Successive Control Barrier Functions for Nonlinear Systems

Rameez Wajid
rameez.wajid@colorado.edu
University of Colorado Boulder
USA

Sriram Sankaranarayanan
first.lastname@colorado.edu
University of Colorado Boulder
USA

Abstract

We present the idea of successive control barrier functions for nonlinear (polynomial) control systems. Control Barrier Functions (CBFs) can be used to maintain safety properties for a system through the online modification of control inputs to ensure that the state remains inside a controlled invariant set that excludes a set of unsafe states. However, the synthesis of CBFs is quite difficult in practice, especially for nonlinear dynamical systems. Computationally inexpensive approaches employ relaxed control barrier conditions that result in relatively small control invariant sets. In turn, this can result in unnecessary modification of the nominal control input to keep the dynamics inside this set.

In this paper, we propose the concept of "successive" CBFs. Rather than rely on a single CBF, our approach uses a hierarchy of functions wherein functions at one level of a hierarchy become active only if the functions at the previous levels have "failed". Using a well-known approach to finding barrier functions for polynomial dynamical systems using sum of squares optimization, we show how to adapt it to synthesize successive barrier functions. We also provide "transit time" guarantees to construct a "chattering-free" runtime enforcement scheme that avoids collisions with fixed obstacles. We demonstrate our approach on a set of interesting nonlinear benchmarks, while comparing it with state of the art approaches for synthesizing CBFs.

Keywords

Control Barrier Functions, Sum Of Squares Programming, Nonlinear Hybrid Systems.

ACM Reference Format:
Rameez Wajid and Sriram Sankaranarayanan. 2025. Successive Control Barrier Functions for Nonlinear Systems. In *28th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3716863.3718043

1 Introduction

In this work, we present algorithms for the computation of control invariant sets for nonlinear polynomial dynamical systems using the notion of *successive control barrier functions.* Control Barrier Functions (CBFs) represent a promising approach to enforce safety in autonomous systems through the continuous or intermittent modification of the nominal control input, obtained from a controller for which safety is not guaranteed, to maintain the system inside a control invariant region that excludes a set of unsafe states [1]. However, the key difficulties include the construction of such functions, especially for nonlinear control systems with constraints on the control inputs. CBFs can be synthesized by a variety of approaches based on those used to synthesize control Lyapunov functions. These include bilinear optimization [36], sum-of-squares approaches using alternating descent [38], solving HJBI PDE [9] and learning such functions from data followed by a post-hoc verification [13, 22]. Despite the proliferation of these approaches, systematically synthesizing CBFs for nonlinear systems is well-known to be a hard problem. We propose the notion of *successive control barrier functions* that define strategies for avoiding a fixed set of unsafe states. The key idea is to derive these functions by combining simple barrier functions. A barrier function is a concept defined for a system without inputs that proves that the system's trajectories will not reach an unsafe set of states, starting from the initial set [32]. We first use a very simple approach for synthesizing CBFs by combining barrier functions. Each such barrier function is designed for a fixed control input $\mathbf{u} = \mathbf{u}_i$ and defines a set of states from which the application of the fixed control input $\mathbf{u}_i$ will avoid visiting the unsafe set. We define the notion of *transit time*: the time taken for the barrier function to go from some fixed threshold value inside the control invariant set to the boundary between controlled (safe) and potentially uncontrolled (unsafe) sets wherein the CBF changes sign. We show how to provide a lower bound for transit time and how such a bound can lead to a periodically sampled computation for monitoring the barrier and modifying the nominal controller to enforce safety. However, such CBFs turn out to be quite conservative, in practice: they produce relatively small control invariant sets as a result of this conservatism. In turn, this results in the runtime enforcement intervening more often to modify the nominal controller unnecessarily.

To alleviate this conservatism, we define successive control barrier functions which define a sequence of CBFs $B_0, \ldots, B_k$ wherein a later CBF satisfies its decrease condition only if the earlier ones are all violated. The key idea here is that when a trajectory violates the CBF B_i, we allow such a violation to happen provided a later B_j for $j > i$ can be used to maintain safety. We demonstrate how to construct such successive CBFs. We then present an approach to use the successive CBFs to maintain control invariance. We use the notion of a transit time to allow for a sampled-data approach which involves periodic sampling using the minimum transit time.

