Information and Computation 289 (2022) 104965

Contents lists available at ScienceDirect

Information and Computation

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

Encoding inductive invariants as barrier certificates: Synthesis via difference-of-convex programming ☆

Qiuye Wang [a,b], Mingshuai Chen [c,*], Bai Xue [a,b], Naijun Zhan [a,b,d,**], Joost-Pieter Katoen [c]

[a] State Key Laboratory of Computer Science, Institute of Software, CAS, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China
[c] RWTH Aachen University, Aachen, Germany
[d] Science and Technology on Integrated Information System Laboratory, Institute of Software, CAS, Beijing, China

ARTICLE INFO

Article history:
Received 15 September 2021
Received in revised form 17 April 2022
Accepted 18 September 2022
Available online 22 September 2022

Keywords:
Barrier certificates
Inductive invariants
Bilinear matrix inequalities
Difference-of-convex programming
Semidefinite programming

ABSTRACT

We present the *invariant barrier-certificate condition* that witnesses unbounded-time safety of differential dynamical systems. The proposed condition is the weakest possible one to attain inductive invariance. We show that discharging the invariant barrier-certificate condition —thereby synthesizing invariant barrier certificates— can be encoded as solving an *optimization problem subject to bilinear matrix inequalities* (BMIs). We further propose a synthesis algorithm based on difference-of-convex programming, which approaches a local optimum of the BMI problem via solving *a series of convex optimization problems*. This algorithm is incorporated in a branch-and-bound framework that searches for the global optimum in a divide-and-conquer fashion. We present a weak completeness result of our method, namely, a barrier certificate is guaranteed to be found (under some mild assumptions) whenever there exists an inductive invariant (in the form of a given template) that suffices to certify safety. Experimental results on benchmarks demonstrate the effectiveness and efficiency of our approach.

1. Introduction

Hybrid systems are mathematical models that capture the interaction between continuous physical dynamics and discrete switching behaviors, and hence are widely used in modeling cyber-physical systems (CPS). These CPS may be complex and safety-critical, with sensitive variables of the environment in its sphere of control. Everyday examples include process control at all scales, ranging from household appliances to nuclear power plants, or embedded systems in transportation domain, such as autonomous driving maneuvers in automotive, aircraft collision-avoidance protocols in avionics, or automatic train control applications, as well as a broad range of devices in health technologies, such as cardiac pacemakers.

☆ This work has been partially funded by the NSFC under grant No. 62192732, 61625206, 61732001, 61872341, and 61836005, by the ERC Advanced Project FRAPPANT under grant No. 787914, by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101008233, and by the CAS Pioneer Hundred Talents Program.

* Corresponding author.
** Corresponding author at: State Key Laboratory of Computer Science, Institute of Software, CAS, Beijing, China.
E-mail addresses: wangqye@ios.ac.cn (Q. Wang), chenms@cs.rwth-aachen.de (M. Chen), xuebai@ios.ac.cn (B. Xue), znj@ios.ac.cn (N. Zhan), katoen@cs.rwth-aachen.de (J.-P. Katoen).

https://doi.org/10.1016/j.ic.2022.104965
0890-5401/

The safety-critical feature of these CPS, with increasingly complex behaviors, has initiated automatic safety or, dually, reachability verification of hybrid systems [1,2]. The problem of reachability verification is undecidable in general [1], albeit with decidable families of sub-classes (see, e.g., [3–7]) identified in the literature. The hard core of the verification problem lies in reasoning about the continuous dynamics, which are often characterized by ordinary differential equations (ODEs). In particular, when nonlinearity arises in the ODEs, the explicit computation of the exact reachable set is usually intractable even for purely continuous dynamics [8].

Therefore in the literature, a plethora of approximation schemes, as surveyed in [2], for reachability analysis of hybrid systems has been developed, including an invariant-style reasoning scheme known as *barrier certificate* [9]. A barrier certificate often serves as an inductive invariant that isolates an unsafe region from the reachable set, thereby witnessing safety of hybrid (polynomial) systems possibly over an infinite time horizon. A common way to synthesize barrier certificates is to reduce the condition defining barrier certificates to a numerical optimization or constraint solving problem. There is, however, a trade-off between the expressiveness of the barrier-certificate condition and the efficiency in discharging the reduced constraints. Hence, to enable efficient algorithmic synthesis of barrier certificates via, e.g., linear programming (LP), second-order cone programming (SOCP), semidefinite programming (SDP) and interval analysis [10,11], the general condition on inductive invariance (that a barrier certificate defines an invariant, see [12,13]) has been strengthened into a spectrum of different shapes, e.g., [14–16,13,12]. It has been, nevertheless, a long-standing challenge *to find a barrier-certificate condition that is as weak as possible while admitting efficient synthesis algorithms*.

In this paper, we present a new condition on barrier certificates, termed the *invariant barrier-certificate condition*, based on the sufficient and necessary condition on being an inductive invariant [17]. Our invariant barrier-certificate condition is the weakest possible condition on barrier certificates to attain inductive invariance. We show, by leveraging Putinar's Positivstellensatz [18], that discharging the invariant barrier-certificate condition —thereby synthesizing invariant barrier certificates— can be encoded as solving an optimization problem subject to *bilinear matrix inequalities* (BMIs). It is known that general BMI problems are NP-hard and non-convex [19]. Existing solvers for BMI problems, e.g., [20,21], are thus considerably less efficient than solvers for (linear) SDP problems. We show that general bilinear matrix-valued functions can be decomposed as a difference of two convex (matrix-valued) functions using matrix decomposition, thus resulting in a synthesis algorithm as per *difference-of-convex programming* (DCP) [22,23], which solves a series of convex sub-problems (in the form of *linear matrix inequalities* (LMIs)) that approaches (arbitrarily close to) a local optimum of the BMI problem. This algorithm is incorporated in a branch-and-bound framework that searches for the global optimum in a divide-and-conquer fashion. We present a weak completeness result of our method: a barrier certificate is guaranteed to be found (under some mild assumptions) whenever there exists an inductive invariant (in the form of a given template) that suffices to certify the system's safety. A similar result on completeness is previously provided only by symbolic approaches, yet to the best of our knowledge, not by methods based on numerical constraint solving, e.g., [15,24,25]. Experiments on a collection of examples suggested that our invariant barrier-certificate condition recognizes more barrier certificates than existing conditions, and that our DCP-based algorithm is more efficient than directly solving the BMIs via off-the-shelf solvers.

Our main contributions in this paper can be summarized as follows.

- We present the invariant barrier-certificate condition, which is the weakest possible condition on barrier certificates to attain inductive invariance.
- We show that synthesizing invariant barrier certificates can be encoded as solving a BMI optimization problem.
- We propose a locally-convergent synthesis algorithm based on difference-of-convex programming.
- We present a weak completeness result by augmenting the local algorithm with a branch-and-bound framework.
- Experimental results suggested that our condition recognizes more barrier certificates than existing ones, and that our DCP-based algorithm is more efficient than directly solving the BMIs.

This article is an extended version of the conference paper [26]. Major extensions include

- two alternative matrix decomposition methods (besides eigendecomposition, cf. Section 5.1) that better exploit matrix sparsity to accelerate various matrix operations;
- a convex relaxation-based method for pruning branches in the branch-and-bound framework (see Algorithm 2 and Section 6.2) to mitigate the effect of exponential blow-up;
- complexity analysis of the DCP iterative procedure (cf. Section 5.3) and potential solutions to circumvent numerical errors in SDP solving (cf. Section 5.5); and
- generalization to hybrid systems (in Section 4.1), additional experimental results, and all the technical proofs.

Paper structure The rest of this paper is structured as follows. Section 2 gives an overview of our approach through a simple example. Section 3 introduces the necessary mathematical preliminaries. Section 4 presents the invariant barrier-certificate condition and shows how to encode it as a BMI optimization problem. Section 5 elucidates an algorithm for solving general BMI optimizations via DCP. Section 6 shows how to incorporate the BMI-solving algorithm into a branch-and-bound framework to attain weak completeness. Section 7 demonstrates our method on a collection of examples. After discussing related work in Section 8, we conclude the paper in Section 9.

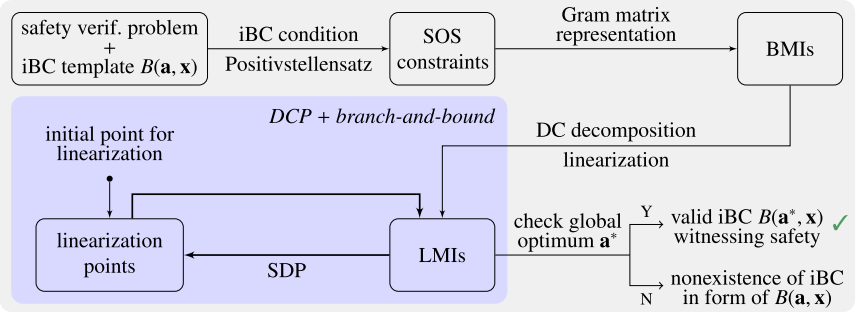


Fig. 1. A sketch of our method for unbounded-time safety verification via invariant barrier certificates (iBC, for short).

2. A bird's-eye perspective

The diagram in Fig. 1 sketches out a bird's-eye view of our method for the unbounded-time safety verification of differential dynamical systems. We use the following example to demonstrate several core steps underneath.

Example 1 *(overview [10]).* Consider the following continuous-time dynamical system modeled by an ordinary differential equation:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ x_1 x_2 - 0.5x_2^2 + 0.1 \end{pmatrix}.$$

The verification obligation is to show that the system trajectory originating from any state in the initial set $\mathcal{X}_0 = \{\mathbf{x} \mid \mathcal{I}(\mathbf{x}) \leq 0\}$ with $\mathcal{I}(\mathbf{x}) = x_1^2 + (x_2 - 2)^2 - 1$ will never enter the unsafe set $\mathcal{X}_u = \{\mathbf{x} \mid \mathcal{U}(\mathbf{x}) \leq 0\}$ with $\mathcal{U}(\mathbf{x}) = x_2 + 1$. $\lhd$

A barrier certificate satisfying our invariant barrier-certificate condition (cf. Definition 4) serves as an inductive invariant that suffices to isolate the unsafe region $\mathcal{X}_u$ from the set of reachable states from $\mathcal{X}_0$, thereby proving safety of the system over an infinite time horizon. To this end, we proceed in the following steps.

1) Encode as sum-of-squares (SOS) constraints We first set a (polynomial) barrier-certificate template, for example, $B(\mathbf{a},\mathbf{x}) = ax_2$ with unknown coefficient $a \in \mathbb{R}$. According to Theorem 1, we only need to consider Lie derivatives up to order $N_{B,f} = 1$, i.e., $\mathcal{L}_f^0 B(\mathbf{a},\mathbf{x}) = ax_2$ and $\mathcal{L}_f^1 B(\mathbf{a},\mathbf{x}) = a(x_1x_2 - 0.5x_2^2 + 0.1)$.

We show that $B(\mathbf{a},\mathbf{x})$ is an invariant barrier certificate if there exists a polynomial $v(\mathbf{x})$, SOS polynomials (i.e., polynomials that can be written as a finite sum of squares of polynomials) $\sigma(\mathbf{x}), \sigma'(\mathbf{x})$ and a constant $\epsilon > 0$ such that

$$-\underbrace{ax_2}_{B} + \sigma(\mathbf{x})\underbrace{\left(x_1^2 + (x_2-2)^2 - 1\right)}_{\mathcal{I}}, \tag{1.1, initial}$$

$$-\underbrace{a\left(x_1x_2 - 0.5x_2^2 + 0.1\right)}_{\mathcal{L}_f^1 B} + v(\mathbf{x})\underbrace{ax_2}_{\mathcal{L}_f^0 B}, \tag{1.2, Lie consecution}$$

$$\underbrace{ax_2}_{B} + \sigma'(\mathbf{x})\underbrace{(x_2+1)}_{\mathcal{U}} - \epsilon \tag{1.3, separation}$$

are SOS polynomials.

2) Reduce to a BMI optimization problem Observe that the above SOS constraints can be formulated as BMI constraints (via the Gram matrix representation, as formalized later). For instance, let us assume that (1.2) is an SOS polynomial of degree at most 2 and $v(\mathbf{s},\mathbf{x}) = s_0 + s_1x_1 + s_2x_2$ is a template polynomial with unknown coefficients $\mathbf{s}$. Then constraint (1.2) is equivalent to the BMI constraint

$$\mathcal{F}_2(\mathbf{a},\mathbf{s}) = -\begin{pmatrix} -0.1a & 0 & 0.5as_0 \\ 0 & 0 & 0.5(as_1 - a) \\ 0.5as_0 & 0.5(as_1 - a) & as_2 + 0.5a \end{pmatrix} \preceq 0$$

meaning that the bilinear matrix (the LHS of $\preceq$) is negative semidefinite. Note that the bilinearity arises due to the coupling of the unknown coefficients $\mathbf{a}$ and $\mathbf{s}$.

Constraints (1.1) and (1.3) can be reduced to BMI constraints in an analogous way,[1] yielding $\mathcal{F}_1$ and $\mathcal{F}_3$. It then follows that, to solve the SOS constraints, we need to find a feasible solution $(\mathbf{a}, \mathbf{s})$ such that[2]

$$\mathcal{F}_1(\mathbf{a}, \mathbf{s}) \preceq 0 \ \wedge \ \mathcal{F}_2(\mathbf{a}, \mathbf{s}) \preceq 0 \ \wedge \ \mathcal{F}_3(\mathbf{a}, \mathbf{s}) \preceq 0 \,. \tag{2}$$

To exploit well-developed optimization techniques, the feasibility problem (2) is transformed to an optimization problem subject to BMI constraints:

$$\begin{array}{ll} \underset{\lambda, \mathbf{a}, \mathbf{s}}{\text{maximize}} & \lambda \\ \text{subject to} & \mathcal{B}_i(\lambda, \mathbf{a}, \mathbf{s}) \ \hat{=} \ \mathcal{F}_i(\mathbf{a}, \mathbf{s}) + \lambda I \ \preceq \ 0, \quad i = 1, 2, 3 \end{array} \tag{3}$$

where I is the identity matrix with compatible dimensions. Note that problem (2) has a feasible solution if and only if the optimal value λ^* in (3) is non-negative.

3) Decompose as difference-of-convex problems The problem (3) contains non-convex constraints and hence does not admit efficient (polynomial-time) algorithms tailored for convex optimizations. However, using our DCP-based technique, a non-convex function $\mathcal{B}_i(\lambda, \mathbf{a}, \mathbf{s})$ can be decomposed as the difference of two (positive semidefinite) convex matrix-valued functions:

$$\mathcal{B}_i(\lambda, \mathbf{a}, \mathbf{s}) \ = \ \mathcal{B}_i^+(\lambda, \mathbf{a}, \mathbf{s}) - \mathcal{B}_i^-(\lambda, \mathbf{a}, \mathbf{s}) \,. \tag{4}$$

The decomposition of $\mathcal{B}_2(\lambda, \mathbf{a}, \mathbf{s})$ (via eigendecomposition), for instance, gives

$$\begin{aligned} &\mathcal{B}_2^+(\lambda, \mathbf{a}, \mathbf{s}) \\ &= \frac{1}{8} \begin{pmatrix} 8\lambda + 0.08a + a^2 + 0.408s_0^2 & 0.408s_0s_1 & -2as_0 + 0.816s_0s_2 \\ 0.408s_0s_1 & 8\lambda + a^2 + 0.408s_1^2 & 4a - 2as_1 + 0.816s_1s_2 \\ -2as_0 + 0.816s_0s_2 & 4a - 2as_1 + 0.816s_1s_2 & 8\lambda - 4a + 2.449a^2 - 4as_2 + s_0^2 + s_1^2 + 1.632s_2^2 \end{pmatrix} \\ &\mathcal{B}_2^-(\lambda, \mathbf{a}, \mathbf{s}) = \frac{1}{8} \begin{pmatrix} a^2 + 0.408s_0^2 & 0.408s_0s_1 & 2as_0 + 0.816s_0s_2 \\ 0.408s_0s_1 & a^2 + 0.408s_1^2 & 2as_1 + 0.816s_1s_2 \\ 2as_0 + 0.816s_0s_2 & 2as_1 + 0.816s_1s_2 & 2.449a^2 + 4as_2 + s_0^2 + s_1^2 + 1.632s_2^2 \end{pmatrix} . \end{aligned}$$

4) Solve a series of convex sub-problems Now, we apply a standard iterative procedure in difference-of-convex programming [27] as follows. Given a feasible solution $\mathbf{z}^k = (\lambda^k, \mathbf{a}^k, \mathbf{s}^k)$ to the BMI optimization problem (3), the concave part $-\mathcal{B}_i^-(\lambda, \mathbf{a}, \mathbf{s})$ in (4) is linearized around $\mathbf{z}^k$, thus yielding a series of convex programs ($k = 0, 1, \ldots$):

$$\begin{array}{ll} \underset{\lambda, \mathbf{a}, \mathbf{s}}{\text{maximize}} & \lambda \\ \text{subject to} & \mathcal{B}_i^+(\mathbf{z}) - \mathcal{B}_i^-\left(\mathbf{z}^k\right) - \mathcal{DB}_i^-\left(\mathbf{z}^k\right)\left(\mathbf{z} - \mathbf{z}^k\right) \ \preceq \ 0, \quad i = 1, 2, 3 \end{array} \tag{5}$$

where $\mathcal{DB}_i^-(\mathbf{z}^k)(\cdot)$ denotes the derivative of the matrix-valued function $\mathcal{B}_i^-(\cdot)$ at $\mathbf{z}^k$.

The soundness of our approach asserts that *the feasible set of the linearized program* (5) *under-approximates the feasible set of the original BMI program* (3). Therefore, if $\lambda^k \geq 0$ after iteration k, we can safely claim that $(\mathbf{a}^k, \mathbf{s}^k)$ is a feasible solution to (2). A barrier certificate $B(\mathbf{x})$ is then obtained by substituting $\mathbf{a}^k$ in $B(\mathbf{a}, \mathbf{x})$. Moreover, if we take the optimum $\mathbf{z}^{*,k}$ of (5) to be the next linearization point $\mathbf{z}^{k+1}$, the solution sequence $\{\mathbf{z}^k\}_{k \in \mathbb{N}}$ converges to a local optimum of (3).

We show that the linearized program (5) is equivalent to an LMI optimization problem admitting polynomial-time algorithms [28], say the well-known *interior-point methods* supported by most off-the-shelf SDP solvers. Our iterative procedure starts with a strictly feasible initial solution $\mathbf{z}^0$ to program (3) and terminates after iteration $k = 2$ with $\lambda^2 \geq 0$ (subject to numerical round-off) and $a^2 = -0.00363421$, yielding the barrier certificate

$$B(\mathbf{a}^2, \mathbf{x}) \ = \ -0.00363421x_2 \ \leq \ 0 \,.$$

Fig. 2 depicts the system dynamics and the synthesized barrier certificate.

We remark that the aforementioned iterative procedure on solving a series of convex optimizations converges only to a local optimum of the BMI problem (3). This means that, in some cases, it may miss the global optimum that induces a non-negative λ^*. We will present in Section 6 a solution to this problem by incorporating our iterative procedure into a branch-and-bound framework that searches for the global optimum in a divide-and-conquer fashion.

[1] Despite that no bilinearity is involved in constraints (1.1) and (1.3), they can be processed in the same way as (1.2), yielding LMI constraints.

[2] Extra constraints on $\sigma(\mathbf{x})$ and $\sigma'(\mathbf{x})$ being SOS polynomials can be encoded analogously in the feasibility problem, yet are omitted here for the sake of simplicity.

