# Under-Approximating Backward Reachable Sets by Semialgebraic Sets

Bai Xue[1] and Zhikun She[2] and Arvind Easwaran[3]

***Abstract*—Under-approximations of backward reachable sets play an important role in controller synthesis and trajectory analysis for constrained nonlinear dynamical systems, but there are few methods available to compute them. Given a nonlinear system, a target region of simply connected compact type and a time duration, we present a method using boundary analysis to compute an under-approximation of the backward reachable set. The under-approximation is represented as a semi-algebraic set, formed by what we term polynomial $\texttt{level} - \texttt{set}$ functions. The polynomial $\texttt{level} - \texttt{set}$ function is a semi-definite positive function with one real root, such that the interior and closure of a semi-algebraic set formed by it are both simply connected and have the same boundary. The function can be computed by solving a convex program, which is constructed based on sum-of-squares decomposition and linear interval inequalities. We test our method on several examples and compare them with existing methods. The results show that our method can obtain better estimations more efficiently in terms of time for these special examples.**

***Index Terms*—Under-approximation; Boundary Analysis; Convex Programming; Semi-algebraic Sets**

## I. Introduction

Reachability analysis, which involves constructing reachable sets, plays an important role not only in automatic verification and falsification of safety properties for continuous nonlinear and hybrid systems [1], [2], but also in the design and synthesis of safe controllers for these systems [3]. Due to the fact that the exact reachable set of nonlinear systems is very hard to compute, an approximation is often computed. Generally, the approximation can be classified into two major types: over-approximations and under-approximations. For over-approximations, tremendous attention has been given and significant advances have been reported in the literature. Examples include methods based on invariant generation ([4], [5], [6]), barrier certificates [7], abstracting the nonlinear dynamics into piecewise linear or polynomial dynamics ([8], [9], [10], [11]) and Taylor series expansions ([12], [13]). However, much less attention has been given to the problem of finding under-approximations, mainly because the problem is more difficult than the one of computing over-approximations [14]. In this paper, we focus on computing under-approximations of backward reachable sets of continuous nonlinear systems. It is a set such that all trajectories originating from it will enter a specified target region after a time duration.

Under-approximations of reachable sets are important to compute because they have many potential applications in engineering domains. For example, they can be used to prove attractive properties by checking if all the trajectories originating from them will stay in them forever and eventually enter some specified desired sets [15], [16], [17], [18]. They can also be used for falsification by checking if the under-approximation intersects the unsafe sets [19][1]. Recently, they have also been used for designing robust artificial pancreas. Set inversion via interval analysis algorithms has been applied to obtain an under-approximation of the feasible set of basal and bolus insulin dose that for a given meal plan, mathematically guarantees a postprandial response fulfilling the International Diabetes Federation guidelines [20]. It has been pointed out that under-approximations are very useful to quantify over-approximations in the design of an artificial pancreas system [21].

Motivated by the above applications, we in this paper propose a convex programming based approach for computing under-approximations of the simply connected compact type, under the assumption that the target region is a simply connected compact set. Notice that many sets that arise in practice can be closely approximated by semi-algebraic sets [22]. Therefore, we would like to represent the under-approximations by semi-algebraic sets. The basic idea for computing the under-approximation first involves computing an enclosure of the boundary of the backward reachable set. Then, we formulate the under-approximation problem as a convex programming problem based on the obtained enclosure. By solving the convex programming problem, an under-approximation is obtained. The contributions of this paper are summarized as follows:

1) We show how the boundary of the backward reachable set can be enclosed through the use of the given target region's boundary rather than the entire target region. That is, by solving a constraint satisfaction problem based on interval arithmetic [23], we first obtain an over-approximation of the target region's boundary; Then based on the obtained over-approximation and the topological property, we employ interval Taylor methods to compute an enclosure of the backward reachable set's boundary.
2) We show how a simply connected compact set can be

1. Carl von Ossietzky Universität Oldenburg, Germany(bai.xue@uni-oldenburg.de).

2. SKLSDE, LMIB and School of Mathematics and Systems Science, Beihang University, China(zhikun.she@buaa.edu.cn). The work of Zhikun She (co-first author) was supported by NSFC-11422111 and NSFC-11371047.

3. School of Computer Engineering, Nanyang Technological University(arvinde@ntu.edu.sg). The work of Arvind Easwaran have been funded in part by the NTU-NHG Ageing Research Grant (ARG/14015).

[1]If over-approximations intersect the unsafe sets, then we are not able to reach a conclusion that the system is unsafe.

under (over)-approximated by a semi-algebraic set. This is a very challenging issue since deciding the number of connected components for a semi-algebraic set is very hard in general [24]. Hence finding the right connected components which can closely approximate the given set becomes nontrivial, especially for cases where an under-approximation is required. A key feature of our approach is that it relies on what we term a polynomial $\texttt{level}-\texttt{set}$ function. The polynomial $\texttt{level}-\texttt{set}$ function is a semi-definite positive function with one root, such that the interior and closure of a semi-algebraic set formed by it are both simply connected and have the same boundary. The function can be computed by solving a convex program, which is constructed by combining sum-of-squares decomposition and linear interval inequality approaches based on an over-approximation of the given simply connected compact set's boundary.

3) We show how to obtain an under-approximation of the backward reachable set using a semi-algebraic set. Based on the enclosure of the backward reachable set's boundary and the approach for under-approximating a simply connected compact set, we compute an under-approximation of the backward reachable set.
4) Finally, we have implemented our approach using validated ordinary differential equation solver **VNODE-LP** [25] and semidefinite programming solver **CSDP** [26], and tested it on several examples. Comparing with [27] and [18], we show that our approach can obtain better estimations more efficiently in terms of time for these special examples.

### A. Related Work

Several techniques have been proposed for computing under-approximations of reachable sets for linear systems [14], [28], [10], [29], [30], [31]. However, there are few methods available to construct these under-approximations for nonlinear systems. One potential method is based on conservative linearization, e.g., [9]. The under-approximations in this case are represented by convex sets such as ellipsoids and polytopes. It is well known that reachable sets of nonlinear systems are in general far from being convex. Thus, a very pessimistic result is often obtained by the above approach. In contrast to such first order approximation methods, our method in this paper is based on higher order approximations and semi-algebraic sets, which are of a more general form than convex sets, and hence less conservative results can be produced. This is illustrated in the following simple example.

*Example 1:* Consider a model of an electromechanical oscillation of a synchronous machine:

$$\begin{cases} \dot{\delta} = \omega \\ \dot{\omega} = 0.2 - 0.7sin\delta - 0.05\omega \end{cases},$$

where $\texttt{TR} = \{(\delta,\omega) : \delta^2 + (\omega-3)^2 \leq 0.25\}$, and TR is short for 'Target Region'. We can use the Runge-Kutta numerical method to estimate the boundary of the backward reachable set of TR for the time duration $t = 3$ based on some boundary points of TR. The estimated boundary is represented by the red curve in Fig. 1, and it can be seen that the backward reachable set is non-convex. Thus the ellipsoidal under-approximation will be too conservative. However, our method in this paper can help obtain a less conservative non-ellipsoidal under-approximation, as shown in Fig. 1. Details about computing the non-ellipsoidal under-approximation can be found in Example 4 in Subsection III-B.

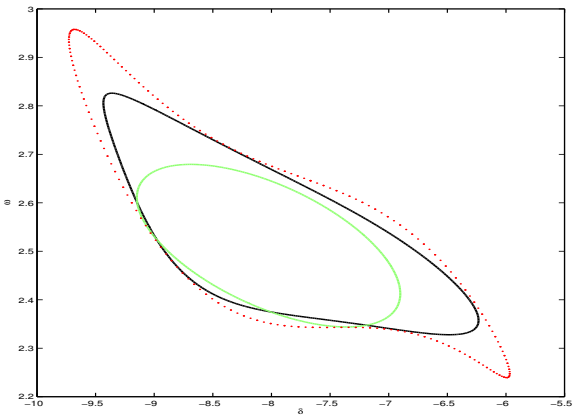


Fig. 1. Under-approximations for the time duration $t = 3$. (red points – the boundary of the backward reachable set based on Runge-Kutta numerical methods; green curve – the boundary of an ellipsoidal under-approximation obtained by the method in this paper; black curve – the boundary of a non-ellipsoidal under-approximation obtained by the method in this paper.)

Subdividing a state space into smaller intervals and then yielding some intervals such that all trajectories originating from them will enter the target region is another technique to compute an under-approximation, denoted by the union of the obtained intervals [27], [20]. This representation for Example 1 is illustrated in Fig. 2, in which the number of intervals is 129. On the other hand, our method is based on the boundary of the backward reachable set, and uses a semi-algebraic set rather than the union of intervals to represent the under-approximation, also illustrated in Fig. 2. This enables us to represent a complicated set in a simpler way since the space required for storing coefficients of a polynomial is much less than the space required for storing several intervals for some cases such as Example 1. More importantly, it is easier to verify whether a point is in the obtained under-approximation or not for these cases, since only basic arithmetic operations are needed.

A method was proposed to compute semi-algebraic inner approximations of reachable sets by solving sum-of-squares programs in [18]. However, this method is restricted to polynomial dynamical systems, our method deals with more general nonlinear systems (e.g., Example 1) than polynomial systems.

The structure of this paper is as follows. We introduce some basic definitions related to backward reachable sets in Subsection II-A, define polynomial $\texttt{level}-\texttt{set}$ functions, which play a crucial role in computing under-approximations, in Subsection II-B, and give a brief introduction to interval Taylor methods and linear interval inequalities in Subsec-

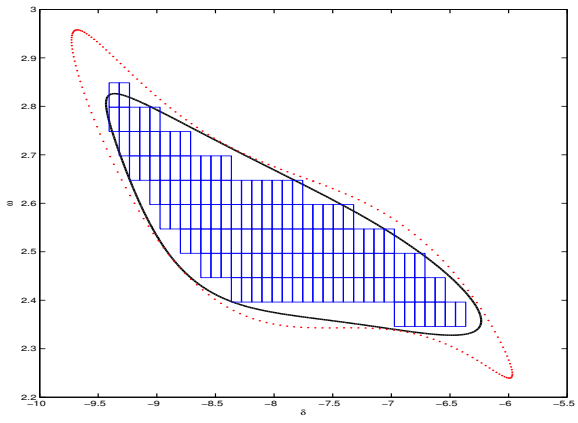

Fig. 2. Under-approximations for the time duration $t = 3$ for Example 1. (red points – the boundary of the backward reachable set obtained based on Runge-Kutta numerical methods; black curve – the boundary of a non-ellipsoidal under-approximation obtained by the method in this paper; blue curve – boundaries of intervals obtained by the method in [27].)

tions II-C and II-D respectively. Based on an approach to under-approximate simply connected compact sets with semi-algebraic sets formed by polynomial $\texttt{level}-\texttt{set}$ functions in Subsection III-A, we design an algorithm to compute under-approximations of backward reachable sets in Subsections III-B and analyze its computational complexity in Subsection III-C. Several numerical examples with a detailed discussion of our approach and comparisons with the method of Lhommeau et al. [27] as well as the method of Korda et al. [18] are provided in Subsections IV-A and IV-B respectively. Finally, we conclude our paper in Section V.

## II. Preliminaries

In this paper, we use the following notations: vectors are denoted by boldface letters (e.g., $\boldsymbol{x}$). $R[\boldsymbol{x}]$ is the set of all polynomials in variables $\boldsymbol{x}$ and $\sum_n$ is the set of sum-of-squares polynomials with $n$ variables, which can be expressed as sum of squares of polynomials. For a set $\Delta$, its complement, interior, closure and boundary are denoted by $\Delta^c$, $\Delta^\circ$, $\overline{\Delta}$ and $\partial\Delta$ respectively. Further, $\mathbb{U}(\boldsymbol{x};\epsilon) = \{\boldsymbol{y} : \|\boldsymbol{y}-\boldsymbol{x}\| < \epsilon, \epsilon > 0\}$ represents an $\epsilon-$neighbourhood of the vector $\boldsymbol{x}$.