Finally, we demonstrate our approach using a set of small but challenging benchmark examples of nonlinear systems ranging from 2 to 7 state variables. We show how the use of successive barrier functions can help add to the control invariant set. We compare our result to a recent approach that uses neural networks as CBFs and has been implemented in the tool FOSSIL [13]. The comparison

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
HSCC '25, Irvine, CA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1504-4/2025/05
https://doi.org/10.1145/3716863.3718043

with FOSSIL demonstrates that our approach, although slower on small examples, can still scale to larger dimensional benchmarks and can obtain control invariant sets that add more controllable states when compared to those included in the CBFs discovered by FOSSIL. The contributions of this paper include; the introduction of the notion of successive control barrier functions for tight over-approximation of unsafe regions for guaranteed safe planning, the synthesis of a runtime safety monitor based on switching between the successive control barrier functions, and empirical evaluation of these on some interesting nonlinear dynamics.

1.1 Related Work

Barrier functions, sometimes called barrier certificates, are used to guarantee a system never reaches an unsafe set of states starting from some initial set [32]. Whereas barrier functions are defined for systems without control inputs, the notion of control barrier functions extend to systems with control inputs. Control Barrier Functions (CBFs) can be used to synthesize feedback laws or modify a nominal feedback in order to ensure a dynamical system's trajectories never leave a specified safe set, or equivalently, never enter an unsafe set [1]. These approaches have been quite beneficial in the verification of cyber-physical systems [12], especially for safety-critical applications [10, 23, 44].

Although CBFs are being widely used to enforce safety properties, computing a CBF for a given nonlinear dynamics is quite challenging. A significant challenge in computing control barrier functions arises due to the relative degree of the function. Using exponential control barrier functions addresses these challenges [27], and higher-order control barrier functions generalize these approaches further [37, 41]. The higher-order approach keeps differentiating a high relative degree constraint function until a control input is found. The CBF conditions are then enforced over the highest-order derivative. However, in practice, many dynamical systems may not obey the relative degree condition. Also, since the approach often starts from the specification of the safety property, it can lead to infeasibility during runtime wherein we may fail to find a control input can be applied to maintain safety.

Lately, many learning-based approaches for CBF computation have emerged. Neural CBFs exploit the universal approximation property of neural networks [22, 46]. Another interesting approach is finding an invariant by learning a hypersurface for each obstacle using support vector machines [42]. In practice, neural CBFs can be learned rapidly but the difficulty arises in the process of verifying a given neural CBF.

Besides neural CBFs, Sum-of-squares(SOS) optimization techniques have shown great promise regarding the efficient computation of control barrier functions [38]. In the case of polynomial dynamical systems with semi-algebraic sets, SOS techniques relax the algebraic geometry problem into semi-definite programs, leading to efficiency in solving the optimization problem [29, 35]. The approach has been well integrated into packages such as SOSTOOLS and the more recent Julia SumOfSquares package [19, 39].

Our approach relies on computing multiple barrier functions. Computing more than one barrier function has recently become a prominent approach for safe planning in challenging environments. The need for multiple barriers primarily arose to counter scenarios where a single barrier function may not be sufficient. Many of these approaches consider environments with multiple obstacles/constraints by synthesizing multiple barriers, wherein each of these barriers enforcing a subset of constraints [16, 21]. One way of computing these viability domains is applying all barriers in a single quadratic program [33, 43], while some approaches decouple the design to ensure joint feasibility under more conservative assumptions [6]. Our approach considers a single unsafe set X_u unlike the other approaches mentioned above. The motivation for considering multiple CBFs lies in improving the control invariant set for a single unsafe set specification.

When dealing with multiple CBFs, a valid concern is how often one needs to switch between barriers. A continuous time monitoring strategy may lead to Zeno behavior. There have been attempts to alleviate this problem by incorporating dwell time constraints when synthesizing barriers [7, 8], where the controllers can only intervene once a minimum dwell time has elapsed. The dwell times are present to compute impulsive timed CBFs. Our work uses a different approach that consists of computing minimum transit times using a notion introduced in this paper, that we call the "slow transit property". The transit property allows our approach to effectively work in a sampled-data manner; i.e., by periodically sampling the state of the system and making a decision on how to modify the nominal control input for the subsequent time interval. Our approach is set up so as to guarantee that the system cannot transit from safe to unsafe within this time interval.

The concept of a controlled invariant set is dual to another interesting concept of Inevitable Collision States (ICS). The ICS concept was first formulated in [14]. A state is an ICS where, no matter what trajectory is executed by the system, a collision with an obstacle will eventually occur [25]. Whereas, the controlled invariant set guarantees that there always exists a control strategy that will ensure collisions with obstacles are not possible. Determining whether a state is ICS for all possible trajectories is intractable [24]. Even for a simple dynamical system, computing ICS for only a few trajectories is computationally intensive [31].

