Under-Approximating Backward Reachable Sets by Polytopes

Bai Xue[1,2(✉)], Zhikun She[1], and Arvind Easwaran[2]

[1] School of Mathematics and Systems Science, Beihang University, Beijing, China
zhikun.she@buaa.edu.cn
[2] Nanyang Technological University, Singapore, Singapore
{xuebai,arvinde}@ntu.edu.sg

Abstract. Under-approximations are useful for falsification of safety properties for nonlinear (hybrid) systems by finding counter-examples. Polytopic under-approximations enable analysis of these properties using reasoning in the theory of linear arithmetic. Given a nonlinear system, a target region of the simply connected compact type and a time duration, we in this paper propose a method using boundary analysis to compute an under-approximation of the backward reachable set. The under-approximation is represented as a polytope. The polytope can be computed by solving linear program problems. We test our method on several examples and compare them with existing methods. The results show that our method is highly promising in under-approximating reachable sets. Furthermore, we explore some directions to improve the scalability of our method.

Keywords: Polytopic under-approximations · Backward reachable sets · Nonlinear systems

1 Introduction

Reachability analysis, which involves constructing reachable sets, is a central component of model checking. It plays an important role in automatic verification and falsification of safety properties for continuous nonlinear and hybrid systems [2,3]. It has been utilized in diverse applications such as artificial pancreas [4,5] and robotic systems [6]. Over the past few years, a lot of attention has been given to construct over-approximations of reachable sets of nonlinear systems, i.e., abstraction methods [7,8], simulation based methods [9] and Taylor series expansions [10,11]. Nevertheless, much less attention has been given to the problem of finding under-approximations. Actually, under-approximations of reachable sets are also important to compute because of a variety of applications in engineering domains. For example, they can be used for designing

B. Xue—This work is based on Bai Xue's Ph.D. thesis from Beihang University [1].
Z. She— The work of Zhikun She was partly supported by NSFC-11422111 and NSFC-11371047.

S. Chaudhuri and A. Farzan (Eds.): CAV 2016, Part I, LNCS 9779, pp. 457–476, 2016.
DOI: 10.1007/978-3-319-41528-4_25

robust artificial pancreas [5,12]. Computing under-approximations of backward reachable sets can help find a set of feasible states such that every trajectory originating from it will definitely enter a specified region (e.g., normal blood glucose ranges) at a specified time instant. 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 [13]. They can also be used for falsification by checking if the under-approximation intersects the unsafe sets[1] [3]. Also, under- and over-approximations of reachable sets can provide an indication of the precision of an estimate of the exact reachability region [4]. In contrast to over-approximation problems, methods for computing under-approximations are far from being developed. One of main reasons may lie in the fact that the problem is more difficult than the one of computing over-approximations [14].

We in this paper propose a linear programming based approach combining validated numerical methods for ordinary differential equations for finding polytopic under-approximations of backward reachable sets, under the assumption that the target region is a simply connected compact set. The basic procedure for computing the under-approximation mainly consists of three steps. The first step is to compute an enclosure of the boundary of the backward reachable set based on validated numerical techniques for ordinary differential equations. The second step is to obtain a polytope, which contains the enclosure obtained in the first step, and the last step is to shrink this polytope based on linear programming to yield an under-approximation of the backward reachable set. The contributions of this paper are summarized as follows:

1. We show how a polytopic under-approximation of the backward reachable set can be obtained by solving linear programming problems. We first construct a polytopic over-approximation of the reachable set based on the reachable set's boundary and validated numerical techniques for ordinary differential equations, then contract this over-approximation to obtain a polytopic under-approximation by solving linear programs.
2. We implement our approach based on linear programming solver GLPK[2] and the validated ordinary differential equation solver VNODE-LP [24], test and compare it with the method of Korda et al. [22] based on several examples. The experiment results show that our approach is highly promising in under-approximating reachable sets for some cases. Furthermore, we explore some directions toward making our method scale well based on an example involving a seven-dimensional biological system.

Related Work

Several techniques have been proposed for computing under-approximations of reachable sets for linear systems, e.g., [14–16]. However, they cannot be easily extended to handle non-linear systems. Under-approximations of reachable sets

[1] If the under-approximation intersects the unsafe sets, then the system is definitely unsafe.

[2] http://www.gnu.org/software/glpk/.

for nonlinear systems have been discussed elsewhere (e.g., [17] and [21]), but a feasible solution is not given. Recently, some methods have been proposed to compute under-approximations of reachable sets for nonlinear systems.

Sum-of-squares programming based methods are proposed to compute inner approximations of reachable sets for polynomial dynamical systems in [22,37]. Unfortunately, the present status of semi-definite programming solvers is not so advanced. The numerical problems produced by these solvers often lead to unreliable results for some cases. On the contrary, our method relies on linear programming and validated numerical methods for ordinary differential equations, thus making our method more reliable. A Taylor model backward flowpipe method is presented to compute under-approximations in [23]. However, the algorithm in [23], in which an interval constraint propagation technique is employed to verify the connectedness of an already obtained basic semi-algebraic set, for finding implicit Taylor models such that the semi-algebraic set formed by them is simply-connected[3] is not complete generally[4]. In our method, the procedure employing interval constraint propagation techniques to enclose the boundary of the reachable set is complete.

As mentioned previously, polytopic under-approximations permits the analysis of some specified properties such as the falsification of safety properties using reasoning in the theory of linear arithmetic. Interval under-approximations received increasing attention recently [18,19]. A method based on modal intervals with affine forms is proposed to under-approximate reachable sets using intervals for continuous nonlinear systems modelled by ordinary differential equations [20]. However, our method provides a way to characterize under-approximations of reachable sets using general polytopes, reducing the conservativeness induced by interval representations in the construction of reachable sets.

The structure of this paper is as follows. Some basic definitions related to backward reachable sets as well as an introduction to convex polytopes is introduced in Sect. 2. Our approach of computing under-approximations, together with its computational complexity, is presented in Sect. 3. Several numerical examples with a detailed discussion of our approach and comparison with the method in [22] are provided in Sect. 4. Finally, we conclude our paper in Sect. 5.

2 Preliminary

In this paper, the following notations are used. Vectors are denoted by boldface letters (e.g., $\boldsymbol{x}$). For a set Δ, its complement, interior, closure and boundary are denoted by Δ^c, Δ°, $\overline{\Delta}$ 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 $\boldsymbol{x}$.

[3] A set is simply connected if there are no holes in it to prevent the continuous shrinking of each closed arc to a point.

[4] An algorithm is complete, implying that it guarantees to find a solution if there is one.

2.1 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 $\boldsymbol{f}(\boldsymbol{x})$: $\mathbb{R}^n \rightarrow \mathbb{R}^n$ 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 $\boldsymbol{\phi}(t; \boldsymbol{x}_0) = \boldsymbol{x}(t)$, where $\boldsymbol{x}(t)$ is the solution of System (1) satisfying the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Furthermore, the backward and forward reachable sets of a simply connected compact set $\mathtt{TR}$ for the time duration T are defined as follows.

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

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

It is intractable to obtain these reachable sets for nonlinear systems since they generally do not have a closed-form solution. However, as mentioned previously, it is sufficient to consider an under-approximation of the backward reachable set, denoted as $\mathtt{UAB}$, for certain applications such as artificial pancreas [12].