### A. Backward Reachable Sets

Consider a nonlinear system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = (x_1, \cdots, x_n)' \in \mathbb{R}^n$, and $\boldsymbol{f}(\boldsymbol{x})$: $\mathbb{R}^n \to \mathbb{R}^n$ is $(p-1)$-time continuously differentiable and $p \geq 1$. We also assume $\boldsymbol{f}$ is locally Lipschitz continuous. Thus for a given set $\mathcal{X}$ that is a simply connected compact set, the existence and uniqueness of the trajectory with $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\boldsymbol{x}_0 \in \mathcal{X}$ will be assured over some time interval $[-\sigma_{\mathcal{X}}, \sigma_{\mathcal{X}}]$ with $\sigma_{\mathcal{X}} > 0$. Further, the trajectory of System (1) is defined to be $\phi(t;\boldsymbol{x}_0) = \boldsymbol{x}(t)$, where $\boldsymbol{x}(t)$ is the solution of System (1) satisfying the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. In addition, we define the backward and forward reachable sets of a simply connected compact set $\texttt{TR}$ for the time duration $T$ as follows.

*Definition 1:* Given System (1), a set $\texttt{TR}$ that is a simply connected compact set and a finite time duration $T \leq \sigma_{\texttt{TR}}$, the backward reachable set of $\texttt{TR}$ for the time duration $T$ is defined to be $\Omega_b(T;\texttt{TR},\boldsymbol{f}) = \{\boldsymbol{x}_0 | \phi(T;\boldsymbol{x}_0) \in \texttt{TR}\}$ and the forward reachable set of $\texttt{TR}$ for the time duration $T$ is defined to be $\Omega_f(T;\texttt{TR},\boldsymbol{f}) = \{\boldsymbol{x} | \boldsymbol{x} = \phi(T;\boldsymbol{x}_0)$ and $\boldsymbol{x}_0 \in \texttt{TR}\}$.

*Remark 1:* According to Definition 1, the map $\phi(t;\cdot)$ : $\texttt{TR} \subseteq \mathbb{R}^n \to \Omega_f(t;\texttt{TR},\boldsymbol{f})$(or, $\Omega_b(t;\texttt{TR},\boldsymbol{f}) \to \texttt{TR}$) is bijective and continuous for $t \in [0,T]$ under the Lipschitz condition of $\boldsymbol{f}$.

It is challenging to obtain these reachable sets for nonlinear systems since they generally do not have a closed-form solution. However, as mentioned in the introduction, it is sufficient to consider an under-approximation of the backward reachable set, denoted as $\texttt{UAB}$, for certain applications.

*Definition 2:* Given System (1), a set $\texttt{TR}$ that is a simply connected compact set and a finite time duration $T \leq \sigma_{\texttt{TR}}$, an $\texttt{UAB}$ of $\texttt{TR}$ for the time duration $T$ is a nonempty subset of $\Omega_b(T;\texttt{TR},\boldsymbol{f})$.

Obviously, all trajectories originating from $\texttt{UAB}$ will definitely enter $\texttt{TR}$ after a time duration $T$, although there may be trajectories not in $\texttt{UAB}$ that also enter $\texttt{TR}$ after the time duration $T$. This is illustrated in Fig. 3.

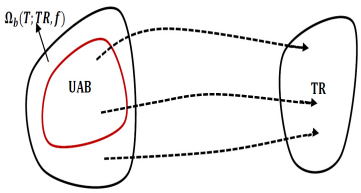


Fig. 3. An illustration for an $\texttt{UAB}$.

### B. Polynomial $\texttt{Level}-\texttt{Set}$ Functions

In this subsection we will introduce the concept of a polynomial $\texttt{level}-\texttt{set}$ function, which plays a crucial role in under-approximating backward reachable sets with semi-algebraic sets in this paper.

*Definition 3:* A continuous function $V(\boldsymbol{x})$ : $\mathbb{R}^n \to \mathbb{R}$ is called as a polynomial $\texttt{level}-\texttt{set}$ function if it satisfies the following properties:

1) $V(\boldsymbol{x}) \in R[\boldsymbol{x}]$;
2) $V(\boldsymbol{x}) \geq 0$ and $V(\boldsymbol{x})$ has exactly one root;
3) For $\forall \alpha > 0$, $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : V(\boldsymbol{x}) \leq \alpha\}$ is a compact set satisfying the condition that $\Omega$ and $\Omega^\circ = \{\boldsymbol{x} \in \mathbb{R}^n : V(\boldsymbol{x}) < \alpha\}$ are both simply connected sets with the same boundary $\{\boldsymbol{x} \in \mathbb{R}^n : V(\boldsymbol{x}) = \alpha\}$.

From the above definition, one may wonder how a polynomial $\texttt{level}-\texttt{set}$ function $V(\boldsymbol{x})$ can be computed efficiently since its conditions are very strong. Fortunately, we propose a method to compute it by building a connection between $\texttt{level}-\texttt{set}$ functions and traditional *Lyapunov* functions, as shown in the following lemma.

*Lemma 1:* Assume $V(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is a polynomial of degree $d$ and $\boldsymbol{x}_0$ is a globally asymptotically stable equilibrium point of system $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$. If $V(\boldsymbol{x})$ is a Lyapunov function with respect to $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$, that is,

1) $V(\boldsymbol{x}_0) = 0$ and $V(\boldsymbol{x}) > 0$ for $\forall \boldsymbol{x} \neq \boldsymbol{x}_0$;
2) $\frac{d}{dt}V(\boldsymbol{x}) = \frac{\partial V}{\partial \boldsymbol{x}}' \boldsymbol{g}(\boldsymbol{x}) < 0$ for $\boldsymbol{x} \neq \boldsymbol{x}_0$, and $\frac{d}{dt}V(\boldsymbol{x}_0) = 0$,

then $V(\boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function.

*Proof:* Since $V(\boldsymbol{x})$ is a polynomial of degree $d$, $lim_{\|\boldsymbol{x}\|\to+\infty} V(\boldsymbol{x}) = +\infty$. From known conditions, we obtain that $V(\boldsymbol{x})$ is a global Lyapunov function for the system $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$. Thus, $\{\boldsymbol{x} : V(\boldsymbol{x}) \leq \alpha\}$ for $\forall \alpha > 0$ is an estimated attraction domain of the equilibrium point $\boldsymbol{x}_0$ for the system $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$. The attraction domain is a set such that all trajectories originating from it will tend towards $\boldsymbol{x}_0$ as time progresses. We obtain that $\Omega$ and $\Omega^\circ$ are simply connected sets with the same boundary $\{x : V(x) = \alpha\}$ (e.g., [32], [33]). Therefore we can conclude that $V(\boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function. ■

According to Lemma 1, given an arbitrary system of the form $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$ with a globally asymptotically stable equilibrium point $\boldsymbol{x}_0$, if we can find a polynomial Lyapunov function $V(\boldsymbol{x})$ satisfying Lemma 1, then $V(\boldsymbol{x})$ is also a polynomial $\texttt{level} - \texttt{set}$ function. Under the assumption that $\boldsymbol{g}(\boldsymbol{x})$ is a polynomial vector field, a polynomial $\texttt{level} - \texttt{set}$ function can be efficiently found by solving a semidefinite program derived from sum-of-squares decomposition [34], [16]. It is worth noticing that $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$ is only for computing a polynomial $\texttt{level} - \texttt{set}$ function and is unrelated to System (1) under consideration in this paper.

### C. Interval Taylor Methods

For nonlinear systems involving initial states that are uncertain but can be represented by intervals, Interval Taylor methods can be used to determine a validated enclosure of all possible solutions to this initial value problem with uncertainty. In this paper, we will employ this technique to compute an enclosure of the backward reachable set's boundary. Its basic framework for performing computation is introduced as follows. For more details, please refer to [12], [35].

Given System (1), an initial interval $\boldsymbol{x}_0^I$ and a time duration $T_1$, the interval Taylor method can be used to compute an interval enclosing the set $\Omega_f(T_1; \boldsymbol{x}_0^I, \boldsymbol{f})$ as follows. Firstly, a time grid $0 = \tau_0 < \tau_1 < \ldots < \tau_N = T_1$ is adopted and for all $j \in \{0, \ldots, N-1\}$ and all $t \in [\tau_j, \tau_{j+1}]$, the Banach fixed point theorem and the Picard-Lindelöf operator are used to compute a prior enclosure $s_j$ such that $\cup_{t\in[\tau_j,\tau_{j+1}]}\Omega_f(t; \boldsymbol{x}_0^I, \boldsymbol{f}) \subseteq s_j$. Secondly, a tighter enclosure $\boldsymbol{x}_{\tau_{j+1}}^I$ of $\Omega_f(\tau_{j+1}; \boldsymbol{x}_0^I, \boldsymbol{f})$ is computed according to the interval Taylor expansion of $\boldsymbol{x}(t)$ at time instant $t = \tau_j$,

$$\boldsymbol{x}_{\tau_{j+1}}^I = \boldsymbol{x}_{\tau_j}^I + \sum_{i=1}^{p-1}(\tau_{j+1}-\tau_j)^i \boldsymbol{f}^{[i]}(\boldsymbol{x}_{\tau_j}^I) + (\tau_{j+1}-\tau_j)^p \boldsymbol{f}^{[p]}(s_j).$$

In above equation $\boldsymbol{x}_{\tau_j}^I$ is an already computed interval-enclosure of $\Omega_f(\tau_j; \boldsymbol{x}_0^I, \boldsymbol{f})$, and the coefficients $\boldsymbol{f}^{[i]}$ are the Taylor coefficients of the solution $\boldsymbol{x}(t)$ which can be computed either numerically by automatic differentiation or analytically via formal methods. Finally, $\boldsymbol{x}_{\tau_N}^I \supseteq \Omega_f(T_1; \boldsymbol{x}_0^I, \boldsymbol{f})$ is obtained, where $\tau_N = T_1$.

### D. Linear Interval Inequalities

Linear interval inequalities emerge in global optimization when linearization techniques are used and in mathematical programming under uncertainty [36]. In this paper, by employing an approach that uses linear interval inequalities to encode polynomial non-negativity over intervals by means of linear constraints, we will construct convex programs to find polynomial $\texttt{level} - \texttt{set}$ functions required in our approach.

A system of linear interval inequalities is formulated as follows:

$$A^I \boldsymbol{y} \leq b^I, \tag{2}$$

where $A^I = \{A : \underline{A} \leq A \leq \overline{A}\}$ (component-wise inequalities) is an $m \times n$ interval matrix and $b^I = \{b : \underline{b} \leq b \leq \overline{b}\}$ (component-wise inequalities) is an $m-$dimensional interval vector. A $\boldsymbol{y}_0$ is called a strong solution to System (2) if it satisfies $A\boldsymbol{y}_0 \leq b$ for each $A \in A^I$ and $b \in b^I$. We denote the set of all strong solutions of System (2) by $Y$, and $Y$ is given as follows.

*Theorem 1 ([37]):* $Y = \{\boldsymbol{y}_1 - \boldsymbol{y}_2 : \overline{A}\boldsymbol{y}_1 - \underline{A}\boldsymbol{y}_2 \leq \underline{b}, \boldsymbol{y}_1 \geq \boldsymbol{0}, \boldsymbol{y}_2 \geq \boldsymbol{0}\}$.

According to Theorem 1, we find that a strong solution to System (2) can be computed by solving a linear programming problem. Based on this, for a parametric polynomial of the form $V(\boldsymbol{x}, \boldsymbol{c}) = \sum_{\alpha\in\mathcal{M}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$, where $c_{\boldsymbol{\alpha}}$'s are parametric coefficients, a linear programming relaxation of the condition on the parametric coefficients making $V(\boldsymbol{x}, \boldsymbol{c})$ non-positive over an interval $\boldsymbol{x} \in I$, can be obtained in the following way.

1) For each monomial $\boldsymbol{x}^\alpha (\alpha \in \mathcal{M})$, we use interval arithmetic to obtain its lower and upper bounds $I^{\alpha-}$ and $I^{\alpha+}$ respectively over the interval $I$, and yield a linear interval inequality $\sum_{\alpha\in\mathcal{M}}[I^{\alpha-}, I^{\alpha+}]c_\alpha \leq 0$.
2) According to Theorem 1, by replacing each variable $c_\alpha$ with a difference of two variables $c_{\alpha 1}$ and $c_{\alpha 2}$, where $c_{\alpha 1} \geq 0$ and $c_{\alpha 2} \geq 0$, we can replace $[I^{\alpha-}, I^{\alpha+}]c_\alpha$ by $I^{\alpha+}c_{\alpha 1} - I^{\alpha-}c_{\alpha 2}$ and arrive at a linear inequality $\sum_{\alpha\in\mathcal{M}}[I^{\alpha+}c_{\alpha 1} - I^{\alpha-}c_{\alpha 2}] \leq 0$, denoted as $\psi[\boldsymbol{c}]$.