There have been few notable attempts to approximate ICS. One of the earliest attempts was a generic ICS checker implemented in [25, 26]. Bekris et al [2, 3] introduced sampling-based planners for avoiding ICS efficiently. Another approach was to employ backward reachability for ICS approximation [31]. Lim et al. [20] use precomputed offsets for safe navigation for fixed-wing aerial vehicles. In this paper, we present a new guaranteed over-approximation technique that uses sum of squares optimization and exploits the link between ICS and control barrier functions.

2 Preliminaries

We will present key preliminary notions including dynamical system models for autonomous vehicles, control barrier functions [1, 32], and sum-of-square (SOS) optimization [18, 29, 30, 35]. We will use boldface notation to denote vectors. For a vector $\mathbf{x} \in \mathbb{R}^n$, the i^{th} component is denoted x_i.

Let $\mathbf{x} \in \mathbb{R}^n$ denote the state of the system and $\mathbf{u} \in \mathbb{R}^m$ denote the control inputs. We assume that $\mathbf{u} \in U$, wherein U is a hyper-rectangle of the form $[l(u_1), h(u_1)] \times \cdots \times [l(u_m), h(u_m)]$. Let $\mathbb{R}[\mathbf{x}]$ denote the set of multivariate polynomials over $\mathbf{x}$ with real-valued coefficients and $\mathbb{R}_d[\mathbf{x}]$ denote polynomials whose terms have degree at most $d \in \mathbb{N}$. A basic *semi-algebraic* set is defined by a conjunction of polynomial inequalities of the form $R = \{\mathbf{x} \mid \bigwedge_{i=1}^{k} p_i(\mathbf{x}) \geq 0\}$ wherein $p_1, \ldots, p_k \in \mathbb{R}[\mathbf{x}]$.

Definition 2.1 (Polynomial Dynamical System). A polynomial dynamical system Σ over some state-space $X \subseteq \mathbb{R}^n$ is a system of ordinary differential equations (ODE) of the form $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, wherein $f : \mathbb{R}^n \to \mathbb{R}^n$ is a polynomial vector field of the form $f(\mathbf{x}, \mathbf{u}) = (p_1(\mathbf{x}, \mathbf{u}), \ldots, p_n(\mathbf{x}, \mathbf{u}))$ for polynomials $p_1, \ldots, p_n$.

A trajectory of a system over time T for a continuous input signal $\mathbf{u} : [0, T) \to U$ is a continuous and differentiable function $\varphi : [0, T) \to X$ such that for all time $t \in [0, T)$, $\frac{d\varphi}{dt} = f(\varphi(t), \mathbf{u}(t))$.

Example 2.2. Consider a polynomial system with 3 state variables (x, y, θ) and control input u. The state space $X = \{(x, y, \theta) | \theta \in [-\pi, \pi]\}$. The dynamics are given as

$$\dot{x} = \left(1 - \frac{\theta^2}{2}\right),\ \dot{y} = \left(\theta - \frac{\theta^3}{3}\right),\ \dot{\theta} = u,\ \text{wherein } u \in [-0.05, 0.05]\,.$$

2.1 Barrier and Control Barrier Functions

Barrier functions (or barrier certificates) are often used in control applications to realize safety properties of closed-loop autonomous systems. The concept of a barrier function was first introduced by Prajna and Jadbabaie [32] and later extended by many others [4, 17, 27, 45]. The concept of a control barrier function extends that of a barrier for systems with control inputs [1, 40]. Let $\dot{\mathbf{x}} = f(\mathbf{x})$ be a differential equation for Lipschitz continuous function f (note: no control inputs). Let X_u represent the unsafe set.

