Assessing Safety for Control Systems Using Sum-of-Squares Programming

Han Wang, Kostas Margellos, Antonis Papachristodoulou

Assessing Safety for Control Systems Using Sum-of-Squares Programming

Han Wang, Kostas Margellos, Antonis Papachristodoulou

Abstract In this chapter we introduce the concept of *safety* for control systems in both continuous and discrete time form. Given a system and a safe set, we say the system is safe if the system state remains inside the safe set for all initial conditions starting from the initial set. Control invariance can be employed to verify safety and design safe controllers. To this end, for general polynomial systems with semi-algebraic safe/initial sets, we show how Sum-of-Squares (SOS) programming can be used to construct invariant sets. For linear systems, evaluating invariance can be much more efficient by using ellipsoidal techniques and dealing with a series of SOS constraints. Following invariance analysis, safe control design and safety verification methods are proposed. We conclude this chapter by showing invariant set construction for both nonlinear and linear systems, and provide MATLAB code for reference.

1 Introduction

The problem of establishing safety of a control system is a topic of significant research interest. Given a system, a predefined safe set and an initial set, safety requires that the system trajectories which start from an initial set stay inside the safe set over a defined time interval. Naturally, verifying safety for a given system and designing a safe control input under saturation constraints are important tasks. As the safe set is defined over a state space, the safety requirement can be formulated as a state constraint in a control verification or design program. Combined with a performance objective and a receding horizon, model predictive control (MPC) [1–4]

Han Wang
University of Oxford, UK e-mail: han.wang@eng.ox.ac.uk

Kostas Margellos
University of Oxford, UK e-mail: kostas.margellos@eng.ox.ac.uk

Antonis Papachristodoulou
University of Oxford, UK e-mail: antonis@eng.ox.ac.uk

is a powerful technique for safe controller design with stability guarantees. However, there are two problems of interest: the first problem is analysis. Safety is a well-defined property like stability. Certifying safety for a given system, an initial set and a safe set through a converse theorem is of interest. The second problem is reducing computational complexity. Finding an online or offline method to design a safe control input with low computational complexity, even for high dimensional nonlinear systems is significant.

For the first problem, reach-avoid games based on optimal control offer a direct interpretation of safety [5–7]. In such a formulation, the cost function is the value function that encodes the safe set. By computing the least worst value function achieved by the control input over a given time interval, we can certify safety. The super (or sub) zero-level set of the value function encodes all the states from which the system is safe over the time interval under consideration. The set consisting of these states is referred to as the viability kernel or invariance kernel of the safe set. Safety is therefore analyzed by the existence of the invariance kernel and an inclusion relationship between the kernel and the initial set. The corresponding optimal control input renders the system safe. Numerical computation for the value function is generally hard, and is obtained as the viscosity solution of a Hamilton-Jacobian-Isaac PDE [8,9]. Interestingly, sets with similar properties as the invariance kernel of the safe set, referred as invariant sets, may not be unique. For some cases, finding an inner or outer approximation of the invariance kernel is sufficient for safety. The construction and approximation of these sets can be quite efficient using numerical methods.

Construction and approximation methods for the invariance kernel can be summarized into three categories: the recursive method, optimisation and using neural networks. Proposed in the pioneering work [10, 11] the recursive method deals with discrete-time systems by backward state propagation from the safe set. Subsequent work ensures that the propagation could terminate with a finite recursion, and calculating the set intersection efficiently [12–14]. The main limitation of this method is that it is only applicable to discrete-time systems. The second method is using convex optimisation techniques to solve an algebraic geometry problem, i.e., finding a set of which the tangent cone at every point contains the system vector field [15]. For polynomial systems with semi-algebraic sets, the algebraic geometry problem can be relaxed and solved efficiently by sum-of-squares programming [16–24]. A SOS program is equivalent to a semi-definite program, which can be solved efficiently with interior-point methods [25]. As it is a convex programming problem, the SOS programming based method is efficiently implementable for invariant set construction. This method is also very scalable for different types of polynomial nonlinear systems. The last method emerged in recent years using deep learning [26–28]. The invariant set is parameterized by a neural network and data of system flows and derivation of invariant function is generated by an outside demonstrator using other control methods. Although more applicable to large scale nonlinear systems, the neural network based method currently lacks theoretical guarantees. Safety is a very critical property which requires rigorous guarantees for most applications.

1.1 Organization

The organization of this chapter is as follows: Section 2 introduces safety in control systems, and reveals the relationship between invariance and safety. Section 3 shows the numerical methods using sum-of-squares programming for constructing invariant sets. Invariance for linear systems is discussed in Section 4. Applications of the proposed sum-of-squares techniques is elaborated in Section 5. Section 6 concludes the chapter.

1.2 Notation

$\mathbb{R}^n$ denotes the space of n-dimensional real numbers. $\mathbb{R}_+$, $\mathbb{N}_+$ denote the space of positive real and integer numbers, respectively. For a set $\mathcal{X}$, $\mathcal{X}'$ is its complement, $\mathrm{cl}(X)$ is its closure.

2 Safety in Control Systems

Consider a nonlinear system

$$x^\dagger = f(x, u), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the n-dimensional state, and $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ is the m-dimensional control input, for any time instance t to be specified in the sequel. Here we use $x^\dagger$ to represent the state transition for both continuous and discrete time system. For continuous-time, we consider

$$\dot{x}(t) = f(x(t), u(t)),$$

and for discrete time

$$x(t+1) = f(x(t), u(t)).$$

For such a system, with a slight abuse of notation, $x(t, u, x_0)$ denotes the state at time $t \geq 0$ (for discrete-time systems we also require t to be an integer), with inputs u starting from $x_0 \in \mathcal{I}$, where $\mathcal{I} \subset \mathbb{R}^n$ is the initial set from which the system starts and is assumed to be non-empty. Here we assume that $x(t, u, x_0)$ is unique for any $t \in \mathcal{L}$, where $\mathcal{L} = [0, +\infty)$ for continuous-time systems and $\mathcal{L} = \mathbb{N}_+$ for discrete time ones. To ease notation, in some occurrences we drop the dependence of u on time.

Safety is a system-set property which models whether a dynamical system can stay within a set, $\mathcal{S} \subset \mathbb{R}^n$. Such a set is usually determined by application requirements, such as collision avoidance in robotic applications. We assume that $\mathcal{S}$ is non-empty and compact, and is the super zero-level set of a differentiable function $s(x) : \mathbb{R}^n \to \mathbb{R}$, i.e.

$$\mathcal{S} := \{x | s(x) \geq 0\}. \tag{2}$$

Only time-invariant safe sets are considered in this chapter. Naturally, we should have $\mathcal{I} \subseteq \mathcal{S}$, such that system starts from the safe set.

Definition 1 Consider system (1) and the safe set $\mathcal{S}$ and initial set $\mathcal{I}$. The system is *safe* if for any $x_0 \in \mathcal{I}$, $t \in \mathcal{T} := [0, T]$, there exists control input with $u(t) \in \mathcal{U}$ such that $x(t, u, x_0) \in \mathcal{S}$. Such a control input is called a *safe control input*.

Given the definition of safety we present here two questions of interest. The first question is to verify safety for a given system, safe and initial sets.

? Question 1: safety verification

Given a dynamical system (1), control set $\mathcal{U}$, initial set I and safe set $\mathcal{S}$, verify whether (1) is safe.

The second question is to design a safe control input.

? Question 2: safe control input design

Given a dynamical system (1), control set $\mathcal{U}$, initial set $\mathcal{I}$ and safe set $\mathcal{S}$, design a safe control input.