For convenience, we denote the above procedure as $\texttt{linear\_interval\_inequalities}(V(\boldsymbol{c}, \boldsymbol{x}), I)$.

If a solution $(c_{\alpha 1}, c_{\alpha 2})_{\alpha\in\mathcal{M}}$, where there exists at least an index $\alpha \in \mathcal{M}$ such that $c_{\alpha 1} - c_{\alpha 2} \neq 0$, can be found, the polynomial $V$ is obtained by substituting $c_\alpha$ with $c_{\alpha 1} - c_{\alpha 2}$. To illustrate the above procedure, we present a simple example with $x \in [1, 2]$. We wish to find a solution $(a, b, c)$ such that $ax^2 + bx + c \leq 0$ over the interval $x \in [1, 2]$. We relax $ax^2 + bx + c \leq 0$ to the linear interval inequality $[1, 4]a + [1, 2]b + [1, 1]c \leq [0, 0]$, which is equivalent to the following linear constraint according to Theorem 1.

$$\begin{cases} 4a_1 + 2b_1 + c_1 - (a_2 + b_2 + c_2) \leq 0 \\ a_i \geq 0, b_i \geq 0, c_i \geq 0, i = 1, 2 \end{cases}. \tag{3}$$

If a solution $(a_1, a_2, b_1, b_2, c_1, c_2)$ such that not all $a_1 - a_2$, $b_1 - b_2$ and $c_1 - c_2$ are equal to zero, is found, $(a_1 - a_2)x^2 + (b_1 - b_2)x + (c_1 - c_2) \leq 0$ holds over the interval $[1, 2]$. Clearly,

$(1,2,1,2,1,3)$ is a solution and $(a,b,c)=(-1,-1,-2)$ is what we want.

This above procedure has been used to find Lyapunov-like functions for nonlinear systems [15].

## III. Under-Approximations for Backward Reachable Sets

In this section, we present a novel approach to compute an UAB represented by a semi-algebraic set. Our approach is iterative with time step $h$, and is mainly composed of two parts in every iteration. The first part is to use Interval Taylor methods to compute a set of intervals, which encloses the boundary of the backward reachable set of TR for the time duration $h$. The second part is to compute an UAB of TR for the time duration $h$ based on the intervals obtained in the first part. In the second part, our approach relies on polynomial $\texttt{level}-\texttt{set}$ functions, which can be found by solving a convex programming problem. We use the yielded UAB as a target region for the computations in the step that follows. In such an iterative way, an UAB of the target region for the time duration $T$ is obtained.

In the following, before detailing our approach for computing an UAB of TR for the time duration $T$ in Subsection III-B, we will first present a detailed introduction on how to compute under-approximations of a simply connected compact set using some intervals covering the set's boundary and polynomial $\texttt{level}-\texttt{set}$ functions in Subsection III-A. The approach in Subsection III-A plays an important role in computing an UAB. Finally, we discuss the computational complexity of our method in Subsection III-C.

### A. Under-Approximations for Simply Connected Compact Sets

Semi-algebraic sets have many applications in statistics, control theory and robotics [38]. One of the underlying reasons is that many sets in $\mathbb{R}^n$ that arise in practice can be closely approximated by semi-algebraic sets [22]. In this subsection we will propose an approach to solve the following problem based on the given set's boundary and polynomial $\texttt{level}-\texttt{set}$ functions.

*Definition 4 (Under-Approximation (UA) Problem): Given a simply connected compact set $\Omega$, find a semi-algebraic set $\Omega_u$ which can closely under-approximate the set $\Omega$.*

However, as mentioned in the introduction, to under-approximate a given simply connected set with semi-algebraic sets is a very challenging problem generally. In order to solve the *UA* problem, we will resort to polynomial $\texttt{level}-\texttt{set}$ functions defined in Subsection II-B.

The following lemma shows that if the boundary of the set $\Omega$ is a subset of the closure of the complement of a semi-algebraic set $\Omega_u$, and the intersection of the set $\Omega$ and the semi-algebraic set $\Omega_u$ is not empty, where $\Omega_u$ is formed by a polynomial $\texttt{level}-\texttt{set}$ function, then the semi-algebraic set $\Omega_u$ is an under-approximation of the set $\Omega$.

*Lemma 2:* Assume $\Omega$ is a simply connected compact set and $V(\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function. If $\partial\Omega\subseteq\overline{\Omega_u^c}$ and $\Omega^\circ\cap\Omega_u^\circ\neq\emptyset$, where $\Omega_u=\{\boldsymbol{x}:V(\boldsymbol{x})\leq 1\}$[2], then $\Omega_u\subseteq\Omega$.

*Proof:* Since $\Omega_u=\{\boldsymbol{x}\in\mathbb{R}^n:V(\boldsymbol{x})\leq 1\}$, then $\Omega_u^\circ=\{\boldsymbol{x}\in\mathbb{R}^n:V(\boldsymbol{x})<1\}$ and $\Omega_u$ are compact simply connected sets with the same boundary $\{\boldsymbol{x}\in\mathbb{R}^n:V(\boldsymbol{x})=1\}$.

Assuming that there exists one point $y\in\Omega_u$ such that $y\notin\Omega$, we derive a contradiction. Since $V(\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function, there exists $\boldsymbol{z}_{\boldsymbol{y},\epsilon}\in\Omega_u^\circ\cap\mathbb{U}(\boldsymbol{y};\epsilon)$ for $\forall\epsilon>0$. And since $\boldsymbol{y}\notin\Omega$, there exists $\mathbb{U}(\boldsymbol{y};\delta)$ such that $\mathbb{U}(\boldsymbol{y};\delta)\nsubseteq\Omega$. Thus, there exists a point $\boldsymbol{z}_{\boldsymbol{y};\delta}\in\Omega_u^\circ\cap\mathbb{U}(\boldsymbol{y};\delta)$ and $\boldsymbol{z}_{\boldsymbol{y};\delta}\notin\Omega$. Let $\boldsymbol{y}_0\in\Omega^\circ\cap\Omega_u^\circ$. Since $\Omega_u^\circ$ is a simply connected set, we obtain that there exists a path $P$ in $\Omega_u^\circ$, connecting $\boldsymbol{y}_0$ and $\boldsymbol{z}_{\boldsymbol{y};\delta}$. So there must exist a $\boldsymbol{y}_1\in P$ such that $\boldsymbol{y}_1\in\partial\Omega$ and $\boldsymbol{y}_1\in\Omega_u^\circ$, contradicting the fact that $\partial\Omega\subseteq\overline{\Omega_u^c}$. Therefore, $\Omega_u\subseteq\Omega$. ■

From Lemma 2 and under the assumption that $\Omega_u=\{\boldsymbol{x}:V(\boldsymbol{c},\boldsymbol{x})\leq 1\}$, where $V(\boldsymbol{c},\boldsymbol{x})$ is of the form $\sum_{\alpha\in\mathcal{M}}c_\alpha\boldsymbol{x}^\alpha$ and $\boldsymbol{c}=(c_\alpha)_{\alpha\in\mathcal{M}}$ is a set of unknown coefficients, the *UA* problem can be formulated as follows: Determine a value for $\boldsymbol{c}$ such that $V(\boldsymbol{c},\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function, $\partial\Omega\subseteq\overline{\Omega_u^c}$ and $\Omega^\circ\cap\Omega_u^\circ\neq\emptyset$.

For $\partial\Omega\subseteq\overline{\Omega_u^c}$, we do not obtain the exact $\partial\Omega$ here, but instead compute an over-approximation of $\partial\Omega$ by solving a constraint satisfaction problem based on interval arithmetic (e.g. [23]). Obviously, if the over-approximation is a subset of $\overline{\Omega_u^c}$, $\partial\Omega\subseteq\overline{\Omega_u^c}$ holds. We denote the over-approximation by $\cup_{i=1}^N s_i$, where $s_i$ is of the interval form and $N\geq 1$ is an integer, and will use $\cup_{i=1}^N s_i$ instead of $\partial\Omega$ to perform computations. For $\Omega^\circ\cap\Omega_u^\circ\neq\emptyset$, we compute an arbitrary point $\boldsymbol{x}_0=(x_{0,1},\ldots,x_{0,n})'\in\Omega\setminus\cup_{i=1}^N s_i$. The point $\boldsymbol{x}_0$ can also be obtained by solving a constraint satisfaction problem. After obtaining $\cup_{i=1}^N s_i$ and $\boldsymbol{x}_0$, we determine $\boldsymbol{c}$ by taking the following steps:

1) For $i=1,\ldots,N$, ensure the non-positivity of $1-V(\boldsymbol{c},\boldsymbol{x})$ over each interval $s_i$ based on the procedure $\texttt{linear\_interval\_inequalities}(1-V(\boldsymbol{c},\boldsymbol{x}),s_i)$ in Subsection II-D and yield a linear constraint $\psi_i[\boldsymbol{c}]$ over $\boldsymbol{c}$;
2) Choose a polynomial dynamical system $\dot{\boldsymbol{x}}=\boldsymbol{g}(\boldsymbol{x})$ with the globally asymptotically stable equilibrium point $\boldsymbol{x}_0=(x_{0,1},\ldots,x_{0,n})'$, and then solve the following semidefinite programming problem involving the vari-

[2]We use a set of the form $\{\boldsymbol{x}:V(\boldsymbol{x})\leq 1\}$ rather than $\{\boldsymbol{x}:V'(\boldsymbol{x})\leq\alpha\}$ to represent the set $\Omega_u$, where $\alpha$ is an arbitrary but fixed positive number. The underlying reason is that a set of the form $\{\boldsymbol{x}:V'(\boldsymbol{x})\leq\alpha\}$ can be equivalently reformulated as a set of the form $\{\boldsymbol{x}:V(\boldsymbol{x})\leq 1\}$, where $V(\boldsymbol{x})=\frac{V'(\boldsymbol{x})}{\alpha}$.

able $\boldsymbol{c}$,

$$min \sum_{i=1}^{N}(V(\boldsymbol{c}, \boldsymbol{x}_{mid,i}) - 1) \tag{4}$$

subject to

$$V(\boldsymbol{c}, \boldsymbol{x}) - h_1 \sum_{i=1}^{n}(x_i - x_{0,i})^2 \in \sum\nolimits_n, \tag{5}$$

$$-\frac{\partial V(\boldsymbol{c}, \boldsymbol{x})}{\partial \boldsymbol{x}}^{'} \boldsymbol{g}(\boldsymbol{x}) - h_2 \sum_{i=1}^{n}(x_i - x_{0,i})^2 \in \sum\nolimits_n, \tag{6}$$

$$\psi_0[\boldsymbol{c}], \tag{7}$$

$$\wedge_{i=1}^{N} \psi_i[\boldsymbol{c}], \tag{8}$$

where $\boldsymbol{x}_{mid,i}$ is the center of the interval $s_i$ ($i = 1, \cdots, N$), the constraints (5) and (6) are semi-definite constraints derived from the sum-of-squares decomposition (e.g., [34]), the linear constraint (7) over the variable $\boldsymbol{c}$ is derived from the constraint $V(\boldsymbol{c}, \boldsymbol{x}_0) = 0$ and the linear constraint (8) is derived from the procedure $\texttt{linear\_interval\_inequalities}(1 - V(\boldsymbol{c}, \boldsymbol{x}), s_i)$ in Subsection II-D, and $h_1, h_2$ are two small positive numbers. The semidefinite program (4) $\sim$ (7) is a convex program problem, which can be solved by interior point methods efficiently (e.g., [26]).

For convenience, we denote the above process as $\texttt{Under} - \texttt{Approximation}(\cup_{i=1}^{N} s_i, \boldsymbol{x}_0)$. The basic idea behind the above procedure for computing $\Omega_u$ is illustrated in Fig. 4, and the obtained $\Omega_u$ is an under-approximation of $\Omega$, as stated in Proposition 1.