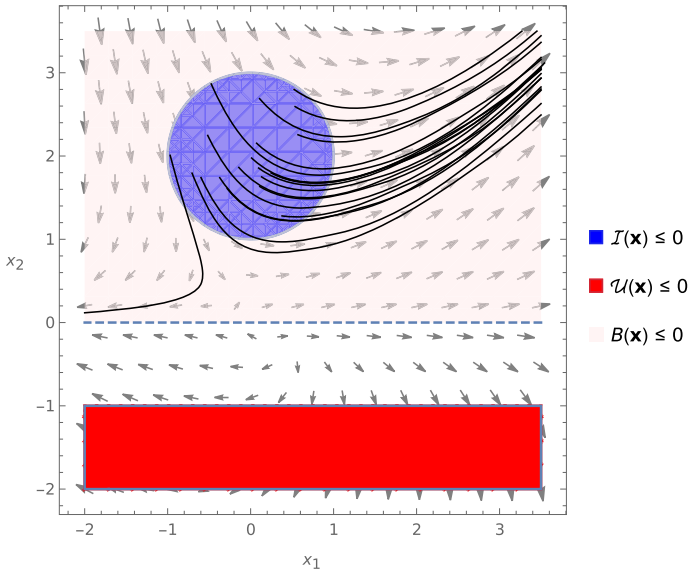


Fig. 2. Phase portrait of the system in Example 1. The arrows indicate the vector field and the solid curves are randomly sampled trajectories.

3. Mathematical foundations

Notations Let $\mathbb{N}$, $\mathbb{N}^+$, $\mathbb{R}$, $\mathbb{R}^+$ and $\mathbb{R}_0^+$ be respectively the set of natural, positive natural, real, positive real and non-negative real numbers. For a vector $\mathbf{x} \in \mathbb{R}^n$, x_i refers to its i-th component and $\|\mathbf{x}\|$ denotes the ℓ^2-norm; we write $diag(\mathbf{x}) \in \mathbb{R}^{n\times n}$ for a diagonal matrix with x_i being the i-th diagonal element. For a matrix $A \in \mathbb{R}^{n\times m}$, $A(i,j)$ refers to its (i,j)-th element; for a square matrix $A \in \mathbb{R}^{n\times n}$, its trace is $tr(A) = \sum_{i=1}^{n} A(i,i)$. Given two matrices $A \in \mathbb{R}^{a\times b}$ and $B \in \mathbb{R}^{c\times d}$, their *Kronecker product* is $A \otimes B \hat{=} [A(1,1)B, \ldots, A(1,b)B; \cdots; A(a,1)B, \ldots, A(a,b)B] \in \mathbb{R}^{ac\times bd}$. $\mathcal{S}^n$ denotes the space of $n \times n$ real, symmetric matrices. For $A \in \mathcal{S}^n$, $A \succeq 0$ means that A is *positive semidefinite* (PSD, for short), i.e., $\forall \mathbf{x} \in \mathbb{R}^n \colon \mathbf{x}^{\mathsf{T}} A \mathbf{x} \ge 0$. More generally, for $A, B \in \mathcal{S}^n$, $A \preceq B$ indicates that $B - A$ is positive semidefinite. A matrix-valued function $\mathcal{B} \colon \mathbb{R}^n \to \mathcal{S}^m$ is *PSD-convex* on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$ if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}. \forall \mu \in (0,1) \colon \mathcal{B}(\mu\mathbf{x}_1 + (1-\mu)\mathbf{x}_2) \preceq \mu\mathcal{B}(\mathbf{x}_1) + (1-\mu)\mathcal{B}(\mathbf{x}_2)$.

SOS, LMIs, and BMIs Let $\mathbb{R}[\mathbf{x}]$ be the polynomial ring in $\mathbf{x}$ over the field $\mathbb{R}$. A polynomial $h \in \mathbb{R}[\mathbf{x}]$ is *sum-of-squares* (SOS) iff there exist polynomials $g_1, \ldots, g_k \in \mathbb{R}[\mathbf{x}]$ such that $h = \sum_{i=1}^{k} g_i^2$. We denote by $\Sigma[\mathbf{x}] \subset \mathbb{R}[\mathbf{x}]$ the set of SOS polynomials over $\mathbf{x}$. A *linear matrix inequality* (LMI) is a constraint of the form $\mathcal{L}(\mathbf{x}) \hat{=} F + \sum_{i=1}^{m} x_i H_i \preceq 0$, where $\mathbf{x} \in \mathbb{R}^m$ is a vector of variables and $F, H_i \in \mathcal{S}^p$ are constant symmetric matrices. LMIs are convex and hence admit polynomial-time algorithms to find feasible solutions (or prove the infeasibility) given the desired precision [28]. A *bilinear matrix inequality* (BMI) is a constraint of the form $\mathcal{B}(\mathbf{x},\mathbf{y}) \hat{=} F + \sum_{i=1}^{m} x_i H_i + \sum_{j=1}^{n} y_j G_j + \sum_{i=1}^{m}\sum_{j=1}^{n} x_i y_j F_{i,j} \preceq 0$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ are vectors of variables and $F, H_i, G_j, F_{i,j} \in \mathcal{S}^p$ are constant symmetric matrices. Solving general BMIs is NP-hard due to the non-convex nature of the constraints [19].

Differential dynamical systems We consider a class of continuous dynamical systems modeled by ordinary differential equations of the autonomous type:

$$\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}) \tag{6}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the *state* vector, $\dot{\mathbf{x}}$ denotes its temporal derivative $\mathrm{d}\mathbf{x}/\mathrm{d}t$, with $t \in \mathbb{R}_0^+$ modeling time, and $\boldsymbol{f} \colon \mathbb{R}^n \to \mathbb{R}^n$ is a polynomial *flow field* (or *vector field*) that governs the evolution of the system. A polynomial vector field is local Lipschitz, and hence for some $T \in \mathbb{R}^+ \cup \{\infty\}$, there exists a unique *solution* (or *trajectory*) $\boldsymbol{\zeta}_{\mathbf{x}_0} \colon [0,T) \to \mathbb{R}^n$ originating from any initial state $\mathbf{x}_0 \in \mathbb{R}^n$ such that (1) $\boldsymbol{\zeta}_{\mathbf{x}_0}(0) = \mathbf{x}_0$, and (2) $\forall \tau \in [0,T) \colon \frac{\mathrm{d}\boldsymbol{\zeta}_{\mathbf{x}_0}}{\mathrm{d}t}\big|_{t=\tau} = \boldsymbol{f}(\boldsymbol{\zeta}_{\mathbf{x}_0}(\tau))$. We assume in the sequel that T is the maximal instant up to which $\boldsymbol{\zeta}_{\mathbf{x}_0}$ exists for all $\mathbf{x}_0$.

Remark 1. Our techniques on synthesizing barrier certificates in this paper focus on differential dynamics of the form (6). However, we will show that there is no substantial difficulty in extending the results to multi-mode hybrid systems where extra constraints on the system evolution, e.g., guards, are present.

Safety verification problem Given a domain set $\mathcal{D} \subseteq \mathbb{R}^n$ and an initial set $\mathcal{X}_0 \subseteq \mathcal{D}$, the *reachable set* of a dynamical system of the form (6) at time instant $t \in [0,T)$ is defined as $\mathcal{R}_{\mathcal{X}_0}(t) \hat{=} \{\boldsymbol{\zeta}_{\mathbf{x}_0}(t) \mid \mathbf{x}_0 \in \mathcal{X}_0\}$. We denote by $\mathcal{R}_{\mathcal{X}_0}$ the aggregated

reachable set, i.e., the union of $\mathcal{R}_{\mathcal{X}_0}(t)$ over $t \in [0, T)$. Given an unsafe set $\mathcal{X}_u \subseteq \mathcal{D}$, the system is said to be *safe* iff $\mathcal{R}_{\mathcal{X}_0} \cap \mathcal{X}_u = \emptyset$, and *unsafe* otherwise. For simplicity, we consider $\mathcal{D} = \mathbb{R}^n$ unless explicitly stated otherwise.

To avoid the explicit computation of the exact reachable set, which is usually intractable for nonlinear hybrid systems (cf., e.g., [2]), barrier-certificate methods make use of a partial differential operator, termed the *Lie derivative*, to capture the evolution of a barrier function along the vector field:

Definition 1 *(Lie derivative* [29]*).* Given a vector field $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ over $\mathbf{x}$, the *Lie derivative* of a polynomial $B \in \mathbb{R}[\mathbf{x}]$ along $\boldsymbol{f}$, $\mathcal{L}_{\boldsymbol{f}}^k B : \mathbb{R}^n \to \mathbb{R}$ of order $k \in \mathbb{N}$, is

$$\mathcal{L}_{\boldsymbol{f}}^k B(\mathbf{x}) \;\hat{=}\; \begin{cases} B(\mathbf{x}), & k = 0\,, \\ \left\langle \frac{\partial}{\partial \mathbf{x}} \mathcal{L}_{\boldsymbol{f}}^{k-1} B(\mathbf{x}),\, \boldsymbol{f}(\mathbf{x}) \right\rangle, & k > 0 \end{cases}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of vectors, i.e., $\langle \mathbf{u}, \mathbf{v} \rangle \hat{=} \sum_{i=1}^n u_i v_i$ for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.

The Lie derivative $\mathcal{L}_{\boldsymbol{f}}^k B(\mathbf{x})$ is essentially the k-th temporal derivative of the (barrier) function $B(\mathbf{x})$, and thus captures the change of $B(\mathbf{x})$ over time. In fact, given a polynomial vector field, one can use (high-order) Lie derivatives to identify the tendency of its trajectories in terms of a polynomial function $B(\mathbf{x})$, as exemplified in Appendix A.

An *inductive invariant* $\Psi \subseteq \mathbb{R}^n$ of a dynamical system is a set of states such that all the trajectories starting from within Ψ remain in Ψ:

Definition 2 *(Inductive invariant* [30]*).* Given a system (6), a set $\Psi \subseteq \mathbb{R}^n$ is an *inductive invariant* of system (6) if and only if

$$\forall \mathbf{x}_0 \in \Psi.\ \forall t \in [0, T) :\ \boldsymbol{\zeta}_{\mathbf{x}_0}(t) \in \Psi\,. \tag{7}$$

In the sequel, we refer to inductive invariants simply as invariants. In [17], a sufficient and necessary condition on being a polynomial invariant is proposed:

Theorem 1 *(Invariant condition* [17]*). Given a polynomial $B \in \mathbb{R}[\mathbf{x}]$, its* zero sub-level set $\{\mathbf{x} \mid B(\mathbf{x}) \leq 0\}$ *is an invariant of system* (6) *if and only if*[3]

$$B \leq 0 \implies \bigvee_{i=0}^{N_{B,\boldsymbol{f}}} \left(\left(\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^j B = 0 \right) \wedge \mathcal{L}_{\boldsymbol{f}}^i B < 0 \right) \vee \bigwedge_{i=0}^{N_{B,\boldsymbol{f}}} \mathcal{L}_{\boldsymbol{f}}^i B = 0 \tag{8}$$

where $N_{B,\boldsymbol{f}} \in \mathbb{N}^+$ is the completeness threshold, i.e., a positive integer that bounds the order of Lie derivatives.

Remark 2. $N_{B,\boldsymbol{f}}$ is the minimal index i such that $\mathcal{L}_{\boldsymbol{f}}^{i+1} B$ is in the polynomial ideal generated by $\mathcal{L}_{\boldsymbol{f}}^0 B, \ldots, \mathcal{L}_{\boldsymbol{f}}^i B$. The ideal membership can be decided by computing the Gröbner basis of this ideal [17]. The complexity of computing $N_{B,\boldsymbol{f}}$ will be discussed in the complexity analysis of our approach (see Section 5.3).

In contrast, a *barrier certificate* is a function whose zero sub-level set isolates an unsafe region $\mathcal{X}_u$ from the reachable set $\mathcal{R}_{\mathcal{X}_0}$ w.r.t. some initial set $\mathcal{X}_0$ (the sub-level set can be non-zero in general):

Definition 3 *(Semantic barrier certificate* [12]*).* Given a system (6), an initial set $\mathcal{X}_0$ and an unsafe set $\mathcal{X}_u$, a *barrier certificate* of (6) is a differentiable function $B : \mathbb{R}^n \to \mathbb{R}$ satisfying

$$\forall \mathbf{x}_0 \in \mathcal{X}_0.\ \forall t \in [0, T) :\ B\left(\boldsymbol{\zeta}_{\mathbf{x}_0}(t)\right) \leq 0 \quad \text{and} \quad \forall \mathbf{x} \in \mathcal{X}_u :\ B(\mathbf{x}) > 0\,. \tag{9}$$

The existence of such a barrier certificate trivially implies safety of the system. Moreover, one may readily verify that if some set $\Psi = \{\mathbf{x} \mid B(\mathbf{x}) \leq 0\}$ is an invariant and satisfies $(\mathcal{X}_0 \subseteq \Psi) \wedge (\Psi \cap \mathcal{X}_u = \emptyset)$, then $B(\mathbf{x})$ is a barrier certificate.

As observed in [12], however, the semantic statement in Definition 3 encodes merely the general *principle of barrier certificates* [13], yet in itself is not that useful for safety verification because it explicitly involves the system solutions. Therefore, in order to enable efficient synthesis, the semantic condition on barrier certificates has been strengthened into a handful of different shapes (see, e.g., [9,14,15,13]) which all imply inductive invariance.[4] It has been yet a long-standing challenge *to find a barrier-certificate condition that is as weak as possible while admitting efficient synthesis algorithms*.

Our BMI encoding of the invariant barrier-certificate condition roots in Putinar's Positivstellensatz, which characterizes positivity of polynomials on a semi-algebraic set defined by a system of polynomial inequalities:

[3] In (8), $\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^j B = 0$ is true for $i = 0$ by default. This applies in the sequel. Moreover, the sub-level set of B can be non-zero in general.

[4] An exception is known as the t-barrier certificate condition [31], which is a continuous analogy to k-induction, thus more general than (classical) inductive invariance. However, this condition also explicitly involves the system solutions, and hence does not admit efficient synthesis.

Theorem 2 *(Putinar's Positivstellensatz [18]). Let* $\mathcal{K} = \{\mathbf{x} \mid \bigwedge_{i=1}^{m} g_i(\mathbf{x}) \geq 0\}$ *be a compact semi-algebraic set defined by* $g_1, \ldots, g_m \in \mathbb{R}[\mathbf{x}]$. *Assume the* Archimedean condition *holds, i.e., there exists* $L \in \mathbb{R}^+$ *such that* $L - \|\mathbf{x}\|^2 = \eta_0(\mathbf{x}) + \sum_{i=1}^{m} \eta_i(\mathbf{x}) g_i(\mathbf{x})$ *for some* $\eta_0, \ldots, \eta_m \in \Sigma[\mathbf{x}]$. *If* $h \in \mathbb{R}[\mathbf{x}]$ *is strictly positive on* $\mathcal{K}$, *then*

$$h(\mathbf{x}) \;=\; \sigma_0(\mathbf{x}) + \sum\nolimits_{i=1}^{m} \sigma_i(\mathbf{x}) g_i(\mathbf{x})$$

holds for some SOS polynomials $\sigma_0, \ldots, \sigma_m \in \Sigma[\mathbf{x}]$.

Remark 3. The Archimedean condition can be met by adding a (redundant) constraint $g_{m+1}(\mathbf{x}) = L_0 - \|\mathbf{x}\|^2 \leq 0$, provided that a bound $L_0 \in \mathbb{R}^+$ is known such that $\forall \mathbf{x} \in \mathcal{K}\colon\ L_0 - \|\mathbf{x}\|^2 \geq 0$. See [18, Chapter 2] for more details on the Archimedean condition.

We now recall a key technique used in our reduction to semidefinite optimizations. Given a symmetric matrix $X \in \mathcal{S}^n$ partitioned as $X = \begin{pmatrix} A & C \\ C^\mathsf{T} & D \end{pmatrix}$ with invertible A, the *Schur complement* of A in X is defined as $X/A \,\hat{=}\, D - C^\mathsf{T} A^{-1} C$. An important property of the Schur complement X/A is that it characterizes the positive semidefiniteness of the block matrix X (which will be used later to transform nonlinear convex constraints into linear constraints):

Theorem 3 *(Schur complement [32]). If* $A \succ 0$, *then* $X \succeq 0$ *iff* $X/A \succeq 0$.

4. Invariant barrier-certificate condition as BMIs

In this section, we present our *invariant barrier-certificate condition* based on the necessary and sufficient condition on being an inductive invariant (cf. Theorem 1), and show how to encode it as BMI constraints.

4.1. Invariant barrier-certificate condition

Definition 4 *(Invariant barrier certificate).* Given a system (6), an initial set $\mathcal{X}_0$ and an unsafe set $\mathcal{X}_u$, a polynomial function $B\colon \mathbb{R}^n \to \mathbb{R}$ is an *invariant barrier certificate* of system (6) if and only if

1. (initial): $\forall \mathbf{x} \in \mathcal{X}_0\colon\ B(\mathbf{x}) \;\leq\; 0$;
2. (consecution): $\forall \mathbf{x} \in \mathbb{R}^n\colon\ \bigwedge_{i=1}^{N_{B,\boldsymbol{f}}} \left(\left(\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x}) = 0 \right) \implies \mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) \leq 0 \right)$;
3. (separation): $\forall \mathbf{x} \in \mathcal{X}_u\colon\ B(\mathbf{x}) \;>\; 0$.

Notice that the consecution constraint in Definition 4 involves Lie derivatives of orders up to $N_{B,\boldsymbol{f}} \in \mathbb{N}^+$, as is the case in Theorem 1. Our invariant barrier-certificate condition hence generalizes existing conditions on barrier certificates, e.g., [15,33,25], which consider Lie derivatives only up to the first order.

The following lemma states that *the consecution condition in Definition 4 is in fact equivalent to the invariant condition (8) in Theorem 1.*

Lemma 1 *(Equivalence of Lie consecution). The consecution condition in Definition 4 holds if and only if the invariant condition (8) in Theorem 1 holds.*

Proof. We prove both the "if" and the "only if" part by contradiction.

For the "if" part, suppose that the invariant condition (8) holds but the consecution condition is invalid. The latter implies that for some $\mathbf{x}_0 \in \mathbb{R}^n$ and $1 \leq i_0 \leq N_{B,\boldsymbol{f}}$,

$$\left(\bigwedge\nolimits_{j=0}^{i_0-1} \mathcal{L}_{\boldsymbol{f}}^j(\mathbf{x}_0) = 0 \right) \;\wedge\; \mathcal{L}_{\boldsymbol{f}}^{i_0} B(\mathbf{x}_0) \;>\; 0\,. \tag{10}$$

Note that (10) implies $B(\mathbf{x}_0) = 0$. From (8), it follows that either

$$\bigwedge\nolimits_{i=0}^{N_{B,\boldsymbol{f}}} \mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}_0) \;=\; 0 \tag{11}$$

holds, or there exists $0 \leq i_1 \leq N_{B,\boldsymbol{f}}$ such that

$$\left(\bigwedge\nolimits_{j=0}^{i_1-1} \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x}_0) = 0 \right) \;\wedge\; \mathcal{L}_{\boldsymbol{f}}^{i_1} B(\mathbf{x}_0) \;<\; 0 \tag{12}$$

holds. However,

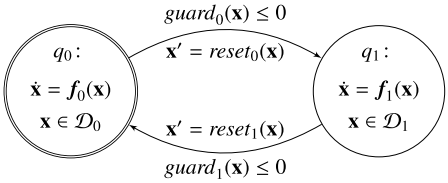


Fig. 3. A simple (symbolic) hybrid automaton.