These questions involve solving analysis and control synthesis problems under input and state constraints; one typical method is to formulate and solve an optimal control problem as follows: for any (x, t),

$$\begin{aligned} V(x,t) = \max_{u(\cdot)} \min_{\tau \in [t,T]} \; & s(\xi(\tau)) \\ \text{subject to } & \dot{\xi} = f(\xi, u), \\ & u(\tau) \in \mathcal{U}, \quad \text{for all } \tau \in [t, T], \\ & \xi(0) = x. \end{aligned} \tag{3}$$

The cost of the optimal control problem (3) involves choosing the control input $u(\tau)$ that maximizes the minimum value of $s(\xi)$ over the time interval $\mathcal{T}$. This is because a larger $s(\xi)$ implies "safer" since the safe set is defined over the super-level set of $s(\xi)$. $V(x, t) : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ is the value function of (3). Given x, if $V(x, 0) \geq 0$, we conclude that the system is safe starting from this state. It is now clear that the set

$$\mathcal{V} := \{x | V(x, 0) \geq 0\} \tag{4}$$

contains all the initial states x_0 from which (1) is safe. This is because for any $x_0 \in \mathcal{V}$ and $t \in \mathcal{T}$, there exists $u \in \mathcal{U}$ such that $s(x(t, u, x_0)) \geq 0$. The following statement based on $\mathcal{V}$ provides an answer to Questions 1 and 2.

Answer to Questions 1&2

If $\mathcal{I} \subseteq \mathcal{V}$, then system (1) is safe in $\mathcal{S}$. Moreover, the control input $u^*(\cdot)$ obtained from (3) renders the system safe.

If $\mathcal{I} \subseteq \mathcal{V}$, we have that for any $x_0 \in \mathcal{I}, V(x_0, 0) \geq 0$, safety is ensured. Conversely, $\mathcal{V}$ is non-empty if the system is safe, since $\mathcal{I} \subseteq \mathcal{V}$ and $\mathcal{I}$ is non-empty. The optimal control problem (3) is well-posed, but is hard to solve in practice as it is equivalent to solving a partial differential equation [8]. Our goal is to construct or approximate the set $\mathcal{V}$ efficiently with alternative methods. We conclude and prove that, it $\mathcal{V}$ is non-empty, then

Properties of $\mathcal{V}$

1. $\mathcal{V} \subseteq \mathcal{S}$.
2. If $T < \infty$, then for all $t \in [0, T], x_0 \in \mathcal{V}$, there exists $u(t) \in \mathcal{U}$, such that $x(t, u, x_0) \in \mathcal{S}$.
3. If $T \to \infty$ and $\mathcal{V}$ is closed, then for all $t \geq 0, x_0 \in \mathcal{V}$, there exists $u(t) \in \mathcal{U}$, such that $x(t, u, x_0) \in \mathcal{V}$.

The first two properties are straightforward following the aforementioned discussion. We formalize and prove the last property as it is the backbone of the analysis in the sequel.

Lemma 1 *Consider system* (1)*, initial set $\mathcal{I}$, and safe set $\mathcal{S}$, with $T \to \infty$. For any (x, t), let $V(x, t)$ be the optimal value function of* (3)*, and define $\mathcal{V}$ as in* (4)*. If $\mathcal{V}$ is non-empty, then Property 3 holds.*

Proof For any $x_0 \in \mathcal{I}$, there always exists a trajectory starting from $x_0 \in \mathcal{V}$ and for any $t \geq 0$, $x(t, u, x_0) \in \mathcal{S}$. Suppose that for one of the trajectories and for $T \geq 0$, we have $x(T, u, x_0) \in \mathcal{S} \backslash \mathcal{V}$. Since $x(t, u, x_0)$ is unique for any $t \geq 0$, we have that for any $\tau \geq 0$, $x(T + \tau, u, x_0) = x(\tau, u, x(T, u, x_0)) \in \mathcal{S}$. This indicates that $x(T, u, x_0) \in \mathcal{V}$, since the system is safe starting from $x(T, u, x_0)$. For $\tau \to \infty$ and $x(T+\tau, u, x_0) \to \mathcal{V}'$, we must have $x(T+\tau, u, x_0) \in \text{cl}(\mathcal{V}) = \mathcal{V}$ (since $\mathcal{V}$ is closed), thus $x(T, u, x_0) \in \mathcal{V}$. This leads to a contradiction. □

Property 2 is called *invariance* for a set with respect to a system. The existence of such a set reveals the safety property for a given system and safe set.

2.1 Relationship between Invariance and Safety

Definition 2 Consider system (1) and a set $\mathcal{K}$. A set $\mathcal{B} \subset \mathbb{R}^n$ is called a *T-invariance kernel* of $\mathcal{K}$ if for any $t \in [0, T]$, $x_0 \in \mathcal{B}$, there exists $u(t) \in \mathcal{U}$, such that $x(t, u, x_0) \in \mathcal{K}$.

Note that $\mathcal{V}$ is the maximal T-invariance kernel for system (1) and safe set $\mathcal{S}$. This is a corollary of Lemma 1, as one can see that any point exhibiting the invariance property will be within $\mathcal{V}$. As we can see here, if a system is safe, then we can always construct a set $\mathcal{V}$ by solving the optimal control problem (3). The relationship between the existence of a T-invariance kernel $\mathcal{B}$ of $\mathcal{S}$ and safety is shown in the following converse theorem.

Theorem 1 *Consider system* (1)*, initial set $\mathcal{I}$, safe set $\mathcal{S}$ and time interval* $[0, T]$. *If the system is safe, then there exists a T-invariance kernel $\mathcal{B}$ of $\mathcal{S}$, such that $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$. Conversely, if there exists a T-invariance kernel $\mathcal{B}$, such that $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$, then the system is safe. The same arguments hold for $T \to \infty$ if $\mathcal{S}$ is closed.*

Proof We first consider $T < \infty$. For the sufficiency part of the proof suppose the system is safe, and for every $\mathcal{B}$ such that $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$, $\mathcal{B}$ is not a T-invariance kernel of $\mathcal{S}$. Then there exists at least one $x_0 \in \mathcal{B}$ and $t \in \mathcal{T}$, such that $x(t, u, x_0) \notin \mathcal{S}$. Let $\mathcal{B} = \mathcal{I}$, this contradicts to the assumption that the system is initially safe. For the necessity part we have that for such a $\mathcal{B}$, we have that for any $x_0 \in \mathcal{B}$ and $t \in \mathcal{T}$, $x(t, u, x_0) \in \mathcal{B} \subseteq \mathcal{S}$.

We now consider the case where $T \to \infty$. The sufficiency follows the same arguments while the limit points are bypassed by the closeness of $\mathcal{S}$. For the necessity part, consider a set $\mathcal{B}$ constructed by a collection of points $\mathcal{B} := \{x(t, u, x_0) | t \geq 0, x_0 \in \mathcal{I}, x(t, u, x_0) \in \mathcal{S}\}$. Then following the arguments in Lemma 1, we have that $\mathcal{B}$ is an invariance kernel of $\mathcal{S}$, and hence $\mathcal{B} \in \mathcal{S}$. □

With Theorem 1 in hand, the safety verification problem is equivalent to an existence problem. By constructing a T-invariance kernel and the corresponding control input u, we solve the safety problem. Invariance introduced in Definition 2 is related to a safe set, as it requires that trajectories starting from the invariance kernel stay in the safe set. However, there is no safe set appearing in the third property. In fact, with $T \to \infty$, every trajectory starting from $\mathcal{V}$ can stay within $\mathcal{V}$ for all time. We pay specific attention to the case $T \to \infty$ since this commonly holds for general nonlinear systems. The set $\mathcal{V}$ obtained by (3) is also called a *control invariant set* if it is non-empty.

Definition 3 A set $\mathcal{B}$ is called a control invariant set for system (1) if for any $x_0 \in \mathcal{B}$, and $t \geq 0$, there exists $u(t) \in \mathcal{U}$ such that $x(t, u, x_0) \in \mathcal{B}$.

If $\mathcal{B}$ is a control invariant set sucht that $\mathcal{B} \subseteq \mathcal{S}$, then it is for sure a T-invariance kernel of $\mathcal{S}$ by definition. Control invariance is an isolated property for a set $\mathcal{B}$ compared with invariance kernel, which also depends on $\mathcal{S}$. The existence of a control invariant set $\mathcal{B}$ also reveals safety.