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

Obviously, all trajectories originating from $\mathtt{UAB}$ will definitely enter $\mathtt{TR}$ after a time duration T, although there may be trajectories not in $\mathtt{UAB}$ that also enter $\mathtt{TR}$ after the time duration T. The under-approximation is equivalent to a region attracting to a target region, but a variant of the classical region of attraction containing an equilibrium.

2.2 Convex Polytopes

Convex polyhedra over reals (rationals) are a natural representation of sets of states for the verification of hybrid systems [25–27]. A convex polytope is a set in $\mathbb{R}^l$ that can be regarded as the set of solutions to the system of linear inequalities $A\boldsymbol{w} + C \leq B$,[5] where $A = (a_{ij})_{m \times l}$ is a $m \times l$ matrix, $\boldsymbol{w} = (w_1, \ldots, w_l)'$ is a $l \times 1$ vector, $C = (c_1, \ldots, c_m)'$ and $B = (b, \ldots, b)'$ are both $m \times 1$ vectors.

[5] A convex polytope is formulated in this form for the convenience of the presentation of our approach in Sect. 3.

A convex polytope $P = \{\boldsymbol{w} : A\boldsymbol{w}+C \leq B\}$ has the following property, where the matrix A is full row rank.

Property 1. Let P be compact and its interior P° be not empty, then P and P° are both simply connected sets with the same boundary $\partial P = \{\boldsymbol{w} \in P : \vee_{i=1}^{m}[\sum_{j=1}^{l} a_{ij}w_j + c_i = b]\}$.

Based on Property 1, the following two lemmas can be obtained, which are further illustrated in Fig. 1.

Lemma 1. *Assume $P = \{\boldsymbol{w} : A\boldsymbol{w} + C \leq B\}$ is a compact convex polytope. If U is a compact set such that its boundary is a subset of the compact convex polytope P, then P is an over-approximation of the set U.*

Proof. Since U is a compact set, there exists $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{il})' \in U$ such that $\sum_{j=1}^{l} a_{ij}w_j + c_i$ reaches its maximum value $\mathtt{MAX}_i$ in U at this point, where $i = 1, \ldots, m$. Obviously, $U \subseteq P$ is equivalent to $\mathtt{MAX}_i \leq b$ for $i = 1, \ldots, m$. Thus it is enough to prove that $\mathtt{MAX}_i \leq b$ for $i = 1, \ldots, m$.

Assuming that there exists an index $i \in \{1, \ldots, m\}$ such that $\mathtt{MAX}_i > b$, we derive a contradiction as follows. Since $\partial U \subseteq P$ and $A\boldsymbol{w} + C \leq B$ for $\forall \boldsymbol{w} \in P$, then $\boldsymbol{y}_i \in U^\circ$. If $U^\circ = \emptyset$, a contradiction is obtained; Otherwise, let $\Omega = \{\boldsymbol{w} : A\boldsymbol{w} + C \leq \mathtt{MAX}\}$, where $\mathtt{MAX} = (\mathtt{MAX}_i, \ldots, \mathtt{MAX}_i)'$. By Property 1, we obtain that $\boldsymbol{y}_i \in \partial\Omega$. Thus for an arbitrary but fixed positive number ϵ, there exists $\boldsymbol{z} = (z_1, \ldots, z_l)' \in \mathbb{U}(\boldsymbol{y}_i; \epsilon)$ such that $\sum_{j=1}^{l} a_{ij}z_j + c_i > \mathtt{MAX}_i$. Also, since $\boldsymbol{y}_i \in U^\circ$, there exist $\epsilon_1 > 0$ and $\boldsymbol{w}_0 = (w_{01}, \ldots, w_{0l})' \in \mathbb{U}(\boldsymbol{y}_i; \epsilon_1) \subseteq U$ such that $\sum_{j=1}^{l} a_{ij}w_{0j} + c_i > \mathtt{MAX}_i$, contradicting the fact that $\sum_{j=1}^{l} a_{ij}w_j + c_i$ reaches its maximum $\mathtt{MAX}_i$ in U at the point $\boldsymbol{y}_i$. Thus, $\mathtt{MAX}_i \leq b$ for $i = 1, \ldots, m$. That is, P is an over-approximation of the set U.

Lemma 2. *Assume O is a simply connected compact set and $P = \{\boldsymbol{w} : A\boldsymbol{w} + C \leq B\}$ is a compact convex polytope. If the boundary of the set O is a subset of the enclosure of the complement of the polytope P, and the intersection of the interior of the set O and the interior of the set P is not empty, then the set P is an under-approximation of the set O.*

Proof. Since $P = \{\boldsymbol{w} : A\boldsymbol{w}+C \leq B\}$ is compact, P° and P are simply connected sets with the same boundary $\partial P = \{\boldsymbol{w} \in P : \vee_{i=1}^{m} \sum_{j=1}^{l} a_{ij}w_j + c_i = b\}$.

Assuming that $\boldsymbol{y} \in P$ is a point such that $\boldsymbol{y} \notin O$, we derive a contradiction as follows.

Case 1: $\boldsymbol{y} \in P^\circ$. Since $O^\circ \cap P^\circ \neq \emptyset$, there exists $\boldsymbol{y}_0 \in O^\circ \cap P^\circ$. Thus there exists a path q in P°, connecting $\boldsymbol{y}$ and $\boldsymbol{y}_0$. Due to the assumption that $\boldsymbol{y} \notin O$, there exists $\boldsymbol{y}_1 \in q$ such that $\boldsymbol{y}_1 \in \partial O$ and $\boldsymbol{y}_1 \in P^\circ$, contradicting the assumption that $\partial O \subseteq \overline{P^c}$.

Case 2: $\boldsymbol{y} \in \partial P$. Since $\boldsymbol{y} \notin O$ and O is compact, there exists a $\delta > 0$ such that $P^\circ \cap \mathbb{U}(\boldsymbol{y};\delta) \neq \emptyset$ and $\mathbb{U}(\boldsymbol{y};\delta) \cap O = \emptyset$. Thus there exists $\boldsymbol{z}_1$ such that $\boldsymbol{z}_1 \in P^\circ \cap \mathbb{U}(\boldsymbol{y};\delta)$ and $\boldsymbol{z}_1 \notin O$. Then, similar to the above case, a contradiction is derived.

Thus, we conclude that the set P is an under-approximation of the set O.

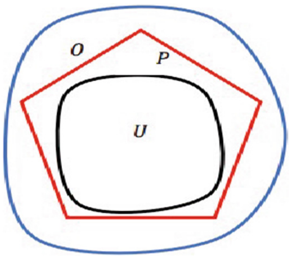

Fig. 1. An illustration for Lemmas 1 and 2. (blue curve – the boundary of the set O in Lemma 1; red curve – the boundary of the convex polytope P; black curve – the boundary of the set U in Lemma 2.) (Color figure online)

Based on the above two lemmas, an approach to compute a polytopic UAB is proposed in the section that follows.

3 Under-Approximating Backward Reachable Sets

In this section an approach is proposed to compute an UAB of a compact simply connected target region TR after the time duration T. The UAB is represented by a polytope.

3.1 Computing Under-Approximations

In this subsection an approach for computing an UAB of TR for the time duration T is detailed. The framework to compute an UAB of a simply connected compact set TR for the time duration T in our method involves the following steps,

1. a time grid $0 = t_0 < t_1 < \ldots < t_N = T$ is adopted with a step size h;
2. starting with $U_0 =$ TR, we compute a compact polytope U_1, which is an UAB of TR for the time duration h;
3. starting from the k^{th} UAB, we advance our approximation to a compact polytopic UAB U_{k+1};
4. U_N is what we want to obtain.