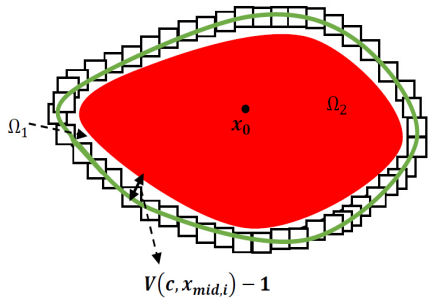


Fig. 4. An illustration for computing an under-approximation $\Omega_u$ of $\Omega$. (red region – an under-approximation $\Omega_u$; green curve – $\partial\Omega$; white boxes – an over-approximation of $\partial\Omega$.)

*Proposition 1:* Assume that $\boldsymbol{c}_0$ satisfies the convex program (4) $\sim$ (7), $\Omega_u$ is an under-approximation of the simply connected compact set $\Omega$, where $\Omega_u = \{\boldsymbol{x} : V(\boldsymbol{c}_0, \boldsymbol{x}) \leq 1\}$.

*Proof:* Since $\boldsymbol{c}_0$ satisfies $\psi_i[\boldsymbol{c}]$, we get $V(\boldsymbol{c}_0, \boldsymbol{x}) \geq 1$ for $\boldsymbol{x} \in \cup_{i=1}^{N} s_i$. Due to the fact that $\partial\Omega \subseteq \cup_{i=1}^{N} s_i$, $V(\boldsymbol{c}_0, \boldsymbol{x}) \geq 1$ for $\boldsymbol{x} \in \partial\Omega$.

According to Lemma 1, $V(\boldsymbol{c}_0, \boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function. Moreover, according to Definition 3, $\Omega_u$ and $\Omega_u^{\circ}$ are simply connected compact sets with the same boundary $\{\boldsymbol{x} : V(\boldsymbol{c}_0, \boldsymbol{x}) = 1\}$. So we get $\partial\Omega \subseteq \overline{\Omega_u^c}$. According to the fact that $\boldsymbol{x}_0 \in \Omega_u^{\circ} \cap \Omega^{\circ}$ and Lemma 2, we get $\Omega_u \subseteq \Omega$. ■

Thus we have solved the *UA* problem posted at the beginning of this subsection. This approach presented in this subsection will be employed to compute an UAB of TR in the subsection that follows.

*Remark 2:*
1) Since the computational complexity of semidefinite programs derived from sum-of-squares decomposition increases rapidly with the number of state variables and degrees of the sum-of-squares polynomials [39], we choose $\boldsymbol{g}(\boldsymbol{x}) = (-x_1 + x_{0,1}; \ldots; -x_n + x_{0,n})$, which is the simplest form satisfying Lemma 1. This is a trade-off between efficiency and accuracy.
2) In the semidefinite program (4) $\sim$ (7), since $V(\boldsymbol{c}, \boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function that will be computed, $\{\boldsymbol{x} : V(\boldsymbol{c}, \boldsymbol{x}) \leq 1\}$ is a simply connected compact set with the boundary $\{\boldsymbol{x} : V(\boldsymbol{c}, \boldsymbol{x}) = 1\}$. Since $\partial\Omega \subseteq \cup_{i=1}^{N} s_i$, the objective function $\sum_{i=1}^{N}(V(\boldsymbol{c}, \boldsymbol{x}_{mid,i}) - 1)$ can be viewed as a measure on how close the two sets are when $s_i (i = 1, \cdots, N)$ is small. This is illustrated in Fig. 4.

*Remark 3:* Since it is also very useful to evaluate the under-approximation by computing an over-approximation of the given set in some engineering fields (e.g., artificial pancreas [21]), we will compute an over-approximation of $\Omega$ based on the already estimated $V(\boldsymbol{c}_0, \boldsymbol{x})$. The computation is based on the following lemma, which shows that if the given simply connected compact set's boundary is a subset of a semi-algebraic set formed by a polynomial $\texttt{level} - \texttt{set}$ function, then the semi-algebraic set is an over-approximation of the given set.

*Lemma 3:* Assume $\Omega$ is a simply connected compact set and $V(\boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function. If $\partial\Omega \subseteq \Omega_o = \{\boldsymbol{x} : V(\boldsymbol{x}) \leq 1\}$, then $\Omega \subseteq \Omega_o$.

*Proof:* Since $\Omega$ is a compact set, there exists a point $\boldsymbol{x}_1 \in \Omega$ such that $V(\boldsymbol{x}_1) = \max_{\boldsymbol{x} \in \Omega} V(\boldsymbol{x})$. Clearly, $\Omega \subseteq \Omega_o$ is equivalent to $V(\boldsymbol{x}_1) \leq 1$. Thus it is enough to prove that $V(\boldsymbol{x}_1) \leq 1$.

Assuming that $V(\boldsymbol{x}_1) > 1$, we will derive a contradiction. Let $\Omega' = \{\boldsymbol{x} : V(\boldsymbol{x}) \leq V(\boldsymbol{x}_1)\}$. Since $V(\boldsymbol{x})$ is a polynomial $\texttt{level} - \texttt{set}$ function, $\partial\Omega' = \{\boldsymbol{x} : V(\boldsymbol{x}) = V(\boldsymbol{x}_1)\}$ and $\boldsymbol{x}_1 \in \partial\Omega'$. Thus, there exists $\boldsymbol{z}_\epsilon \in \mathbb{U}(\boldsymbol{x}_1; \epsilon)$ for $\forall \epsilon > 0$ such that $V(\boldsymbol{z}_\epsilon) > V(\boldsymbol{x}_1)$. Since $\partial\Omega \subseteq \Omega_o$ and $V(\boldsymbol{x}) \leq 1$ for $\boldsymbol{x} \in \partial\Omega$, then $\boldsymbol{x}_1 \in \Omega^{\circ}$. Thus there exist $\mathbb{U}(\boldsymbol{x}_1; \epsilon_1) \subseteq \Omega$ and $\boldsymbol{z}_{\epsilon_1} \in \mathbb{U}(\boldsymbol{x}_1; \epsilon_1)$ such that $V(\boldsymbol{z}_{\epsilon_1}) > V(\boldsymbol{x}_1)$, contradicting the fact that $V(\boldsymbol{x}_1) = \max_{\boldsymbol{x} \in \Omega} V(\boldsymbol{x})$. Therefore $V(\boldsymbol{x}_1) \leq 1$ and $\Omega \subseteq \Omega_o$. ■

According to Lemma 3 and the already estimated $\Omega_u$, an over-approximation of $\Omega$ can be computed by expanding $\Omega_u$. For each $s_i$ ($i = 1, \cdots, N$), we use interval arithmetic to find an upper bound $\overline{V}_i$ of $V(\boldsymbol{c}_0, \boldsymbol{x})$ over $\boldsymbol{x} \in s_i$, and then get the maximum MAX of the finite set $\{\overline{V}_i, i = 1, \ldots, N\}$. Then $\Omega_o = \{\boldsymbol{x} : V(\boldsymbol{c}_0, \boldsymbol{x}) \leq \texttt{MAX}\}$ is an over-approximation of the set $\Omega$. This can be assured by the following Proposition.

*Proposition 2:* $\Omega \subseteq \Omega_o$, where $\Omega_o = \{\boldsymbol{x} : V(\boldsymbol{c}_0, \boldsymbol{x}) \leq \texttt{MAX}\}$ and $\boldsymbol{c}_0$ satisfies the convex program (4) $\sim$ (7).

*Proof:* For $j = 1, \ldots, N$, we yield that $V(\boldsymbol{c}_0, \boldsymbol{x}) \leq \texttt{MAX}$ for $\boldsymbol{x} \in \cup_{i=1}^{N} s_i$. Since $\partial\Omega \subseteq \cup_{i=1}^{N} s_i$, $V(\boldsymbol{c}_0, \boldsymbol{x}) \leq \texttt{MAX}$ for $\boldsymbol{x} \in \partial\Omega$.

According to Lemma 3, $\Omega \subseteq \Omega_o$ since $V(\boldsymbol{c}_0, \boldsymbol{x})$ is a

polynomial $\texttt{level}-\texttt{set}$ function. ■

Thus, $\frac{\texttt{MAX}-1}{\texttt{MAX}}$ can be used to evaluate the obtained under-approximation. As the $\frac{\texttt{MAX}-1}{\texttt{MAX}}$ becomes smaller, the resulting under-approximation becomes less conservative.

The basic idea behind the above procedure for computing $\Omega_o$ is illustrated in Fig. 5.

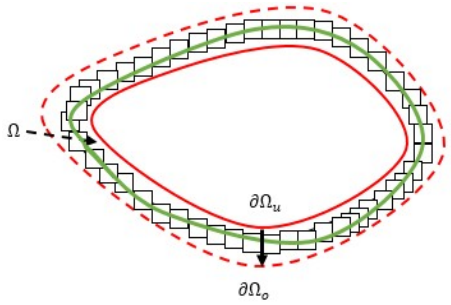


Fig. 5. An illustration for computing an over-approximation $\Omega_o$ of $\Omega$. (red dash curve – $\partial\Omega_o$; red solid curve – $\partial\Omega_u$; green solid curve – $\partial\Omega$; white boxes – an over-approximation of $\partial\Omega$.)

### B. Under-Approximating Backward Reachable Sets

In this subsection, we will employ the approach proposed in Subsection III-A to compute an $\texttt{UAB}$ of $\texttt{TR}$ for the time duration $T$. The $\texttt{UAB}$ is represented by a semi-algebraic set of the form $\{\boldsymbol{x} : V(\boldsymbol{x}) \leq 1\}$, where $V(\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function of degree $d \geq 2$.

The framework to compute an $\texttt{UAB}$ of $\texttt{TR}$ for the time duration $T$ involves the following steps:

1) a time grid $0 = t_0 < t_1 < \ldots < t_N = T$ is adopted with a fixed time step $h$;
2) starting with $U_{t_0} = \texttt{TR}$, we compute $U_{t_1}$, which is an $\texttt{UAB}$ of $\texttt{TR}$ for the time duration $t_1$;
3) starting from the $k^{th}$ $\texttt{UAB}$ $U_{t_k}$, we advance our approximation to an $\texttt{UAB}$ $U_{t_{k+1}}$;
4) $U_{t_N}$ is what we want to obtain.

Assume that we have already obtained a simply connected compact set $U_{t_k} = \{\boldsymbol{x} : V_{t_k}(\boldsymbol{x}) \leq 1\}$, where $V_{t_k}(\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function with $V_{t_k}(\boldsymbol{x}_k) = 0$, and $U_{t_k}$ is an $\texttt{UAB}$ of $\texttt{TR}$ for the time duration $t_k$. We now construct an $\texttt{UAB}$ for the time duration $t_{k+1}$ through the following steps:

a) Compute $\Omega_{t_{k+1}}$, which is the union of a collection of intervals, such that $\partial\Omega_b(h; U_{t_k}, \boldsymbol{f}) \subseteq \Omega_{t_{k+1}}$, as discussed below;
b) Use $\boldsymbol{x}_k$ to compute a point $\boldsymbol{x}_{k+1}$ such that $\boldsymbol{x}_{k+1} \in (\Omega_b(h; U_{t_k}, \boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$, as discussed below;
c) Determine a polynomial $\texttt{level}-\texttt{set}$ function $V_{t_{k+1}}(\boldsymbol{x})$ based on the procedure $\texttt{Under}-\texttt{Approximation}(\Omega_{t_{k+1}}, \boldsymbol{x}_{k+1})$ in Subsection III-A.
d) Set $U_{t_{k+1}} := \{\boldsymbol{x} : V_{t_{k+1}}(\boldsymbol{x}) \leq 1\}$.

In order to prove that $U_{t_{k+1}}$ obtained by the above steps is also a simply connected compact set and is a subset of $\Omega_b(h; U_{t_k}, \boldsymbol{f})$, we first introduce a property.

*Property 1:* If $\Delta \subseteq \mathbb{R}^n$ is a simply connected compact set, $\Omega_b(h; \Delta, \boldsymbol{f})$ is also a simply connected compact set satisfying $\partial\Omega_b(h; \Delta, \boldsymbol{f}) = \Omega_b(h; \partial\Delta, \boldsymbol{f})$ and $(\Omega_b(h; \Delta, \boldsymbol{f})^\circ = \Omega_b(h; \Delta^\circ, \boldsymbol{f})$.