Theorem 2 *Consider system* (1)*, initial set* $\mathcal{I}$ *and safe set* $\mathcal{S}$*. If there exists a control invariant set* $\mathcal{B}$ *such that* $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$*, then the system is safe. Conversely, if the system is safe, then there exists a control invariant set* $\mathcal{B}$ *such that* $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$*.*

Proof of Theorem 4 is analogous to that of Theorem 1.

2.2 Control Invariance for Continuous-Time Systems

In this section we introduce and analyze control invariance for continuous-time systems. We first exploit conditions for control invariance with set representation, and give alternative conditions with function representation.

2.2.1 Control Invariance with Set Representation

Control invariance is closely related to the concept of *tangent cone*, which is defined in a point-wise manner over the set.

Definition 4 Let $\mathcal{D}$ be a compact set. The tangent cone of $\mathcal{D}$ at x is the set

$$\wp_{\mathcal{D}}(x) = \left\{ z \in \mathbb{R}^n : \liminf_{h \to 0} \frac{\text{dist}(x + hz, \mathcal{D})}{h} = 0, \right\} \tag{5}$$

where

$$\text{dist}(x, \wp) = \inf_{y \in \wp} ||x - y||, \tag{6}$$

is a distance function and $|| \cdot ||$ denotes the Euclidian norm.

The tangent cone to a set $\mathcal{D}$ at x is shown in Figure 1. We only illustrate the case of $x \in \partial\mathcal{D}$ since $\wp_{\mathcal{D}}(x) = \mathbb{R}^n$ if $x \in \text{Int}(\mathcal{D})$, and $\wp_{\mathcal{D}}(x) = \emptyset$ if $x \in \mathcal{D}'$. In the geometric sense, for a convex set $\mathcal{D}$, every vector $f(x, u) \in \wp_{\mathcal{D}}(x)$ points inside $\mathcal{D}$, or at least is tangent to the boundary curve of $\mathcal{D}$ at x. The tangent cone clearly relates to control invariance.

Theorem 3 (Nagumo's Theorem [15])

Consider a system $\dot{x} = f(x, u)$*. Let* $\mathcal{B}$ *be a compact and convex set. Then the set* $\mathcal{B}$ *is a control invariant set for the system if and only if*

$$\forall x \in \mathcal{B}, \exists u \in \mathcal{U}, \text{ such that } f(x, u) \in \wp_{\mathcal{B}}(x).$$

For continuous system dynamics, $f(x, u)$ is required to point inside the set $\mathcal{B}$ only for $x \in \partial\mathcal{B}$. We note here that in Theorem 3, $\mathcal{B}$ needs to be convex. One can construct a counter example involving a nonconvex set that is not control invariant even if it satisfies the requirement in Theorem 3.

The tangent cone is defined point-wise in x. For special sets such as ellipsoidal control invariant or polyhedral control invariant sets, the tangent cone can be directly

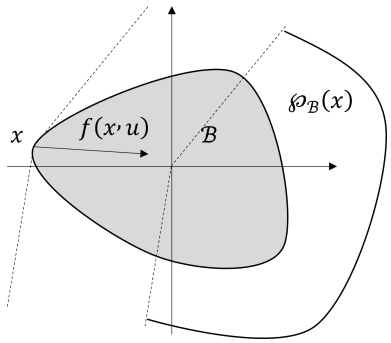

Fig. 1 Tangent cone for a compact set $\mathcal{B}$ at $x \in \partial\mathcal{B}$.

calculated. However, there is no explicit form for general compact sets $\mathcal{B}$. Verifying condition (5) and designing the control input are still challenging tasks.

2.2.2 Control Invariance with Function Representation

In this section we consider the invariance condition by the set representation of a set $\mathcal{B}$. Similarly to the definition of $\mathcal{S}$, suppose $\mathcal{B}$ is the zero super-level set of a function $b(x) : \mathbb{R}^n \to \mathbb{R}$:

$$\mathcal{B} := \{x | b(x) \geq 0\}.$$

Then the control invariance condition is summarized in the following theorem.

Theorem 4 *Consider a system $\dot{x} = f(x,u)$ and a compact set $\mathcal{B}$ described by the zero super-level set of a continuously differentiable function $b(x)$. Then $\mathcal{B}$ is a control invariant set for the system if*

$$\forall x \in \partial\mathcal{B}, \exists u \in \mathcal{U}, \text{ such that } \frac{\partial b(x)}{\partial x} f(x,u) > 0. \tag{7}$$

Theorem 4 certifies control invariance by checking a Lyapunov-like derivative condition for every x on the boundary of the set $\mathcal{B}$. Expanding $b(x)$ at $x(t)$ with respect to t we have

$$b(x(t+\delta t)) = b(x(t)) + \dot{b}(x)|_{x=x(t)}\delta t + o(x(t)),$$

where $o(x(t))$ is a small residual term. If $b(x(t)) = 0$ for $x(t) \in \partial\mathcal{B}$, then $b(x(t+\delta t)) > 0$ if $\dot{b}(x) > 0$. We note here that $\dot{b}(x) = \frac{\partial b(x)}{\partial x} f(x,u)$ is necessary to be strictly positive to bypass the case $o(x(t)) < 0$.

Although condition (7) is only sufficient for control invariance and requires the function $b(x)$ to be continuously differentiable, it is easy to be checked numerically. For a given state $x(t) \in \partial\mathcal{B}$ and function $b(x)$, checking (7) can be done by solving a feasibility optimisation problem

$$\text{find } u \in \mathcal{U}$$
$$\text{subject to } \frac{\partial b(x)}{\partial x} f(x,u) > 0, \forall x \in \partial\mathcal{B}.$$

Checking (7) for any $x \in \partial\mathcal{B}$ is arduous as there are infinite conditions to check. We show how to deal with this using sum-of-squares programming in Section 3.

2.3 Control Invariance for Discrete-Time Systems

The control invariance condition for discrete-time systems is slightly different from that for continuous-time systems. To check the control invariance of a given set $\mathcal{B}$, one needs to check the state transition for every $x \in \mathcal{B}$ but not only $x \in \partial\mathcal{B}$. This is because for a discrete-time system $x(t+1) = f(x(t), u(t))$, there may be the case that $f(\partial\mathcal{B}, u) \in \mathcal{B}$, but $f(\mathcal{B}, u) \notin \mathcal{B}$. The control invariance condition for discrete-time systems is formalized in the following theorem, as a natural counterpart of the Nagumo's Theorem 3.

Theorem 5 *Consider a system $x(t+1) = f(x(t), u(t))$ and a compact set $\mathcal{B}$ defined by the zero super-level set of a function $b(x)$. Then $\mathcal{B}$ is a control invariant set for the system if and only if*

$$\forall x \in \mathcal{B}, \exists u \in \mathcal{U}, \text{such that } b(f(x,u)) \geq 0. \tag{8}$$

2.4 Summary

In this section we revealed the relationship between invariance and safety. Given a safe set $\mathcal{S}$, initial set $\mathcal{I}$ and system dynamics as in (1), safety is equivalent to the existence of a T-invariance kernel $\mathcal{B}$ of the safe set $\mathcal{S}$. In the case where $T \to \infty$, safety is equivalent to control invariance of a set $\mathcal{B}$, where $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$. Control invariance provides an easier way of verifying safety since it certifies a derivative condition for a compact subset of $\mathcal{B}$. On the contrary, T-invariance kernel is built on an inclusion relationship over trajectories, which is hard to be cast in an algebraic form. In the next section we introduce sum-of-squares programming, which is a powerful tool to analyze control invariance under safety, and design a safe control input.