Assume that we have already obtained a compact polytope U_k, where U_k is an UAB of TR for the time duration t_k. A compact polytopic UAB for the time duration $k+1$ is constructed through the following steps:

(a) compute a set Ω_{k+1}, which is an union of a collection of intervals, such that $\partial\Omega_b(h; U_k, \boldsymbol{f}) \subseteq \Omega_{k+1}$, as discussed below;
(b) compute a compact polytope $O_{k+1} = \{\boldsymbol{x} : A\boldsymbol{x} + C \leq B\}$ such that $\Omega_{k+1} \subseteq O_{k+1}$;
(c) contract O_{k+1} to obtain $U_{k+1} = \{\boldsymbol{x} : A\boldsymbol{x}+C \leq B^u\}$ such that $\Omega_{k+1} \subseteq \overline{U_{k+1}^c}$ and $U_{k+1}^{\circ} \cap (\Omega_b(h; U_k, \boldsymbol{f}))^{\circ} \neq \emptyset$.

In order to prove that U_{k+1} obtained by the steps (a) $\sim$ (c) is also a simply connected compact set and is a subset of $\Omega_b(h; U_k, \boldsymbol{f})$, we first introduce a fundamental theorem behind our method based on the fact that $\boldsymbol{\phi}(t; \cdot) : \Omega_b(t; \Delta, \boldsymbol{f}) \mapsto \Delta$ is a homeomorphism between two topological spaces $(\Delta, \mathcal{T}_\Delta)$ and $(\Omega_b(t; \Delta, \boldsymbol{f}), \mathcal{T}_{\Omega_b(t;\Delta,\boldsymbol{f})})$.

Theorem 1. [28,29] *If $\Delta \subseteq \mathbb{R}^n$ is a simply connected compact set, then $\Omega_b(t; \Delta, \boldsymbol{f})$ is also a simply connected compact set and $\partial\Omega_b(t; \Delta, \boldsymbol{f}) = \Omega_b(t; \partial\Delta, \boldsymbol{f})$.*

Based on Theorem 1, we have the following lemma stating that U_{k+1} is a simply connected compact UAB of $\mathtt{U_k}$ for the time duration h.

Lemma 3. *If U_k is a simply connected compact set, then U_{k+1} obtained by our framework is also a simply connected compact set satisfying $U_{k+1} \subseteq \Omega_b(h; U_k, \boldsymbol{f})$.*

Proof. Since U_k is a simply connected compact set, $\Omega_b(h; U_k, \boldsymbol{f})$ is also a simply connected compact set according to Theorem 1. Also, since O_{k+1} in our framework is a simply connected compact set, we obtain that U_{k+1} is a simply connected compact set.

Regarding $\partial\Omega_b(h; U_k, \boldsymbol{f}) \subseteq \Omega_{k+1} \subseteq \overline{U_{k+1}^c}$ and $U_{k+1}^{\circ} \cap (\Omega_b(h; U_k, \boldsymbol{f}))^{\circ} \neq \emptyset$, we conclude that $U_{k+1} \subseteq \Omega_b(h; U_k, \boldsymbol{f})$ according to Lemma 2.

From Lemma 3, we can deduce that U_N is an UAB of TR for the time duration T, as stated in Theorem 2.

Theorem 2. *Given a nonlinear system of the form (1), if $U_0 =$ TR is a simply connected compact set, U_N obtained by our computational framework is an UAB of TR for the time duration $t = T$.*

In the sections that follow, we detail how to compute Ω_{k+1}, O_{k+1} and U_{k+1} in the steps (a) $\sim$ (c).

3.1.1 Computing Ω_{k+1} and O_{k+1}

In this subsection, we describe how to compute Ω_{k+1} and O_{k+1} in the steps (a) and (b) respectively in our computational framework.

Firstly, we introduce a proposition stating that the backward reachable set of System (1) can be obtained by computing the corresponding forward reachable set of its reverse system, as described in the following.

Proposition 1. [21] $\Omega_f(h;\mathcal{X},-\boldsymbol{f})$=$\Omega_b(h;\mathcal{X},\boldsymbol{f})$, *where* $\mathcal{X}\subseteq\mathbb{R}^n$.

From Proposition 1, we observe that $\Omega_f(h;U_k,-\boldsymbol{f})$ instead of $\Omega_b(h;U_k,\boldsymbol{f})$ can be used for performing computations in our computational framework, where $k=0,\ldots,N-1$. Thus, we can equivalently compute a set Ω_{k+1} such that $\partial\Omega_f(h;U_k,-\boldsymbol{f})\subseteq\Omega_{k+1}$. Also, the fact that the boundary of $\Omega_f(h;U_k,-\boldsymbol{f})$ corresponds to the boundary of U_k under the map $\boldsymbol{\phi}(h;\cdot)$ according to Theorem 1 is observed. Thus Ω_{k+1} is obtained based on ∂U_k. According to these observations, an approach to computing Ω_{k+1} is presented, as described in the following.

1. For a given ϵ_M, we use the interval Branch and Bound methods (e.g., [30]) to obtain a set of compact intervals $\{s_j,j=1,\ldots,M_k\}$ such that $\partial U_k\subseteq\cup_{j=1}^{M_k}s_j$, where M_k is the number of intervals and each interval s_j is of the form $[\underline{x}_1,\overline{x}_1]\times\ldots\times[\underline{x}_n,\overline{x}_n]$ satisfying $|\overline{x}_l-\underline{x}_l|\leq\epsilon_M$.
2. For $j=1,\ldots,M_k$, we use interval reachability analysis based methods (e.g., [24]) to obtain a compact interval I_j such that $\Omega_f(h;s_j,-\boldsymbol{f})\subseteq I_j$. Thus, $\Omega_{k+1}=\cup_{j=1}^{M_k}I_j$ is what we want.

The above procedure for computing Ω_{k+1} is denoted by $\texttt{Boundary}(h,U_k,\epsilon_M)$.

Remark 2. In the procedure $\texttt{Boundary}(h,U_k,\epsilon_M)$, ϵ_M is used to restrict the size of boxes enclosing ∂U_k. As ϵ_M becomes smaller, the volume of the obtained boxes becomes smaller and the resulting Ω_{k+1} becomes less conservative, but the computational burden increases.

The procedure $\texttt{Boundary}(h,U_k,\epsilon_M)$ for computing Ω_{k+1} is illustrated through the following example.

Example 1. Consider a model of an electromechnical oscillation of s synchronous machine [31],

$$\begin{cases}\dot{x_1}=x_2\\ \dot{x_2}=0.2-0.7sinx_1-0.05x_2\end{cases},$$

where $\texttt{TR}=[-0.1,0.1]\times[2.9,3.1]$ and $T=0.5$.

Computing Ω_1 when $h=0.5$ and $\epsilon_M=0.05$ is illustrated in Fig. 2.

Next, we compute a convex hull O_{k+1} such that $O_{k+1}\supseteq\Omega_{k+1}$, where $\Omega_{k+1}=\cup_{j=1}^{M_k}I_j$. Let v_j be the set of vertices of the interval I_j and $v=\cup_{j=1}^{M_k}v_j$. We get a polytope $O_{k+1}=\{\boldsymbol{x}:A\boldsymbol{x}+C\leq B\}$ of v using convex hull algorithm (e.g., [33]), where $A=(a_{ij})_{m\times n}$ and $B=(b,\ldots,b)'$. This procedure for computing O_{k+1} is denoted by $\texttt{Polytope}(\Omega_{k+1})$.

