p_x}\\ \dot{v_x}\\ \dot{p_y}\\ \dot{v_y}\\ \dot{\phi}\\ \dot{\omega}\\ \dot{T_1}\\ \dot{T_2}\end{bmatrix} = \begin{bmatrix} v_x \\ -\frac{C_D^v v_x}{m} \\ v_y \\ \frac{-mg - C_D^v v_y}{m} \\ \omega \\ -\frac{C_D^\phi \omega}{I_{yy}} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ D_x \\ 0 \\ D_y \\ 0 \\ D_\phi \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{-\sin\phi}{m} & \frac{-\sin\phi}{m} \\ 0 & 0 \\ \frac{\cos\phi}{m} & \frac{\cos\phi}{m} \\ 0 & 0 \\ \frac{l}{I_{yy}} & \frac{l}{I_{yy}} \\ 0 & \alpha \\ -\alpha & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix},$$

where we set the coefficients in the system be the same as Example 7, i.e. $C_D^v = 0.1, m = 5, g = 9.8, C_D^\phi = 0.1, l = 0.5, I_{yy} = 10$, and $\alpha = 1$. We set the initial set as $\mathbf{X}_0 = \{\mathbf{x} = (p_x, v_x, \ldots, T_1, T_2) \mid$



(a) Over-approx. of $\mathbf{Reach}_{\mathbf{f},g}^{max\ T}$ by **CORA** (b) Over-approx. of $\mathbf{Reach}_{\mathbf{f},g}^{max\ T}$ by **Flow*** (c) Over-approx. of $\mathbf{Reach}_{\mathbf{f},g}^{max\ T}$ by [35]

Fig. 12. Results of Example 8 with Higher Parameters Setting.

Table 3
Parameters setting for **CORA**.

CORA	options.timeStep 0.05/0.01/0.005/0.001	options.zonotopeOrder 10/30/50	options.taylorTerms 5/50/100

Table 4
Parameters setting for **Flow***.

Flow*	Fixed steps 0.001/0.0005	Adaptive orders 16/32/64

Table 5
Parameters setting for method in [37].

Method in [37]	Sampling-time 0.02/0.01/0.005	Order 3/30/50

$\|\mathbf{x} - (0,2,0,1,\frac{\pi}{6},0.1,10,-10)^T\|_2 \leq 0.1\}$ and let $\mathbf{u} = (D_x, D_y, D_\phi) \in [-0.02, 0.02] \times [-0.02, 0.02] \times [-0.02, 0.02]$. We want to compute the approximations of reachable sets with $t \in [0,1]$. Fig. 11(a) shows the obtained over- and under- approximations of reachable sets corresponding to $\mathbf{u} = (\pm 0.02, \pm 0.02, \pm 0.02)$ at $t = 0, 0.1, \ldots, 1$ onto the corresponding $p_x - p_y$ planes defined by letting $(v_x, v_y, \phi, \omega, T_1, T_2)$ equal to the simulated trajectory with $(D_x, D_y, D_\phi) = (0,0,0)$ starting from $\mathbf{x}_0 = (0,2,0,1,\frac{\pi}{6},0.1,10,-10)^T$.

The results given by **Flow*** and **CORA** with the smallest box containing $\mathbf{X}_0$ are shown in Fig. 11(b) and 11(c). In Fig. 11(a), $p_x(1) \in [2, 2.5]$, which shows that our method has higher precision than **CORA** and **Flow***.

Moreover, we also experimented with all the combinations of parameter settings listed in Tables 3 and 4 for **CORA** and **Flow*** for Examples 8 and 9, and Table 5 for the method in [37] for Example 8. However, the results did not change significantly. To intuitively understand this, we show the results of Example 8 with options.timeStep = 0.001, options.zonotopeOrder = 50 and options.taylorTerms = 100 for **CORA**, fixed steps = 0.0005 and adaptive orders = 64 for **Flow***, and sampling-time = 0.005 and order = 50 for the method in [37] in Fig. 12. Clearly, from Figs. 10, 11 and 12, we can see that to some extent, the method in [37] has higher precision than **Flow***, **CORA** has higher precision than the method in [37], and our method has higher precision than **CORA**. Intuitively, this is because **OURS** does not directly use the interval Taylor model but uses a finite truncation of the exact Taylor expansion of evolution functions w.r.t. t and the lower/upper bound of the corresponding remainder, which is not an interval but a function of t obtained by interval arithmetics, as the explicit algebraic expressions for over-/under- approximation of evolution function.