3 Sum-of-Squares Programming

Consider a polynomial optimisation problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \ f(x) \\ &\text{subject to } g(x) \leq 0, \\ &\qquad\quad h(x) = 0, \end{aligned} \tag{9}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$, $g(x) : \mathbb{R}^n \to \mathbb{R}$, $h(x) : \mathbb{R}^n \to \mathbb{R}$ are polynomial functions. For convex $f(x)$, $g(x)$ and $h(x)$, (9) is a convex optimisation problem that can be solved efficiently. If any of these functions is non-convex, we can consider the following problem instead:

$$\begin{aligned} &\max_{\gamma} \ \gamma \\ &\text{subject to } \gamma - f(x) \leq 0, \forall x \in \mathcal{K} := \{x | g(x) \leq 0, h(x) = 0\}. \end{aligned} \tag{10}$$

Now the decision variable is γ. The new constraint $\gamma - f(x) \leq 0$ is a nonpositivity constraint for a polynomial, and scales linearly in γ. However, checking positivity of polynomials of degree higher than 4 is NP-hard. To deal with this issue, sum-of-squares decomposition is proposed.

3.1 Sum-of-Squares Decomposition

Definition 5 A polynomial $f(x)$ is said to be a sum-of-squares polynomial if there exists polynomials $f_i(x)$, such that

$$f(x) = \sum_i f_i(x)^2. \tag{11}$$

We also call (11) a sum-of-squares decomposition for $f(x)$. Clearly, if a function $f(x)$ has a sum-of-squares decomposition, then it is non-negative for all $x \in \mathbb{R}^n$. One question here is whether all positive polynomials admit a sum-of-squares decomposition - the answer is no. As an example, the Motzkin polynomial $1 + x^2y^4 + x^4y^2 - 3x^2y^2$ is nonnegative, but has no sum-of-squares decomposition. In the sequel, we use $\mathbb{R}[x]$ to denote the set of real polynomials in x, and $\Sigma[x]$ to denote the set of sum-of-squares polynomials in x.

Computing the sum-of-squares decomposition (11) can be efficient as it is equivalent to a positive semidefinite feasibility program.

Lemma 2 *Consider a polynomial $f(x)$ of degree $2d$ in $x \in \mathbb{R}^n$. Let $z(x)$ be a vector of all monomials of degree less than or equal to d. Then $f(x)$ admits a sum-of-squares decomposition if and only if*

$$f(x) = z(x)^\top Q z(x), Q \succeq 0. \tag{12}$$

In Lemma 2, $z(x)$ is a user-defined monomial basis if d and n are fixed. In the worst case, $z(x)$ has $\binom{n+d}{d}$ components, and Q is a $\binom{n+d}{d} \times \binom{n+d}{d}$ squared

matrix. The necessity of Lemma 2 is natural from the definition of positive semi-definite matrix, considering the monomial $z(x)$ as a vector of new variables z_i. The sufficiency is shown by factorizing $Q = L^\top L$. Then $z(x)^\top Q z(x) = (Lz(x))^\top Lz(x) = ||Lz(x)||_2^2 \geq 0$.

Given $z(x)$, finding Q to decompose $f(x)$ following (12) is a semi-definite program, which can be solved efficiently using interior point methods. Selecting the basis $z(x)$ depends on the structure of $f(x)$ to be decomposed.

Going back to problem (10), γ should satisfy that the intersected set $\{x|\gamma - f(x) \geq 0\} \bigcap \mathcal{K} \bigcap \{x|\gamma - f(x) = 0\}$ is empty. Here the condition $\{x|\gamma - f(x) > 0\}$ is expressed by $\{x|\gamma - f(x) \geq 0\} \bigcap \{x|\gamma - f(x) = 0\}$. The intersected set has a special structure: it is defined by a series of polynomial equality and inequality constraints.

Definition 6 A set $\mathcal{X} \subset \mathbb{R}^n$ is semi-algebraic if it can be represented using polynomial equality and inequality constraints. If there are only equality constraints, the set is algebraic.

Three types of polynomials are defined based on a series of polynomials $f_1(x), \ldots, f_m(x)$.

Definition 7 The *monoid* generated by $f_1(x), \ldots, f_m(x)$ is denoted by

$$\text{monoid}(f_1(x), \ldots, f_m(x)) = \prod_{i=1}^{m} f_i(x)^{k_i}, k_i \in \mathbb{N},$$

where $\mathbb{N}$ is the set of non-negative integers.

Definition 8 The *ideal* generated by polynomials $f_1(x), \ldots, f_m(x)$ is denoted by

$$\text{ideal}(f_1(x), \ldots, f_m(x)) = \sum_{i=1}^{m} h_i(x) f_i(x),$$

where $h_1(x), \ldots, h_m(x)$ are polynomials in x.

Definition 9 The *cone* generated by polynomials $f_1(x), \ldots, f_m(x)$ is denoted by

$$\text{cone}(f_1(x), \ldots, f_m(x)) = \sum_{i=1}^{r} g_i(x) t_i(x),$$

where $t_i(x), \ldots, t_r(x) \in \text{monoid}(f_1(x), \ldots, f_m(x))$, and $g_1(x), \ldots, g_r(x) \in \Sigma[x]$.

The cone and ideal are closely related to the emptiness of semi-algebraic sets. Specifically, the ideal is related to algebraic sets.

Lemma 3

$$-1 \in t(x) + \text{ideal}(f_1(x), \ldots, f_m(x)) \Leftrightarrow \{x \in \mathbb{R}^n | f_i(x) = 0, \forall i = 1, \ldots, m\} = \emptyset,$$

where $t(x) \in \Sigma[x]$.

The cone is related to the sets defined by polynomial inequality constraints.

Lemma 4

$$-1 \in \text{cone}(f_1(x), \ldots, f_m(x)) \Leftrightarrow \{x \in \mathbb{R}^n | f_i(x) \geq 0, \forall i = 1, \ldots, m\} = \emptyset.$$

Lemmas 3 and 4 are known as *Nullstellensatz*. Based on these results, we have the main result of this section, called the *Positivstellensatz* theorem.

Theorem 6 ([29, Theorem 4.4.2])

Let $\mathcal{K}$ *be a semi-algebraic set,* $\mathcal{K} := \{x \in \mathbb{R}^n | f_i(x) \geq 0, g_j(x) = 0, h_k(x) \neq 0, \ \forall i = 1, \ldots, m, j = 1, \ldots, p, k = 1, \ldots, q\}$. *We have*

$$0 \in \text{cone}(f_1(x), \ldots, f_m(x)) + \text{ideal}(g_1(x), \ldots, g_r(x)) + \text{monoid}(h_1(x), \ldots, h_q(x)) \Leftrightarrow \mathcal{K} = \emptyset. \tag{13}$$

The *Positivstellensatz* gives a necessary and sufficient condition to test whether a semi-algebraic set is empty or not. Testing whether $0 \in \text{cone}(f_1(x), \ldots, f_m(x)) + \text{ideal}(g_1(x), \ldots, g_r(x)) + \text{monoid}(h_1(x), \ldots, h_q(x))$ can be done using sum-of-squares programming. One thing worth mentioning here is the choice of polynomial multipliers $h_1(x), \ldots, h_m(x)$ and sum-of-squares polynomial multipliers $t_1(x), \ldots, t_r(x)$ in the cone and ideal. If there is no sum-of-squares decomposition for the *Positivstellensatz* condition, this does not necessarily imply that $\mathcal{K} = \emptyset$, but can also be due to improperly chosen multipliers. In this case one can increase the degree of the multipliers and repeat the test (which is of non-decreasing accuracy) but this will result in a larger semidefinite programme. The following lemma gives relaxed conditions to test the emptyness of a semi-algebraic set.