Since I_j is compact for $j=1,\ldots,M_k$, v is a bounded set, and as a consequence O_{k+1} is bounded and thus compact. Also, since every box I_j is also a convex hull of v_j, every $\boldsymbol{x}\in I_j$ can be formulated as $\sum_{l=1}^{2^n}\lambda_l\boldsymbol{v}_{j,l}$, where $\boldsymbol{v}_{j,l}\in v_j$, $\lambda_l\geq 0$ for $l=1,\ldots,2^n$ and $\sum_{l=1}^{2^n}\lambda_l=1$. Thus $\boldsymbol{x}\in O_{k+1}$ holds, implying that $\cup_{j=1}^{M_k}I_j\subseteq O_{k+1}$. Now we conclude that O_{k+1} in the step (b) is computed.

Remark 3. According to Lemma 1 in Subsect. 2.2, the convex hull O_{k+1} is an over-approximation of the backward reachable set of U_k for the time duration h.

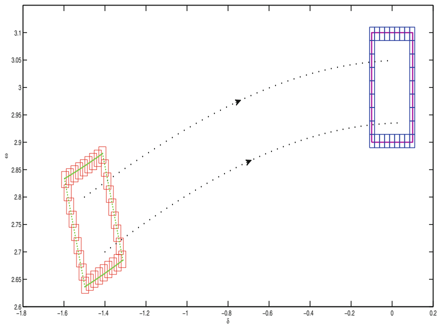

Fig. 2. An illustration for computing Ω_1. (red boxes – Ω_1 including $\partial\Omega_b(T;\mathtt{TR},\boldsymbol{f})$; green points – $\partial\Omega_b(T;\mathtt{TR},\boldsymbol{f})$ obtained by simulation methods; black points – some simulation trajectories originating from $\Omega_b(T;\mathtt{TR},\boldsymbol{f})$ over the time interval $[0,0.5]$; purple curve – $\partial\mathtt{TR}$; blue boxes – $\cup_j s_j$ including ∂TR.) (Color figure online)

3.1.2 Computing an Under-Approximation U_{k+1}

This section focuses on computing a polytopic under-approximation U_{k+1} (step (c) in our computational framework) by solving linear programming problems.

After obtaining $\Omega_{k+1} = \cup_{j=1}^{M_k} I_j$ and $O_{k+1} = \{\boldsymbol{x} : A\boldsymbol{x} + C \leq B\}$ in steps (a) and (b) based on the method in Subsect. 3.1, we shrink O_{k+1} to yield U_{k+1} by solving linear programming problems. The computations consist of two steps, as described below.

1. For $j = 1,\ldots,M_k$, we solve the following linear optimization problem:

$$\begin{aligned} \text{minimize} \quad & b_j \\ \text{s. t.} \quad & A\boldsymbol{x} + C \leq B_j, \\ & b_j \leq b, \\ & \boldsymbol{x} \in I_j, \end{aligned} \tag{2}$$

   where $B_j = (b_j,\ldots,b_j)'$. Since $b_j \leq b$, we can obtain that $\{\boldsymbol{x} : A\boldsymbol{x} + C \leq B_j\} \subseteq \{\boldsymbol{x} : A\boldsymbol{x} + C \leq B\}$.
2. We denote $min\{b_j, j = 1,\ldots,M_k\}$ by b^u and $(b^u,\ldots,b^u)'$ by B^u respectively. If $\{\boldsymbol{x} : A\boldsymbol{x} + C \leq B^u\} \neq \emptyset$, it is denoted by U_{k+1}. The case that U_{k+1} is empty is discussed in Sect. 4. Note that U_{k+1} is just a candidate of what we want.

The above procedure for U_{k+1} is denoted by $\mathtt{Contraction}(\Omega_{k+1}, O_{k+1})$, which is illustrated in the following example.

Example 2. For Example 1, computing U_1 when $\epsilon_M = 0.05$ and $h = 0.5$ is illustrated in Fig. 3, where $T = 0.5$.

Since $U_{k+1} \subseteq O_{k+1}$, U_{k+1} is compact. However, we cannot conclude that U_{k+1} is an UAB of U_k for the time duration h. In order to further ensure that U_{k+1} is an under-approximation of $\Omega_b(h;U_k,\boldsymbol{f})$, we need to verify whether U_{k+1}

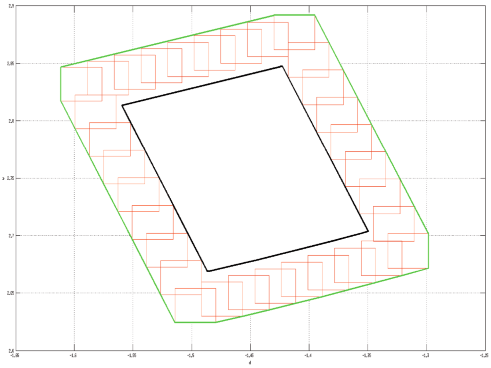

Fig. 3. An illustration for computing Ω_1. (red boxes – Ω_1 including $\partial\Omega_b(T;\mathtt{TR},\boldsymbol{f})$; green curve – ∂O_1; black curve – ∂U_1.) (Color figure online)

satisfies the condition as described in the step (c) in our computational framework, i.e., verify whether $\Omega_{k+1}\subseteq\overline{U^c_{k+1}}$ and $U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ}\neq\emptyset$ holds.

For the constraint $\Omega_{k+1}\subseteq\overline{U^c_{k+1}}$, we can ensure it by the following lemma.

Lemma 4. *$\Omega_{k+1}\subseteq\overline{U^c_{k+1}}$, where Ω_{k+1} and U_{k+1} are respectively obtained based on the procedures* $\mathtt{Boundary}(h,U_k,\epsilon_M)$ *and* $\mathtt{Contraction}(\Omega_{k+1},O_{k+1})$*.*

Proof. Since $U_{k+1}=\{\boldsymbol{x}:A\boldsymbol{x}+C\leq B^u\}$, where $A=\big(a_{ij}\big)_{m\times n}$, $C=(c_1,\ldots,c_m)'$, $B^u=(b^u,\ldots,b^u)'$, $b^u=\min\{b_j,j=1,\ldots,M_k\}$ and b_j is obtained by solving the optimization problem (2), we can obtain that for every $\boldsymbol{x}=(x_1,\ldots,x_n)'\in\cup_{j=1}^{M_k}I_j$, there exists an index $i\in\{1,\ldots,m\}$ such that $\sum_{j=1}^{n}a_{ij}x_j+c_i\geq b^u$, implying that $\boldsymbol{x}\notin\{\boldsymbol{x}:A\boldsymbol{x}+C<B^u\}$. Thus, $\Omega_{k+1}=\cup_{j=1}^{M_k}I_j\subseteq\overline{U^c_{k+1}}$.

In order to check whether $U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ}\neq\emptyset$ holds, we first take a point $\boldsymbol{x}\in U^{\circ}_{k+1}=\{\boldsymbol{x}:A\boldsymbol{x}+C<B^u\}$, then apply interval methods (e.g., [24]) to get an interval enclosure $s_{\boldsymbol{x}}$ of $\boldsymbol{\phi}(h;\boldsymbol{x})$, and check whether $s_{\boldsymbol{x}}\subseteq U_k$ holds. If the answer is positive, $U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ}\neq\emptyset$ holds, as stated in Lemma 5. The procedure for checking $U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ}\neq\emptyset$ is denoted by $\mathtt{Verification}(U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ})$.