- (11) cannot hold as $\mathcal{L}_{\boldsymbol{f}}^{i_0} B(\mathbf{x}_0) = 0$ in (11) but $\mathcal{L}_{\boldsymbol{f}}^{i_0} B(\mathbf{x}_0) > 0$ in (10);
- for $i_1 \le i_0$, (12) cannot hold as $\mathcal{L}_{\boldsymbol{f}}^{i_1} B(\mathbf{x}_0) < 0$ in (12) but $\mathcal{L}_{\boldsymbol{f}}^{i_1} B(\mathbf{x}_0) \ge 0$ in (10);
- for $i_1 > i_0$, (12) cannot hold as $\mathcal{L}_{\boldsymbol{f}}^{i_0} B(\mathbf{x}_0) = 0$ in (12) but $\mathcal{L}_{\boldsymbol{f}}^{i_0} B(\mathbf{x}_0) > 0$ in (10).

For the "only if" direction, suppose that the consecution condition in Definition 4 holds but the invariant condition (8) is invalid. The latter implies that there exists $\mathbf{x}_0$ such that $B(\mathbf{x}_0) \le 0$ and

$$\neg \left(\left(\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^{j} B(\mathbf{x}_0) = 0 \right) \wedge \mathcal{L}_{\boldsymbol{f}}^{i} B(\mathbf{x}_0) < 0 \right) \tag{13}$$

holds for any $0 \le i \le N_{B,\boldsymbol{f}}$.

For $i = 0$, (13) yields that $B(\mathbf{x}_0) \ge 0$. Together with the premise $B(\mathbf{x}_0) \le 0$, we have $B(\mathbf{x}_0) = \mathcal{L}_{\boldsymbol{f}}^{0} B(\mathbf{x}_0) = 0$. Now, by taking the case $i = 1$ in the consecution condition, we deduce $\mathcal{L}_{\boldsymbol{f}}^{1} B(\mathbf{x}_0) \le 0$. Meanwhile, for $i = 1$, (13) yields $\mathcal{L}_{\boldsymbol{f}}^{1} B(\mathbf{x}_0) \ge 0$. It thus follows that $\mathcal{L}_{\boldsymbol{f}}^{1} B(\mathbf{x}_0) = 0$. Analogously, by taking $i = 2, \ldots, N_{B,\boldsymbol{f}}$, we conclude $\mathcal{L}_{\boldsymbol{f}}^{i} B(\mathbf{x}_0) = 0$ for all $0 \le i \le N_{B,\boldsymbol{f}}$. This is exactly encoded in (8) (the rightmost conjunctive clause) and hence contradicts the assumption that (8) is invalid. Therefore, the consecution condition implies (8). □

Lemma 1 reveals the relation between an inductive invariant and an invariant barrier certificate:

Theorem 4 *(Inductive invariance). Given a system (6), an initial set $\mathcal{X}_0$ and an unsafe set $\mathcal{X}_u$. (1) If polynomial $B(\mathbf{x})$ is an invariant barrier certificate, then $\Psi = \{\mathbf{x} \mid B(\mathbf{x}) \le 0\}$ is an invariant. Conversely, (2) if $\Psi = \{\mathbf{x} \mid B(\mathbf{x}) \le 0\}$ is an invariant satisfying $\mathcal{X}_0 \subseteq \Psi$ and $\Psi \cap \mathcal{X}_u = \emptyset$, then $B(\mathbf{x})$ is an invariant barrier certificate.*

Proof. The claim is an immediate consequence of Lemma 1. □

It follows from Theorem 4 that *our invariant barrier-certificate condition is the least conservative (and in fact the weakest possible) one on barrier certificates to attain inductive invariance*.

Remark 4. We do not employ the invariant condition (8) in Theorem 1 as the constraint on the consecution of Lie derivatives. This is because our consecution condition in Definition 4 is simpler, and in particular, amenable to more straightforward transformations to SOS constraints via Putinar's Positivstellensatz, as shown later in Section 4.2.

Remark 5. For a fixed $0 < \mathfrak{N} < N_{B,\boldsymbol{f}}$, the consecution condition in Definition 4 can be strengthened in the following way while preserving inductive invariance:

$$\forall \mathbf{x} \in \mathbb{R}^n : \bigwedge_{i=1}^{\mathfrak{N}-1} \left(\left(\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^{j} B(\mathbf{x}) = 0 \right) \implies \mathcal{L}_{\boldsymbol{f}}^{i} B(\mathbf{x}) \le 0 \right) \wedge \left(\left(\bigwedge_{j=0}^{\mathfrak{N}-1} \mathcal{L}_{\boldsymbol{f}}^{j} B(\mathbf{x}) = 0 \right) \implies \mathcal{L}_{\boldsymbol{f}}^{\mathfrak{N}} B(\mathbf{x}) < 0 \right)$$

where for the $\mathfrak{N}$-th Lie derivative, one needs $\mathcal{L}_{\boldsymbol{f}}^{\mathfrak{N}} B(\mathbf{x}) < 0$ (rather than $\mathcal{L}_{\boldsymbol{f}}^{\mathfrak{N}} B(\mathbf{x}) \le 0$). In practice, using such a strengthened consecution condition —with less sub-constraints to solve— may yield more efficient synthesis.

Generalization to hybrid systems Our invariant barrier-certificate condition can be readily generalized to multi-mode hybrid systems exhibiting both continuous dynamics and discrete transitions in the same vein as in [9,25]. We illustrate such generalization by a simple (symbolic) hybrid automaton [2] as depicted in Fig. 3. The system has two modes q_0 (initial mode) and q_1 governed respectively by polynomial flow fields $\boldsymbol{f}_0(\mathbf{x})$ and $\boldsymbol{f}_1(\mathbf{x})$ and mode domains $\mathcal{D}_0$ and $\mathcal{D}_1$. The system may evolve continuously in mode q_k (for $k = 0, 1$) within $\mathcal{D}_k$ or jump to mode q_{1-k} when $guard_k(\mathbf{x}) \le 0$ is satisfied. In the latter case, the system state will be set to $\mathbf{x}' = reset_k(\mathbf{x}) \in \mathcal{D}_{1-k}$ after the jump. We aim to verify that no trajectory originating from an initial set $\mathcal{X}_0 \subseteq \mathcal{D}_0$ will ever visit states in the unsafe sets $\mathcal{X}_{u,k} \subseteq \mathcal{D}_k$. To this end, our invariant barrier-certificate condition (cf. Definition 4) can be augmented to recognize an invariant barrier certificate $B_k(\mathbf{x})$ for each mode q_k:

1. (initial): $\forall \mathbf{x} \in \mathcal{X}_0 :\ B_0(\mathbf{x}) \ \leq\ 0$;
2. (consecution): $\forall k \in \{0,1\}.\ \forall \mathbf{x} \in \mathcal{D}_k :\ \bigwedge_{i=1}^{N_{B_k,\boldsymbol{f}_k}} \left(\left(\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}_k}^j B_k(\mathbf{x}) = 0 \right) \implies \mathcal{L}_{\boldsymbol{f}_k}^i B_k(\mathbf{x}) \leq 0 \right)$;
3. (transition): $\forall k \in \{0,1\}.\ \forall \mathbf{x} \in \mathcal{D}_k :\ \left(B_k(\mathbf{x}) \leq 0 \wedge guard_k(\mathbf{x}) \leq 0 \right) \implies B_{1-k}(reset_k(\mathbf{x})) \leq 0$;
4. (separation): $\forall k \in \{0,1\}.\ \forall \mathbf{x} \in \mathcal{X}_{u,k} :\ B_k(\mathbf{x}) \ >\ 0.$

The existence of $B_k(\mathbf{x})$ satisfying the above constraints ensures safety of the hybrid system model. In fact, all these constraints (with polynomial guards and resets as well as domains described by polynomials) can be encoded in a BMI optimization problem and thereby solved by our DCP-based algorithm without substantial changes. For simplicity, however, we present our techniques for single-mode dynamical systems based on the invariant barrier-certificate condition given in Definition 4.

4.2. Encoding as BMI optimizations

Next, we show how to encode the synthesis of an invariant barrier certificate as an optimization problem subject to BMIs. To this end, we first recast the invariant barrier-certificate condition into a collection of SOS constraints. For simplicity, we assume that $\mathcal{X}_0$ and $\mathcal{X}_u$ are both captured by a single polynomial. Our formulations, however, apply also to cases with basic semi-algebraic $\mathcal{X}_0$ or $\mathcal{X}_u$.

Theorem 5 *(Sufficient condition for invariant barrier certificate). Given a system (6), an initial set $\mathcal{X}_0 = \{\mathbf{x} \mid \mathcal{I}(\mathbf{x}) \leq 0\}$ and an unsafe set $\mathcal{X}_u = \{\mathbf{x} \mid \mathcal{U}(\mathbf{x}) \leq 0\}$. A polynomial $B \in \mathbb{R}[\mathbf{x}]$ is an invariant barrier certificate of (6) if for some $\epsilon \in \mathbb{R}^+$, there exist polynomials $v_{i,j} \in \mathbb{R}[\mathbf{x}]$ and SOS polynomials $\sigma(\mathbf{x}), \sigma'(\mathbf{x}) \in \Sigma[\mathbf{x}]$ s.t.*

1. $-B(\mathbf{x}) + \sigma(\mathbf{x})\mathcal{I}(\mathbf{x})$,
2. *for all* $1 \leq i \leq N_{B,\boldsymbol{f}}$, $-\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) + \sum_{j=0}^{i-1} v_{i,j}(\mathbf{x}) \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x})$,
3. $B(\mathbf{x}) + \sigma'(\mathbf{x})\mathcal{U}(\mathbf{x}) - \epsilon$

are SOS polynomials in $\Sigma[\mathbf{x}]$.

Proof. It can be shown that the k-th condition in Theorem 5 implies the k-th condition in Definition 4, for $k = 1, 2, 3$. For instance, the second condition in Theorem 5 requires that $-\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) + \sum_{j=0}^{i-1} v_{i,j}(\mathbf{x}) \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x})$ is an SOS polynomial (and thus non-negative) for all $1 \leq i \leq N_{B,\boldsymbol{f}}$, we therefore have $\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) \leq \sum_{j=0}^{i-1} v_{i,j}(\mathbf{x}) \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x})$ for all $1 \leq i \leq N_{B,\boldsymbol{f}}$. It follows that for all $\mathbf{x}$, when $\mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x}) = 0$ with $0 \leq j \leq i-1$, we have $\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) \leq 0$, which is the consecution condition in Definition 4. A similar argument applies to the other two conditions. □

By enforcing the Archimedean condition and applying Putinar's Positivstellensatz, we further derive a necessary condition of invariant barrier certificate:

Theorem 6 *(Necessary condition for invariant barrier certificate). Given a system (6), an initial set $\mathcal{X}_0 = \{\mathbf{x} \mid \mathcal{I}(\mathbf{x}) \leq 0\}$ and an unsafe set $\mathcal{X}_u = \{\mathbf{x} \mid \mathcal{U}(\mathbf{x}) \leq 0\}$. If $B \in \mathbb{R}[\mathbf{x}]$ is an invariant barrier certificate of (6), then for some $\epsilon \in \mathbb{R}^+$, there exist polynomials $v_{i,j} \in \mathbb{R}[\mathbf{x}]$ and SOS polynomials $\sigma(\mathbf{x}), \sigma'(\mathbf{x}), \rho(\mathbf{x}), \rho'(\mathbf{x}), \rho_i''(\mathbf{x}) \in \Sigma[\mathbf{x}]$ s.t. for any $L \in \mathbb{R}^+$,*

1. $-B(\mathbf{x}) + \rho(\mathbf{x})(\|\mathbf{x}\|^2 - L) + \sigma(\mathbf{x})\mathcal{I}(\mathbf{x}) + \epsilon$,
2. *for all* $1 \leq i \leq N_{B,\boldsymbol{f}}$, $-\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) + \rho_i''(\mathbf{x})(\|\mathbf{x}\|^2 - L) + \sum_{j=0}^{i-1} v_{i,j}(\mathbf{x}) \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x}) + \epsilon$,
3. $B(\mathbf{x}) + \rho'(\mathbf{x})(\|\mathbf{x}\|^2 - L) + \sigma'(\mathbf{x})\mathcal{U}(\mathbf{x})$

are SOS polynomials in $\Sigma[\mathbf{x}]$.

Proof. The invariant barrier-certificate condition in Definition 4 characterizes positivity of polynomials over certain sets. By adding a "ball" constraint $\|\mathbf{x}\|^2 - L \leq 0$ to those sets (thus achieving the Archimedean condition), we can apply Putinar's Positivstellensatz to rewrite those polynomials into SOS forms.

For instance, the consecution condition in Definition 4 implies that $-\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) + \epsilon$ is strictly positive on $\mathcal{K} = \{\mathbf{x} \mid (\bigwedge_{j=0}^{i-1} \mathcal{L}_{\boldsymbol{f}}^0 B(\mathbf{x}) = 0) \wedge -(\|\mathbf{x}\|^2 - L) \geq 0\}$ for all $1 \leq i \leq N_{B,\boldsymbol{f}}$. Putinar's Positivstellensatz can then be applied to show that $-\mathcal{L}_{\boldsymbol{f}}^i B(\mathbf{x}) + \epsilon = \sigma_i(\mathbf{x}) - \rho_i''(\mathbf{x})(\|\mathbf{x}\|^2 - L) - \sum_{j=0}^{i-1} v_{i,j}(\mathbf{x}) \mathcal{L}_{\boldsymbol{f}}^j B(\mathbf{x})$ holds for some SOS polynomials $\sigma_i(\mathbf{x})$, $\rho_i''(\mathbf{x})$ and some polynomials $v_{i,j}(\mathbf{x})$ for $1 \leq i \leq N_{B,\boldsymbol{f}}$ and $0 \leq j \leq i-1$. The second condition in Theorem 6 then follows immediately.

A similar argument applies to the other two conditions. □

Notice that a polynomial $B(\mathbf{x})$ satisfying the sufficient condition in Theorem 5 suffices as an invariant barrier certificate that witnesses safety of the system. In contrast, a polynomial $B(\mathbf{x})$ satisfying the necessary condition in Theorem 6 may serve as a candidate invariant barrier certificate, and safety of the system can be concluded via a posterior check of $B(\mathbf{x})$ per Definition 4. Such a check inherits decidability of the first-order theory over real-closed fields [34].

Next *we show how to encode an SOS constraint of the shape "$h(\mathbf{x}) \in \Sigma[\mathbf{x}]$" in Theorems 5 and 6 as a BMI constraint*. To this end, we first set a *template polynomial* $B(\mathbf{a}, \mathbf{x})$ parameterized by unknown real coefficients $\mathbf{a}$ as the barrier certificate (required to be linear in its parameters $\mathbf{a}$). We then proceed by setting templates for the remaining unknown polynomials (e.g., $v_{i,j}(\mathbf{x})$) and SOS polynomials (e.g., $\sigma(\mathbf{x})$ and $\rho(\mathbf{x})$) in $h(\mathbf{x})$, with all the parameters in these templates grouped in $\mathbf{s}$. Observe that the parameterized SOS polynomial $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ is of a bilinear form on the parameter spaces, i.e., $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ is linear in $\mathbf{a}$ and $\mathbf{s}$ separately. However, nonlinearity arises in the combined parameter space $(\mathbf{a}, \mathbf{s})$ due to the product couplings of $\mathbf{a}$ and $\mathbf{s}$, i.e., $v_{i,j}(\mathbf{s}_{i,j}, \mathbf{x})\mathcal{L}_f^j B(\mathbf{a}, \mathbf{x})$ in the consecution constraint.

Now the problem of synthesizing an invariant barrier certificate boils down to searching for an instantiation of the parameters $\mathbf{a}$ and $\mathbf{s}$ such that the sufficient condition in Theorem 5 holds (or alternatively, the necessary condition in Theorem 6 holds and the posterior check of Definition 4 passed). Such an instantiation of $\mathbf{a}$ (making $B(\mathbf{a}, \mathbf{x})$ an invariant barrier certificate) will be called *valid* in the sequel.

Suppose that a parameterized SOS polynomial $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ is of degree at most $2d$, with user-specified $d \in \mathbb{N}$. Then $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ can always be written in *quadratic form* as $h(\mathbf{a}, \mathbf{s}, \mathbf{x}) = \mathbf{b}^{\mathsf{T}} Q(\mathbf{a}, \mathbf{s})\mathbf{b}$, where $\mathbf{b} = (1, x_1, x_2, x_1x_2, \ldots, x_n^d)$ is the *basis vector* of size $p = \binom{n+d}{n}$ containing all monomials of degree up to d, and $Q(\mathbf{a}, \mathbf{s}) \in \mathcal{S}^p$ is a parameterized real symmetric matrix known as the *Gram matrix* [35].[5] An important fact states that $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ *is SOS if and only if* $Q(\mathbf{a}, \mathbf{s}) \succeq 0$.

Let $\mathcal{F}(\mathbf{a}, \mathbf{s}) = -Q(\mathbf{a}, \mathbf{s})$. As per $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$, the matrix-valued function $\mathcal{F}(\mathbf{a}, \mathbf{s})$ is bilinear in $(\mathbf{a}, \mathbf{s})$. Observe that $h(\mathbf{a}, \mathbf{s}, \mathbf{x})$ *is SOS if and only if the BMI constraint* $\mathcal{F}(\mathbf{a}, \mathbf{s}) \preceq 0$ *holds*. See Example 1 for an illustration of this BMI encoding.

In general, $\mathcal{F}(\mathbf{a}, \mathbf{s})$ can be flattened in an expanded bilinear form as

$$\mathcal{F}(\mathbf{a}, \mathbf{s}) \;=\; F + \sum\nolimits_{i=1}^{m} a_i H_i + \sum\nolimits_{j=1}^{n} s_j G_j + \sum\nolimits_{i=1}^{m}\sum\nolimits_{j=1}^{n} a_i s_j F_{i,j}$$

where m and n are the size of $\mathbf{a}$ and $\mathbf{s}$, respectively; $F, H_i, G_j, F_{i,j} \in \mathcal{S}^p$ are constant matrices. Discharging the conditions of invariant barrier certificates hence amounts to solving the BMI feasibility problem of finding $\mathbf{a}$ and $\mathbf{s}$ s.t.

$$\mathcal{F}_\iota(\mathbf{a}, \mathbf{s}) \;\preceq\; 0, \quad \iota = 1, 2, \ldots, l\,. \tag{14}$$

Here $\mathcal{F}(\mathbf{a}, \mathbf{s})$ is indexed by ι and l is the number of SOS constraints involved.

To exploit well-developed techniques in optimization, the feasibility problem (14) is transformed to an optimization problem subject to BMI constraints:

$$\begin{aligned} &\underset{\lambda, \mathbf{a}, \mathbf{s}}{\text{maximize}} \quad \lambda \\ &\text{subject to} \quad \mathcal{F}_\iota(\mathbf{a}, \mathbf{s}) + \lambda I \;\preceq\; 0, \quad \iota = 1, 2, \ldots, l\,. \end{aligned} \tag{15}$$

A solution $(\lambda, \mathbf{a}, \mathbf{s})$ to (15) is *feasible* if it satisfies the BMIs in (15), and *strictly feasible* if all the BMIs are satisfied with strict inequalities. We sometimes drop the λ component in the solution when it is clear from the context. Notice that *problem* (14) *has a feasible solution if and only if the optimal value* λ^* *in the BMI optimization problem* (15) *is non-negative*.

To achieve (weak) completeness of our method in subsequent sections on solving the BMI optimization problem, we make the following assumption on the boundedness of the search space $(\mathbf{a}, \mathbf{s})$ of the optimization.