*Proof:* Since $\phi(h; \cdot) : \Omega_b(h; \Delta, \boldsymbol{f}) \to \Delta$ is a homeomorphism between topological spaces $(\Delta, \mathcal{T}_\Delta)$ and $(\Omega_b(h; \Delta, \boldsymbol{f})), \mathcal{T}_{\Omega_b(h;\Delta,\boldsymbol{f})})$, where $\mathcal{T}_\Delta$ and $\mathcal{T}_{\Omega_b(h;\Delta,\boldsymbol{f})}$ are topologies for $\Delta$ and $\Omega_b(h; \Delta, \boldsymbol{f})$ respectively, the conclusion holds. ■

Based on Property 1, we have the following lemma.

*Lemma 4:* If $U_{t_k}$ is a simply connected compact set, then $U_{t_{k+1}}$ obtained by the above steps is also a simply connected compact set and $U_{t_{k+1}} \subseteq \Omega_b(h; U_{t_k}, \boldsymbol{f})$.

*Proof:* According to Definition 3, we can obtain that $U_{t_{k+1}}$ is a simply connected compact set since $V_{t_{k+1}}(\boldsymbol{x})$ is a polynomial $\texttt{level}-\texttt{set}$ function. Since $U_{t_k}$ is a simply connected compact set, $\Omega_b(h; U_{t_k}, \boldsymbol{f})$ is also a simply connected compact set due to Property 1. Further, according to Proposition 1 in Subsection III-A, $U_{t_{k+1}} \subseteq \Omega_b(h; U_{t_k}, \boldsymbol{f})$. ■

From Lemma 4, we can deduce that for $k = 0, \ldots, n-1$, $U_{t_{k+1}}$ is an $\texttt{UAB}$ of $U_{t_k}$ for all $k$.

In the following, we will discuss how to compute $\Omega_{t_{k+1}}$ and $\boldsymbol{x}_{k+1}$ in steps a) and b) in the above procedure. From Property 1, we find that the boundary of $\Omega_b(h; U_{t_k}, \boldsymbol{f})$ corresponds to the boundary of $U_{t_k}$ and an interior point of $\Omega_b(h; U_{t_k}, \boldsymbol{f})$ corresponds to an interior point of $U_{t_k}$ under the map $\phi(h; \cdot)$. So we will obtain $\Omega_{t_{k+1}}$ and $\boldsymbol{x}_{k+1}$ based on $\partial U_{t_k}$ and $\boldsymbol{x}_k$ respectively.

*1) Computing $\Omega_{t_{k+1}}$:* For a given $\epsilon_{l,M}$, $l = 1, \ldots, n$, we first obtain a set of compact intervals $\{s_{k,j}, j = 1, \ldots, M_k\}$ by solving a constraint satisfaction problem such that $\partial U_{t_k} \subseteq \cup_{j=1}^{M_k} s_{k,j}$, where $M_k$ is the number of obtained intervals and $s_{k,j}$ is of the form $[\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_n, \overline{x}_n]$ satisfying $|\overline{x}_l - \underline{x}_l| \leq \epsilon_M$ for $l = 1, \ldots, n$. Then for $j = 1, \ldots, M_k$, we use Interval Taylor methods to obtain a compact interval $s'_{k,j}$ such that $\Omega_b(h; s_{k,j}, \boldsymbol{f}) \subseteq s'_{k,j}$. $s'_{k,j}$ is obtained by considering the reverse system $\dot{\boldsymbol{x}} = -\boldsymbol{f}(\boldsymbol{x})$ since it is well known that $\Omega_f(h; U_{t_k}, -\boldsymbol{f}) = \Omega_b(h; U_{t_k}, \boldsymbol{f})$ [33]. Thus, $\Omega_{t_{k+1}} = \cup_{j=1}^{M_k} s'_{k,j}$ is what we want.

*Remark 4:* $\epsilon_{l,M}$'s are used to restrict the size of intervals enclosing $\partial U_{t_k}$. As the $\epsilon_{l,M}$'s become smaller, the resulting set $\Omega_{t_{k+1}}$ becomes tighter, but the computational cost increases.

*Example 2:* For Example 1, computing $\Omega_T$ when $\epsilon_{l,M} = 0.1$ and $h = 3$ is illustrated in Fig. 6, where $T = 3$.

*2) Computing $\boldsymbol{x}_{k+1}$:* One way to compute $\boldsymbol{x}_{k+1} \in (\Omega_b(h; U_{t_k}, \boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$ involves the following steps:

a) Use Interval Taylor methods to get an interval enclosure $s_{\boldsymbol{x}_k}$ of $\Omega_b(h; \boldsymbol{x}_k, \boldsymbol{f})$;
b) Take the geometric center point $\frac{\sum_{l=1}^{2^n} \boldsymbol{v}_{kl}}{2^n}$ of $s_{\boldsymbol{x}_k}$, where $\boldsymbol{v}_{kl}$ is the $l^{th}$ vertex of interval $s_{\boldsymbol{x}_k}$. If $\frac{\sum_{l=1}^{2^n} \boldsymbol{v}_{kl}}{2^n} \notin \Omega_{t_{k+1}}$, then go to c, else go to d;
c) Use Interval Taylor methods to get an interval enclosure $s_{k+1}$ of $\phi(h; \frac{\sum_{l=1}^{2^n} \boldsymbol{v}_{kl}}{2^n})$. If $s_{k+1} \subseteq U_{t_k}$, set $\boldsymbol{x}_{k+1} := \frac{\sum_{l=1}^{2^n} \boldsymbol{v}_{kl}}{2^n}$, else go to d;
d) In the interval $s_{\boldsymbol{x}_k}$, linearly search for $\boldsymbol{x}_{k+1}$ satisfying $\boldsymbol{x}_{k+1} \notin \Omega_{t_{k+1}}$ and $s_{k+1} \subseteq U_{t_k}$ along some coordinate direction starting from $\frac{\sum_{l=1}^{2^n} \boldsymbol{v}_{kl}}{2^n}$, where $s_{k+1}$ is an interval enclosure of $\phi(h; \boldsymbol{x}_{k+1})$ obtained by Interval Taylor

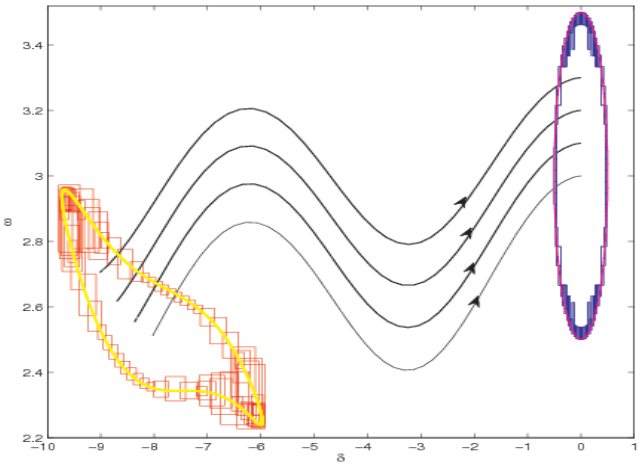

Fig. 6. An illustration for computing $\Omega_T$. (red boxes – $\Omega_T$ including $\partial\Omega_b(T;\mathtt{TR},\boldsymbol{f})$; blue boxes – $\cup_j s_{0,j}$ including $\partial\mathtt{TR}$; yellow curve – $\partial\Omega_b(T;\mathtt{TR},\boldsymbol{f})$ obtained by simulation methods; purple curve – $\partial\mathtt{TR}$; black curve – some simulation trajectories originating from $\Omega_b(T;\mathtt{TR},\boldsymbol{f})$ over the time interval $[0,3]$.)

methods.

*Proposition 3:* If $\boldsymbol{x}_{k+1}$ is obtained by the above steps a) $\sim$ d), then $\boldsymbol{x}_{k+1} \in (\Omega_b(h;U_{t_k},\boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$.

*Proof:* From the above steps a) $\sim$ d), $\boldsymbol{x}_{k+1} \notin \Omega_{t_{k+1}}$. Since $s_{k+1} \subseteq U_{t_k}$, $\boldsymbol{x}_{k+1} \in \Omega_b(h;U_{t_k},\boldsymbol{f})$. Using the fact that $\partial\Omega_b(h;U_{t_k},\boldsymbol{f}) \subseteq \Omega_{t_{k+1}}$, we get $\boldsymbol{x}_{k+1} \in (\Omega_b(h;U_{t_k},\boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$. ■

Computing $\boldsymbol{x}_{k+1} \in (\Omega_b(h;U_{t_k},\boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$ is illustrated in Fig. 7 and Example 3. Note that $\Omega_{t_{k+1}}$, which is the union of intervals including $\partial\Omega_b(h;U_{t_k},\boldsymbol{f})$, is not presented in Fig. 7.

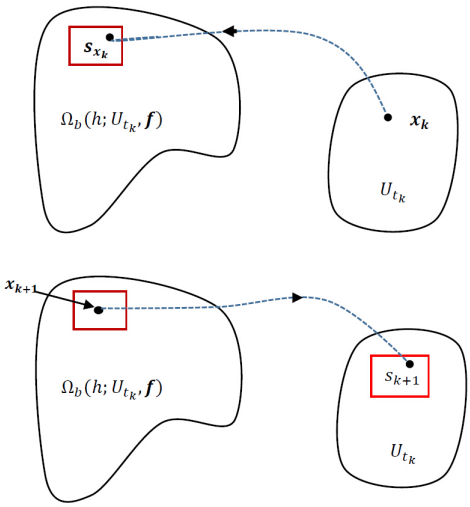


Fig. 7. An illustration for computing a $\boldsymbol{x}_{k+1} \in (\Omega_b(h;U_{t_k},\boldsymbol{f}))^\circ \setminus \Omega_{t_{k+1}}$.

*Example 3:* For Example 1, when $\boldsymbol{x}_0 = (0,3)'$ is used, $s_{\boldsymbol{x}_0} = [-8.03,-0.801] \times [2.50,2.52]$ is obtained. Thus $\boldsymbol{x}_1 = (-0.802,2.51)'$ is found since $s_1 = [-0.0034,-0.0033] \times [2.9977,2.9978] \subseteq \mathtt{TR}$.

*Remark 5:* If a point $\boldsymbol{x}_{k+1}$ satisfying $\boldsymbol{x}_{k+1} \notin \Omega_{t_{k+1}}$ and $s_{k+1} \subseteq U_{t_k}$ is not found, one would re-compute with smaller $\epsilon_{l,M}$'s and\or a smaller step size $h$.

Thus we have presented our approach to compute an UAB in details. In order to enhance the understanding of our approach, an example is given below to illustrate our approach.

*Example 4:* Consider the system in Example 1 again. In the introduction, we provided a non-ellipsoidal under-approximation of $\Omega_b(3;\mathtt{TR},\boldsymbol{f})$. Here, we will present some computational details. Assume that the time step $h$ is equal to 3 and $\epsilon_{l,M} = 0.001$. Firstly, based on the method described in Subsubsection III-B1, we compute $\Omega_T = \cup_j s'_{1,j}$ such that $\partial\Omega_b(3;\mathtt{TR},\boldsymbol{f}) \subseteq \Omega_T$, where $s'_{1,j}$ is of the interval form. Secondly, based on the method described in Subsubsection III-B2, we compute $\boldsymbol{x}_1 \in (\Omega_b(3;\mathtt{TR},\boldsymbol{f}))^\circ \setminus \Omega_T$ based on $\boldsymbol{x}_0$. Since $V_0 = 4\delta^2 + 4(\omega-3)^2$, $\boldsymbol{x}_0 = (0,3)$. Thus, $\boldsymbol{x}_1 = (-8.02,2.51) \in (\Omega_b(T;\mathtt{TR},\boldsymbol{f}))^\circ \setminus \Omega_T$ can be obtained. Thirdly, when the degree $d$ of polynomial $\mathtt{level-set}$ functions is set to 4, based on the procedure $\mathtt{Under-Approximation}(\Omega_T,\boldsymbol{x}_1)$, a computed UAB for the time duration $T = 3$ is $\{(\delta,\omega) : 46.55(-2.51+\omega)^2 - 51.85(-2.51+\omega)^3 + 29.91(-2.51+\omega)^4 + 7.28(-2.51+\omega)(8.02+\delta) + 12.37(-2.51+\omega)^2(8.02+\delta) - 23.53(-2.51+\omega)^3(8.02+\delta) + 0.64(8.02+\delta)^2 + 0.57(-2.51+\omega)(8.02+\delta)^2 + 0.47(-2.51+\omega)^2(8.02+\delta)^2 - 0.31(8.02+\delta)^3 + 2.42(-2.51+\omega)(8.02+\delta)^3 + 0.30(8.02+\delta)^4 \leq 1\}$, whose boundary is depicted in Fig. 8.