Lemma 5. *If $s_{\boldsymbol{x}}\subseteq U_k$, then $\boldsymbol{x}$[6]$\in U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ}$ holds, where $s_{\boldsymbol{x}}$ and U_{k+1} are respectively computed based on the procedures* $\mathtt{Verification}(U^{\circ}_{k+1}\cap(\Omega_b(h;U_k,\boldsymbol{f}))^{\circ})$ *and* $\mathtt{Contraction}(\Omega_{k+1},O_{k+1})$*.*

[6] Although $\boldsymbol{x}$ can be an arbitrary point belonging to U°_{k+1}, $\boldsymbol{x}$ has to be a point being away from ∂U_{k+1} due to the fact that $s_{\boldsymbol{x}}$ is an interval box rather a point and $s_{\boldsymbol{x}}\subseteq U_k$. This can be done by taking $\boldsymbol{x}$ being in $\{\boldsymbol{x}:A\boldsymbol{x}+C\leq B^u-\delta\}$, where $\delta>0$.

Proof. Since $s_{\boldsymbol{x}} \subseteq U_k$, $\boldsymbol{x} \in \Omega_f(h; U_k, -\boldsymbol{f})$ and thus $\boldsymbol{x} \in \Omega_b(h; U_k, \boldsymbol{f})$ holds. Also, according to the fact that $\partial\Omega_b(h; U_k, \boldsymbol{f}) \subseteq \Omega_{k+1}$ and $\Omega_{k+1} \subseteq \overline{U^c_{k+1}}$, we obtain that $U^{\circ}_{k+1} \cap \partial\Omega_b(h; U_k, \boldsymbol{f}) = \emptyset$, implying that $\boldsymbol{x} \notin \partial\Omega_b(h; U_k, \boldsymbol{f})$. Thus, $\boldsymbol{x} \in U^{\circ}_{k+1} \cap (\Omega_b(h; U_k, \boldsymbol{f}))^{\circ}$.

Thus, if the boolean value returned by $\texttt{Verification}(U^{\circ}_{k+1} \cap (\Omega_b(h; U_k, \boldsymbol{f}))^{\circ})$ is true, i.e., $U^{\circ}_{k+1} \cap (\Omega_b(h; U_k, \boldsymbol{f}))^{\circ} \neq \emptyset$, then U_{k+1} obtained by the procedure $\texttt{Contraction}(\Omega_{k+1}, O_{k+1})$ is an `UAB` of $\Omega_b(h; U_k, \boldsymbol{f})$.

Remark 4. In the procedure $\texttt{Contraction}(\Omega_{k+1}, O_{k+1})$, $|\frac{b-b^u}{b-d}|$ can be used to evaluate the obtained `UAB` U_k, where d is the supremum such that $\{\boldsymbol{x} : A\boldsymbol{x} + C < D\} = \emptyset$ and $D = (d, \dots, d)'$[7]. As it approaches one, the under-approximation becomes increasingly conservative.

Thus our approach for computing a compact polytopic `UAB` is elucidated. We formally formulate our approach for computing an `UAB` of `TR` for the time duration T as Algorithm 1.

Algorithm 1. Computing an Under-Approximation

Input: Given system (1), a target region: `TR`, a time duration: T, a time step h such that $\frac{T-0}{h} \geq 1$ is an integer, ϵ_M: the size of intervals enclosing the boundaries, and ϵ: local error bounds.
Output: an `UAB` of `TR` for the time duration T.

$U_0 :=$ `TR`;
for $i = 0 : 1 : N-1$ **do**
  $\Omega_{i+1} := \texttt{Boundary}(h, U_i, \epsilon_M)$;
  $O_{i+1} := \texttt{Polytope}(\Omega_{i+1})$;
  $U_{i+1} := \texttt{Contraction}(\Omega_{i+1}, O_{i+1})$;
  if $\texttt{Verification}(U^{\circ}_{i+1} \cap (\Omega_b(h; U_i, \boldsymbol{f}))^{\circ})$ is false or $|\frac{b-b^u}{b-d}| > \epsilon$ **then**
    return "failed to obtain an `UAB`" and terminate;
  end if
end for
return an `UAB` U_N.

Remark 5. Our method, as formalised in Algorithm 1, can be applied to under-approximate forward reachable sets by performing forward computations on initial sets.

In order to enhance the understanding of our approach, an example is employed to illustrate Algorithm 1 as follows.

Example 3. Consider a model of an electromechanical oscillation of s synchronous machine,

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = 0.2 - 0.7 sin x_1 - 0.05 x_2 \end{cases},$$

where `TR` $= [-0.1, 0.1] \times [2.9, 3.1]$ and $T = 3$.

[7] d can be obtained by solving the linear program: min d, s.t., $A\boldsymbol{x} + C \leq D$.

Let $h = 3$, $\epsilon_M = 0.0001$ and $\epsilon = 0.5$. Firstly, we compute $\Omega_1 = \cup_j I_j$ such that $\partial\Omega_b(T; \mathtt{TR}, \boldsymbol{f}) \subseteq \Omega_1$ based on the procedure $\mathtt{Boundary}(h, \mathtt{TR}, \epsilon_M)$ in Subsect. 3.1, where I_j is of the interval form. Secondly, we compute O_1 based on the procedure $\mathtt{Polytope}(\Omega_1)$ in Subsect. 3.1 such that $\Omega_1 \subseteq O_1$. Thirdly, we contract O_1 to obtain U_1 based on the procedure $\mathtt{Contraction}(\Omega_1, O_1)$ in Subsect. 3.1. Finally, we find a point $\boldsymbol{x} = (-8.08, 2.52) \in U_1^\circ$ and obtain $s_{\boldsymbol{x}} = [0.0082, 0.0083] \times [3.0181, 3.0182]$ based on the procedure $\mathtt{Verification}(U_1^\circ \cap (\Omega_b(h; \mathtt{TR}, \boldsymbol{f}))^\circ)$ in Subsect. 3.1. Since $s_{\boldsymbol{x}} \subseteq \mathtt{TR}$ and $|\frac{b-b^u}{b-d}| \approx 0.246621 \leq \epsilon$, where $b = 0$, $b^u = -0.008260$ and $d = -0.0334927$, U_1 is an $\mathtt{UAB}$ of $\mathtt{TR}$ for the time duration $T = 3$. The boundary of U_1 is depicted in Fig. 4.