Assumption 1 *(Boundedness on the parameters).* Every feasible solution $(\mathbf{a}, \mathbf{s})$ to the BMI problem (15) is in a compact set with non-empty interior, i.e.,

$$(\mathbf{a}, \mathbf{s}) \;\in\; \mathcal{C}_\mathbf{a} \times \mathcal{C}_\mathbf{s} \;=\; \left\{(\mathbf{a}, \mathbf{s}) \;\middle|\; \|\mathbf{a}\|^2 \le L_\mathbf{a}, \|\mathbf{s}\|^2 \le L_\mathbf{s}\right\}$$

for some known bounds $L_\mathbf{a}, L_\mathbf{s} \in \mathbb{R}^+$.

Remark 6. The boundedness on $\mathbf{a}$ in Assumption 1 makes sense in practice since we usually prefer barrier certificates with bounded coefficients. Moreover, when the bilinear functions $\mathcal{F}_\iota(\mathbf{a}, \mathbf{s})$ in (15) are affine in $\mathbf{a}$ and $\mathbf{s}$, i.e., with a zero constant matrix F, the parameters $\mathbf{a}$ and $\mathbf{s}$ can be scaled independently by any positive factor. Therefore in this case, w.l.o.g., one may simply set $L_\mathbf{a} = L_\mathbf{s} = 1$.

[5] Extracting the Gram matrix amounts to solving a system of linear equations resulting from coefficient matching. The derived Gram matrix may contain extra unknowns if the system of linear equations admits multiple solutions, which nevertheless can be encoded in our subsequent workflow by enumerating the basis of its null space.

5. Solving BMI optimizations via DCP

The BMI optimization problem (15), derived from the synthesis problem, is known to be NP-hard and contains non-convex constraints [19], and hence is not amenable to efficient (polynomial-time) algorithms in contrast to convex optimization. In this section, we present an algorithm for solving general BMI optimizations via difference-of-convex programming [22,23], which solves a series of convex sub-problems that approaches a local optimum of (15).

For brevity, we consider optimization problems with a single BMI constraint (whereas multiple BMI constraints can be joined as a single BMI in a block-diagonal fashion):

$$\begin{array}{ll} \text{maximize} & g(\mathbf{z}) \\ \mathbf{z}=(\mathbf{x},\mathbf{y}) & \\ \text{subject to} & \mathcal{B}(\mathbf{x},\mathbf{y}) \;\hat{=}\; F+\sum_{i=1}^{m} x_i H_i+\sum_{j=1}^{n} y_j G_j+\sum_{i=1}^{m}\sum_{j=1}^{n} x_i y_j F_{i,j} \;\preceq\; 0 \end{array} \tag{16}$$

where the objective function $g\colon \mathbb{R}^{m+n}\to\mathbb{R}$ is linear in $\mathbf{z}=(\mathbf{x},\mathbf{y})$; $F, H_i, G_j, F_{i,j}\in\mathcal{S}^p$ are constant symmetric matrices.

5.1. Difference-of-convex decomposition

The key challenge in solving the BMI problem (16) is its non-convexity, that is, the matrix-valued function $\mathcal{B}(\mathbf{x},\mathbf{y})$ is, in general, not PSD-convex.

There have been attempts, most pertinently in [27], to decompose a bilinear function as a difference between two PSD-convex functions, known as the *difference-of-convex* (DC) *decomposition*, such that the optimization in its decomposed form enjoys well-established techniques in difference-of-convex programming [22,23]. The DC decomposition in [27], however, is confined to BMIs of a specific structure, namely, $X^{\mathsf{T}}Y+Y^{\mathsf{T}}X\preceq 0$, where X and Y are matrix variables containing variables x_i and y_j, respectively. The more general bilinear function $\mathcal{B}(\mathbf{x},\mathbf{y})$ in (16) does unfortunately not admit straightforward forms of decomposition such as those in [27, Lemma 3.1].

In this subsection, we first show how to formulate a difference-of-convex decomposition of the matrix-valued function $\mathcal{B}(\mathbf{x},\mathbf{y})$ using matrix decomposition (inspired by [36]), and then present three different ways to obtain such a matrix decomposition. These decomposition methods compete with each other in terms of theoretical simplicity, generality, and the exploitation of matrix sparsity.

First, observe that the function $\mathcal{B}(\mathbf{x},\mathbf{y})$ can be written as

$$\mathcal{B}(\mathbf{x},\mathbf{y}) \;=\; \begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}^{\mathsf{T}}\begin{pmatrix}0 & \Gamma\\ \Gamma^{\mathsf{T}} & 0\end{pmatrix}\begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}+\begin{pmatrix}\Omega_1 & \Omega_2\end{pmatrix}\begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}+F \tag{17}$$

where 0 represents the zero matrices with compatible dimensions and

$$\Gamma \;=\; \frac{1}{2}\begin{pmatrix}F_{1,1} & \dots & F_{1,n}\\ \vdots & \ddots & \vdots\\ F_{m,1} & \dots & F_{m,n}\end{pmatrix},\quad \Omega_1 \;=\; \begin{pmatrix}H_1 & \dots & H_m\end{pmatrix},\quad \Omega_2 \;=\; \begin{pmatrix}G_1 & \dots & G_n\end{pmatrix}.$$

The form of (17) implies that $\mathcal{B}(\mathbf{x},\mathbf{y})$ *is PSD-convex if the matrix* $M=\begin{pmatrix}0 & \Gamma\\ \Gamma^{\mathsf{T}} & 0\end{pmatrix}$ *is positive semidefinite*. Unfortunately, as [36, Theorem 1] points out, for a non-trivial bilinear function $\mathcal{B}(\mathbf{x},\mathbf{y})$, M may not be positive semidefinite.

Nevertheless, the matrix M can always be decomposed as $M=M_1-M_2$ with $M_1, M_2\succeq 0$, i.e., a difference between two PSD-matrices. This, in turn, leads to a DC decomposition of $\mathcal{B}(\mathbf{x},\mathbf{y})$:

Theorem 7 *(DC decomposition by matrix decomposition). Suppose $M=M_1-M_2$ with $M_1, M_2\succeq 0$. Then, the form*

$$\mathcal{B}(\mathbf{x},\mathbf{y}) \;=\; \mathcal{B}^{+}(\mathbf{x},\mathbf{y})-\mathcal{B}^{-}(\mathbf{x},\mathbf{y}) \tag{18}$$

where

$$\begin{aligned}\mathcal{B}^{+}(\mathbf{x},\mathbf{y}) &\;=\; \begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}^{\mathsf{T}} M_1\begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}+\begin{pmatrix}\Omega_1 & \Omega_2\end{pmatrix}\begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}+F\\ \mathcal{B}^{-}(\mathbf{x},\mathbf{y}) &\;=\; \begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}^{\mathsf{T}} M_2\begin{pmatrix}\mathbf{x}\otimes I\\ \mathbf{y}\otimes I\end{pmatrix}\end{aligned}$$

is a difference-of-convex decomposition of $\mathcal{B}(\mathbf{x},\mathbf{y})$, i.e., the matrix-valued functions $\mathcal{B}^{+}(\mathbf{x},\mathbf{y})$ and $\mathcal{B}^{-}(\mathbf{x},\mathbf{y})$ are PSD-convex on $\mathbb{R}^{m+n}$.

Proof. We first show the PSD-convexity of $\mathcal{B}^+(\mathbf{x},\mathbf{y})$. Let $\mathbf{z}=(\mathbf{x},\mathbf{y})\in\mathbb{R}^{m+n}$. According to [37, Proposition 1], $\mathcal{B}^+(\mathbf{z})=\mathcal{B}^+(\mathbf{x},\mathbf{y})$ is PSD-convex if (and only if) for any $\mathbf{v}\in\mathbb{R}^p$, the function $\phi_{\mathbf{v}}(\mathbf{z})=\mathbf{v}^\mathsf{T}\mathcal{B}^+(\mathbf{z})\mathbf{v}$ is convex. Note that

$$\begin{aligned}\phi_{\mathbf{v}}(\mathbf{z}) &= \mathbf{v}^\mathsf{T}\left(\mathbf{z}\otimes I\right)^\mathsf{T} M_1\left(\mathbf{z}\otimes I\right)\mathbf{v}+\mathbf{v}^\mathsf{T}\begin{pmatrix}\Omega_1 & \Omega_2\end{pmatrix}\left(\mathbf{z}\otimes I\right)\mathbf{v}+\mathbf{v}^\mathsf{T}F\mathbf{v}\\ &= (\mathbf{z}\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}\otimes\mathbf{v})+\mathbf{v}^\mathsf{T}\begin{pmatrix}\Omega_1 & \Omega_2\end{pmatrix}(\mathbf{z}\otimes\mathbf{v})+\mathbf{v}^\mathsf{T}F\mathbf{v}\,.\end{aligned}$$

Then, for any $\mu_1\in(0,1)$ and $\mu_2=1-\mu_1$, we have, for any $\mathbf{z}_1,\mathbf{z}_2\in\mathbb{R}^{m+n}$,

$$\begin{aligned}&\phi_{\mathbf{v}}(\mu_1\mathbf{z}_1+\mu_2\mathbf{z}_2)-(\mu_1\phi_{\mathbf{v}}(\mathbf{z}_1)+\mu_2\phi_{\mathbf{v}}(\mathbf{z}_2))\\ =\ &(\mu_1(\mathbf{z}_1\otimes\mathbf{v})+\mu_2(\mathbf{z}_2\otimes\mathbf{v}))^\mathsf{T}M_1(\mu_1(\mathbf{z}_1\otimes\mathbf{v})+\mu_2(\mathbf{z}_2\otimes\mathbf{v}))-\mu_1(\mathbf{z}_1\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_1\otimes\mathbf{v})-\mu_2(\mathbf{z}_2\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_2\otimes\mathbf{v})\\ =\ &\mu_1\mu_2(\mathbf{z}_2\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_1\otimes\mathbf{v})+\mu_1\mu_2(\mathbf{z}_1\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_2\otimes\mathbf{v})-\mu_1\mu_2(\mathbf{z}_1\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_1\otimes\mathbf{v})\\ &-\mu_1\mu_2(\mathbf{z}_1\otimes\mathbf{v})^\mathsf{T}M_1(\mathbf{z}_1\otimes\mathbf{v})\\ =\ &-\mu_1\mu_2((\mathbf{z}_1-\mathbf{z}_2)\otimes\mathbf{v})^\mathsf{T}M_1((\mathbf{z}_1-\mathbf{z}_2)\otimes\mathbf{v})\\ \le\ &0 \qquad\qquad (\text{positive semidefiniteness of } M_1)\end{aligned}$$

which means that $\phi_{\mathbf{v}}(\mathbf{z})$ is convex. Thus, $\mathcal{B}^+(\mathbf{x},\mathbf{y})$ is PSD-convex.

The PSD-convexity of $\mathcal{B}^-(\mathbf{x},\mathbf{y})$ can be shown in an analogous way. □

It remains to find a matrix decomposition of M. In what follows, we present three different ways to decompose the matrix $M\in\mathcal{S}^{(m+n)p}$ as a difference between two PSD-matrices. Notice that M is a real symmetric matrix and thus only has real eigenvalues.

5.1.1. Decompose M via eigendecomposition

A (real symmetric) matrix is positive semidefinite if and only if all of its eigenvalues are non-negative. Although the matrix M may have both non-negative and negative eigenvalues, we can "group" them respectively in PSD-matrices M_1 and M_2 such that $M=M_1-M_2$.

One way to do so is to use the *eigendecomposition* of M. That is, $M=V^\mathsf{T}DV$, where the orthogonal matrix V contains the eigenvectors of M, and D is a diagonal matrix whose diagonal elements are the eigenvalues of M.

Let D^+ be the matrix obtained by setting all negative elements of D to zero, and $D^-=D^+-D$. Then,

$$M = \underbrace{V^\mathsf{T}D^+V}_{M_1}-\underbrace{V^\mathsf{T}D^-V}_{M_2}\,. \tag{19}$$

It follows from construction that $M_1,M_2\succeq 0$, and therefore, by Theorem 7, we obtain a DC decomposition of $\mathcal{B}(\mathbf{x},\mathbf{y})$.

5.1.2. Decompose M via bounds on eigenvalues

The eigendecomposition-based DC decomposition is theoretically simple, yet does not benefit from the sparsity nature of M: The matrix $M=\begin{pmatrix}0 & \Gamma\\ \Gamma^\mathsf{T} & 0\end{pmatrix}\in\mathcal{S}^{(m+n)p}$ in (17) is often highly sparse, which is potentially a useful feature in accelerating many matrix operations. However, sparsity is of little value when *all* of the eigenvalues and eigenvectors are needed, which typically takes time cubic in the matrix size [38]. In particular, the decomposed matrices M_1 and M_2 may not be as sparse as M is, thus slowing down almost all the subsequent matrix manipulations.

The key observation here is that, *to obtain a DC decomposition, one does not need to compute all the eigenvalues*. In fact, *it suffices to find a bound on the eigenvalues*: Let $\lambda_u\in\mathbb{R}_0^+$ be an upper-bound on all the eigenvalues of M (the symbol λ shall not be confused with those used in optimization problems). We have

$$M = \underbrace{\lambda_u I}_{M_1}-\underbrace{(\lambda_u I-M)}_{M_2}\,. \tag{20}$$

Here, $M_1\succeq 0$ trivially holds as $\lambda_u\geq 0$. The positive semidefiniteness of $M_2=\lambda_u I-M$ can be shown by considering the eigendecomposition of M:

$$M_2 = \lambda_u I-V^\mathsf{T}DV = V^\mathsf{T}(\lambda_u I-D)\,V$$

where the diagonal matrix $\lambda_u I-D$ contains the eigenvalues of M_2. Since λ_u upper-bounds all the eigenvalues of M (diagonal elements in D), $\lambda_u I-D$ contains only non-negative values, and thus we conclude that $M_2\succeq 0$.

In order to obtain the upper-bound λ_u, it suffices to compute only the *largest eigenvalue* of M, which can be done substantially more efficient than conducting the full eigendecomposition, especially for sparse M [39, Chapter VI]. Moreover, the decomposed matrices M_1 and M_2 given in (20) are guaranteed to be as sparse as M is.

We remark, however, that the derived matrices M_1 and M_2 in (20) have inevitably larger entries than those built from eigendecomposition. In practice, larger entries in M_2 may increase the linearization error (in the transformation to convex sub-problems, cf. Section 5.2), thereby slowing down the convergence of the iterative DCP procedure.

Remark 7. Apart from using an upper-bound $\lambda_u \geq 0$ on the eigenvalues of M, a DC decomposition can also be obtained by using a lower bound $\lambda_l \leq 0$ on the eigenvalues of M. In that case, we have $M_1 = M - \lambda_l I$ and $M_2 = -\lambda_l I$.

5.1.3. Decompose M via SDP

The problem of decomposing the matrix M as a difference between two PSD-matrices can alternatively be modeled as an SDP problem:

$$\begin{aligned} \underset{M_2}{\text{minimize}} \quad & tr\,(M_2) \\ \text{subject to} \quad & M + M_2 \succeq 0\,, \\ & M_2 \succeq 0\,. \end{aligned} \tag{21}$$

A feasible solution to (21) clearly induces a matrix decomposition (with $M_1 = M + M_2$) as required in Theorem 7. The objective function (i.e., the trace of M_2) in (21) intuitively measures the magnitude of the (undesired) "concave part" $-\mathcal{B}^-(\mathbf{x}, \mathbf{y})$ in (18). As argued previously, minimizing such an objective may reduce the linearization error and thus expedite the DCP procedure.[6]

Although it would seem to be more time-consuming to solve an SDP problem than to perform the eigendecomposition, the specific SDP instance (21) can often be solved rather efficiently by exploiting the sparsity pattern of M, e.g., the chordal sparsity [42]. Alternatively, one can improve the performance by imposing a certain sparsity structure (e.g., to be diagonal) on M_1 or M_2. For instance, one possible formulation using diagonal matrix $M_1 = diag(\mathbf{c})$ is

$$\begin{aligned} \underset{\mathbf{c}}{\text{minimize}} \quad & tr\,(diag\,(\mathbf{c}) - M) \\ \text{subject to} \quad & c_i \geq 0, \quad i = 1, 2, \ldots, (m+n)p\,, \\ & diag(\mathbf{c}) - M \succeq 0 \end{aligned}$$

which can be further rewritten as a (sparse) LMI problem:

$$\begin{aligned} \underset{\mathbf{c}}{\text{minimize}} \quad & \sum\nolimits_i c_i \\ \text{subject to} \quad & c_i \geq 0, \quad i = 1, 2, \ldots, (m+n)p\,, \\ & \sum\nolimits_i c_i\, \mathbf{e}_i^{\mathsf{T}} \mathbf{e}_i - M \succeq 0 \end{aligned} \tag{22}$$

where $\mathbf{e}_i$ denotes a row vector with 1 in its i-th column and 0's elsewhere. When M admits a specific sparsity pattern, the LMI problem (22) can be solved extremely efficiently (see, e.g., [43], for solving LMIs with thousands of variables in minutes).

In a nutshell, the eigendecomposition-based method is theoretically simple, yet does not benefit from the sparsity nature of M. Decomposing M via bounds on eigenvalues exploits the sparsity nature of M —thereby yielding considerably faster matrix operations, but may slow down the convergence of the iterative DCP procedure. The SDP-based decomposition may expedite the DCP procedure, but is theoretically more involved and stands out only when M admits specific sparsity patterns. We will compare these different DC decomposition methods empirically in Section 7.

5.2. Reduction to LMIs

On top of a DC decomposition (cf. Theorem 7), we can now apply a standard iterative procedure in difference-of-convex programming [27] to solve the BMIs.

The core idea of the procedure is to iteratively solve a series of convex sub-problems. More specifically, given a feasible solution $\mathbf{z}^k = (\mathbf{x}^k, \mathbf{y}^k)$ to the BMI optimization problem (16), the "concave part" $-\mathcal{B}^-(\mathbf{x}, \mathbf{y})$ in (18) is linearized around $\mathbf{z}^k$, thereby yielding a series of convex programs ($k = 0, 1, \ldots$):

$$\begin{aligned} \underset{\mathbf{z} = (\mathbf{x}, \mathbf{y})}{\text{maximize}} \quad & g(\mathbf{z}) + \frac{1}{2}\delta \left\| \mathbf{z} - \mathbf{z}^k \right\|^2 \\ \text{subject to} \quad & \mathcal{B}^+(\mathbf{z}) - \mathcal{B}^-\left(\mathbf{z}^k\right) - \mathcal{D}\mathcal{B}^-\left(\mathbf{z}^k\right)\left(\mathbf{z} - \mathbf{z}^k\right) \preceq 0 \end{aligned} \tag{23}$$

[6] A good DC decomposition should make the concave part (locally) "as affine as possible". Such "affineness" can be measured by the Hessian matrix for scalar-valued functions (see [40]). For matrix-valued functions, the Hessian is in fact a 4-th rank tensor, but its norm can still be bounded by the norm of a certain matrix (cf. [41]). That matrix, in our case, is exactly the matrix M_2.

Algorithm 1: BMI-DC: solving BMIs based on DC decomposition.

input: A BMI optimization problem (16) with a strictly feasible initial solution $\mathbf{z}^0$.
output: A sequence of feasible solutions $S = \{\mathbf{z}^0, \ldots, \mathbf{z}^k\}$ to the BMI optimization.