Lemma 5 (S-procedure)

Suppose $t(x) \in \Sigma[x]$, *then*

$$f(x) - t(x)g(x) \in \Sigma[x] \Rightarrow f(x) \geq 0, \forall x \in \{x | g(x) \geq 0\}. \tag{14}$$

Suppose $p(x) \in \mathbb{R}[x]$, *then*

$$f(x) - p(x)g(x) \in \Sigma[x] \Rightarrow f(x) \geq 0, \forall x \in \{x | g(x) = 0\}. \tag{15}$$

Compared with *Positivstellensatz*, the S-procedure only gives a sufficient condition for the emptiness of a semi-algebraic set. However, a good feature is that there is no multiplier for $f(x)$. This is especially useful when $f(x)$ is a parameterized function to be constructed. We also highlight here that the S-procedure is sufficient

and necessary for quadratic $f(x)$ and $g(x)$. In this case, $t(x)$ degenerates into a positive scalar.

Using *Positivestellensatz* for linear functions results in Farkas Lemma.

Lemma 6 ([30, Farkas Lemma])

The set $\{x \in \mathbb{R}^n | Ax + b \geq 0, Cx + d = 0\}$ *is empty if and only if there exist* $\lambda \geq 0$ *and* μ *such that*

$$\lambda^\top A + \mu^\top C = 0, \lambda^\top b + \mu^\top d = -1.$$

We have now shown the necessary basic results of sum-of-squares decomposition, and how to use it to characterize the emptiness of semi-algebraic sets. In the next part of this section we will leverage Theorem 6 and Lemma 5 to verify safety, as well as design a safe control input for polynomial systems over semi-algebraic sets.

3.2 Convex Optimisation for Safety

We still separate our discussion into two parts, first on safety for continuous-time systems and then on safety for discrete-time systems. We restrict attention to polynomial control-affine systems and semi-algebraic safe/initial sets.

3.3 Safety for Continuous-Time Systems

Consider a polynomial control affine system

$$\dot{x} = f(x) + g(x)u, \tag{16}$$

where $x(t) \in \mathbb{R}^n$ is the state and $u(t) \in \mathbb{R}^m$ is the control input. Here $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ and $g(x) : \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^m$ are polynomial functions in x. The initial set $\mathcal{I}$ and safe set $\mathcal{S}$ are defined by the zero super-level sets of polynomial functions $h(x)$ and $s(x)$, respectively.

To verify safety and design a safe control input for system (16), a control invariant set $\mathcal{B}$ which satisfies $\mathcal{I} \subseteq \mathcal{B} \subseteq \mathcal{S}$ is useful. Theorem 6 gives sufficient and necessary conditions to construct such a set, as well as the corresponding safe control input u. To do this, we use the function representation result in Section 2.2.2. Suppose $\mathcal{B}$ is the zero super-level set of a polynomial function $b(x)$.

Initial Condition

The initial condition requires $x \in \mathcal{B}$ for any $x \in \mathcal{I}$. Equivalently we have

$$\{x | h(x) \geq 0, b(x) < 0\} = \emptyset,$$

which can be re-written into a standard algebraic set condition

$$\{x| - b(x) \geq 0, b(x) \neq 0, h(x) \geq 0\} = \emptyset.$$

Given that $h(x)$, $b(x)$ are both polynomials, using Positivstellensatz we have

$$\sigma_0(x) - \sigma_1(x)b(x) + b(x)^2 + \sigma_2(x)h(x) = 0,$$

where $\sigma_0(x), \sigma_1(x), \sigma_2(x) \in \Sigma[x]$ are sum-of-squares multipliers. As $b(x)$ is an unknown polynomial to be determined, the bilinear term $\sigma_1(x)b(x)$ and $b(x)^2$ can not be transformed into a matrix semi-denifite problem. One way we follow here is to set $\sigma_1(x) = 1$; then we have

$$b(x) - s_2(x)h(x) = s_0(x) + b(x)^2,$$

which is further relaxed to

$$b(x) - \sigma_2(x)h(x) \in \Sigma[x]. \tag{17}$$

Condition (17) can be cast into a sum-of-squares program with variables $b(x) \in \mathbb{R}[x]$ and $\sigma_2(x) \in \Sigma[x]$. We also remark here that (17) can also be derived from S-procedure, Lemma 5.

Safety Condition

The safety condition requires $\forall x \in \mathcal{B}$, $x \in \mathcal{S}$. Following the safe steps for the initial condition, we have the following sum-of-squares constraint

$$-b(x) + \sigma_1(x)s(x) \in \Sigma[x], \tag{18}$$

where $\sigma_1(x) \in \Sigma[x]$.

Control Invariance Condition

The last condition to be handled is the control invariance condition, given in (7). In function representation it is

$$\forall x \text{ s.t. } b(x) = 0, \exists u(x), \text{such that } \frac{\partial b(x)}{\partial x}(f(x) + g(x)u(x)) > 0,$$

where $u(x) \in \mathbb{R}^m[x]$ is a polynomial control input. In standard algebraic set form we have

$$\{x|b(x) = 0, -\frac{\partial b(x)}{\partial x}(f(x) + g(x)u(x)) \geq 0\} = \emptyset.$$

Using Positivstellensatz we have

$$t(x)b(x) + \sigma_4(x)\left(-\frac{\partial b(x)}{\partial x}(f(x) + g(x)u(x))\right) + \sigma_3(x) = 0,$$

where $t(x) \in \mathbb{R}[x], \sigma_3(x), \sigma_4(x) \in \Sigma[x], \sigma_3(x)$ is a predefined small term. Following the same simplification steps as above, by letting $\sigma_4(x) = 1$ we have

$$\frac{\partial b(x)}{\partial x}(f(x) + g(x)u(x)) - t(x)b(x) + \sigma_3(x) \in \Sigma[x]. \tag{19}$$

Here $\frac{\partial b(x)}{\partial x}$ is a polynomial in x. Unlike (17) and (18), (19) has bilinearity in $\frac{\partial b(x)}{\partial x}g(x)u(x)$ and $t(x)b(x)$ which cannot be removed easily. Summarizing (17), (18), (19), we have the following sum-of-squares program.

$$\begin{aligned} &\text{find } b(x), u(x), \sigma_1(x), \sigma_2(x), t(x) \\ &\text{subject to (17), (18), (19)} \\ &\quad b(x), u(x), t(x) \in \mathbb{R}[x], \sigma_1(x), \sigma_2(x) \in \Sigma[x]. \end{aligned} \tag{20}$$

As noted already, the sum-of-squares program (20) can not be directly transformed into a semi-definite program due to the bilinearity in (19). A typical method to bypass this difficulty is to solve it iteratively by fixing part of the variables.

Alternating Direction Methodology

Let $b^k(x)$, $u^k(x)$ and $t^k(x)$ be the solutions at the k-th iteration for $b(x), u(x), t(x)$. Utilizing the methodology of alternating direction algorithm, the sum-of-squares program (20) at the k-th iteration is divided into three programs: updating $b(x)$, updating $u(x)$ and updating $t(x)$. In each program, the other two variables are fixed polynomials.

Updating $b(x)$ at the k-th iteration with fixed $u(x) = u^{k-1}(x)$, and $t(x) = t^{k-1}(x)$ we have the following feasibility optimisation problem

$$\begin{aligned} &\text{find } b(x) \\ &\text{subject to (17), (18)} \\ &\quad \frac{\partial b(x)}{\partial x}(f(x) + g(x)u^{k-1}(x)) - t^{k-1}(x)b(x) + \sigma_3(x) \in \Sigma[x]. \end{aligned}$$

Updating $u(x)$ at the k-th iteration with fixed $b(x) = b^k(x)$, $t(x) = t^{k-1}(x)$ we have the following feasibility optimisation problem

$$\begin{aligned} &\text{find } u(x) \\ &\text{subject to } \frac{\partial b^k(x)}{\partial x}(f(x) + g(x)u(x)) - t^{k-1}(x)b^k(x) + \sigma_3(x) \in \Sigma[x]. \end{aligned}$$

Updating $t(x)$ at the k-th iteration with fixed $b(x) = b^k(x), u(x) = u^k(x)$ we have

$$\begin{aligned} &\text{find } t(x) \\ &\text{subject to } \frac{\partial b^k(x)}{\partial x}(f(x) + g(x)u^k(x)) - t(x)b^k(x) + \sigma_3(x) \in \Sigma[x]. \end{aligned}$$

Solving the above three programs iteratively until convergence, we complete the safety synthesis. All the three programs are standard sum-of-squares programs, i.e., convex optimisation problems. There are several points we want to highlight here:

- The iterative algorithm is not guaranteed to converge to a feasible solution.
- Selection of monomial parameterization basis for $b(x)$, $u(x)$ and $t(x)$ influences the result. If the iterative algorithm does not converge, one can chose other polynomial basis, especially increase the degree of the monomial basis.