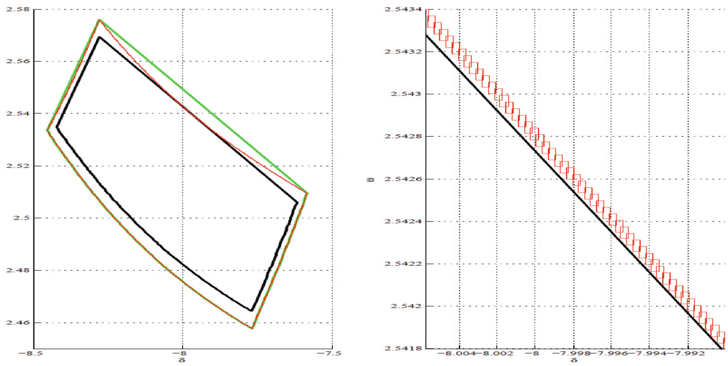

Fig. 4. An $\mathtt{UAB}$ for Example 3. (left: red boxes – Ω_1 including $\partial\Omega_b(3; \mathtt{TR}, \boldsymbol{f})$; green curve – ∂O_1; black curve – ∂U_1; right: a zoomed-in portion of the left figure.) (Color figure online)

3.2 Computational Complexity

In this subsection, the computational complexity of Algorithm 1 is discussed briefly. In the k^{th} step, the branch-and-bound method for the problem of yielding some interval subdivisions to enclose ∂U_k is of exponential complexity $\mathcal{O}(\xi^n)$, where $\xi = \mathcal{O}(\frac{1}{\epsilon_M})$. The underlying interval Taylor series method is of polynomial complexity: the work is $\mathcal{O}(p^2)$ to compute the Taylor coefficients, where p is the order of the used Taylor expansion, and $\mathcal{O}(n^3)$ for performing linear algebra [32]. The complexity of applying simplex algorithms to solve the linear program (2) is $\mathcal{O}(nm_k)$ generally, where m_k is the number of linear constraints. The computational complexity of the convex hull algorithm (e.g., [33]) is $\mathtt{Con_k} = \mathcal{O}(2^n M_k\ logr)$ for $n \leq 3$ and $\mathcal{O}(2^n M_k f_r/r + f_r)$ when $n > 3$, where $r \leq 2^n M_k$ is the number of vertices of O_{k+1}, $f_r = \mathcal{O}(r^{\lfloor \frac{n}{2} \rfloor}/\lfloor \frac{n}{2} \rfloor !)$ and $\lfloor \frac{n}{2} \rfloor$ is the floor function of $\frac{n}{2}$. Therefore, the total computational complexity of our method is $\sum_{k=0}^{N-1} \left(\mathcal{O}(\xi_k^n) + M_k(\mathcal{O}(p^2) + \mathcal{O}(n^3)) + M_k\mathcal{O}(nm_k) + \mathtt{Con_k}\right)$.

4 Examples, Discussions and Comparisons

Our approach is implemented based on the floating point linear programming solver GLPK running the Simplex algorithm and the validated ordinary differential equation solver VNODE-LP [24]. We evaluate it using five examples and compare it with the method of Korda et al. [22]. The results for Examples 4–7 can be found in Figs. 5, 6, 7 and 8 respectively. Table 1 presents details on parameters that control our approach. All these computations are performed on an i5-3337U 1.8 GHz CPU with 4 GB RAM running Ubuntu Linux 13.04.

4.1 Examples and Discussions

In this subsection our approach is evaluated using Examples 4–8, and parameters that control our approach are discussed using the first four examples. The results are illustrated in Figs. 4, 5, 6 and 7. Regarding the computational complexity analysis in Subsect. 3.2, our approach suffers from dimensional curse. In order to overcome this problem, we explore some future directions to make our approach more practical through Example 8.

Table 1. Performance of Algorithm 1 on Examples. Each benchmark is indexed by its example number. `TR`: target region, ϵ_M: bound for the size of intervals in the procedure `Boundary`(h, U_k, ϵ_M); ϵ: bound for $|\frac{b-b^u}{b-d}|$ in the procedure `Contraction`(Ω_{k+1}, O_{k+1}); h: step size; T :a specified time duration for `UAB`; Time: CPU time cost (seconds).

Ex	`TR`	ϵ_M	ϵ	h	T	Time
4	$[-0.1, 0.1] \times [-0.1, 0.1]$	0.001	0.5	0.5	10	34.29
4	$[-0.1, 0.1] \times [-0.1, 0.1]$	0.0002	0.5	0.5	10	266.58
5	$[0.3, 0.4] \times [0.5, 0.7]$	0.001	0.5	0.05	1.1	55.23
5	$[0.3, 0.4] \times [0.5, 0.7]$	0.0002	0.5	0.05	1.1	410.13
6	$[1.2, 1.5] \times [0.8, 1.1]$	0.001	0.5	0.5	10	23.04
6	$[1.2, 1.5] \times [0.8, 1.1]$	0.0001	0.5	0.5	10	911.40
7	$x_i \in [-0.1, 0.1]$, $i = 1, \ldots, 3$	0.003	0.5	0.5	2.5	450.32
7	$x_i \in [-0.1, 0.1]$, $i = 1, \ldots, 3$	0.003	0.5	2.5	2.5	66.56
8	$x_i \in [-0.015, 0.001]$, $i = 1, \ldots, 7$	0.016	0.5	0.01	0.2	0.67

Example 4. Consider the system in Example 1 again

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = 0.2 - 0.7 sin x_1 - 0.05 x_2 \end{cases}.$$

Example 5. Consider the Brusselator model [10],

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

Example 6. Consider the Van-der-Pol system,

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

Example 7. Consider the 3D-Lotka-Volterra System,

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

Note that $\Omega_b(2.5; \mathtt{TR}, \boldsymbol{f}) \subseteq O_1$ in Fig. 8 according to Remark 3.

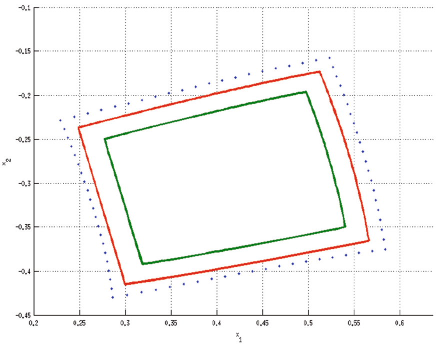

Fig. 5. ∂UAB for Example 4.(blue points – $\partial\Omega_b(10; \mathtt{TR}, \boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – ∂U_{20} when $\epsilon_M = 0.0002$; green curve – ∂U_{20} when $\epsilon_M = 0.001$.) (Color figure online)

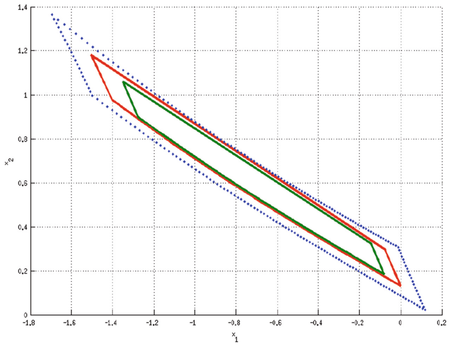

Fig. 6. ∂UAB for Example 5. (blue points – $\partial\Omega_b(1.1; \mathtt{TR}, \boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – ∂U_{22} when $\epsilon_M = 0.0002$; green curve – ∂U_{22} when $\epsilon_M = 0.001$.) (Color figure online)

From the above four examples, we first observe that polytopes can represent reachable sets well for some nonlinear systems, e.g., Examples 4–7. Also, we observe that (1) when h is fixed, the resulting UAB becomes less conservative as ϵ_M becomes smaller (Examples 4–6); (2) when ϵ_M is fixed, a smaller h may lead to large errors. The underlying reason is that the under-approximation error in every iterative step will propagate through the computations (Example 7), similar to the well known wrapping effect in over-approximating reachable sets. The errors in the construction of under-approximations of reachable sets using our method result from three parts in every iteration. The first one is the computation of interval boxes enclosing the boundary of the target region. The second one is the computation of interval boxes enclosing the boundary of the