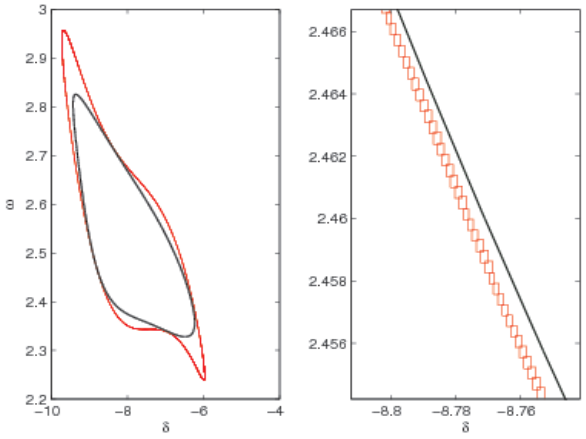

Fig. 8. An UAB for Example 4. (left: red boxes – $\Omega_T$ including $\partial\Omega_b(3;\mathtt{TR},\boldsymbol{f})$; black curve – $\partial U_3$ obtained when $d=4$ and $h=3$; right: a zoomed-in portion of the left figure.)

### C. Computational Complexity

In this subsection, we will discuss the computational complexity of our method briefly. In the $k^{th}$ step, the interval branch-and-bound method for the problem of yielding some intervals to enclose $\partial U_{t_k}$ is of exponential complexity $\mathcal{O}(\xi_k^n)$, where $\xi_k = \mathcal{O}(\frac{1}{\epsilon_{k,M}})$. The underlying Taylor series method is of polynomial complexity [40]: the work is $\mathcal{O}(p^2)$ to complete the Taylor coefficients, and $\mathcal{O}(n^3)$ for performing linear algebra, where $p$ is the order of Taylor series

expansion. The size of the matrix is $\begin{pmatrix} d \\ n+d \end{pmatrix} + M_k + 2\begin{pmatrix} d \\ n+d \end{pmatrix}$ and the number of variables is $2\begin{pmatrix} d \\ n+d \end{pmatrix}$ in the semidefinite program (4) $\sim$ (7). Thus, the complexity of applying interior point methods to solve the semidefinite program (4)$\sim$(7) is $\texttt{SDP}_k = \mathcal{O}\left(2\left(\begin{pmatrix} d \\ n+d \end{pmatrix} + M_k + 2\begin{pmatrix} d \\ n+d \end{pmatrix}\right)\begin{pmatrix} d \\ n+d \end{pmatrix}\right)$ [41]. Therefore, the total computational complexity of our method is $\sum_{k=0}^{N-1}\left(\mathcal{O}(\xi_k^n) + \mathcal{O}(p^2) + \mathcal{O}(n^3) + \texttt{SDP}_k\right)$.

## IV. Examples and Comparisons

In this section, we evaluate our method using four interesting examples and compare it with [27] and [18]. These results can be found in Fig. 9 $\sim$ 18. All the computations are performed on an i5-3337U 1.8GHz CPU with 4GB RAM running Ubuntu Linux 13.04. Table I presents details on the parameters that control our approach.

### A. Examples and Discussions

In this subsection, we evaluate our approach using four examples, three of which are two-dimensional and one of them is three-dimensional. We also discuss parameters that control our approach in the context of these examples. These parameters include $h$, which is the time step, $\epsilon_M$, which represents the size of intervals enclosing boundaries, and $d$, which denotes the degree of polynomial $\texttt{level}-\texttt{set}$ functions. According to the computational complexity analysis of our approach in Subsection IV-B, our approach suffers from dimensional curse. Nevertheless, we can still explore some future research directions based on the three two-dimensional examples to make our approach truly practical.

TABLE I
Performance of our approach on examples. $T$ : the given time instant for UAB; $h$: step size; $d$: degree of polynomial $\texttt{level}-\texttt{set}$ functions; $\epsilon_M : \epsilon_{i,M}(i = 1, \cdots, n)$ in computing $\Omega_{t_{k+1}}$; Time: CPU execution time cost(seconds).

| Examples | T | h | $d$ | $\epsilon_M$ | TIME |
|---|---|---|---|---|---|
| 5 | 25 | 5 | 2 | 0.01 | 3 |
| 5 | 25 | 5 | 2 | 0.001 | 119 |
| 5 | 25 | 5 | 4 | 0.01 | 4 |
| 5 | 25 | 5 | 4 | 0.001 | 223 |
| 6 | 0.2 | 0.2 | 2 | 0.005 | 5 |
| 6 | 0.2 | 0.05 | 4 | 0.005 | 191 |
| 6 | 0.2 | 0.2 | 4 | 0.005 | 8 |
| 7 | 5 | 0.5 | 2 | 0.005 | 45 |
| 7 | 5 | 0.5 | 4 | 0.005 | 120 |
| 8 | 0.2 | 0.1 | 2 | 0.008 | 1227 |

Below we present the three two-dimensional examples that were used to evaluate our approach. The corresponding results are shown in Fig.9 $\sim$ Fig. 11.

*Example 5:* Consider a model of an electromechanical oscillation of a synchronous machine [48],

$$\begin{cases} \dot{\delta} = \omega \\ \dot{\omega} = 0.2 - 0.7sin\delta - 0.05\omega \end{cases},$$

where $\texttt{TR} = \{(\delta, \omega) : \delta^2 + \omega^2 \leq 0.01\}$.

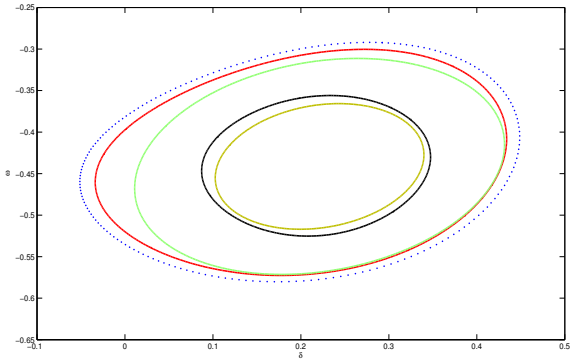

Fig. 9. UABs for Example 5. (blue points – $\partial\Omega_b(25; \texttt{TR}, \boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_{25}$ obtained when $d = 4$ and $\epsilon_{iM} = 0.001$; green curve – $\partial U_{25}$ obtained when $d = 2$ and $\epsilon_{iM} = 0.001$; black curve – $\partial U_{25}$ obtained when $d = 4$ and $\epsilon_{iM} = 0.01$; yellow curve – $\partial U_{25}$ obtained when $d = 2$ and $\epsilon_{iM} = 0.01$.)

*Example 6:* Consider the Brusselator model [13],

$$\begin{cases} \dot{x_1} = 1 + x_1^2 x_2 - 1.5x_1 - x_1 \\ \dot{x_2} = 1.5x_1 - x_1^2 x_2 \end{cases},$$

where $\texttt{TR} = \{\boldsymbol{x} : (x_1 - 0.9)^2 + (x_2 - 0.3)^2 <= 0.25\}$.

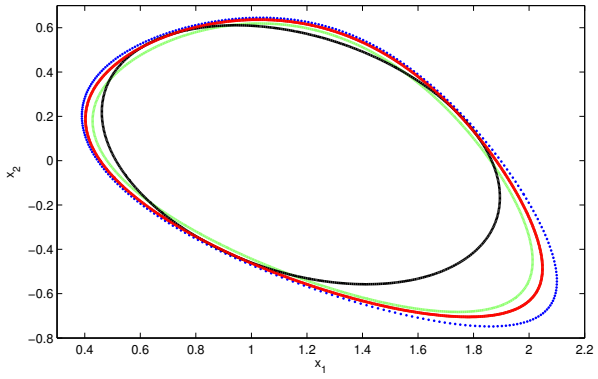

Fig. 10. UABs for Example 6. (blue points – $\partial\Omega_b(0.2; \texttt{TR}, \boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_{0.2}$ obtained when $d = 4$ and $h = 0.2$; green line – $\partial U_{0.2}$ obtained when $d = 4$ and $h = 0.05$; black curve – $\partial U_{0.2}$ obtained when $d = 2$ and $h = 0.2$.)

*Example 7:* Consider the Van-der-Pol system [9],

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -0.2(x_1^2 - 1)x_2 - x_1 \end{cases},$$

where $\texttt{TR} = \{\boldsymbol{x} : (x_1 - 1)^2 + (x_2 - 1)^2 <= 0.25\}$.

From the above examples, we observe that

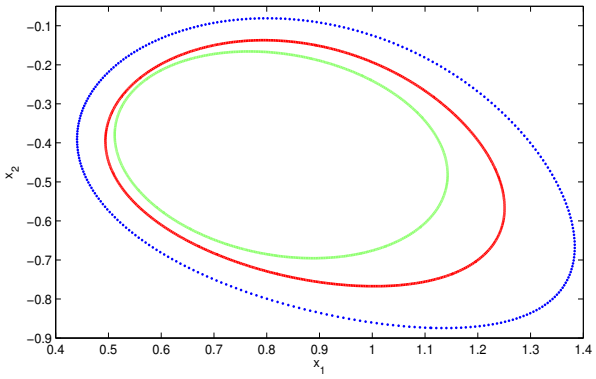


Fig. 11. UABs for Example 7. (blue points – $\partial\Omega_b(5;\mathtt{TR},\boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_5$ obtained when $d=4$; green curve – $\partial U_5$ obtained when $d=2$.)

1) when $d$ and $h$ are fixed, the resulting under-approximation becomes less conservative as $\epsilon_M$ becomes smaller (Example 5);
2) when $\epsilon_M$ and $d$ are fixed, a smaller time-step $h$ may lead to larger errors. The underlying reason is that the under-approximation error in every iterative step will propagate through the computations (Example 6), similar to the well-known wrapping effect in over-approximating reachable sets;
3) when $\epsilon_M$ and $h$ are fixed, a better under-approximation will be found as $d$ increases (Examples 5 $\sim$ 7).

Thus, in case that the returned under-approximation is empty in some iterative step, we have to try a smaller $\epsilon_M$ and/or a higher degree $d$ and a different time step $h$, but the computational cost increases. A smaller $\epsilon_M$ will help to obtain a tighter $\Omega_{t_{k+1}}$, eventually resulting in a less conservative under-approximation. Therefore, in order to obtain a tighter $\Omega_{t_{k+1}}$, either methods different from Interval Taylor methods, or reducing the wrapping effect caused by interval Taylor methods must be considered. As the degree $d$ increases, we have to solve a larger semidefinite programming problem. Although there are some existing semidefinite program solvers that utilize the power of parallel computing and show good performance for large-sized problems, however, the present status of semidefinite program software packages is not as advanced as that of their linear programming counterparts. An interesting topic is to obtain a polynomial $\mathtt{level-set}$ function in a local compact region based on a linear program, which is derived based on Handelman representation [42], rather than a semidefinite program. This could also be an interesting topic for future research.

Note that when $d=2$, our approach can also be used to compute ellipsoidal estimations of reachable sets. This is different from conservative linearization based methods, which can be seen as a special first-order case of our approach. Further, as mentioned in the introduction, since reachable sets of nonlinear systems are in general far from being convex, the idea of using ellipsoidal representations for approximating reachable sets is questionable. Our approach on the other hand can produce non-ellipsoidal or non-convex representations when $d>2$, and this can lead to better results (Examples 5 $\sim$ 7).

*Example 8:* Below we present a three-dimensional example [43] to illustrate the challenges of our approach for high dimensional systems. The results of our approach for this example are shown in Fig. 12.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = x_3 \\ \dot{x_3} = -4x_1 - 3x_2 - 3x_3 + x_1^2x_2 + x_1^2x_3 \end{cases},$$

where $\mathtt{TR} = \{\boldsymbol{x} : (x_1-1)^2 + (x_2-1)^2 + (x_3-1)^2 \le 0.01\}$.