3.4 Safety for Discrete-Time Systems

From the discussion in Section 2.3, safe control invariant set synthesis for discrete-time system goes beyond just checking a condition at the boundary of the safe set. Consider a discrete-time control affine system

$$x(t+1) = f(x(t)) + g(x(t))u(t), \tag{21}$$

where $x(t) \in \mathbb{R}^n$ and $u \in \mathcal{U}$. The initial and safe sets are constructed in the same way as those in the continuous-time case. Using the result of Theorem 5 yields the following SOS program:

$$\begin{aligned} &\text{find } b(x), u(x), \sigma_1(x), \sigma_2(x), \sigma_3(x) \\ &\text{subject to (17), (18)}, \sigma_1(x), \sigma_2(x), \sigma_3(x) \in \Sigma[x] \\ &\qquad b(f(x) + g(x)u(x)) - \sigma_3(x)b(x) + \sigma_4(x) \in \Sigma[x]. \end{aligned} \tag{22}$$

Beyond the bilinearity experienced in the continuous-time program case (20), unavoidable non-convexity occurs in the term $b(f(x)+g(x)u(x))$, since $b(x)$ and $u(x)$ are both parameterized functions. The iterative algorithm cannot be used for this type of problems. To make computation efficient, a special form of control invariant sets, polyhedral sets, will be considered:

$$b(x) = px + q, \tag{23}$$

where $p \in \mathbb{R}^m \times \mathbb{R}^n$, $q \in \mathbb{R}^m$ and $b(x) : \mathbb{R}^n \to \mathbb{R}^m$ is a vector function. Using the polyhedral invariant set, the SOS program is

$$\begin{aligned} &\text{find } b(x), u(x), \sigma_1(x), \sigma_2(x), \sigma_3(x) \\ &\text{subject to } -px - q + \sigma_1(x)h(x) \in \Sigma^m[x], \\ &\qquad px + q - \sigma_2(x)s(x) \in \Sigma^m[x], \\ &\qquad pf(x) + pg(x)u(x) + q - \sigma_3(x)b(x) + \sigma_4(x) \in \Sigma^m[x], \\ &\qquad \sigma_1(x), \sigma_2(x) \in \Sigma[x], \sigma_3(x) \in \Sigma^m[x]. \end{aligned} \tag{24}$$

Here $\Sigma^m[x]$ is the set of m-dimensional SOS polynomial vectors. The alternative directional algorithm is now applicable to (24).

3.5 Summary

In this section we introduced sum-of-squares programming and its application to safety assessment. For polynomial systems with semi-algebraic initial and safe sets, invariance conditions proposed in Section 2 can be encoded as algebraic constraints in convex programs, and solved iteratively.

4 Safety for Linear Systems with Constrained Inputs

In this section we pay special interest to continuous-time linear systems under unit peak input. The system dynamics are given by

$$\dot{x} = Ax + Bu, \tag{25}$$

where $x(t) \in \mathbb{R}^n$ denotes the state, and $u(x) \in \mathbb{R}^m$ the control input, while the matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. Here we define u as feedback control input of state x. Define the compact semi-algebraic set $\mathcal{S} := \{x|s(x) \geq 0\}$ as the safe set. Throughout the paper, we assume that the system can be safely controlled within $\mathcal{S}$. For the sake of brevity, we assume the system is fully observable without state and output noise.

4.1 Unit Peak Input

Unit peak input implies a constraint on the Euclidean norm. In the meaning of 2-norm, the control input u is bounded by $||u||_2^2 \leq u_{\max}$, where $u_{\max} > 0$ is a positive scalar. The system is assumed to be locally stabilizable around $x^* \in \mathcal{S}$. Among different types of invariance sets, we select the ellipsoidal invariance set as a candidate. Our choice stems from the fact that, if the system (25) is stabilizable (or locally stabilizable) with state feedback control law $u = kx$, then there exists a quadratic Lyapunov function $V(x) = x^\top Px$ with $P \succ 0$ [31, Theorem 4.17]. Then every sub-level set of $V(x)$ serves as an invariant set. Under this set-up, we use

$$b(x) = -x^\top Px + l, \tag{26}$$

where P and l are parameters to be determined for a candidate function for the safe invariance set.

According to Theorem 2, $b(x)$ satisfies:

$$\exists u \in \{u | ||u||_2^2 \leq u_{\max}\},\ \frac{\partial b(x)}{\partial x}(Ax + Bu) \geq 0, \forall\, x \text{ such that } b(x) = 0, \tag{27a}$$

$$b(x) < 0, \text{for any } x \text{ such that } s(x) < 0. \tag{27b}$$

To guarantee condition (27a) holds, we require that

$$\sup_{||u||_2^2 \leq u_{\max}} \frac{\partial b(x)}{\partial x}(Ax + Bu) \geq 0, \text{for any } x \text{ such that } b(x) = 0. \tag{28}$$

By (26), we have $\frac{\partial b(x)}{\partial x} = 2Px$. The maximizing u satisfies

$$\left\langle B^\top \frac{\partial b(x)}{\partial x}^\top, u \right\rangle = \left\| \frac{\partial b(x)}{\partial x} B \right\| \cdot ||u||, \tag{29}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operator. We directly have $u = -\zeta B^\top P^\top x$, where ζ is a positive scaled scalar. Without control limitation, ζ should be large enough to maximize $\frac{\partial b(x)}{\partial x} Bu$. Under the assumption that u is bounded, then ζ is a function of both state x and matrix P. Condition (27a) turns out to be

$$x^\top(-PA + \zeta PBB^\top P^\top)x \geq 0, \tag{30}$$

which is equivalent to

$$-PA - A^\top P + 2\zeta PBB^\top P^\top \geq 0. \tag{31}$$

We omit the boundary condition $b(x) = 0$ since $\mathcal{B}$ is convex and compact [32]. In fact, for a point y such that $b(y) \neq 0$, we can always find $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$, such that $x = ty$ and $b(x) = 0$. We directly have that $x^\top(-PA + \zeta PBB^\top P^\top)x \geq 0 \Leftrightarrow y^\top(-PA + \zeta PBB^\top P^\top)y \geq 0$.

To overcome the bilinear term $\zeta PBB^\top P^\top$ in (31), we propose the following conditions:

$$-PA - A^\top P + 2\zeta \hat{P} \geq 0, \tag{32a}$$

$$\begin{bmatrix} \hat{P} & PB \\ B^\top P^\top & I \end{bmatrix} \geq 0. \tag{32b}$$

Theorem 7 states that (32) is sufficient for (31).

Theorem 7 *If there exist $\zeta > 0$, $P \geq 0$ and $\hat{P}$ satisfying* (32) *then P and ζ satisfy* (31)*.*

$$\max_{P, \hat{P}, l, \sigma} \ l \tag{33a}$$

$$\text{subject to } x^\top P x - l + \sigma s(x) \in \Sigma[x], \tag{33b}$$

$$\text{satisfying (32)} \tag{33c}$$

$$\max_{\zeta' \geq 0, \sigma} \ \zeta' \tag{34a}$$

$$\text{subject to } -\zeta' x^\top P B B^\top P^\top x - \sigma(-x^\top P x + l) + u_{\max} \in \Sigma[x], \tag{34b}$$

Here we use $\Sigma[x]$ to represent the set of sum-of-squares polynomials in x, and $\sigma \in \Sigma[x]$ to be a sum-of-squares multiplier. The programs (33) and (34) are computed iteratively following the alternating directional algorithm. Solving (33) requires to fix ζ which is initialized to be a small constant. We remark that in (34) we use a new nonnegative decision variable ζ' instead of ζ^2 to obtain a linear program.