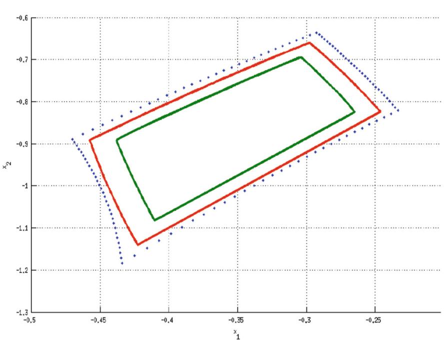

Fig. 7. ∂UAB for Example 6. (blue points – $\partial\Omega_b(10; \texttt{TR}, \boldsymbol{f})$ obtained by Runge-Kutta methods; red curve – ∂U_{20} when $\epsilon_M = 0.0001$; green curve - ∂U_{20} when $\epsilon_M = 0.001$.) (Color figure online)

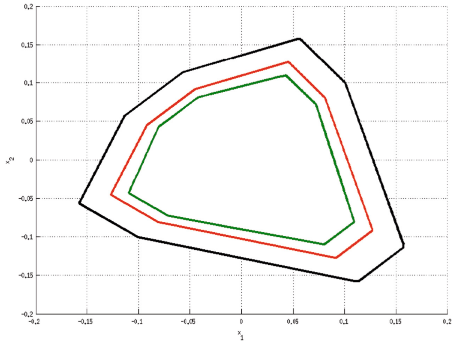

Fig. 8. ∂UAB for Example 7. (black curve – ∂O_1 when $h = 2.5$; red curve – ∂U_1 when $h = 2.5$; green curve – ∂U_5 when $h = 0.5$.) (Color figure online)

backward reachable set based on the interval Taylor-series method and the last one is the computation of an polytopic under-approximation. It is well known that reachable sets of nonlinear systems are in general far from being convex, the last one contributes to the total error mainly. Especially, for the case that the returned under-approximation is empty in some iterative step, we could try a smaller ϵ_M and/or a different time step h. A smaller ϵ_M, which mitigates the error from the first source, will help to obtain a tighter Ω_{k+1}, eventually leading to a less conservative UAB. However, the computational cost increases. Therefore, in order to obtain a tighter Ω_{k+1}, reachability analysis methods which better control the wrapping effect should be considered (e.g., [10,27]). This corresponds to the reduction of the error from the second source. As to the last error source resulting from polytopic approximations, an under-approximation of the semi-algebraic form instead of the polytopic form will be contemplated in our future study.

Example 8. Consider a seven-domensional biological system[8],

$$\begin{cases} \dot{x}_1 = -0.4x_1 + 5x_3x_4 \\ \dot{x}_2 = 0.4x_1 - x_2 \\ \dot{x}_3 = x_2 - 5x_3x_4 \\ \dot{x}_4 = 5x_5x_6 - 5x_3x_4 \\ \dot{x}_5 = -5x_5x_6 + 5x_3x_4 \\ \dot{x}_6 = 0.5x_7 - 5x_5x_6 \\ \dot{x}_7 = -0.5x_7 + 5x_5x_6 \end{cases}.$$

[8] The model is from http://ths.rwth-aachen.de/research/hypro/biological-model-i/.

Using an interval hull rather than a convex hull in every iterative step of Algorithm 1, we obtain that an `UAB` for the time duration $t = 0.2$ is $[-0.0152, 0.000] \times [-0.0169, 0.0011] \times [-0.0140, 0.0030] \times [-0.0141, 0.0001] \times [-0.0141, 0.0001] \times [-0.0138, 0.0014] \times [-0.0155, 0.000]$.

From Example 8, we observe that our approach scales well to systems with a large number of variables by using an interval hull instead of a convex hull in every iteration. However, this results in more conservative results, compared to that based on polytopic representations. In order to reduce the conservativeness brought by interval representations, while making our approach scale well, we will explore using oriented rectangular hulls [25], zonotopes [15] or symbolic orthogonal projections [34] to construct under-approximations in our future work. Furthermore, regarding that the boundary of a polytope is piecewise of the zonotope form, therefore the exact boundary of the polytope rather than interval subdivisions enveloping it obtained by Branch and Bound methods in every iteration can be used for computations directly using methods in [11,27], thereby reducing the computational cost and further improving the scalability of our method.

4.2 Comparisons

In this section we will compare our method with the method of Korda et al. [22]. Due to a lot of input parameters such as sum-of-squares multipliers being coordinated in the method of Korda et al. [22], it is not trivial to find an optimal combination, thereby making fair comparisons difficult. Therefore, we try to explore some potential benefits of our method by comparing with this method.

Firstly, the method of Korda et al. [22] 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. In contrast, our method is not restricted to polynomial dynamical systems. That is, our method can deal with more general nonlinear systems such as Example 4 in Subsect. 4.1. Secondly, we compare the performances of the two methods based on Examples 5–8. Assume that the specified constraint sets for the four examples are $\{\boldsymbol{x} : 1.25^2 - (x_1 + 0.75)^2 - (x_2 - 0.65)^2 \geq 0\}$, $\{\boldsymbol{x} : 4 - x_1^2 - x_2^2 \geq 0\}$, $\{\boldsymbol{x} : 0.04 - x_1^2 - x_2^2 - x_3^2 \geq 0\}$ and $\{\boldsymbol{x} : 0.0125^2 - \sum_{i=1}^{7}(x_i + 0.0075)^2 \geq 0\}$ respectively. Actually, they are respectively the over-approximations of backward reachable sets of the target regions for these four examples. Using the method of Korda et al. [22], we can not obtain feasible solutions to any of the above examples based on the sum-of-squares programming solver YALMIP [35] with Sedumi [36]. Since there are a lot of sum-of-squares multipliers that are coordinated in advance, their degrees should be determined in advance for computations, improper mixing will result in unreliable results. The main underlying reason is that the present status of semi-definite programming solvers is not so advanced, as pointed out in [37]. The numerical problems produced by these solvers often result in unreliable results for some cases. We use Example 5 to illustrate this.

Although the solver YALMIP returns a "feasible" solution as shown in Fig. 9 for some mixing of sum-of-squares multipliers, the result is incorrect actually. On the contrary, our method relies on Interval methods to locate the boundary of the backward reachable set and linear programs to obtain an under-approximation in every iterative step, making our method more reliable.

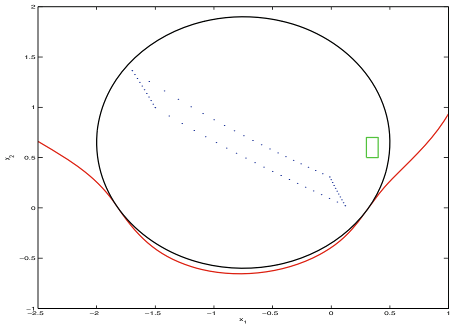

Fig. 9. An incorrect UAB for Example 5 obtained by the method of Korda et al. [22] due to numerical problems. (black curve – $\{\boldsymbol{x} : 1.25^2 - (x_1 + 0.75)^2 - (x_2 - 0.65)^2 \geq 0\}$; red curve – the boundary of an incorrect under-approximation of $\Omega_b(1.1, \mathtt{TR}, \boldsymbol{f})$; green curve - $\partial\mathtt{TR}$; blue points – $\partial\Omega_b(1.1, \mathtt{TR}, \boldsymbol{f})$ obtained by Runge Kutta methods.) (Color figure online)

5 Conclusion

Given a nonlinear system and a target region of the simply connected compact type, we in this paper proposed a method by performing boundary analysis to obtain an UAB of the target region for a specified time duration. The UAB is represented as a polytope. The polytope can be obtained by combining validated numerical methods for ordinary differential equations and linear programs. Numerical results and comparisons with the method of Korda et al. [22] based on five examples were given to illustrate the benefits of our approach. The results show that our method has some significant benefits in under-approximating reachable sets for some cases. Furthermore, we explore some directions toward improving the scalability of our method.