$k \leftarrow 0$; $S \leftarrow \{\mathbf{z}^0\}$;
$M \leftarrow$ reformulation of (16) as (17);
$(M_1, M_2) \leftarrow$ matrix decomposition of M as in Theorem 7;
repeat
  Construct the convex sub-problem (23) out of (M_1, M_2) linearized around $\mathbf{z}^k$;
  $\mathbf{z}^{k+1} \leftarrow$ optimum of the program (23);
  $S \leftarrow S \cup \{\mathbf{z}^{k+1}\}$; ▷ S keeps track of visited points
  $k \leftarrow k + 1$;
until $\|\mathbf{z}^k - \mathbf{z}^{k-1}\| < \varepsilon$ *for a given tolerance* $\varepsilon \in \mathbb{R}_0^+$;
return S;

where $\mathcal{D}\mathcal{B}^-(\mathbf{z})\colon \mathbb{R}^{m+n} \to \mathcal{S}^p$ is the derivative of the matrix-valued function $\mathcal{B}^-$ at $\mathbf{z}$, i.e., a linear mapping from a vector $\mathbf{u} \in \mathbb{R}^{m+n}$ to a matrix in $\mathcal{S}^p$:

$$\mathcal{D}\mathcal{B}^-(\mathbf{z})(\mathbf{u}) \;\hat{=}\; \sum\nolimits_{i=1}^{n+m} u_i \frac{\partial \mathcal{B}^-}{\partial z_i}(\mathbf{z}) \,.$$

An extra regularization term $\frac{1}{2}\delta\|\mathbf{z} - \mathbf{z}^k\|^2$ with $\delta < 0$ is added in (23) to enforce that $g(\mathbf{z})$ strictly increases after each iteration until it stabilizes, which can be encoded as a second-order cone constraint and embedded in SDP solving.

Note that the linearized problem (23) is convex and therefore can be solved efficiently (see, e.g., [44]). Furthermore, Theorem 3 can also be used to reformulate (23) as an LMI problem:

Theorem 8 *(Reduction to LMIs). The quadratic matrix inequality (QMI) constraint*

$$\mathcal{B}^+(\mathbf{z}) - \mathcal{B}^-\left(\mathbf{z}^k\right) - \mathcal{D}\mathcal{B}^-\left(\mathbf{z}^k\right)\left(\mathbf{z} - \mathbf{z}^k\right) \;\preceq\; 0$$

in (23) *is equivalent to the LMI constraint (of the size* $(m+n+1)p$*)*

$$\begin{pmatrix} -I & N(\mathbf{z}\otimes I) \\ (\mathbf{z}\otimes I)^{\mathsf{T}} N^{\mathsf{T}} & -\mathcal{B}^-\left(\mathbf{z}^k\right) - \mathcal{D}\mathcal{B}^-\left(\mathbf{z}^k\right)\left(\mathbf{z} - \mathbf{z}^k\right) + \Omega(\mathbf{z}\otimes I) + F \end{pmatrix} \;\preceq\; 0$$

where N *is the square root matrix of* M_1*, i.e.,* $M_1 = N^{\mathsf{T}}N$*, and* $\Omega = \begin{pmatrix} \Omega_1 & \Omega_2 \end{pmatrix}$.

Proof. Note that the square root matrix N of M_1 exists since $M_1 \succeq 0$.[7] The claim then follows immediately by applying the Schur complement in Theorem 3. □

Theorem 8 entails that the series of linearized convex sub-problems of the form (23) can be solved alternatively by most off-the-shelf SDP solvers designated for discharging LMIs via polynomial-time algorithms [28], say the interior-point methods. Furthermore, by taking the optimum of the k-th sub-problem to be the next linearization point $\mathbf{z}^{k+1}$, we obtain an iterative procedure for solving general BMIs, as depicted in Algorithm 1.

Algorithm 1 falls into the DCP framework [27] and thus enjoys useful properties, e.g., soundness, termination and convergence as follows.

Theorem 9 *(Soundness). Every solution* $\mathbf{z}^i = (\mathbf{x}^i, \mathbf{y}^i) \in S$ *with* $i = 0, \ldots, k$ *returned by Algorithm 1 is a feasible solution to the original BMI problem* (16)*.*

Proof. We prove by induction on i. The base case holds as $\mathbf{z}^0$ is assumed to be a feasible solution to (16). For the induction step, we show that $\mathbf{z}^{i+1}$ is a feasible solution to (16) if $\mathbf{z}^i$ is a feasible solution to (16). Since $\mathbf{z}^{i+1}$ is a feasible solution to (23) linearized at $\mathbf{z}^i$, it suffices to show that the feasible set of (23) is a subset (or, an under-approximation) of the feasible set of (16).

Theorem 7 shows that $\mathcal{B}^-(\mathbf{z})$ is PSD-convex, then by [27, Lemma 2.2 (b)], we have

$$\mathcal{B}^-(\mathbf{z}) - \mathcal{B}^-\left(\mathbf{z}^i\right) \;\succeq\; \mathcal{D}\mathcal{B}^-\left(\mathbf{z}^i\right)\left(\mathbf{z} - \mathbf{z}^i\right) \,. \tag{24}$$

In the meantime, $\mathbf{z}^i$ is a feasible solution to (23) and thus fulfills

[7] In case we have $M_1 = V^{\mathsf{T}}D^+V$ (with only non-negative eigenvalues in D^+) from the eigendecomposition of M, the matrix N can be computed as $N = V^{\mathsf{T}}(D^+)^{1/2}V$, where $(D^+)^{1/2}$ is the diagonal matrix whose diagonal elements are square roots of those in D^+. For the other decomposition methods as presented in Section 5.1, N can be obtained via Cholesky decomposition of M_1.

$$\mathcal{B}^+(\mathbf{z}) - \mathcal{B}^-\left(\mathbf{z}^i\right) - \mathcal{D}\mathcal{B}^-\left(\mathbf{z}^i\right)\left(\mathbf{z} - \mathbf{z}^i\right) \preceq 0\,. \tag{25}$$

Combining (24) and (25), we have $\mathcal{B}(\mathbf{x},\mathbf{y}) = \mathcal{B}^+(\mathbf{z}) - \mathcal{B}^-(\mathbf{z}) \preceq 0$ which is exactly the BMI constraint of (16). This completes the proof. □

The result below states termination and convergence of Algorithm 1 in terms of *KKT points* of (16), i.e., solutions fulfilling the KKT conditions [32] of (16). The KKT conditions, short for Karush-Kuhn-Tucker conditions, are used to determine the optimality of a solution to a constrained nonlinear optimization problem. Addressing these conditions in detail falls outside the scope of this paper.

Theorem 10 *(Termination and convergence). If (16) has finitely many KKT points, then (1) for $\varepsilon \in \mathbb{R}^+$, Algorithm 1 terminates; (2) for $\varepsilon = 0$, Algorithm 1 visits an infinite sequence of solutions converging to a KKT point.*

Proof. Let $\bar{S} = \{\mathbf{z}^i\}_{i\in\mathbb{N}}$ be the infinite sequence of visited points for $\varepsilon = 0$.

We first show that (2) implies (1). Assume that (2) holds, i.e., $\bar{S}$ converges (to a KKT point of (16)), then by Cauchy's criterion for convergence, we have $\forall\varepsilon \in \mathbb{R}^+.\, \exists k \in \mathbb{N}^+ \colon \|\mathbf{z}^k - \mathbf{z}^{k-1}\| < \varepsilon$ (with $\mathbf{z}^k, \mathbf{z}^{k-1} \in \bar{S}$). Algorithm 1 thus terminates.

It then remains to show that $\bar{S}$ converges to a KKT point of (16) if the set of KKT points of (16) is finite. This is in fact a straightforward corollary of [27, Theorem 4.3], by noticing that the assumptions thereof can be readily verified. For simplicity, we highlight the validity of only a few of these assumptions: Since $\mathbf{z}^0$ in Algorithm 1 is a strictly feasible solution to (16), the relative interior of the feasible set of (16) is non-empty and thus Assumption A1 in [27] holds; Our Assumption 1 on the boundedness of the search space ensures that $g(\mathbf{z})$ in (16) is bounded from above over a bounded feasible set, and therefore the boundedness assumptions in [27, Theorem 4.3] hold. □

We remark that, under some sufficient KKT conditions and regularity conditions [32], a KKT point suffices as a local optimum. In this case, the infinite sequence $\{\mathbf{z}^i\}_{i\in\mathbb{N}}$ of points visited by Algorithm 1 (for $\varepsilon = 0$) converges to a local optimum of (16).

It is also worth noting that, in [45], the authors presented a DC-based approach to synthesizing parameters in parametric Markov decision processes, which integrates (probabilistic) model checking into the DCP procedure, thereby yielding possibly earlier termination and numerically more stable results in practice. It is our future interest to investigate a similar idea in the context of barrier-certificate synthesis for hybrid systems.

5.3. Complexity of Algorithm 1

We discuss ingredients for establishing the time complexity of Algorithm 1, which concerns (1) computing the DC decomposition; (2) performing a single iteration; and (3) conducting a number of iterations (up to a desired precision).

Recall that the matrix M to be decomposed (cf. Theorem 7) is of the size $(m+n)p$, where m and n are the number of parameters in the template barrier certificate (i.e., size of $\mathbf{a}$) and other template polynomials (i.e., size of $\mathbf{s}$), respectively; $p = \binom{r+d}{r}$ bounds the size of the basis vector $\mathbf{b}$ (where r is the system dimension and the SOS polynomial is of degree at most $2d$). All the three DC decomposition methods in Section 5.1 can be done in polynomial time, e.g., $O((m+n)^3p^3)$ for the eigendecomposition of M [38].

Performing a single iteration in Algorithm 1 amounts to solving an LMI instance with $k+2$ constraints (derived from Definition 4) where k is the order of Lie derivatives considered (bounded by $N_{B,\boldsymbol{f}}$). Computing $N_{B,\boldsymbol{f}}$ is non-elementary in theory (described in terms of the fast-growing hierarchy [46] or an explicit Ackermannian function [47,48]), yet it is relatively small in practice and can be obtained offline. Each LMI constraint involves matrices in $\mathcal{S}^{(m+n+1)p}$ (see Theorem 8), which can be solved in $O(((m+n)p)^{6.5})$ [49]. Note that, in practice, the computation time is often significantly less than this theoretical bound especially when the matrices in the LMI instance admit specific sparsity patterns (see, e.g., [43], for solving LMIs with thousands of variables in minutes).

Bounding or even estimating the number of iterations required to achieve a desired precision is non-trivial: one needs to determine the *rate of convergence* of the sequence of solutions produced by the iterative procedure. Since Algorithm 1 essentially builds first-order approximations of the original BMI optimization problem, one may reasonably assume that it is at least linearly convergent. However, to the best of our knowledge, proving linear convergence for general difference-of-convex algorithms remains an open problem [50], albeit with some known results on typical subclasses [51]. In practice, nonetheless, difference-of-convex algorithms often converge to a local optimum within a few iterations, as can be observed in our experiments in Section 7.

5.4. Finding the initial solution

The iterative procedure in Algorithm 1 starts with a fed-by-oracle strictly feasible initial solution $\mathbf{z}^0$ to the BMI problem (16). Finding such an initial solution, however, is non-trivial in general due to the non-convexity of (16). We argue though, that a strictly feasible initial solution can be obtained for the BMI problem of the form (15) induced by the barrier-certificate synthesis problem.

Recall that in the BMI problem (15), bilinearity arises from the multiplication of $B(\mathbf{a}, \mathbf{x})$ with some unknown multiplier polynomials parameterized by $\mathbf{s}$. One way to reduce the BMI constraints to LMIs is to fix every multiplier polynomial to be a non-negative constant, thereby yielding a linear program:

$$\begin{aligned} &\underset{\lambda, \mathbf{a}}{\text{maximize}} \quad \lambda \\ &\text{subject to} \quad \mathcal{F}_\iota(\mathbf{a}, \mathbf{s})\big|_{\mathbf{s}=(c_\iota, 0, \ldots, 0)} + \lambda I \preceq 0, \quad \iota = 1, 2, \ldots, l \end{aligned} \tag{26}$$

where $\mathbf{s}$ in $\mathcal{F}_\iota(\mathbf{a}, \mathbf{s})$ is substituted by $(c_\iota, 0, \ldots, 0)$ with $c_\iota \in \mathbb{R}_0^+$, which encodes a non-negative constant multiplier polynomial. Observe that no $\mathbf{s}$-variable is involved in (26) and the constraints therein are linear in $\mathbf{a}$.

Evidently, a strictly feasible solution $(\lambda, \mathbf{a})$ to (26) induces a strictly feasible solution $(\lambda, \mathbf{a}, (c_\iota, 0, \ldots, 0))$ to (15) as well. Moreover, we have

Lemma 2. *The LMI program (26) always has a strictly feasible solution.*

Proof. Let $\Lambda_{\mathbf{a}} \hat{=} \min_{1 \le \iota \le l} -\rho\left(\mathcal{F}_\iota(\mathbf{a}, \mathbf{s})\big|_{\mathbf{s}=(c_\iota, 0, \ldots, 0)}\right)$, where $\rho(A)$ denotes the spectral radius of matrix A, i.e., the largest absolute value of the eigenvalues of A. It follows that program (26) has a strictly feasible solution if $\lambda < \Lambda_{\mathbf{a}}$.

Furthermore, under Assumption 1 on the boundedness of parameter $\mathbf{a} \in \mathcal{C}_{\mathbf{a}}$, $\Lambda_{\mathbf{a}}$ can be shown to be bounded by the well-known Gershgorin circle theorem.

Therefore, by taking an interior point of $\mathcal{C}_{\mathbf{a}}$ as $\tilde{\mathbf{a}}$, and $\tilde{\lambda} = \Lambda_{\tilde{\mathbf{a}}} - \epsilon$ for some $\epsilon \in \mathbb{R}^+$, we obtain a strictly feasible solution $(\tilde{\lambda}, \tilde{\mathbf{a}})$ to program (26). □

As a consequence, *a strictly feasible solution to the BMI problem (15) can be obtained by solving the LMI problem (26)*. In fact, when considering Lie derivatives only up to the first order, solving (the feasibility counterpart of) (26) is exactly the procedure to synthesize either an *exponential barrier certificate* [14] (with $c_\iota \in \mathbb{R}^+$) or a *convex barrier certificate* [9] (with $c_\iota = 0$). Algorithm 1 therefore subsumes existing synthesis techniques in the sense that any valid barrier certificate synthesized by methods in [14,9] can also be discovered by Algorithm 1. Moreover, an alternative way to reduce the BMI constraints to LMIs is to fix the multipliers to be some given non-trivial (SOS) polynomials [16].

Remark 8. Different choices of the multiplier constants c_ι in (26) may lead to different initial solutions fed to Algorithm 1, thereby considerably different numbers of iterations until termination. In practice, techniques like randomization are worth exploring when choosing these multiplier constants.

5.5. Numerical errors in SDP solving and potential solutions

Most of the existing off-the-shelf SDP solvers are based on numerical computations. The underlying numerical errors caused by, e.g., floating-point computations, may hence lead to unsound results in SDP-based verification or synthesis. To circumvent this issue, three different types of solution have been presented in the literature:

- Validated SDP solving: In [52], Roux et al. presented verified SDPs, where the basic idea is to compute a suitable bound $\epsilon \in \mathbb{R}^+$ and replace all matrix-inequality constraints of the form $A \preceq 0$ by the corresponding ϵ-strengthened versions $A + \epsilon I \preceq 0$. In [53], the authors further developed this idea to guarantee the soundness of SDP-based synthesis of nonlinear Craig interpolants.
- Posterior check by symbolic methods: The soundness of numerical SDP-based approaches can be retrieved by performing a posterior check via symbolic methods, e.g., quantifier elimination [54] and SMT solving [55].
- Exact SDP solving: Henrion et al. presented in [56] an exact algorithm based on symbolic homotopy for solving SDP problems. This algorithm, as noted by the authors, can solve SDP instances only of small sizes.

In this article, we exploit the second approach to perform a posterior verification of the synthesized candidate barrier certificates via both the quantifier-elimination procedure in Wolfram Mathematica and the SMT solver Z3 [57].

6. Incorporating in a branch-and-bound framework

The aforementioned iterative procedure on solving a series of convex optimizations converges only to a local optimum of the BMI problem (15) (or more generally, (16)). This means that, in some cases, it may miss the global optimum that induces a non-negative λ^*. We present in this section a solution to this problem by incorporating the iterative procedure into a branch-and-bound framework that searches for the global optimum in a divide-and-conquer fashion, as is a common technique in non-convex optimizations.

Algorithm 2: Branch-and-Bound: searching for a valid parameter $\bar{\mathbf{a}}$.

input: A BMI optimization problem of the form (15) with $\mathcal{C}_{\mathbf{a}} = \{\mathbf{a} \mid \|\mathbf{a}\|^2 \leq L_{\mathbf{a}}\}$.
output: A valid parameter $\bar{\mathbf{a}}$, or otherwise $\perp$ indicating a failure.

1 **if** $L_{\mathbf{a}} < \eta$ **then return** $\perp$; ▷ abort on fine-enough partitions ($\eta \in \mathbb{R}^+$)
2 $\hat{\lambda} \leftarrow$ an upper-bound on the objective value λ of (15) over $(\mathcal{C}_{\mathbf{a}}, \mathcal{C}_{\mathbf{s}})$;
3 **if** $\hat{\lambda} < 0$ **then** ▷ skip branches inducing only negative objective values
4 **return** $\perp$ ▷ if Theorem 6 is used
5 ‖ **goto** Line 12; ▷ if Theorem 5 is used
/* sample-and-check (Line 6 − 7) is not necessary if Theorem 6 is used */
6 $\bar{\mathbf{a}} \leftarrow$ a randomly-sampled point in $\mathcal{C}_{\mathbf{a}}$;
7 **if** $\bar{\mathbf{a}}$ *is valid* **then return** $\bar{\mathbf{a}}$; ▷ check validity (inductive invariance)
8 **if** $proj_{\mathbf{a}}(S_{glb}) \cap \mathcal{C}_{\mathbf{a}} = \emptyset$ **then** ▷ S_{glb} contains a global set of visited points
9 $S \leftarrow$ apply BMI-DC in Algorithm 1 to (15) with initial solution in $(\mathcal{C}_{\mathbf{a}}, \mathcal{C}_{\mathbf{s}})$;
10 $S_{glb} \leftarrow S_{glb} \cup S$;
/* checking validity is not necessary if Theorem 5 is used */
11 **if** *a valid parameter* $\bar{\mathbf{a}} \in proj_{\mathbf{a}}(S)$ *is found* **then return** $\bar{\mathbf{a}}$;
12 $(\mathcal{C}^1_{\mathbf{a}}, \mathcal{C}^2_{\mathbf{a}}) \leftarrow bisect(\mathcal{C}_{\mathbf{a}})$; ▷ partition the parameter space
13 $\bar{\mathbf{a}} \leftarrow$ Branch-and-Bound($\mathcal{C}^1_{\mathbf{a}}$);
14 **if** $\bar{\mathbf{a}} \neq \perp$ **then return** $\bar{\mathbf{a}}$;
15 **else return** Branch-and-Bound($\mathcal{C}^2_{\mathbf{a}}$);

6.1. The branch-and-bound algorithm

The basic idea is as follows. We first try to solve the BMI problem (15) by Algorithm 1 over the compact parameter space $(\mathcal{C}_{\mathbf{a}}, \mathcal{C}_{\mathbf{s}})$. If a valid solution, (i.e., a solution that contains a valid parameter $\bar{\mathbf{a}} \in \mathcal{C}_{\mathbf{a}}$ such that $B(\bar{\mathbf{a}}, \mathbf{x})$ is an invariant barrier certificate) is found, then the corresponding barrier certificate can be obtained. Otherwise, we keep bisecting $\mathcal{C}_{\mathbf{a}}$ and apply Algorithm 1 over each bisection (note that the validity of $\bar{\mathbf{a}} \in \mathcal{C}_{\mathbf{a}}$ does not depend on $\mathbf{s}$, thus we do not partition $\mathcal{C}_{\mathbf{s}}$). The procedure, as depicted in Algorithm 2 in a recursive manner, terminates when a valid parameter is found or the partition is fine enough.