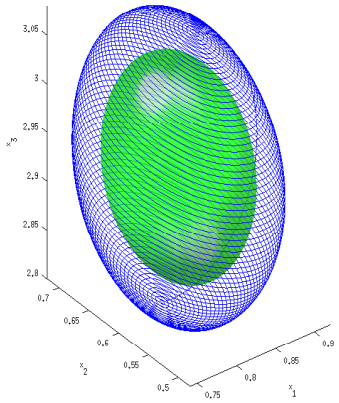


Fig. 12. An UAB for Example 8. (green curve – $\partial U_{0.2}$; blue curve – $\partial\Omega_b(0.2;\mathtt{TR},\boldsymbol{f})$ obtained by Runge-Kutta methods.)

From this example, we find that the computational burden increases significantly as the system dimension increases. The underlying reason is that the computational complexity of our method increases rapidly with the system dimension. In our computations, the computational burden mainly lies in solving the convex program (4) $\sim$ (7). In our future work, we will focus on improving the computational complexity of our approach by solving linear programs instead of semidefinite programs, as described above.

## B. Comparisons

There are few methods available to compute under-approximations of backward reachable sets for nonlinear systems. Recently, an approach to compute over- and under-approximations of the capture basin of a given target region was proposed in [27]. The capture basin is a set such that all trajectories originating from it will enter the target region in finite time. The under-approximations obtained by the method in [27] are respectively presented in Fig. 13 $\sim$ 15 for Examples 5 $\sim$ 7. The corresponding computation times are 240, 317 and 111 seconds, respectively. Note that the computation times for under-approximations obtained by our approach in Fig. 13 $\sim$

Fig. 15 are respectively 119, 8 and 120 seconds[3].

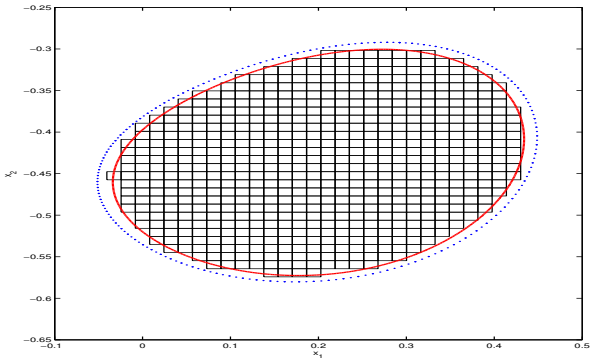

Fig. 13. UABs for Example 5. (blue points – $\partial\Omega_b(25;\mathtt{TR},\boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_{25}$ obtained when $d=4$ and $\epsilon_M = 0.001$; black curve – the boundaries of intervals used to represent an under-approximation obtained by the method in [27].)

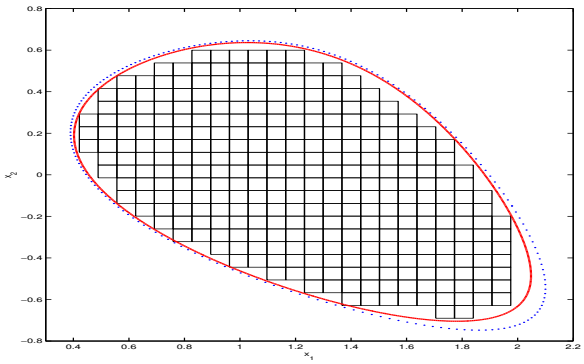

Fig. 14. UABs for Example 6.(blue points – $\partial\Omega_b(0.2;\mathtt{TR},\boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_{0.2}$ obtained when $d=4$ and $h = 0.2$; black curve – the boundaries of intervals used to represent an under-approximation obtained in [27].)

Comparing computation times of the two approaches, we can observe that our method is more efficient for Examples 5 and 6. But this is not the most important contribution of our method. Our main contribution lies in the fact that we use a semi-algebraic set rather than the union of intervals to represent an under-approximation. The benefits of this representation are as follows.

1) The storage space for under-approximations can be reduced dramatically since only the coefficients of the obtained polynomial $\mathtt{level-set}$ function are stored for some cases. This can be illustrated by Examples 5 $\sim$ 8. When the method in [27] is used, we have to store 607, 317 and 93 intervals for the first three examples respectively. However, the numbers of coefficients in our method are 15, 15 and 15 respectively.
2) It is easier to verify whether a point is in the obtained under-approximation or not since only basic arithmetic operations are needed for some cases such as Examples 5 $\sim$ 8. However, the worst case in verifying whether a point is in the under-approximation obtained by the method in [27] is that all the intervals obtained have to be used for performing verification. But for our representation, the computations involve performing basic arithmetic operations to yield the polynomial $\mathtt{level-set}$ function value at this point and comparing this value against the value 1.

[3]The result computed by the method in [27] for Example 8 is not presented here since it is difficult to quantitatively compare the results in a three dimensional space. It is difficult to tell the difference between the results.

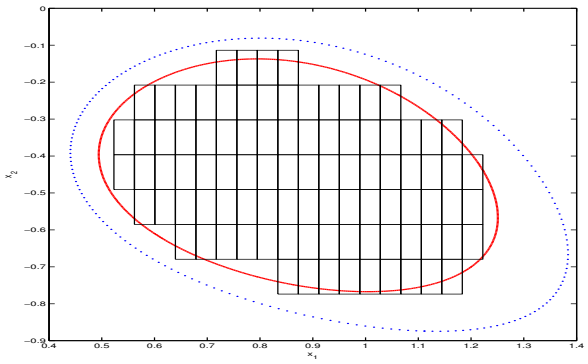

Fig. 15. UABs for Example 7. (blue points – $\partial\Omega_b(5;\mathtt{TR},\boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – $\partial U_5$ obtained when $d=4$; black curve – the boundaries of intervals used to represent an under-approximation obtained by the method in [27].)

Besides, note that $\boldsymbol{g}(\boldsymbol{x})$ can be an arbitrary polynomial vector field such that $\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x})$ has a globally asymptotically stable equilibrium point $\boldsymbol{x}_0 = (x_{0,1},\ldots,x_{0,n})$ in the semi-definite program (4) $\sim$ (7) although, we have employed $\boldsymbol{g}(\boldsymbol{x})$s of the form $(-x_1+x_{0,1};\ldots;-x_n+x_{0,n})$ as mentioned in Remark 2 to perform computations for the above examples. In the following we will employ other forms, $(-(x_1-x_{0,1})-2(x_2-x_{0,2});(x_1-x_{0,1})-(x_2-x_{0,2}))$ for Examples 5 $\sim$ 7 and $(-(x_1-x_{0,1})-2(x_2-x_{0,2});(x_1-x_{0,1})-(x_2-x_{0,2});-x_3+x_{0,3})$ for Example 8, to perform computations. The obtained results for Examples 5 and 8 are similar to those obtained using the former form, based on the same parameters presented in Table 1. Thus, we only present the results for Example 6 and 7 here. The results, together with the ones obtained using the former form, are illustrated in Fig. 16 and 17 respectively. The corresponding computation times are about 10 seconds and 131 seconds respectively. From the two examples, we can conclude that different results can be obtained using different $\boldsymbol{g}(\boldsymbol{x})$s, and the results obtained using $\boldsymbol{g}(\boldsymbol{x})$ of a complex form are not always better than the ones obtained using a simple form. As for how to choose a form that gives good results, we will make an investigation in our future work.

Finally, we compare our method with the method in [18], which aims to compute inner approximations of the region of attraction for polynomial dynamical systems by solving sum-of-squares programming problems. The region of attraction is the set of all states that end in the target set at a given time without leaving a constraint set. Under the condition that the related constraint sets are specified or computed in advance, our method can also be used to compute inner approximations

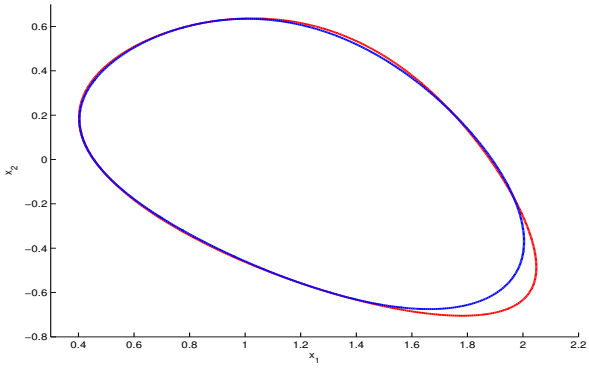

Fig. 16. UABs for Example 6.(red curve – $\partial U_{0.2}$ obtained using $\boldsymbol{g}(\boldsymbol{x})$ of the form $(-x_1+x_{0,1}; -x_2+x_{0,2})$ when $d=4$, $h=0.2$ and $\epsilon_M=0.005$; blue curve – $\partial U_{0.2}$ obtained using $\boldsymbol{g}(\boldsymbol{x})$ of the form $(-(x_1-x_{0,1})-2(x_2-x_{0,2}); (x_1-x_{0,1})-(x_2-x_{0,2}))$ when $d=4$, $h=0.2$ and $\epsilon_M=0.005$.)

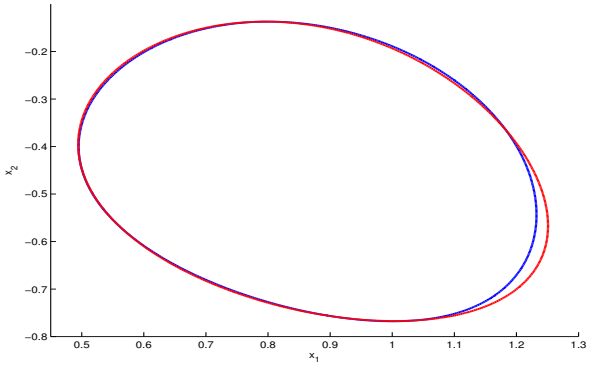

Fig. 17. UABs for Example 7. (red curve –$\partial U_{0.2}$ obtained using $\boldsymbol{g}(\boldsymbol{x})$ of the form $(-x_1+x_{0,1}; -x_2+x_{0,2})$ when $d=4$, $h=0.5$ and $\epsilon_M=0.005$; blue curve – $\partial U_{0.2}$ obtained using $\boldsymbol{g}(\boldsymbol{x})$ of the form $(-(x_1-x_{0,1})-2(x_2-x_{0,2}); (x_1-x_{0,1})-(x_2-x_{0,2}))$ when $d=4$, $h=0.5$ and $\epsilon_M=0.005$.)

of the region of attraction as defined in above work. Since the method in [18] is restricted to polynomial systems, we will compare our method with the method in [18] based on Example 6∼8. Assume that the constraint sets for these three examples are $\{\boldsymbol{x} : 1.1-(x_1-1.3)^2-x_2^2 \geq 0\}$, $\{\boldsymbol{x} : 4-x_1^2-x_2^2 \geq 0\}$ and $\{\boldsymbol{x} : 2.25-(x_1-0.9)^2-(x_2-0.9)^2-(x_3-2)^2 \geq 0\}$ respectively. Using the method in [18], a solution was obtained for Example 6 based on the sum-of-squares programming solver YALMIP [45] with Sedumi [46], SDPT3 [47], CSDP [26] or Mosek[4]. The solution, which is defined by a polynomial of degree 6, is illustrated in Fig. 18 for Example 6. The corresponding computation time is around 10.04 seconds. Combining the results displayed in Fig. 18, we observe that our method can obtain a better estimation more efficiently in terms of time compared with the method in [18] for this example. Note that the computation time for the result in Fig. 18 obtained using our method is around 8 seconds. Our method avoids the complexity of choosing appropriate state constraint sets to solve the corresponding semi-definite programming problems, and can deal with more general nonlinear systems such as Example 5 in Subsection IV-A, thus making our method effective.

[4]This software can be downloaded from https://mosek.com/products/beta-versions.

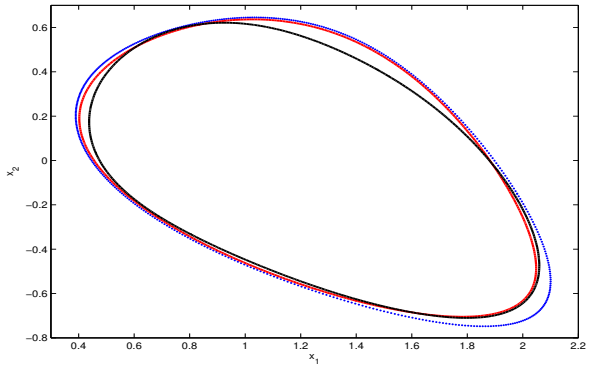

Fig. 18. UABs for Example 6. (red curve – $\partial U_{0.2}$ obtained by our method when $d=4$ and $h=0.2$; black curve – the boundary of an under-approximation of $\Omega_b(0.2; \texttt{TR}, \boldsymbol{f})$ obtained by the method in [18]; blue points – $\partial\Omega_b(0.2; \texttt{TR}, \boldsymbol{f})$ obtained by Runge Kutta methods.)