6. Conclusion

In this paper, we have presented the **MATLAB** tool **OURS** to construct over- and under-approximations of evolution functions, leading to over- and under-approximations of reachable sets, via estimating the corresponding remainder of the partial sum of t-expansion with interval arithmetics. We have also confirmed the efficacy and promise of **OURS** by comparing to other methods with examples. Moreover, **OURS** has been extended to deal with a class of time-invariant differential equations with disturbances described by uncertain parameters and produce over- and under- approximations of both minimal and maximal reachable sets of the uncertain systems. The performance of **OURS** in dealing with parametric uncertainties has also been shown by examples with comparisons and discussions.

In the future, we will investigate switched and even hybrid systems. Moreover, it is also interesting to study the control synthesis problem based on the approximation of reachable set [38–40].

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.

Uncited references

[1], [23]

Acknowledgments

The authors would like to thank the three anonymous reviewers for their insightful suggestions and comments on the earlier SETTA-2021 submission. Especially, the authors deeply thank the two anonymous reviewers of the journal submission for their detailed comments, which greatly help the authors to improve the quality of the paper.

References

[1] R. Hu, M. Li, Z. She, *OURS*: OVer- and under-approximating reachable sets for analytic time-invariant differential equations, in: Qin S. Woodcock J. Zhang W (Ed.), SETTA 2021, in: Lecture Notes in Computer Science, vol. 13071, 2021, pp. 261–278.

[2] E. Plaku, L. Kavraki, M. Vardi, Hybrid systems: from verification to falsification by combining motion planning and discrete search, Form. Methods Syst. Des. 34 (2009) 157–182.

[3] E. Goubault, S. Putot, Inner and outer reachability for the verification of control systems, in: HSCC'19, 2019, pp. 11–22.

[4] M. Li, Z. She, Over- and under-approximations of reachable sets with series representations of evolution functions, IEEE Trans. Automat. Control 66 (3) (2021) 1414–1421.

[5] M. Li, P.N. Mosaad, M. Fränzle, Z. She, B. Xue, Safe over- and under-approximation of reachable sets for autonomous dynamical systems, in: FORMATS 2018, 2018, pp. 252–270.

[6] S. Ratschan, Z. She, Safety verification of hybrid systems by constraint propagation-based abstraction refinement, ACM Trans. Embedd. Comput. Syst. 6 (1) (2007) 8, 1–23.

[7] A.A. Kurzhanskiy, P. Varaiya, Ellipsoidal toolbox (ET), in: Proceedings of the 45th IEEE Conference on Decision and Control, (2006) pp. 1498–1503.

[8] A. Girard, C. Le Guernic, O. Maler, Efficient computation of reachable sets of linear time-invariant systems with inputs, in: HSCC'06, 2006, pp. 257–271.

[9] T. Dang, O. Maler, R. Testylier, Accurate hybridization of nonlinear systems, in: HSCC'10, 2010, pp. 11–20.

[10] M. Althoff, Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets, in: HSCC'13, 2013, pp. 173–182.