Algorithm 2 takes as input a BMI problem of the form (15) that encodes either the sufficient condition in Theorem 5 or the necessary condition in Theorem 6 for invariant barrier certificates. In the former case, a sample-and-check process (Line 6–7) is necessary to attain (weak) completeness (see Theorem 11). The conditional statement in Line 8 rules out parameter (sub-)spaces that have already been explored, which is the case when the projection of some visited point in S_{glb} (a global set that keeps track of visited points by Algorithm 1, initialized as $\emptyset$) onto $\mathbf{a}$ is in the current parameter space.

To further improve the performance, Algorithm 2 is complemented by an operation (Line 2–5) that prunes branches inducing only negative objective values. This is witnessed by a negative upper-bound on the objective value of (15) over the current parameter space. We defer the computation of such an upper-bound to Section 6.2. When Theorem 5 is used to form (15), however, the partition of the parameter space (Line 12–15) is still necessary to attain completeness, as a negative objective value of (15) encoding the sufficient condition for invariant barrier certificate may still induce a valid parameter. In practice, one may choose to preferentially explore (partition) branches with larger $\hat{\lambda}$.

The following theorem claims a weak completeness result: our method guarantees to find a barrier certificate when there exists an inductive invariant (in the form of a given template) that suffices to certify safety of the system.

Theorem 11 *(Weak completeness). Algorithm 2 returns a valid parameter $\bar{\mathbf{a}} \in \mathcal{C}_{\mathbf{a}}$, if (1) the partition granularity is fine enough (i.e., small enough $\eta \in \mathbb{R}^+$), (2) the degrees of multiplier polynomials and SOS polynomials used to form (15) are large enough, and (3) there exists, for the given template $B(\mathbf{a}, \mathbf{x})$, a strictly valid parameter $\hat{\mathbf{a}} \in \mathcal{C}_{\mathbf{a}}$ (i.e., any parameter in some neighborhood of $\hat{\mathbf{a}}$ is valid).*

Proof. When the assumptions (1)–(3) hold, Algorithm 2 will eventually visit a branch wherein any parameter is valid (in case a valid parameter has not been found yet). If the necessary condition in Theorem 6 is used to form the BMI problem (15), Line 11 ensures to return a valid parameter $\bar{\mathbf{a}} \in \mathcal{C}_{\mathbf{a}}$; Otherwise if the BMI problem (15) encodes the sufficient condition in Theorem 5 which strengthens the invariant barrier-certificate condition in Definition 4, a valid parameter $\bar{\mathbf{a}}$ may not induce a non-negative objective value of (15). In this case, however, any parameter sampled and returned by Line 6–7 in the branch is valid, as it contains only valid parameters. □

6.2. Computing an upper-bound $\hat{\lambda}$ by convex relaxation

The bisection operation in Algorithm 2 incurs —in the worst case— an exponential blow-up in the number of branches. In practice, however, one can prune branches inducing only negative objective values, which can be evidenced by a negative upper-bound $\hat{\lambda}$ on the objective value of (15) over the current parameter space (Line 2–5 in Algorithm 2). Such an upper-bound can be computed by *over-approximating the BMI problem* (in contrast to under-approximations pursued by Algorithm 1) via, e.g., convex relaxation [58]. Moreover, the efficiency of Algorithm 2 greatly depends on the tightness of the upper-bound.

In this subsection, we show how to obtain a preferably tight upper-bound (on the objective value) of a BMI program by a classical semidefinite relaxation. Interested readers may refer to [58] for more established results on this topic.

To better illustrate the idea, we stick to the BMI optimization problem of the general form (16). As the non-convexity comes from the quadratic terms $x_i y_i F_{i,j}$, a straightforward convex relaxation is

$$\begin{array}{ll} \underset{\substack{\mathbf{z}=(\mathbf{x},\mathbf{y}),\\ Z=(Z(i,j))_{m\times n}}}{\text{maximize}} & g(\mathbf{z}) \\ \text{subject to} & F+\sum_{i=1}^{m} x_i H_i+\sum_{j=1}^{n} y_j G_j+\sum_{i=1}^{m}\sum_{j=1}^{n} Z(i,j)F_{i,j} \preceq 0\,. \end{array} \tag{27}$$

That is, we replace each quadratic term $x_i y_i$ with a new variable $Z(i,j)$, which constitutes a matrix $Z=(Z(i,j))_{m\times n}$ of fresh variables. The resulting constraint in (27) becomes an LMI that can be solved by SDP.

Notice that the convex program (27) may lead to excessively coarse over-approximations, as the relation $Z(i,j)=x_i y_j$ is completely abstracted away in the relaxation. However, by adding extra convex constraints, one can obtain better over-approximations of the feasible set and thereby tighter upper-bounds (despite the fact that finitely many convex constraints can never precisely capture a non-convex constraint): The classical SDP relaxation replaces the non-convex constraints $Z(i,j)=x_i y_j$, with $i=1,\ldots,m;\ j=1,\ldots,n$ by

$$\begin{pmatrix} 0 & Z \\ Z^{\mathsf{T}} & 0 \end{pmatrix} - \mathbf{z}^{\mathsf{T}}\mathbf{z} \preceq 0\,. \tag{28}$$

Schur complement in Theorem 3 implies that constraint (28) is equivalent to the LMI constraint

$$\begin{pmatrix} 1 & \mathbf{x} & \mathbf{y} \\ \mathbf{x}^{\mathsf{T}} & 0 & Z \\ \mathbf{y}^{\mathsf{T}} & Z^{\mathsf{T}} & 0 \end{pmatrix} \succeq 0\,. \tag{29}$$

By adding the LMI (29) as an additional constraint to (27) and solving the consequent LMI optimization problem, one obtains an upper-bound (on the objective value) of the BMI program (16).

7. Experimental results

We have carried out a prototypical implementation[8] of our synthesis techniques in Wolfram Mathematica, which was selected due to its built-in primitives for SDP, polynomial algebra and matrix operations. Given a safety verification problem as input, our implementation works toward discovering an invariant barrier certificate (in the form of a given template) that witnesses unbounded-time safety of the system. A collection of benchmark examples (detailed in Appendix B) has been evaluated on a 2.10 GHz Xeon processor with 376 GB RAM running 64-bit CentOS Linux 7.

Table 1 reports the empirical results. BMI-DC concerns our locally-convergent Algorithm 1 for solving BMIs (encoding the sufficient condition in Theorem 5) via the eigendecomposition-based DC decomposition (a comparison to other decomposition methods will be presented later). We compare our approach with PENLAB [69] —an off-the-shelf solver in Matlab for directly discharging the same BMI problems (with no guarantee on convergence)— and SOSTOOLS [70] —for solving LMIs derived from Prajna and Jadbabaie's original barrier-certificate condition [9]. The comparison is performed under the same problem configurations.[9] Due to numerical errors caused by floating-point computations and the fact that reaching the local/global optimum does not necessarily yield a valid barrier certificate, we additionally perform a posterior check, via both the quantifier-elimination procedure in Mathematica and the SMT solver Z3 [57], of the synthesized candidate barrier certificate per Definition 4.

Table 1 shows that BMI-DC suffices to synthesize valid barrier certificates in most of the examples within a reasonable number of iterations (i.e., the number of convex sub-problems solved by SDP). This however does not cover all the cases: (1) For the focus example, the solution is close enough to a local optimum (after 100 iterations) but yields still an invalid barrier certificate. This problem can be solved (if there exists an invariant barrier certificate as specified) by enforcing the branch-and-bound framework as presented in Section 6; (2) For examples sys-bio1, sys-bio2, and quadcopter, neither quantifier elimination in Mathematica nor nonlinear reasoning in Z3 can conclude the validity of the synthesized barrier certificates within 15 minutes due to the relatively high system dimensionality (thus marked as **?**; the same applies to PENLAB and SOSTOOLS). The validity for all the other examples is either verified (✓) or refuted (✗) within 10 seconds. The phase portraits of a selected set of examples and the synthesized invariant barrier certificates are depicted in Fig. 4.

[8] Available at https://github.com/Chenms404/BMI-DC.

[9] For PENLAB and SOSTOOLS, we use their optimized, built-in criteria for termination and finding initial solutions.

Table 1
Empirical results on benchmark examples (time in seconds).

Example name	n_{sys}	d_{flow}	d_{BC}	BMI-DC			PENLAB		SOSTOOLS	
				#iter.	time	validity	time	validity	time	validity
overview [10]	2	2	1	2	**0.03**	✓	0.31	✓	0.07	✓
contrived	2	1	2	0	**0.01**	✓	0.48	✓	0.75	✓
lie-der [17]	2	2	1	0	**0.01**	✓	0.22	✓	0.04	✓
lorenz [10]	3	2	2	8	**2.37**	✓	75.11	✗	1.47	✗
lti-stable [59]	2	1	2	0	**0.01**	✓	0.23	✓	0.14	✓
lotka-volterra [60]	3	2	1	3	**0.07**	✓	0.36	✓	0.21	✓
clock [61]	2	3	1	0	**0.01**	✓	0.88	✗	0.18	✗
lyapunov [62]	3	3	2	4	1.25	✓	56.98	✗	**0.35**	✓
arch1 [63]	2	5	2	0	**0.01**	✓	33.76	✗	0.31	✓
arch2 [63]	2	2	2	5	**0.37**	✓	0.38	✗	0.17	✗
arch3 [63]	2	3	2	1	**0.07**	✓	0.54	✓	0.18	✓
arch4 [63]	2	2	1	2	0.09	✓	0.49	✗	**0.06**	✓
barr-cert1 [9]	2	3	2	12	**0.85**	✓	2.53	✗	0.09	✗
barr-cert2 [10]	2	2	2	6	**1.57**	✓	1.16	✗	0.15	✓
barr-cert3 [33]	2	2	1	0	**0.01**	✓	0.20	✓	0.11	✗
barr-cert4 [33]	2	3	2	13	**0.96**	✓	0.89	✗	0.23	✗
fitzhugh-nagumo [64]	2	3	2	2	**0.16**	✓	1.24	✓	0.25	✗
stabilization [65]	3	2	2	9	2.88	✓	55.22	✓	**0.11**	✓
lie-high-order	2	1	2	32	**4.12**	✓	1.56	✗	0.25	✗
raychaudhuri [66]	4	2	2	34	**9.51**	✓	33.64	✗	0.14	✗
focus [67]	2	1	4	100	54.89	✗	0.95	✗	0.48	✗
sys-bio1 [68]	7	2	2	2	73.22	?	101.95	?	1.35	?
sys-bio2 [68]	9	2	1	1	1.03	?	15.54	?	0.16	?
quadcopter [59]	12	1	1	0	0.03	?	65.42	?	0.36	?

n_{sys}: system dimension; d_{flow}: maximal flow-field degree; d_{BC}: degree of the template barrier certificate.
#iter.: number of DCP iterations. 0 means that the initial solution (cf. Section 5.4) is valid.
validity: the synthesized barrier certificate is valid (✓), invalid (✗), or inconclusive within 15 minutes (?, beyond the capability of quantifier elimination in MATHEMATICA and nonlinear reasoning in Z_3).
time: CPU-time, excluding that for casting the BMIs/LMIs. Boldface marks the winner among ✓'s.

Causes of invalid results (✗) by PENLAB and SOSTOOLS Numerical issues are a common (yet minor) cause of invalid results produced by all the tools in Table 1. Whereas the major causes we observed in PENLAB and SOSTOOLS are (1) PENLAB employs non-convex optimization techniques that yield no guarantee on the convergence to local optimums; and (2) SOSTOOLS solves Prajna and Jadbabaie's original, convex barrier-certificate condition [9] which is too conservative to recognize the otherwise valid barrier certificates. In fact, most of the invalid results returned by SOSTOOLS have a rather low "feasibility ratio" (reported by the underlying SDP solver SeDuMi [71]) indicating that SOSTOOLS fails to find barrier certificates adhering to the convex barrier-certificate condition.

Comparison to SOSTOOLS and PENLAB[10] The comparison in Table 1 suggests that (1) *Our invariant barrier-certificate condition recognizes more barrier certificates than the original (more conservative) condition as implemented in* SOSTOOLS. In particular, the lie-high-order example does admit an inductive invariant in the form of the given template, but none of the existing barrier-certificate conditions [15,33,25] —concerning Lie derivatives only up to the first order— recognizes it, since we have $\mathcal{L}^1_f B(\mathbf{x}) = 0$ for some $\mathbf{x}$ on the boundary of B and hence it requires to exploit the second-order Lie derivative[11]; (2) *Our DCP-based synthesis algorithm finds more barrier certificates in less time than directly solving the BMI problems via non-convex optimization techniques as implemented in* PENLAB.

Note that, in our setting, the *volumes* of the invariant sets identified by different approaches are not of primal concern: our goal is to find an invariant that suffices to prove safety of the system instead of a set that "best" over-/under-approximates the reachable set (cf. [72,73]). However, it would be an interesting future step to investigate the connection between, e.g., robustness, and the volumes of the synthesized invariant sets à la [74,75].

We remark that symbolic, monolithic methods based on, e.g., quantifier elimination [17] or nonlinear reasoning in SMT, can hardly deal with any of the examples listed in Table 1 due to the prohibitively high computation complexity. Moreover, it would be desirable to pursue a comparison with the augmented Lagrangian method for solving BMIs as proposed in [25], which unfortunately is not yet possible due to the unavailability of the implementation thereof. We will discuss crucial differences to [25] in Section 8.

[10] We remark that, even though we perform the comparison under the same problem configurations, it is arguably not a fair comparison in terms of the computation time, as the tools are implemented in different platforms (e.g., MATHEMATICA, MATLAB) and rely on different SDP solvers.

[11] In fact, we have $N_{B,f} = 2$ for the lie-high-order example. For all the other examples in Table 1, we either have $N_{B,f} = 1$ or apply the strengthened consecution condition as described in Remark 5 with $\mathfrak{R} = 1 < N_{B,f}$ for efficient synthesis.

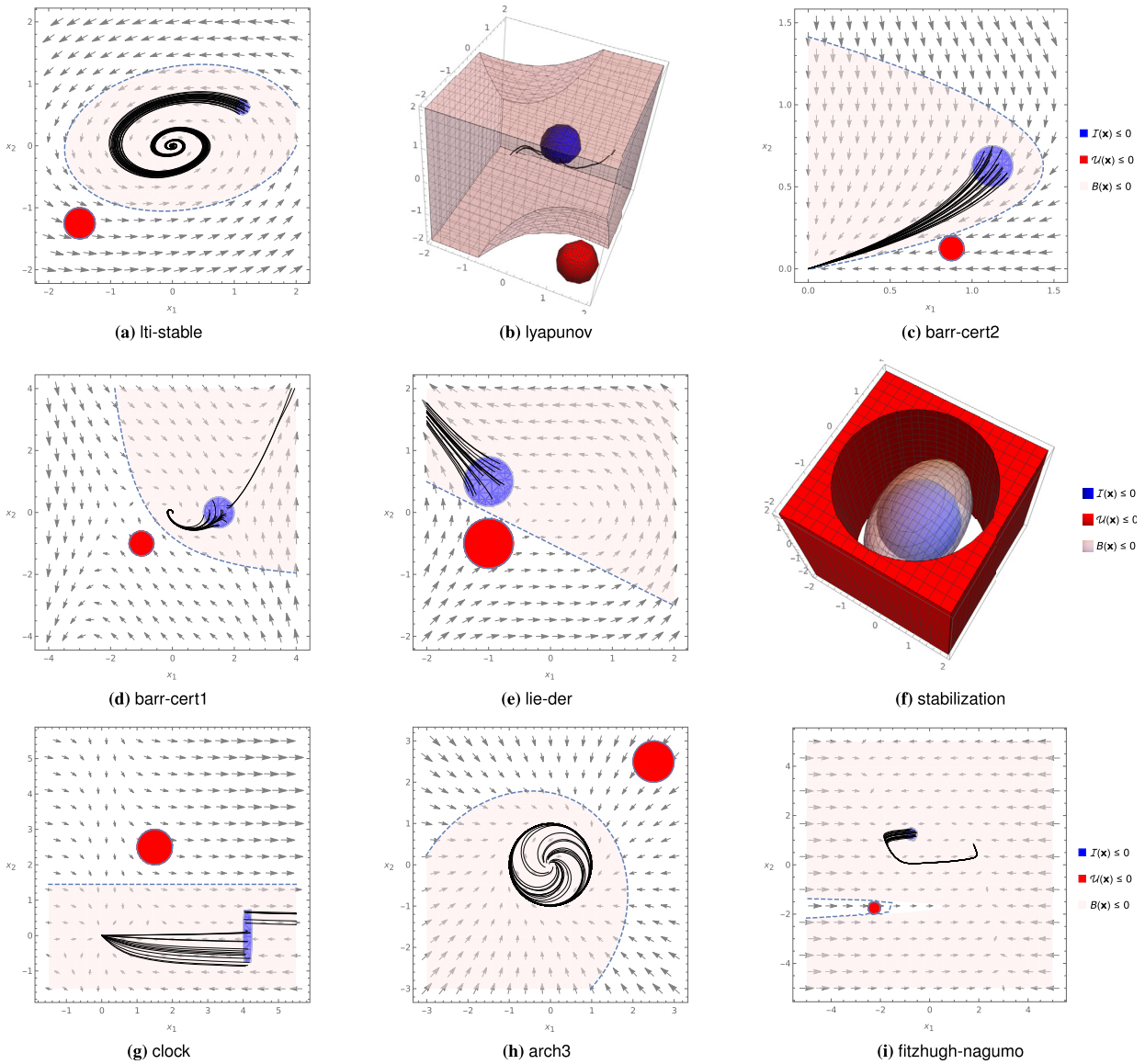
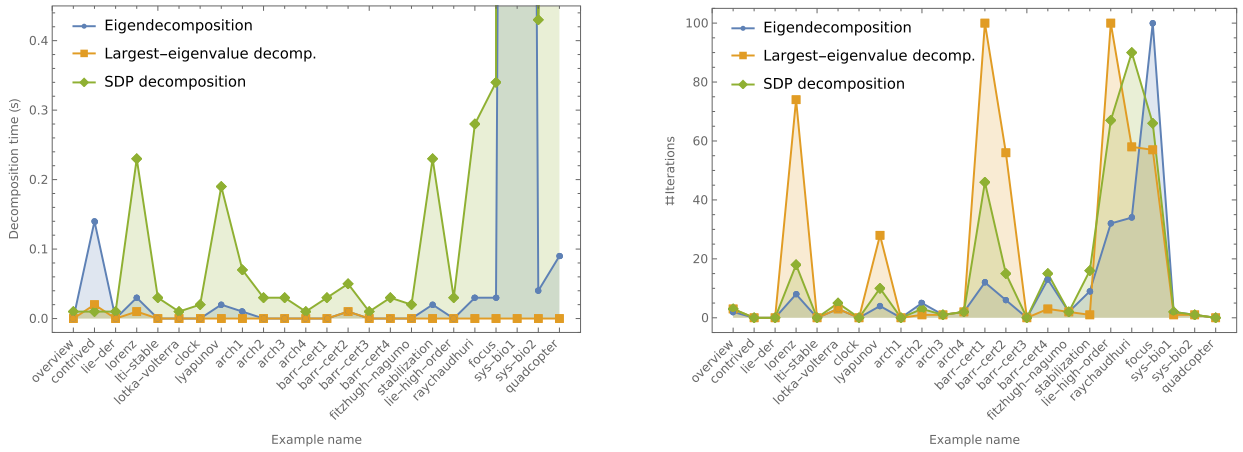


Fig. 4. Phase portraits of a selected set of examples with the synthesized invariant barrier certificates. The arrows indicate the vector field (hidden in 3D-graphics for a clear presentation) and the solid curves are randomly sampled trajectories.