## V. Conclusion

Given a nonlinear system and a simply connected compact target region, in this paper we proposed a numerical method by performing boundary analysis to obtain an under-approximation of the backward reachable set for a given time duration. The UAB is represented as a semi-algebraic set, which relies on a polynomial $\texttt{level}-\texttt{set}$ function. The polynomial $\texttt{level}-\texttt{set}$ function could be computed by solving a semidefinite programming problem. Numerical results and comparisons with [27] and [18] based on four examples were given to illustrate the benefits of our approach. The results show that our method can obtain excellent estimations in a more computational manner for these special examples. Nevertheless, a lot remains to be done to make our method truly practical. For instance, the computational burden is considerable if the state space dimension is large. Although our method can give an estimation of local error bounds for under-approximations, which can be found in Subsection III-A, yielding a global error is still an open problem. These will be considered in our future work.

Extending our method to compute under-approximations of reachable sets for nonlinear hybrid systems with uncertain parameters and inputs is also another interesting topic. Applying our approach for computing under-approximations of backward reachable sets in the design of artificial pancreas is also ongoing.

## VI. Acknowledgement

The authors are grateful to Mr. Milan Korda from École Polytechnique Fédérale de Lausanne, Dr. Xin Chen from RWTH Aachen University for helpful discussions. Also, the authors are especially grateful to the anonymous reviewers for their valuable comments.

## REFERENCES


[1] J. Lunze and F. Lamnabhi-Lagarrigue. Handbook of Hybrid Systems Control: Theory, Tools, Applications. Cambridge University Press, 2009.

[2] S. Ratschan and Z. She. Safety Verification of Hybrid Systems by Constraint Propagation-based Abstraction Refinement. ACM Trans. Embed. Comput. Syst., 6(1), Article No.8, pp. 1–23, 2007.

[3] I. M. Mitchell, A. M. Bayen and C. J. Tomlin. A Time-Dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games. IEEE Trans. Automat. Contr., Vol. 50, pp. 947–957, 2005.

[4] S. Sankaranarayanan, H. B. Sipma and Z. Manna. Fixed Point Iteration for Com- puting the Time Elapse Operator. In 2006 Proc. of the 9th International Workshop on Hybrid Systems: Computation and Control, pp. 537–551, 2006.

[5] M. A. B. Sassi and A. Girard. Computation of polytopic invariants for polynomial dynamical systems using linear programming. Automatica, Vol. 48, pp. 3114–3121, 2012.

[6] J. Liu, N. Zhan and H. Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In Proceedings of the 9th International Conference on Embedded Software, pp. 97–106, 2011.

[7] S. Prajna, A. Jadbabaie and G. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Transactions on Automatic Control, Vol. 52, pp. 1415–1428, 2007.

[8] Z. Han and B. H. Krogh. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In Proceedings of the American Control Conference, pp. 1505–1510, 2006.

[9] M. Althoff, O. Stursberg and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In Proceedings of the 47th IEEE Conference on Decision and Control, Cancun, pp. 4042–4048, 2008.

[10] E. Asarin, O. Bournez, T. Dang and O. Maler. Approximate reachability analysis of piecewise linear dynamical systems. In N. Lynch, and B. Krogh (Eds.), Hybrid systems: Computation and control, LNCS1790. pp. 20–31, 2000.

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

[12] N. S. Nedialkov, K. R. Jackson and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. Applied Mathematics and Computation, Vol. 105, pp. 21–68, 1999.

[13] X. Chen, E. Ábrahám and S. Sankaranarayanan. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. In Proceedings of the 33rd IEEE Real-Time Systems Symposium, IEEE Computer Society, pp. 183–192, 2012.

[14] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis: internal approximation. Systems and Control Letters, Vol. 41, pp. 201–211, 2000.

[15] S. Ratschan and Z. She. Providing a Basin of Attraction to A Target Region of Polynomial Systems by Computation of Lyapunov-like Functions. SIAM J. on Control and Optimization, 48(7): 4377–4394, 2010.

[16] Z. She and B. Xue. Computing an invariance kernel with target by computing Lyapunov-like functions. IET Control Theory and Applications, 7(15): 1932–1940, 2013.

[17] D. Henrion and M. Korda. Convex Computation of the Region of Attraction of Polynomial Control Systems. IEEE Transactions on Automatic Control, Vol. 59, pp. 297–312, 2014.

[18] M. Korda, D. Henrion and N. C. Jones. Inner Approximations of the Region of Attraction for Polynomial Dynamical Systems. IFAC conference on Nonlinear Control Systems, 46(23): 534–539, 2013.

[19] E. Plaku, L. E. Kavraki and M. Y. Vardi. Hybrid systems: from verification to falsification by combining motion planning and discrete search. Formal Methods in System Design, Vol. 34, pp. 157–182, 2009.

[20] J. Bondia, E. Dassau, H. Zisser, R. Calm, J. Vehí, L. Jovanovič and F. J. Doyle. Coordinated basal-bolus infusion for tighter postprandial glucose control in insulin pump therapy. J. Diabetes Sci. Technol., 3(1): 89–97, 2009.

[21] P. Herrero, R. Calm, J. Vehí, J. Armengol, P. Georgiou, N. Oliver and C. Tomazou. Robust fault detection system for insulin pump therapy using continuous glucose monitoring. J. Diabetes Sci. Technol., 6(5): 1131–1141, 2012.

[22] J. Bochnak, M. Coste, M.-F. Roy, Géométrie algébrique réelle, Springer-Verlag, Berlin, Heidelberg, New York, 1987.

[23] L. Granvilliers and F. Benhamou. Realpaver: an Interval Solver using Constraint Satisfaction Techniques. ACM. TOMS., 32(1): 138–156, 2006.

[24] D. Yu. Grigor'ev and N. N. Vorobjov Jr.. Counting connected components of a semialgebraic set in subexponential time. Computational Complexity, Vol. 2, pp. 133–186, 1992.

[25] N. S. Nedialkov. VNODE-LP – a validated solver for initial value problems in ordinary differential equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, 2006.

[26] B. Borchers. CSDP, a C library for semidefinite programming. Optimization Methods and Software, 11(1): 613–623, 1999.

[27] M. Lhommeau, L. Jaulin and L. Hardouin. Capture basin of approximation using interval analysis. International Journal of Adaptive Control and Signal Processing, Vol. 25, pp. 264–272, 2011.

[28] A. Girard, C. Le Guernic and O. Maler. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. Hybrid Systems: Computation and Control Lecture Notes in Computer Science Volume 3927, pp. 257–271, 2006.

[29] J. N. Maidens, S. Kaynama, I. M. Mitchell, M. M. K. Oishi and G. A. Dumont. Lagrangian methods for approximating the viability kernel in high-dimensional systems. Automatica, Vol. 49, pp. 2017–2029, 2013.

[30] A. Hamadeh and J. Goncalves. Reachability analysis of continuous-time piecewise affine systems. Automatica, Vol. 44, pp. 3189–3194, 2008.

[31] A. N. Daryin and A. B. Kurzhanski. Estimation of reachability sets for large-scale uncertain systems: From theory to computation. In Proceedings of the 51st Annual Conference on Decision and Control, pp. 7401–7406, 2012.

[32] D. L. Elliott. Bilinear Control Systems: Matrices in Action. Page 26. Springer, 2009.

[33] R. Genesio, M. Tartaglia and A. Vicino. On the estimation of asymptotic stability regions: State of the art and new proposals. IEEE Transactions on Automatic Control, Vol. 30, pp. 747–755, 1985.

[34] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In Proceedings of the 41st IEEE Conference on Decision and Control, pp. 3482–3487, 2002.

[35] Y. Lin and M. A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. Applied Numerical Mathematics, Vol. 57, pp. 1145–1162, 2007.

[36] M. Hladík and J. Horáček. Interval Linear Programming Techniques in Constraint Programming and Global Optimization. In M. Ceberio and V. Kreinovich, editors, Constraint Programming and Decision Making, Studies in Computational Intelligence Volume 539, pp. 47–59, 2014.

[37] J. Rohn and J. Kreslová. Linear interval inequalities. Linear and Multilinear Algebra, Vol. 38, pp. 79–82, 1994.

[38] N. Munro. Symbolic Methods in Control System Analysis and Design. The Institution of Engineering and Technology, 1999.

[39] W. Tan. Nonlinear Control Analysis and Synthesis using Sum-of-Squares Programming. Ph. D dissertation, University of California, Berkeley, 2006.

[40] N. Ramdani and N. S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. Nonlinear Systems: Hybrid Systems, Vol. 5, pp. 149–162, 2011.

[41] L. Dai, B. Xia and N. Zhan. Generating non-linear interpolants by semidefinite programming. In Proc. of CAV 2013, Lecture Notes in Computer Science 8044, pp. 364–380, 2013.

[42] S. Sankaranarayanan, X. Chen and E. Ábrahám. Lyapunov Function Synthesis Using Handelman Representations. IFAC conference on Nonlinear Control Systems, Vol. 46, pp. 576–581, 2013.

[43] U. Topcu, A. Packard and P. Seiler. Local stability analysis using simulations and sum-of-squares programming. Automatica, Vol. 44, pp. 2669–2675, 2008.

[44] D. Henrion, J. B. Lasserre, and J. Löfberg. Gloptipoly 3: moments, optimization and semidefinite programming. Optimization Methods and Software, Vol. 24. pp. 761–779, 2009.

[45] J. Löfberg. YALMIP, A toolbox for modeling and optimization in MATLAB. In Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design, pp. 284–289, 2004.

[46] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software, Vol. 11, pp. 625–653, 1999.

[47] K. C. Toh, M. J. Todd and R. H. Tütüncü. SDPT3 - a Matlab software package for semidefinite programming, version 2.1. Optimization Methods and Software, pp. 545–581, 1999.

[48] Y. Susuki, T. J. Koo, H. Ebina, T. Yamazaki, T. Ochi, T. Uemura and T. Hikihara. A Hybrid System Approach to the Analysis and Design of Power Grid Dynamic Performance. Proceedings of the IEEE, Vol. 100, pp. 225–239, 2012.

**Bai Xue** is working as a postdoc in the Department für Informatik at Carl von Ossietzky Universität Oldenburg from November, 2015. He received the B.Sc. degree in Information and Computing Science in 2008 from Tianjin University of Technology and Education and the Ph.D. degree in Applied Mathematics in 2014 from Beihang University. Prior to joining in Carl von Ossietzky Universität Oldenburg, he worked as a research fellow in the Centre for High Performance Embedded Systems at Nanyang Technological University from May, 2014 to September, 2015. His research interests involve stability and reachability analysis of (stochastic/probabilistic) hybrid systems. He has published in several leading conferences and journals in these areas.

**Zhikun She** is a professor and vice-dean of the School of Mathematics and Systems Science at Beihang University. He received the B.Sc. degree in 1999 and the Ph.D. degree in 2005 both from Peking University, and worked at MPI für Informatik as a post-doctor from Jan. 2004 to Dec. 2006. His research interests involve the theory, methodologies, and applications of safety verification and stability analysis of hybrid systems. He has been the principal investigator of more than 10 research projects and published more than 40 research papers. In particular, he has been awarded 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.

**Arvind Easwaran** is an assistant professor in the School of Computer Engineering at Nanyang Technological University (NTU), Singapore. He received MSc and PhD degrees from the University of Pennsylvania, USA, and a BE degree from University of Mumbai, India, all in Computer Science & Engineering. Prior to joining NTU in 2013, he has been an Invited Research Scientist at the Polytechnic Institute of Porto, Portugal, and an R & D Scientist at Honeywell Aerospace, USA. His research interests include Cyber-Physical Systems, Real-Time Systems, and Formal Methods. He has published in several leading conferences and journals in these areas, some of which are highly cited and have won awards.