[11] E. Goubault, S. Putot, Robust under-approximations and application to reachability of non-linear control systems with disturbances, IEEE Control Syst. Lett. 4 (4) (2020) 928–933.
[12] G. Frehse, PHAVer: ALgorithmic verification of hybrid systems past HyTech, Int. J. Softw. Tools Technol. Transf. 10 (2008) 263–279.
[13] G. Frehse, C.L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, SpaceEx: Scalable verification of hybrid systems, in: CAV'11, in: LNCS, vol. 6806, 2011, pp. 379–395.
[14] T.C. Wang, S. Lall, M. West, Polynomial level-set method for polynomial system reachable set estimation, IEEE Trans. Automat. Control 58 (10) (2013) 2508–2521.
[15] I.M. Mitchell, A.M. Bayen, C.J. Tomlin, A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games, IEEE Trans. Automat. Control 50 (7) (2005) 947–957.
[16] I.M. Mitchell. The flexible, Extensible and efficient toolbox of level set methods, J. Sci. Comput. 35 (2) (2008) 300–329.
[17] M. Althoff, An introduction to CORA 2015, in: HSCC'15, 2015, pp. 120–151.
[18] N. Kochdumper, M. Althoff, Computing non-convex inner-approximations of reachable sets for nonlinear continuous systems, in: 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 2130–2137.
[19] S. Kaynama, M. Oishi, I.M. Mitchell, G.A. Dumont, The continual reachability set and its computation using maximal reachability techniques, in: 50th IEEE Conference on Decision and Control (CDC), 2011, pp. 6110–6115.
[20] S. Kaynama, J. Maidens, M. Oishi, I.M. Mitchell, G.A. Dumont, Computing the viability kernel using maximal reachable sets, in: HSCC'12, 2012, pp. 55–64.
[21] X. Chen, E. Ábrahám, S. Sankaranarayanan, Flow*: An analyzer for non-linear hybrid systems, in: CAV'13, in: LNCS, vol. 8044, 2013, pp. 258–263.
[23] [Online]. Available: https://flowstar.org/dowloads/.
[22] X. Chen, E. Ábrahám, S. Sankaranarayanan, Taylor model flowpipe construction for non-linear hybrid systems, in: RTSS 2012, 2012, pp. 183–192.
[24] B. Xue, M. Fränzle, N. Zhan, Under-approximating reach sets for polynomial continuous systems, in: HSCC'18, 2018, pp. 51–60.
[25] I.M. Mitchell, Comparing forward and backward reachability as tools for safety analysis, in: HSCC'07, 2007, pp. 428–443.
[26] C.J. Tomlin, J. Lygeros, S.S. Sastry, A game theoretic approach to controller design for hybrid systems, Proc. IEEE 88 (7) (2000) 949–970.
[27] S. Kaynama, M. Oishi, I.M. Mitchell, G.A. Dumont, The continual reachability set and its computation using maximal reachability techniques, in: IEEE Conference on Decision and Control (CDC), 2011, pp. 6110–6115.
[28] K. Margellos, J. Lygeros, Hamilton–Jacobi formulation for reach-avoid differential games, IEEE Trans. Automat. Control 56 (8) (2011) 1849–1861.
[29] B. Xue, M. Fränzle, N. Zhan, Inner-approximating reachable sets for polynomial systems with time-varying uncertainties, IEEE Trans. Automat. Control 65 (4) (2020) 1468–1483.
[30] M. Rungger, M. Zamani, Accurate reachability analysis of uncertain nonlinear systems, in: HSCC'18, 2018, pp. 61–70.
[31] S. Ratschan, Z. She, Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions, SIAM J. Control Optim. 48 (7) (2010) 4377–4394.
[32] N.S. Nedialkov, Implementing a rigorous ode solver through literate programming, in: Modeling, Design, and Simulation of Systems with Uncertainties Mathematical Engineering, Vol. 3, 2011, pp. 3–19.
[33] L. Granvilliers, F. Benhamou, RealPaver: AN interval solver using constraint satisfaction techniques, ACM Trans. Math. Softw. 32 (1) (2006) 138–156.
[34] [Online]. Available: http://www.cs.ubc.ca/~mitchell/ToolboxLS.
[35] A.A. Julius, G.J. Pappas, Trajectory based verification using local finite-time invariance, in: HSCC'09, 2009, pp. 223–236.
[36] M. Chen, S.L. Herbert, M.S. Vashishtha, S. Bansal, C.J. Tomlin, Decomposition of reachable sets and tubes for a class of nonlinear systems, IEEE Trans. Automat. Control 63 (11) (2018) 3675–3688.
[37] E. Goubault, S. Putot, Inner and outer reachability for the verification of control systems, in: HSCC'19, 2019, pp. 11–22.
[38] B. Luo, H. Wu, T. Huang, D. Liu, Reinforcement learning solution for HJB equation arising in constrained optimal control problem, Neural Netw. 71 (11) (2015) 150–158.
[39] Y. Li, J. Liu, ROCS: A Robustly complete control synthesis tool for nonlinear dynamical systems, in: HSCC'18, 2018, pp. 130–135.
[40] N. Kochdumper, F. Gruber, B. Schürmann, V. Gaßmann, M. Klischat, M. Althoff, AROC: A toolbox for automated reachset optimal controller synthesis, in: HSCC'21, 2021, pp. 1–6.

Ruiqi Hu was born in Beijing, China, in 1996. He received the B.Sc. degree in College of Mathematics from Sichuan University, Sichuan, China, in 2018. He is currently pursuing the Ph.D. degree in applied mathematics from Beihang University, Beijing, China. His research interests include reachable set computation for dynamical systems and control synthesis problem for avoidance-guaranteed reachability.

Zhikun She received the B.Sc. degree in Computational Mathematics and the Ph.D. degree in Mathematics from Peking University, Beijing, China, in 1999 and 2005, respectively. He is currently a Professor in the School of Mathematical Sciences, Beihang University, Beijing, China. From January 2004 to December 2006, he was a Postdoctoral Researcher in the MPI für Informatik. His research interests include theory, methodologies, and applications of safety verification and stability analysis of hybrid systems. He has been the Principal Investigator of more than ten research projects and published more than 80 research papers.

Dr. She received the first class of Natural Science Prize of the Ministry of Education of China in 2013 and the National Science Fund for Excellent Young Scholars of China in 2014.