Comparison between different DC decompositions Fig. 5 depicts a comparison of a naive implementation of the three different DC decomposition methods presented in Section 5.1. We observe that, in general, (1) the method based on largest eigenvalues enables faster matrix decompositions, but needs more iterations to achieve the desired precisions and yields valid barrier certificates only for 13 out of 24 benchmark examples; (2) the SDP-based method needs a mild amount of iterations (yielding 14/24 valid barrier certificates), but slows down the matrix decompositions (potentially due to the lack of specific sparsity patterns); and (3) the eigendecomposition-based method leads to less number of iterations (yielding 20/24 valid barrier certificates) within a reasonable amount of decomposition time. In summary, there is no clear winner amongst these DC decomposition methods and the implementation can be improved by carefully exploiting the underlying sparsity patterns of the matrices.

8. Related work

As surveyed in [2], the research community has, over the past three decades, extensively addressed the automatic verification of safety-critical hybrid systems. The almost universal undecidability of the unbounded-time reachability problem [1], however, confines the sound key-press routines to either semi-decision procedures or approximation schemes, most of which address bounded-time verification by, e.g., computing the finite-time image of a set of initial states.



(a) decomposition time for different DC decomposition methods

(b) #iterations for different DC decomposition methods

Fig. 5. Comparison of the three different DC decomposition methods (see Section 5.1) in terms of the decomposition time and the number of DCP iterations induced by the decomposition.

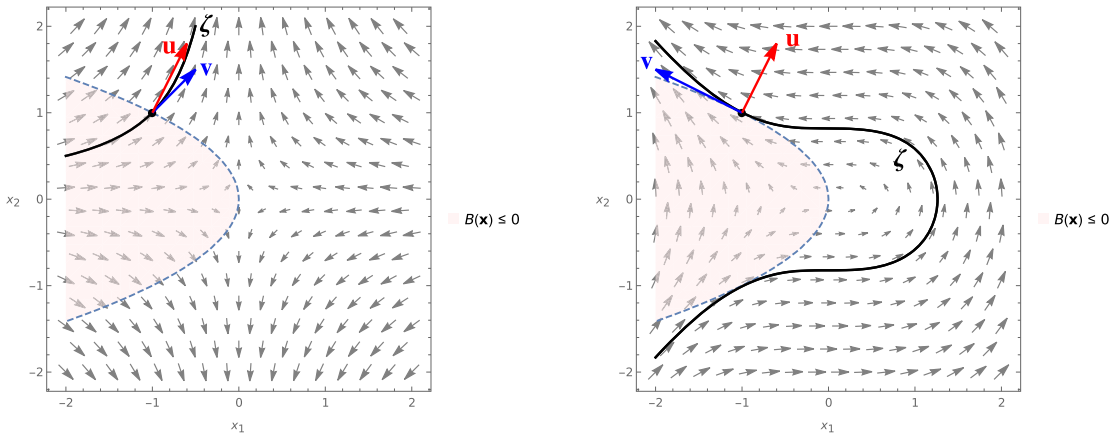
Invariant generation [9,17], amongst others, is a well-established approximation scheme that provides a reliable witness for safety (or equivalently, unreachability) of dynamical systems over an infinite time horizon. Invariants can be constructed in various forms, e.g., barrier certificates [9,12] and differential invariants [30,17]. With a priori specified templates, the invariant synthesis problem can be reduced to numerical optimizations or constraint solving, as in, e.g., [76–79].

Most pertinently, Prajna and Jadbabaie proposed in their seminal work [9] a concept coined *barrier certificate* to encode invariants. To enable efficient synthesis using semidefinite programming, the barrier-certificate condition in [9] strengthens the general condition encoding inductive invariance. Since then, significant efforts have been investigated in developing more relaxed (i.e., weaker) forms of barrier-certificate condition that still admit efficient synthesis, thereby leading to, e.g., exponential-type barrier certificates [14], Darboux-type barrier certificates [16], general barrier certificates [13] and vector barrier certificates [12]. Similar barrier-certificate conditions have been explored to verify systems that address control inputs [80,81], disturbances [47], and stochastic dynamics [82,83]. To attain efficient synthesis, these barrier-certificate conditions share a common property on convexity. That is, if for some $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^m$, $B(\mathbf{a}_1, \mathbf{x})$ and $B(\mathbf{a}_2, \mathbf{x})$ both satisfy the barrier-certificate condition, then for any $0 < \mu < 1$, $B(\mu\mathbf{a}_1 + (1-\mu)\mathbf{a}_2, \mathbf{x})$ must also satisfy the barrier-certificate condition.

However, neither the semantic barrier-certificate condition (9) encoding the general principle of barrier certificates [12,13] nor the inductive invariant condition (8) is convex. This means, when resorting to convex barrier-certificate conditions, one may miss some potential barrier certificates that suffice as inductive invariants witnessing safety. Therefore, non-convex conditions were suggested [15], for which the synthesis problem can be reduced to BMI problems solvable via customized schemes, e.g., the augmented Lagrangian method [25] and the alternating minimization algorithm [33]. Our synthesis techniques also exploit a BMI reduction, with three crucial differences: (1) our invariant barrier-certificate condition is equivalent to the inductive invariant condition in the sense of Theorem 4, and thus is less conservative than all the aforementioned conditions which consider Lie derivatives only up to the first order; (2) our DCP-based techniques for solving BMIs naturally inherit appealing results on convergence and (weak) completeness, which are not (and can hardly be) provided by the approaches in [15,25,33]; (3) our DCP-based iterative procedure visits only feasible solutions to the original BMI problem, and hence whenever a solution that induces a non-negative objective value is found, we can safely terminate the algorithm and claim a feasible solution to the original BMI problem, which may yield a valid barrier certificate. This is not the case for the approaches in [15,25,33].

There are recent efforts in synthesizing barrier certificates via machine learning techniques. Instead of choosing a (polynomial) template and determining the unknown parameters thereof, Zhao et al. [84] proposes to learn a neural network —using generated samples from the target system— as a candidate barrier certificate and do posterior verification via, e.g., SMT or interval analysis. This idea has been further incorporated in a counter-example guided inductive synthesis (CEGIS) framework in [85,86]. Neural networks in these approaches act as implicit template barrier certificates (with a-priori fixed network structures and activation functions whereas the unknown parameters are the weights to be learnt) which can recognize more complex barrier certificates beyond polynomials. Moreover, applying non-convex barrier-certificate conditions in synthesis does not bring extra overheads to these learning-based approaches. On the contrary, these approaches cannot guarantee to find a barrier certificate even if there exists one (recognizable by the neural network). Consequently, when the verification fails, one can only resort to supplying the synthesizer with more samples (or heuristically fine-tuning the network and/or the loss function) but no conclusion about the existence of barrier certificates can be drawn.

Beyond barrier certificates, Wang and Rajamani [36] investigated the feasibility problem of general BMI problems with an application to multi-objective nonlinear observer design. The technique of eigendecomposition was also used therein to



(a) first-order Lie derivative and the gradient

(b) demand for the second-order Lie derivative

Fig. 6. An illustration of how Lie derivatives capture the tendency of trajectories in terms of a polynomial function $B(\mathbf{x})$. ζ: the system trajectory passing through $(-1,1)$; $\mathbf{v}$: the evolution direction per the vector field at $(-1,1)$; $\mathbf{u}$: the gradient of $B(\mathbf{x})$ at $(-1,1)$.

conduct the DC decomposition. The decomposed concave part, however, is simply ignored and no iterative procedure that exhibits convergence to a local optimum can be provided.

The idea of augmenting a locally-convergent algorithm with a branch-and-bound framework to find the global optimum has been exploited in the realm of optimization [87] and control [88]. In contrast, our method is designed for the specific problem of barrier-certificate synthesis, and hence our branch-and-bound algorithm concerns only the parameter space of $\mathbf{a}$, i.e., coefficients of the template barrier certificate.

Finally, we refer interested readers to other approaches to solving BMI problems, e.g., rank minimization [89–91], sequential SDP [92,93], as well as methods committed to general non-convex optimizations, e.g., interior point trust-region [94–96], successive linearization [97] and primal-dual interior point [98].

9. Conclusion

Barrier certificates are a powerful tool to prove time-unbounded safety of hybrid systems. We have presented a new condition on barrier certificates —the invariant barrier-certificate condition, which has been shown as the weakest possible condition on barrier certificates to attain inductive invariance. We showed that our invariant barrier-certificate condition can be reformulated as an optimization problem subject to bilinear matrix inequalities, which can be solved by our locally-convergent algorithm based on difference-of-convex programming. By incorporating this algorithm into a branch-and-bound framework, we obtained a weak completeness result. Experiments on benchmark examples suggested that our invariant barrier-certificate condition recognizes more barrier certificates than existing conditions, and that our DCP-based algorithm is more efficient than directly solving the BMIs via off-the-shelf solvers.

We stress that our techniques for solving BMIs are of a general nature rather than being confined to barrier-certificate synthesis. Interesting future directions include to extend our method to other synthesis problems, e.g., discovering invariants and/or termination proofs of deterministic/probabilistic programs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank Hengjun Zhao for the fruitful discussion on differential dynamics requiring high-order Lie derivatives.

Appendix A. Lie derivatives and the trajectory tendency

Example 2 *(Lie derivatives* [17]*).* Let $B(\mathbf{x}) = x_1 + x_2^2$. Consider the vector field $\boldsymbol{f} = (-x_1, x_2)$ as depicted in Fig. 6a. By Definition 1, we have $\mathcal{L}_{\boldsymbol{f}}^0 B(\mathbf{x}) = x_1 + x_2^2$ and $\mathcal{L}_{\boldsymbol{f}}^1 B(\mathbf{x}) = -x_1 + 2x_2^2$. We exemplify with the point $\mathbf{x} = (-1,1)$ on the parabola $B(\mathbf{x}) = x_1 + x_2^2$ that $\mathcal{L}_{\boldsymbol{f}}^1 B\big|_{(-1,1)} = 3 > 0$ reveals the fact that the system trajectory ζ passing through $(-1,1)$ will escape from the region $B(\mathbf{x}) \leq 0$. In Fig. 6a, the vector $\mathbf{v} = (1,1)$ points to the evolution direction per $\boldsymbol{f} = (-x_1, x_2)$, and the vector

$\mathbf{u} = \frac{\partial}{\partial \mathbf{x}} B\big|_{(-1,1)} = (1, 2)$ denotes the gradient of $B(\mathbf{x})$ at $(-1, 1)$. These two vectors together assert that the trajectory $\boldsymbol{\zeta}$ will enter the region $B(\mathbf{x}) > 0$ immediately after passing through $(-1, 1)$ since the angle formed by $\mathbf{u}$ and $\mathbf{v}$ is less than $\pi/2$, that is, the first-order Lie derivative $\mathcal{L}^1_{\boldsymbol{f}} B\big|_{(-1,1)} = 3$ is positive. Dually, a negative first-order Lie derivative will witness the crossings of a trajectory from the region $B(\mathbf{x}) > 0$ to the region $B(\mathbf{x}) \leq 0$.

However, if the angle between the evolution direction $\mathbf{v}$ and the gradient $\mathbf{u}$ is $\pi/2$ or the gradient is a zero vector, then it is impossible to read off the trajectory tendency via the consequent zero first-order Lie derivative. In this case, we resort to non-zero higher-order Lie derivatives: Consider another vector field $\boldsymbol{f}' = (-2x_2, x_1^2)$ as depicted in Fig. 6b with the same function $B(\mathbf{x})$. We have $\mathcal{L}^0_{\boldsymbol{f}'} B(\mathbf{x}) = x_1 + x_2^2$ and $\mathcal{L}^1_{\boldsymbol{f}'} B(\mathbf{x}) = 2x_1^2 x_2 - 2x_2$, where $\mathcal{L}^1_{\boldsymbol{f}'} B\big|_{(-1,1)} = 0$ as the evolution direction $\mathbf{v}$ is perpendicular to the gradient $\mathbf{u}$. However, since the second-order Lie derivative $\mathcal{L}^2_{\boldsymbol{f}'} B(\mathbf{x}) = 2x_1^4 - 2x_1^2 - 8x_1 x_2^2$ at $(-1, 1)$ is positive, we can conclude that the trajectory passing through $(-1, 1)$ will enter the region $B(\mathbf{x}) > 0$. Notice that, to determine the trajectory tendency, we need to consider Lie derivatives only up to a certain order (as asserted by Theorem 1), e.g., 2 in this example. ◁

Appendix B. Benchmark examples

Example 3 *(contrived).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + x_2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.125)^2 + (x_2 - 0.625)^2 - 0.0125 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 0.875)^2 + (x_2 - 0.125)^2 - 0.0125 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 4 *(lie-der* [17]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -2x_2 \\ x_1^2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 - 0.5)^2 - 0.16 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 + 0.5)^2 - 0.16 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

Example 5 *(lorenz* [10]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 10.0(-x_1 + x_2) \\ -x_2 + x_1(28.0 - x_3) \\ x_1 x_2 - \frac{8}{3} x_3 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1 + 14.5)^2 + (x_2 + 14.5)^2 + (x_3 - 12.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1 + 16.5)^2 + (x_2 + 14.5)^2 + (x_3 - 2.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^3 \mid -20 \leq x_1, x_2, x_3 \leq 20\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 6 *(lti-stable* [59]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -0.1x_1 - 10x_2 \\ 4x_1 - 2x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.125)^2 + (x_2 - 0.625)^2 - 0.125^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1.5)^2 + (x_2 + 1.25)^2 - 0.25^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 7 *(lotka-volterra* [60]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_1(1 - x_3) \\ x_2(1 - 2x_3) \\ x_3(-1 + x_1 + x_2) \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1-1)^2 + (x_2-1)^2 + x_3^2 - 0.64 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + (x_2+1)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^3 \mid -2 \leq x_1, x_2, x_3 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x}) = ax_2$.

Example 8 *(clock* [61]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + 2x_1^2 x_2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (8x_1 - 33)^2 + x_2^2 - 1 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + (x_2 - 2.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -1.5 \leq x_1, x_2 \leq 5.5\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

Example 9 *(lyapunov* [62]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -x_2 \\ -x_3 \\ -x_1 - 2x_2 - x_3 + x_1^3 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1 - 0.25)^2 + (x_2 - 0.25)^2 + (x_3 - 0.25)^2 - 0.25 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^3 \mid (x_1 - 1.5)^2 + (x_2 + 1.5)^2 + (x_3 + 1.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^3 \mid -2 \leq x_1, x_2, x_3 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 10 *(arch1* [63]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + 2x_1^3 x_2^2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + (x_2 - 0.5)^2 - 0.04 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1.5)^2 + (x_2 + 1.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 11 *(arch2* [63]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ 5(x_1 x_2 - 1) \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 0.5)^2 + (x_2 + 0.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1.5)^2 + (x_2 + 1.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 12 *(arch3* [63]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 - x_1^3 + x_2 - x_1 x_2^2 \\ -x_1 + x_2 - x_1^2 x_2 - x_2^3 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 - 0.04 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 2.5)^2 + (x_2 - 2.5)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -3 \leq x_1, x_2 \leq 3\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 13 *(arch4* [63]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -2x_1 + x_1^2 + x_2 \\ x_1 - 2x_2 + x_2^2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 - 0.1^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 0.75)^2 + (x_2 - 0.75)^2 - 0.25^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -0.5 \leq x_1, x_2 \leq 1\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

Example 14 *(barr-cert1* [9]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + \frac{1}{3}x_1^3 - x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 - 0.25 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 + 1)^2 - 0.16 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -4 \leq x_1, x_2 \leq 4\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 15 *(barr-cert2* [10]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + x_1 x_2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.125)^2 + (x_2 - 0.625)^2 - 0.125^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 0.875)^2 + (x_2 - 0.125)^2 - 0.075^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0 \leq x_1, x_2 \leq 1.5\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 16 *(barr-cert3* [33]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + x_1 x_2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 + 1)^2 - 0.25 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + (x_2 - 1)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

Example 17 *(barr-cert4* [33]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + 2x_1^2 x_2 \\ -x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid 9x_1^2 + (2x_2 - 2.25)^2 - 0.75^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 2)^2 + (x_2 - 2)^2 - 0.5^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -1 \leq x_1, x_2 \leq 3\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 18 *(fitzhugh-nagumo* [64]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -1/3x_1^3 + x_1 - x_2 + 0.875 \\ 0.08(x_1 - 0.8x_2 + 0.7) \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 0.75)^2 + (x_2 - 1.25)^2 - 0.25^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 2.25)^2 + (x_2 + 1.75)^2 - 0.25^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -5 \leq x_1, x_2 \leq 5\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 19 *(stabilization* [65]*).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -x_1 + x_2 - x_3 \\ -x_1(x_3 + 1) - x_2 \\ 0.76524x_1 - 4.7037x_3 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 - 1 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^3 \mid -x_1^2 - x_2^2 + 3 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^3 \mid -2 \leq x_1, x_2, x_3 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 20 *(lie-high-order)*. The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.125)^2 + (x_2 - 0.625)^2 - 0.0125 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 0.875)^2 + (x_2 - 0.125)^2 - 0.0125 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid -2 \leq x_1, x_2 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x}) = x_1^2 + a_1 x_2^2 + a_2 x_1 + a_3 x_2 + a_4$.

Example 21 *(raychaudhuri* [66]*)*. The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -0.5x_1^2 - 2(x_2^2 + x_3^2 - x_4^2) \\ -x_1 x_2 - 1 \\ -x_1 x_3 \\ -x_1 x_4 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^4 \mid x_1^2 + (x_2 + 1)^2 - 0.1 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^4 \mid (x_1 + 1)^2 + x_2^2 - 0.1 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^4 \mid -1.5 \leq x_1, \ldots, x_4 \leq 1.5\}$.
- $B(\mathbf{a}, \mathbf{x}) = a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + a_4 x_1 + a_5 x_2 + a_6$.

Example 22 *(focus* [67]*)*. The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 - x_2 \\ x_1 + x_2 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 2.75)^2 + (5x_2 - 10)^2 - 0.25^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 - 2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^2 \mid 1.5 \leq x_1, x_2 \leq 3.5\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 4.

Example 23 *(sys-bio1* [68]*)*. The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} -0.4x_1 + 5x_3 x_4 \\ 0.4x_1 - x_2 \\ x_2 - 5x_3 x_4 \\ 5x_5 x_6 - 5x_3 x_4 \\ -5x_5 x_6 + 5x_3 x_4 \\ 0.5x_7 - 5x_5 x_6 \\ -0.5x_7 + 5x_5 x_6 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^7 \mid \sum_{i=1}^7 (x_i - 1)^2 - 0.01^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^7 \mid \sum_{i=1}^7 (x_i - 1.9)^2 - 0.1^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^7 \mid -2 \leq x_1, \ldots, x_7 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 2.