An alternative methods to overcome the bilinearity in (31) is to multiply P^{-1} on both sides, which yields

$$\begin{aligned} & P^{-1}\{-PA - A^\top P + 2\zeta P B B^\top P^\top\} P^{-1} \geq 0 \\ \Longleftrightarrow & -AQ - QA^\top + 2\zeta B B^\top \geq 0, \end{aligned} \tag{35}$$

where $Q = P^{-1}$. This new condition is a linear matrix inequality in Q and ζ. Here the variables substitution leads to a necessary and sufficient condition since P is full rank.

4.2 Summary

In this section we use ellipsoidal techniques to construct a control invariant set for a linear system with quadratic initial and safe sets, under unit peak input. Compared with general polynomial systems, the computation for linear systems is more efficient since the control invariant set is assumed to be an ellipsoid.

5 Applications

In this section we show how to use the SOS parser SOSTOOLS [33] together with the SDP solver SeDuMi [34] to solve the proposed SOS programs for safe control invariant sets and controllers. Code in MATLAB is provided for reference. Two types of systems are given for example, including a general polynomial control affine system (solving (20)) and a linear system (solving (33), (34)). Both of them are continuous-time systems.

5.1 Nonlinear Control Affine System

Consider a system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}x_2\\ x_1+\frac{1}{3}x_1^3+x_2\end{bmatrix} + \begin{bmatrix}x_1^2+x_2+1\\ x_2^2+x_1+1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}, \tag{36}$$

with unit peak input $u_1 \in [-1.5, 1.5]$, $u_2 \in [-1.5, 1.5]$. The safe set is defined by a disc $S = \{x|x_1^2 + x_2^2 - 3 \leq 0\}$, and initial set defined by $I = \{x|(x_1 - 0.4)^2 + (x_2 - 0.4)^2 - 0.16 \leq 0\}$. $b(x)$ is parameterized by a forth-order polynomial, $u(x)$ is parameterized by a quadratic function. The synthesized control invariant set $\mathcal{B}$ as a subset of $\mathcal{S}$, while enclosing $\mathcal{I}$ as shown in Figure 2.

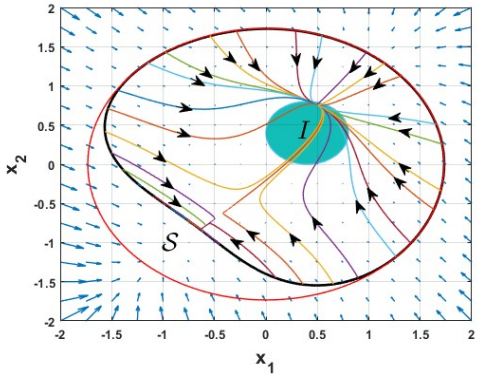

Fig. 2 Control invariant set (inside the black curve) and safe control input. Vector field on $\partial\mathcal{B}$ points inside the set.

5.2 Linear System

Consider a linear system with $A = \begin{bmatrix}0.8 & 0.7\\ -0.4 & -0.6\end{bmatrix}$, $B = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$, and $-1 \leq u \leq 1$. The safe set is the interior of a disc: $\mathcal{S} := \{x|-x_1^2-x_2^2+1 \geq 0\}$. Without a control input, the system's dynamics are an unstable spiral; establishing safety starting from a closed safe set is therefore challenging. Figure 3 and 4 show the synthesized invariant set and value of control input. The control input $u(x)$ satisfies $||u(x)||_2^2 \leq 1$ for $x \in \mathcal{B}$. The sum-of-squares programs (33) and (34) are solved using SOSTOOLS [33] with SeDuMi v1.3.5 [34] as the SDP solver.

By assuming $b(x)$ is an ellipsoidal and $u(x)$ is a linear input, the computation of the control invariant set for linear systems can be fairly efficient.

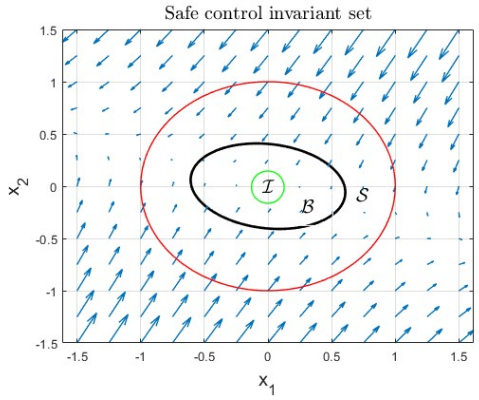

Fig. 3 Ellipsoidal control invariant set for an unstable linear system

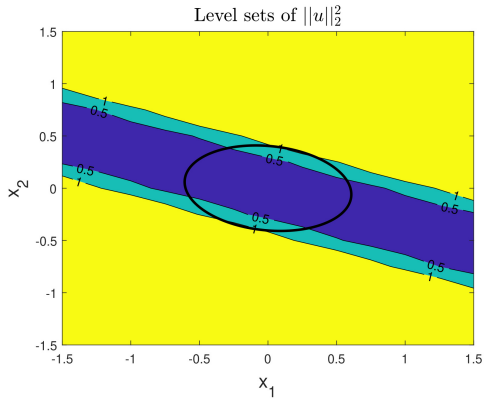

Fig. 4 Level sets of $||u||_2^2$, $u = -\zeta B^\top P^\top x$

6 Conclusion

In this chapter we demonstrate how to assess safety for control systems. The relationship between safety and invariance theory for both continuous and discrete-time systems is illustrated. For polynomial systems with semi-algebraic initial sets and safe sets, safety can be encoded into algebraic conditions, and synthesized by a sum-of-squares program. For linear systems with quadratic initial/safe sets, ellipsoidal techniques can be applied for efficient computation. Numerical simulation of the introduced methodology is performed over a polynomial system and a linear system. A sample code in MATLAB is included.