Extending our method to compute under-approximations of reachable sets for nonlinear systems with time delay (e.g., [38]) is considered in our future work. Moreover, computing a bounded error approximation of the solution over a bounded time is another interesting investigation towards addressing under-approximation problems [39].

Acknowledgements. The authors are grateful to Prof. Martin Fränzle from Carl von Ossietzky Universität Oldenburg, Mr. Milan Korda from École Polytechnique Fédérale de Lausanne, Dr. Xin Chen from University of Colorado at Boulder for helpful discussions. Also, the authors are especially grateful to the anonymous reviewers for their valuable comments.

References

1. Xue, B.: Computing rigor quadratic lyapunov functions and underapproximate reachable sets for ordinary differential equations. Doctoral dissertation, Beihang University (2013)
2. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation-based abstraction refinement. ACM Trans. Embed. Comput. Syst. **6**, 1–23 (2007)
3. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Hybrid systems: from verification to falsification. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 463–476. Springer, Heidelberg (2007)
4. Herrero, P., Calm, R., Vehí, J., Armengol, J., Georgiou, P., Oliver, N., Tomazou, C.: Robust fault detection system for insulin pump therapy using continuous glucose monitoring. J. Diabetes Sci. Technol. **6**, 1131–1141 (2012)
5. Xue, B., Easwaran, A., Cho, N.: Towards robust artificial pancreas based on reachability analysis techniques. In: Workshop on Medical Cyber-Physical Systems (2015)
6. Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Trans. Robot. **30**, 1–16 (2014)
7. Alur, R., Dang, T., Ivančić, F.: Progress on reachability analysis of hybrid systems using predicate abstraction. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 4–19. Springer, Heidelberg (2003)
8. Asarin, E., Dang, T., Girard, A.: Hybridization methods for the analysis of nonlinear systems. Acta Inf. **43**(7), 451–476 (2007)
9. Huang, Z., Mitra, S.: Proofs from simulations and modular annotations. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014), pp. 183–192. ACM, New York (2014)
10. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: Proceedings of the 2012 IEEE 33rd Real-Time Systems Symposium (RTSS 2012), pp. 183–192. IEEE Computer Society, Washington (2012)
11. Althoff, M.: Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC 2013), pp. 173–182. ACM, New York (2013)
12. Revert, A., Calm, R., Vehi, J., Bondia, J.: Calculation of the best basal-bolus combination for postprandial glucose control in insulin pump therapy. IEEE Trans Biomed. Eng. **58**, 274–281 (2011)
13. Ratschan, S., She, Z.: Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions. SIAM J. Control Optim. **48**(7), 4377–4394 (2010)
14. Kurzhanski, A.B., Varaiya, P.: Ellipsoidal techniques for reachability analysis: internal approximation. Syst. Control Lett. **41**, 201–211 (2000)

15. Girard, A., Le Guernic, C., Maler, O.: Efficient computation of reachable sets of linear time-invariant systems with inputs. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 257–271. Springer, Heidelberg (2006)
16. Maidensa, J.N., Kaynamaa, S., Mitchell, I.M., Oishic, M.K., Dumonta, G.A.: Lagrangian methods for approximating the viability kernel in high-dimensional systems. Automatica **49**, 2017–2029 (2013)
17. Benvenuti, L., Bresolin, D., Casagrande, A., Collins, P., Ferrari, A., Mazzi, E., Sangiovanni-Vincentelli, A., Villa, T.: Reachability computation for hybrid systems with Ariadne. In: Proceedings of the 17th IFAC World Congress, vol. 41, pp. 8960–8965. IFAC Papers-OnLine (2008)
18. Goldsztejn, A., Jaulin, L.: Inner approximation of the range of vector-valued functions. Reliable Comput. **14**, 1–23 (2010)
19. Mullier, O., Goubault, E., Kieffer, M., Putot, S.: General inner approximation of vector-valued functions. Reliable Comput. **18**, 117–143 (2013)
20. Goubault, E., Mullier, O., Putot, S., Kieffer, M.: Inner approximated reachability analysis. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC 2014), pp. 163–172. ACM, New York (2014)
21. Mitchell, I.M.: Comparing forward and backward reachability as tools for safety analysis. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 428–443. Springer, Heidelberg (2007)
22. Korda, M., Henrion, D., Jones, N.C.: Inner approximations of the region of attraction for polynomial dynamical systems. In: Proceedings of 9th IFAC Symposium on Nonlinear Control Systems, pp. 534–539 (2013)
23. Chen, X., Sankaranarayanan, S., Ábrahám, E.: Under-approximate flowpipes fornon-linear continuous systems. In: Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design (FMCAD 2014), pp. 59–66. IEEE (2014)
24. Nedialkov, N.S.: 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, Hamilton, Canada, L8S4K1 (2006). VNODE-LP is available at www.cas.mcmaster.ca/nedialk/vnodelp/
25. Stursberg, O., Krogh, B.H.: Efficient representation and computation of reachable sets for hybrid systems. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 482–497. Springer, Heidelberg (2003)
26. Testylier, R., Dang, T.: NLTOOLBOX: a library for reachability computation of nonlinear dynamical systems. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 469–473. Springer, Heidelberg (2013)
27. Eggers, A., Ramdani, N., Nedialkov, N.S., Fränzle, M.: Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. Softw. Syst. Model. **14**, 121–148 (2015)
28. Massey, W.S.: A Basic Course in Algebraic Topology. Springer, New York (1991). Corollary 6.7
29. Khalil, H.K.: Nonlinear Systems, 3rd edn, p. 188. Prentice Hall, Upper Saddle River (2002)
30. Granvilliers, L., Benhamou, F.: Realpaver: an interval solver using constraint satisfaction techniques. ACM TOMS **32**(1), 138–156 (2006)
31. Susuki, Y., Koo, T.J., Ebina, H., Yamazaki, T., Ochi, T., Uemura, T., Hikihara, T.: A hybrid system approach to the analysis and design of power grid dynamic performance. Proc. IEEE **100**, 225–239 (2012)
32. Ramdani, N., Nedialkov, N.S.: Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. Nonlinear Anal. Hybrid Syst. **5**, 149–162 (2011)

33. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**, 469–483 (1996)
34. Hagemann, W.: Reachability analysis of hybrid systems using symbolic orthogonal projections. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 407–423. Springer, Heidelberg (2014)
35. Löfberg, J.: YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference, Taipei, Taiwan, pp. 284–289 (2004)
36. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11**, 625–653 (1999)
37. Wang, T., Lall, S., West, M.: Polynomial level-set method for polynomial system reachable set estimation. IEEE Trans. Autom. Control **58**(10), 2508–2521 (2013)
38. Zou, L., Fränzle, M., Zhan, N., Mosaad, P.N.: Automatic verification of stability and safety for delay differential equations. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9207, pp. 338–355. Springer, Heidelberg (2015)
39. Majumdar, R., Prabhu, V.S.: Computing distances between reach flowpipes. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control (HSCC 2016), pp. 267–276. ACM, New York (2016)