Example 24 *(sys-bio2 [68]).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} 3x_3 - x_1x_6 \\ x_4 - x_2x_6 \\ x_1x_6 - 3x_3 \\ x_2x_6 - x_4 \\ 3x_3 + 5x_1 - x_5 \\ 5x_5 + 3x_3 + x_4 - x_6(x_1 + x_2 + 2x_8 + 1) \\ 5x_4 + x_2 - 0.5x_7 \\ 5x_7 - 2x_6x_8 + x_9 - 0.2x_8 \\ 2x_6x_8 - x_9 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^9 \mid \sum_{i=1}^{9}(x_i - 1)^2 - 0.01^2 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^9 \mid \sum_{i=1}^{9}(x_i - 1.9)^2 - 0.1^2 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^9 \mid -2 \leq x_1, \ldots, x_9 \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

Example 25 *(quadcopter [59]).* The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -7253.4927x_1 + 1936.3639x_{11} - 1338.7624x_4 + 1333.3333x_8 \\ -1936.3639x_{10} - 7253.4927x_2 - 1338.7624x_5 - 1333.3333x_7 \\ -769.2308x_3 - 770.2301x_6 \\ x_{10} \\ x_{11} \\ x_{12} \\ 9.81x_2 \\ -9.81x_1 \\ -16.3541x_{12} - 15.3846x_9 \end{pmatrix}.$$

- $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^{12} \mid \sum_{i=1}^{12} x_i^2 - 0.01 \leq 0\}$.
- $\mathcal{X}_u = \{\mathbf{x} \in \mathbb{R}^{12} \mid (2x_1 - 0.5)^2 + (2x_2 - 0.5)^2 + (2x_3 - 0.5)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 + (x_6 - 1)^2 + (x_7 - 1)^2 + (x_8 + 1)^2 + (x_9 - 1)^2 + (x_{10} - 1)^2 + (x_{11} + 1)^2 + (x_{12} - 1)^2 - 0.25 \leq 0\}$.
- $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^{12} \mid -2 \leq x_1, \ldots, x_{12} \leq 2\}$.
- $B(\mathbf{a}, \mathbf{x})$ includes all monomials up to degree 1.

References

[1] R. Alur, et al., The algorithmic analysis of hybrid systems, Theor. Comput. Sci. 138 (1) (1995) 3–34.
[2] M. Fränzle, M. Chen, P. Kröger, In memory of Oded Maler: automatic reachability analysis of hybrid-state automata, ACM SIGLOG News 6 (1) (2019) 19–39.
[3] G. Lafferriere, G.J. Pappas, S. Yovine, Symbolic reachability computation for families of linear vector fields, J. Symb. Comput. 32 (3) (2001) 231–253.
[4] H. Anai, V. Weispfenning, Reach set computations using real quantifier elimination, in: HSCC, in: LNCS, vol. 2034, Springer, 2001, pp. 63–76.
[5] T. Gan, M. Chen, L. Dai, B. Xia, N. Zhan, Decidability of the reachability for a family of linear vector fields, in: ATVA, in: LNCS, vol. 9364, Springer, 2015, pp. 482–499.
[6] T. Gan, M. Chen, Y. Li, B. Xia, N. Zhan, Computing reachable sets of linear vector fields revisited, in: ECC, IEEE, 2016, pp. 419–426.
[7] T. Gan, M. Chen, Y. Li, B. Xia, N. Zhan, Reachability analysis for solvable dynamical systems, IEEE Trans. Autom. Control 63 (7) (2018) 2003–2018.
[8] W.D. Smith, Church's thesis meets the N-body problem, Appl. Math. Comput. 178 (1) (2006) 154–183.
[9] S. Prajna, A. Jadbabaie, Safety verification of hybrid systems using barrier certificates, in: HSCC, in: LNCS, vol. 2993, Springer, 2004, pp. 477–492.
[10] A. Djaballah, A. Chapoutot, M. Kieffer, O. Bouissou, Construction of parametric barrier functions for dynamical systems using interval analysis, Automatica 78 (2017) 287–296.
[11] S. Kong, A. Solar-Lezama, S. Gao, Delta-decision procedures for exists-forall problems over the reals, in: CAV, in: LNCS, vol. 10982, Springer, 2018, pp. 219–235.
[12] A. Sogokon, K. Ghorbal, Y.K. Tan, A. Platzer, Vector barrier certificates and comparison systems, in: FM, in: LNCS, vol. 10951, Springer, 2018, pp. 418–437.
[13] L. Dai, T. Gan, B. Xia, N. Zhan, Barrier certificates revisited, J. Symb. Comput. 80 (2017) 62–86.
[14] H. Kong, F. He, X. Song, W.N.N. Hung, M. Gu, Exponential-condition-based barrier certificate generation for safety verification of hybrid systems, in: CAV, in: LNCS, vol. 8044, Springer, 2013, pp. 242–257.
[15] Z. Yang, W. Lin, M. Wu, Exact safety verification of hybrid systems based on bilinear SOS representation, ACM Trans. Embed. Comput. Syst. 14 (1) (2015) 1–19.

[16] X. Zeng, W. Lin, Z. Yang, X. Chen, L. Wang, Darboux-type barrier certificates for safety verification of nonlinear hybrid systems, in: EMSOFT, ACM, 2016, pp. 1–10.
[17] J. Liu, N. Zhan, H. Zhao, Computing semi-algebraic invariants for polynomial dynamical systems, in: EMSOFT, ACM, 2011, pp. 97–106.
[18] J.-B. Lasserre, Moments, Positive Polynomials and Their Applications, vol. 1, World Scientific, 2010.
[19] O. Toker, H. Ozbay, On the NP-Hardness of Solving Bilinear Matrix Inequalities and Simultaneous Stabilization with Static Output Feedback, ACC, vol. 4, IEEE, 1995, pp. 2525–2526.
[20] M. Kocvara, M. Stingl, P. GbR, PENBMI user's guide (version 2.0), in: Software Manual, PENOPT GbRSoftware Manual, PENOPT GbR, in: Hauptstrasse A, vol. 31, 2005, 91338.
[21] R. Orsi, LMIRank: software for rank constrained LMI problems, http://users.cecs.anu.edu.au/~robert/lmirank/, 2005, retrieved: April 9, 2022.
[22] P.D. Tao, E.B. Souad, Algorithms for Solving a Class of Nonconvex Optimization Problems. Methods of Subgradients, North-Holland Mathematics Studies, vol. 129, Elsevier, 1986, pp. 249–271.
[23] H.A. Le Thi, T.P. Dinh, DC programming and DCA: thirty years of developments, Math. Program. 169 (1) (2018) 5–68.
[24] Z. Yang, C. Huang, X. Chen, W. Lin, Z. Liu, A linear programming relaxation based approach for generating barrier certificates of hybrid systems, in: FM, in: LNCS, vol. 9995, Springer, 2016, pp. 721–738.
[25] X. Chen, C. Peng, W. Lin, Z. Yang, Y. Zhang, X. Li, A novel approach for solving the BMI problem in barrier certificates generation, in: CAV, in: LNCS, vol. 12224, Springer, 2020, pp. 582–603.
[26] Q. Wang, M. Chen, B. Xue, N. Zhan, J.-P. Katoen, Synthesizing invariant barrier certificates via difference-of-convex programming, in: CAV (I), in: LNCS, vol. 12759, Springer, 2021, pp. 443–466.
[27] Q.T. Dinh, S. Gumussoy, W. Michiels, M. Diehl, Combining convex–concave decompositions and linearization approaches for solving BMIs, with application to static output feedback, IEEE Trans. Autom. Control 57 (6) (2011) 1377–1390.
[28] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory, SIAM, 1994.
[29] I. Kolář, P.W. Michor, J. Slovák, Natural Operations in Differential Geometry, Springer-Verlag, 1993.
[30] A. Platzer, E.M. Clarke, Computing differential invariants of hybrid systems as fixedpoints, in: CAV, in: LNCS, vol. 5123, Springer, 2008, pp. 176–189.
[31] S. Bak, t-Barrier Certificates: A Continuous Analogy to K-Induction, ADHS, vol. 51, Elsevier, 2018, pp. 145–150.
[32] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[33] Y. Zhang, Z. Yang, W. Lin, H. Zhu, X. Chen, X. Li, Safety verification of nonlinear hybrid systems based on bilinear programming, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11) (2018) 2768–2778.
[34] A. Tarski, A Decision Method for Elementary Algebra and Geometry, University of California Press, Berkeley, 1951.
[35] M.-D. Choi, T.Y. Lam, B. Reznick, Sums of Squares of Real Polynomials, Proceedings of Symposia in Pure Mathematics, vol. 58, American Mathematical Society, 1995, pp. 103–126.
[36] Y. Wang, R. Rajamani, Feasibility analysis of the bilinear matrix inequalities with an application to multi-objective nonlinear observer design, in: CDC, IEEE, 2016, pp. 3252–3257.
[37] A. Shapiro, First and second order analysis of nonlinear semidefinite programs, Math. Program. 77 (1997) 301–320.
[38] V.Y. Pan, Z.Q. Chen, The complexity of the matrix eigenproblem, in: STOC, 1999, pp. 507–516.
[39] L.N. Trefethen, D. Bau III, Numerical Linear Algebra, SIAM, 1997.
[40] A.A. Ahmadi, G. Hall, DC decomposition of nonconvex polynomials with algebraic techniques, Math. Program. 169 (1) (2018) 69–94.
[41] Y. Wang, L. Qi, X. Zhang, A practical method for computing the largest M-eigenvalue of a fourth-order partially symmetric tensor, Numer. Linear Algebra Appl. 16 (7) (2009) 589–601.
[42] R.Y. Zhang, J. Lavaei, Sparse semidefinite programs with near-linear time complexity, in: CDC, IEEE, 2018, pp. 1624–1631.
[43] R.Y. Zhang, J. Lavaei, Efficient algorithm for large-and-sparse LMI feasibility problems, in: CDC, IEEE, 2018, pp. 6868–6875.
[44] S. Zhang, J. Ang, J. Sun, An alternating direction method for solving convex nonlinear semidefinite programming problems, Optimization 62 (4) (2013) 527–543.
[45] M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, U. Topcu, Synthesis in pMDPs: a tale of 1001 parameters, in: ATVA, in: LNCS, vol. 11138, Springer, 2018, pp. 160–176.
[46] D. Figueira, S. Figueira, S. Schmitz, P. Schnoebelen, Ackermannian and primitive-recursive bounds with Dickson's lemma, in: LICS, IEEE, 2011, pp. 269–278.
[47] Q. Wang, Y. Li, B. Xia, N. Zhan, Generating semi-algebraic invariants for non-autonomous polynomial hybrid systems, J. Syst. Sci. Complex. 30 (1) (2017) 234–252.
[48] Y. Li, N. Zhan, M. Chen, H. Lu, G. Wu, J.-P. Katoen, On termination of polynomial programs with equality conditions, CoRR, arXiv:1510.05201 [abs].
[49] A. Nemirovski, Interior point polynomial time methods in convex programming, Lect. Notes 42 (16) (2004) 3215–3224.
[50] B.K. Sriperumbudur, G.R.G. Lanckriet, On the convergence of the concave-convex procedure, in: NIPS, vol. 9, Curran Associates, Inc., 2009, pp. 1759–1767.
[51] H.A. Le Thi, V.N. Huynh, T. Pham Dinh, Convergence analysis of difference-of-convex algorithm with subanalytic data, J. Optim. Theory Appl. 179 (1) (2018) 103–126.
[52] P. Roux, Y.-L. Voronin, S. Sankaranarayanan, Validating numerical semidefinite programming solvers for polynomial invariants, Form. Methods Syst. Des. 53 (2) (2018) 286–312.
[53] T. Gan, B. Xia, B. Xue, N. Zhan, L. Dai, Nonlinear Craig interpolant generation, in: CAV, in: LNCS, vol. 12224, Springer, 2020, pp. 415–438.
[54] G.E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in: Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, Springer Berlin Heidelberg, 1975, pp. 134–183.
[55] C.W. Barrett, R. Sebastiani, S.A. Seshia, C. Tinelli, Satisfiability modulo theories, in: Handbook of Satisfiability, in: FAIA, vol. 185, IOS Press, 2009, pp. 825–885.
[56] D. Henrion, S. Naldi, M.S.E. Din, Exact algorithms for semidefinite programs with degenerate feasible set, J. Symb. Comput. 104 (2021) 942–959.
[57] L.M. de Moura, N. Bjørner, Z3: an efficient SMT solver, in: TACAS, in: LNCS, vol. 4963, Springer, 2008, pp. 337–340.
[58] M. Kheirandishfard, F. Zohrizadeh, R. Madani, Convex relaxation of bilinear matrix inequalities part I: theoretical results, in: CDC, IEEE, 2018, pp. 67–74.
[59] S. Gao, et al., Numerically-robust inductive proof rules for continuous dynamical systems, in: CAV, in: LNCS, vol. 11562, Springer, 2019, pp. 137–154.
[60] E. Goubault, J.-H. Jourdan, S. Putot, S. Sankaranarayanan, Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials, in: ACC, IEEE, 2014, pp. 3571–3578.
[61] S. Ratschan, Z. She, Safety verification of hybrid systems by constraint propagation-based abstraction refinement, ACM Trans. Embed. Comput. Syst. 6 (1) (2007) 8.
[62] S. Ratschan, Z. She, Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions, SIAM J. Control Optim. 48 (7) (2010) 4377–4394.
[63] A. Sogokon, K. Ghorbal, T.T. Johnson, Non-linear continuous systems for safety verification (benchmark proposal), in: ARCH @ CPSWeek, in: EPiC Series in Computing, EasyChair, vol. 43, 2016, pp. 42–51.
[64] M.A.B. Sassi, A. Girard, S. Sankaranarayanan, Iterative computation of polyhedral invariants sets for polynomial dynamical systems, in: CDC, IEEE, 2014, pp. 6348–6353.

[65] M.A.B. Sassi, S. Sankaranarayanan, Stability and stabilization of polynomial dynamical systems using Bernstein polynomials, in: HSCC, ACM, 2015, pp. 291–292.
[66] A. Ferragut, A. Gasull, Seeking Darboux polynomials, Acta Appl. Math. 139 (1) (2015) 167–186.
[67] S. Ratschan, Z. She, Constraints for continuous reachability in the verification of hybrid systems, in: AISC, Springer, 2006, pp. 196–210.
[68] E. Klipp, R. Herwig, A. Kowald, C. Wierling, H. Lehrach, Systems Biology in Practice: Concepts, Implementation and Application, Wiley, 2008.
[69] J. Fiala, M. Kočvara, M. Stingl, PENLAB: a MATLAB solver for nonlinear semidefinite optimization, CoRR, arXiv:1311.5240 [abs].
[70] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P.A. Parrilo, SOSTOOLS version 3.00 sum of squares optimization toolbox for MATLAB, CoRR, arXiv:1310.4716 [abs].
[71] J.F. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optim. Methods Softw. 11 (1–4) (1999) 625–653.
[72] M. Korda, D. Henrion, I. Mezic, Convex computation of extremal invariant measures of nonlinear dynamical systems and Markov processes, J. Nonlinear Sci. 31 (1) (2021) 14.
[73] V. Magron, P. Garoche, D. Henrion, X. Thirioux, Semidefinite approximations of reachable sets for discrete-time polynomial systems, SIAM J. Control Optim. 57 (4) (2019) 2799–2820.
[74] D. Henrion, J.B. Lasserre, C. Savorgnan, Approximate volume and integration for basic semialgebraic sets, SIAM Rev. 51 (4) (2009) 722–743.
[75] F. Dabbene, D. Henrion, C.M. Lagoa, Simple approximations of semialgebraic sets and their applications to control, Automatica 78 (2017) 110–118.
[76] A. Tiwari, Approximate reachability for linear systems, in: HSCC, in: LNCS, vol. 2623, Springer, 2003, pp. 514–525.
[77] S. Sankaranarayanan, H.B. Sipma, Z. Manna, Constructing invariants for hybrid systems, in: HSCC, in: LNCS, vol. 2993, Springer, 2004, pp. 539–554.
[78] S. Gulwani, A. Tiwari, Constraint-based approach for analysis of hybrid systems, in: CAV, Springer, 2008, pp. 190–203.
[79] J. Kapinski, J.V. Deshmukh, S. Sankaranarayanan, N. Arechiga, Simulation-guided Lyapunov analysis for hybrid dynamical systems, in: HSCC, ACM, 2014, pp. 133–142.
[80] X. Xu, P. Tabuada, J.W. Grizzle, A.D. Ames, Robustness of control barrier functions for safety critical control, in: ADHS, in: IFAC-PapersOnLine, vol. 48, Elsevier, 2015, pp. 54–61.
[81] A.D. Ames, X. Xu, J.W. Grizzle, P. Tabuada, Control barrier function based quadratic programs for safety critical systems, IEEE Trans. Autom. Control 62 (8) (2016) 3861–3876.
[82] C. Huang, X. Chen, W. Lin, Z. Yang, X. Li, Probabilistic safety verification of stochastic hybrid systems using barrier certificates, ACM Trans. Embed. Comput. Syst. 16 (5s) (2017) 186:1–186:19.
[83] P. Jagtap, S. Soudjani, M. Zamani, Formal synthesis of stochastic systems via control barrier certificates, IEEE Trans. Autom. Control 66 (7) (2020) 3097–3110.
[84] H. Zhao, X. Zeng, T. Chen, Z. Liu, Synthesizing barrier certificates using neural networks, in: HSCC, ACM, 2020, pp. 25:1–25:11.
[85] A. Peruffo, D. Ahmed, A. Abate, Automated and formal synthesis of neural barrier certificates for dynamical models, in: TACAS (I), in: LNCS, vol. 12651, Springer, 2021, pp. 370–388.
[86] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, A. Peruffo, FOSSIL: a software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks, in: HSCC, ACM, 2021, pp. 24:1–24:11.
[87] K.-C. Goh, M.G. Safonov, G.P. Papavassilopoulos, Global optimization for the biaffine matrix inequality problem, J. Glob. Optim. 7 (4) (1995) 365–380.
[88] H.D. Tuan, P. Apkarian, Y. Nakashima, A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities, Int. J. Robust Nonlinear Control: IFAC-Affil. J. 10 (7) (2000) 561–578.
[89] S. Ibaraki, M. Tomizuka, Rank Minimization Approach for Solving BMI Problems with Random Search, ACC, vol. 3, IEEE, 2001, pp. 1870–1875.
[90] R. Orsi, U. Helmke, J.B. Moore, A Newton-like method for solving rank constrained linear matrix inequalities, Automatica 42 (11) (2006) 1875–1882.
[91] B. Recht, M. Fazel, P.A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, SIAM Rev. 52 (3) (2010) 471–501.
[92] R. Correa, A global algorithm for nonlinear semidefinite programming, SIAM J. Optim. 15 (1) (2004) 303–318.
[93] A. Eggers, N. Ramdani, N.S. Nedialkov, M. Fränzle, Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods, Softw. Syst. Model. (2012) 1–28.
[94] J.E. Dennis, M. Heinkenschloss, L.N. Vicente, Trust-region interior-point SQP algorithms for a class of nonlinear programming problems, SIAM J. Control Optim. 36 (5) (1998) 1750–1794.
[95] F. Leibfritz, E.M.E. Mostafa, An interior point constrained trust region method for a special class of nonlinear semidefinite programming problems, SIAM J. Optim. 12 (4) (2002) 1048–1074.
[96] W.-Y. Chiu, Method of reduction of variables for bilinear matrix inequality problems in system and control designs, IEEE Trans. Syst. Man Cybern. Syst. 47 (7) (2016) 1241–1256.
[97] C. Kanzow, C. Nagel, H. Kato, M. Fukushima, Successive linearization methods for nonlinear semidefinite programs, Comput. Optim. Appl. 31 (3) (2005) 251–273.
[98] H. Yamashita, H. Yabe, Local and superlinear convergence of a primal-dual interior point method for nonlinear semidefinite programming, Math. Program. 132 (1–2) (2012) 1–30.