Appendix

Program Code with SOSTOOLS

```
options.solver='sedumi'; % you need to add SeDuMi into work path

% Constructing the vector field
pvar x1 x2;
x = [x1;x2];

f = [          x2;
    +x1+x1^3/3+x2];

gx = [x1^2+x2+1;
        x2^2+x1+1];

limits = 1.5;

% Define the safe set S >= 0 is safe
bias = 0;
S = -(x1-bias)^2-(x2-bias)^2+3; % Avoid a round region

% Define the initial set I >= 0
I = -(x1-bias-0.4)^2-(x2-bias-0.4)^2+0.16; % Start from a round region

% Initialize the SOSP
prog = sosprogram(x);

% The barrier function
[prog,B] = sospolyvar(prog,monomials(x,0:4),'wscoeff');

% The multipliers
[prog,sigma1] = sossosvar(prog,monomials(x,0:1));
[prog,sigma2] = sossosvar(prog,monomials(x,0:1));
[prog,sigma3] = sossosvar(prog,monomials(x,0:1));

% ============================
% Inequality constraints
% Constraint 1: B(x) < 0 when S < 0
prog = sosineq(prog,-B+sigma1*S-0.001); % 0.001 is used for strict inequality

% Constraint 2: B(x) >= 0 when I >= 0
prog = sosineq(prog,B-sigma2*I);

% ============================
% Iterative procedure below, we interatively solve the problem, first find
% a feasible control input
% Initialize the control input
SolU1 = 0;
SolU2 = 0;
```

```
% Initialize the barrier function
g = [gx(1)*SolU1;
       gx(2)*SolU2];

prog_barrier = prog;
expr = diff(B,x1)*(f(1,1)+g(1,1))+diff(B,x2)*(f(2,1)+g(2,1));
prog_barrier = sosineq(prog_barrier,expr);

% Solve the problem
prog_barrier = sossolve(prog_barrier,options);

% Get the resulting invariant set
SolB = sosgetsol(prog_barrier,B);

clear prog_barrier

% Initialize the polynomial multiplier
SolLambda1 = 0;
SolLambda2 = 0;
SolLambda3 = 0;
SolLambda4 = 0;
SolLambda5 = 0;

for i=1:20
    % ============================
    % First fix the multiplier, control input, solve for the invariant set
    % ============================
    % Constraint 1: dB/dx*f(x)+alpha B > 0 when B = 0
    g = [gx(1)*SolU1;
           gx(2)*SolU2];

    prog_barrier = prog;
    expr = diff(B,x1)*(f(1,1)+g(1,1))+diff(B,x2)*(f(2,1)+g(2,1))+SolLambda1*B;
    prog_barrier = sosineq(prog_barrier,expr);

    prog_barrier = sosineq(prog_barrier,-SolU1+limits-SolLambda2*B);
    prog_barrier = sosineq(prog_barrier,SolU1+limits-SolLambda3*B);
    prog_barrier = sosineq(prog_barrier,-SolU2+limits-SolLambda4*B);
    prog_barrier = sosineq(prog_barrier,SolU2+limits-SolLambda5*B);

    % Enlarge the area
    expr = B-sigma3*SolB;
    prog_barrier = sosineq(prog_barrier,expr);

    % Solve the problem
    prog_barrier = sossolve(prog_barrier,options);

    % Get the resulting invariant set
    SolB = sosgetsol(prog_barrier,B);

    % Delete the temporary variable
    clear prog_barrier;

    % ============================
```

```
    % Then fix the invariant set, solve for the control input
    prog_u = sosprogram(x);
    [prog_u,u1] = sospolyvar(prog_u,monomials(x,0:3));
    [prog_u,u2] = sospolyvar(prog_u,monomials(x,0:3));
    g = [gx(1)*u1;
           gx(2)*u2];
    expr = diff(SolB,x1)*(f(1,1)+g(1,1))+diff(SolB,x2)*(f(2,1)+g(2,1))+SolLambda1*SolB;
    prog_u = sosineq(prog_u,expr);

    %  Control limits
    [prog_u,lambda2] = sospolyvar(prog_u,monomials(x,0:2));
    [prog_u,lambda3] = sospolyvar(prog_u,monomials(x,0:2));
    [prog_u,lambda4] = sospolyvar(prog_u,monomials(x,0:2));
    [prog_u,lambda5] = sospolyvar(prog_u,monomials(x,0:2));
    prog_u = sosineq(prog_u,-u1+limits-lambda2*SolB);
    prog_u = sosineq(prog_u,u1+limits-lambda3*SolB);
    prog_u = sosineq(prog_u,-u2+limits-lambda4*SolB);
    prog_u = sosineq(prog_u,u2+limits-lambda5*SolB);

    prog_u = sossolve(prog_u,options);

    % Get the control input
    SolU1 = sosgetsol(prog_u,u1);
    SolU2 = sosgetsol(prog_u,u2);

    % Get the multipliers
    SolLambda2 = sosgetsol(prog_u,lambda2);
    SolLambda3 = sosgetsol(prog_u,lambda3);
    SolLambda4 = sosgetsol(prog_u,lambda4);
    SolLambda5 = sosgetsol(prog_u,lambda5);

    % Delete the temporary variable
    clear prog_u;



    % ============================
    % Then fix the invariant set, solve for the multipler lambda
    prog_lambda = sosprogram(x);
    [prog_lambda,lambda1] = sospolyvar(prog_lambda,monomials(x,0:2));
    g = [gx(1)*SolU1;
           gx(2)*SolU2];
    expr = diff(SolB,x1)*(f(1,1)+g(1,1))+diff(SolB,x2)*(f(2,1)+g(2,1))+lambda1*SolB;
    prog_lambda = sosineq(prog_lambda,expr);

    % Solve the problem
    prog_lambda = sossolve(prog_lambda,options);

    % Get the resulting polynomial multipler
    SolLambda1 = sosgetsol(prog_lambda,lambda1);

    % Delete the temproray variable
    clear prog_lambda;
end
```

References

1. Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business media, 2013.
2. Basil Kouvaritakis and Mark Cannon. *Model predictive control*, volume 38. Switzerland: Springer International Publishing, 2016.
3. Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.
4. S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.
5. Kostas Margellos and John Lygeros. Hamilton–Jacobi formulation for reach–avoid differential games. *IEEE Transactions on Automatic Control*, 56(8):1849–1861, 2011.
6. Claire J Tomlin, John Lygeros, and S Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.
7. John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.
8. John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.
9. Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.
10. Dimitri P Bertsekas and Ian B Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7(2):233–247, 1971.
11. Dimitri Bertsekas. Infinite time reachability of state-space regions by using feedback control. *IEEE Transactions on Automatic Control*, 17(5):604–613, 1972.
12. Matthias Rungger and Paulo Tabuada. Computing robust controlled invariant sets of linear systems. *IEEE Transactions on Automatic Control*, 62(7):3665–3670, 2017.
13. Tzanis Anevlavis and Paulo Tabuada. A simple hierarchy for computing controlled invariant sets. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2020.
14. Zheming Wang, Raphaël M Jungers, and Chong Jin Ong. Computation of invariant sets via immersion for discrete-time nonlinear systems. *Automatica*, 147:110686, 2023.
15. Mitio Nagumo. Über die Lage der Integralkurven gewöhnlicher Differentialgleichungen. *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, 24:551–559, 1942.
16. Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.
17. Andrew Clark. A semi-algebraic framework for verification and synthesis of control barrier functions. *arXiv preprint arXiv:2209.00081*, 2022.
18. Zheming Wang, Raphaël M Jungers, and Chong-Jin Ong. Computation of the maximal invariant set of linear systems with quasi-smooth nonlinear constraints. In *18th European Control Conference (ECC)*, pages 3803–3808. IEEE, 2019.
19. Li Wang, Dongkun Han, and Magnus Egerstedt. Permissive barrier certificates for safe stabilization using sum-of-squares. In *2018 Annual American Control Conference (ACC)*, pages 585–590. IEEE, 2018.
20. Antoine Oustry, Matteo Tacchi, and Didier Henrion. Inner approximations of the maximal positively invariant set for polynomial dynamical systems. *IEEE Control Systems Letters*, 3(3):733–738, 2019.
21. Andrea Iannelli, Andrés Marcos, and Mark Lowenberg. Robust estimations of the region of attraction using invariant sets. *Journal of the Franklin Institute*, 356(8):4622–4647, 2019.

22. Han Wang, Kostas Margellos, and Antonis Papachristodoulou. Safety verification and controller synthesis for systems with input constraints. *arXiv preprint arXiv:2204.09386*, 2022.
23. Elias August and Mauricio Barahona. Finding positively invariant sets and proving exponential stability of limit cycles using sum-of-squares decompositions. *arXiv preprint arXiv:2208.11599*, 2022.
24. Milan Korda, Didier Henrion, and Colin N Jones. Convex computation of the maximum controlled invariant set for polynomial control systems. *SIAM Journal on Control and Optimization*, 52(5):2944–2969, 2014.
25. Christoph Helmberg, Franz Rendl, Robert J Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996.
26. Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.
27. Lars Lindemann, Alexander Robey, Lejun Jiang, Stephen Tu, and Nikolai Matni. Learning robust output control barrier functions from safe expert demonstrations. *arXiv preprint arXiv:2111.09971*, 2021.
28. Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022.
29. Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. *Real algebraic geometry*, volume 36. Springer Science & Business Media, 2013.
30. Julius Farkas. Uber die theorie des einfachen ungleichungen, J. Reine Angew. Mathematik. *Reine Angew*, 1902.
31. Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.
32. Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*, volume 78. Springer, 2008.
33. Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, and Pablo Parrilo. Sostools version 3.00 sum of squares optimization toolbox for matlab. *arXiv preprint arXiv:1310.4716*, 2013.
34. Jos F Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.