Definition 2.3 (Barrier Function). A *barrier function* $B : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function of the state variables such that there exist constants $\epsilon > 0$ and λ satisfying:

$$\forall\, \mathbf{x} \in X_u,\ B(\mathbf{x}) \geq \epsilon,\ \text{and} \ \forall\, \mathbf{x} \in X,\ \ \nabla B(\mathbf{x}) \cdot f(\mathbf{x}) \leq -\lambda B(\mathbf{x})\,.$$

Theorem 2.4. *Let B be a barrier function following Def. 2.3 and $\mathbf{x}_0 \in \mathbb{R}^n$ be such that $B(\mathbf{x}_0) \leq 0$. Any trajectory $\varphi : [0, T) \to \mathbb{R}^n$ with $\varphi(0) = \mathbf{x}_0$ will not intersect the unsafe set X_u.*

Proof. Let $\varphi : [0, T) \to \mathbb{R}^n$ be any trajectory. Since f is continuous, it follows that φ is differentiable. Consider the function $e^{\lambda t} B(\varphi(t))$. For all $t \in [0, T)$, $\frac{d}{dt}\left(e^{\lambda t} B(\varphi(t))\right) = \lambda e^{\lambda t} B(\varphi(t)) + e^{\lambda t} \frac{d}{dt} B \leq \lambda e^{\lambda t} B(\varphi(t)) - \lambda e^{\lambda t} B(\varphi(t)) \leq 0$. Therefore, $e^{\lambda t} B(\varphi(t)) \leq B(\varphi(0))$. Thus, for all $t \geq 0$, $B(\varphi(t)) \leq 0$ if $B(\varphi(0)) \leq 0$. Therefore, $\varphi(t) \notin X_u$ since $B(\mathbf{x}) \geq \epsilon > 0$ for all $\mathbf{x} \in X_u$. □

Example 2.5. Consider the system in Ex. 2.2, wherein we fix the external inputs $u_1 = -0.05$. Let $X_u = \{(x, y) \mid x^2 + y^2 \leq 0.04\}$ be the unsafe set. We obtain a barrier function

$$B_0(x, y, \theta) = 0.25 + 10^{-4} \times \left(\begin{array}{c} -2.42\theta - 23.1y - 9x + 2.7\theta^2 + 49.4y\theta \\ -3.1y^2 - 6.36x\theta - 1.02xy - 5.86x^2 \end{array}\right).$$

We set $\epsilon = 0.25, \lambda = -0.1$. Note that for any state (x, y, θ) s.t. $B_0(x, y, \theta) \leq 0$, we can apply the control $u = -0.05$ to keep the system away from the set X_u.

The concept of a control barrier function extends that of a barrier function for systems with control inputs.

Definition 2.6 (Control Barrier Function). Consider a system $\mathcal{S}$ as in Definition 2.1 and the set $X_u \subseteq X$ be the unsafe set. A *control barrier function* for $\mathcal{S}$ is a differentiable function $B : X \to \mathbb{R}$ such that there exist constants $\epsilon > 0$ and λ satisfying (a) $B(\mathbf{x}) \geq \epsilon$ for all $\mathbf{x} \in X_u$ and (b) for all $\mathbf{x} \in \mathbb{R}^n$ there exists a control input $\mathbf{u} \in U$ such that $\nabla B(\mathbf{x}) \cdot f(\mathbf{x}, \mathbf{u}) \leq -\lambda B(\mathbf{x})$.

Analogous to barrier functions, we can prove the following theorem for control barrier functions [1].

Theorem 2.7. *Let $B(\mathbf{x})$ be a control barrier function following Def. 2.6. For all $\mathbf{x}_0$ such that $B(\mathbf{x}_0) \leq 0$, there exists a control feedback law $\kappa : X \to U$ mapping states to controls, such that the resulting trajectory $\varphi : [0, T) \to \mathbb{R}^n$ will not enter the unsafe set X_u.*

Control barrier functions are used to avoid violations of safety properties by applying a control input to the system that ensures the derivative satisfies $\dot{B}(\mathbf{x}(t)) \leq -\lambda B(\mathbf{x}(t))$.

2.2 Sum Of Squares Programming for Synthesizing Barriers

Barrier functions for polynomial functions can be synthesized using sum-of-squares (SOS) optimization techniques [30, 35]. A polynomial $\sigma \in \mathbb{R}[\mathbf{x}]$ is a *sum of squares* (SOS) iff it can be written as $\sigma = h_1^2 + \cdots + h_k^2$ for some polynomials $h_1, \ldots, h_k \in \mathbb{R}[\mathbf{x}]$. Note that if a polynomial σ is SOS then for all $\mathbf{x} \in \mathbb{R}^n$, $\sigma(\mathbf{x}) \geq 0$ (the converse is not necessarily true). Let the unsafe set X_u be a basic semialgebraic set defined as follows: $X_u = \{\mathbf{x} \in \mathbb{R}^m \mid g_1(\mathbf{x}) \geq 0,\ g_2(\mathbf{x}) \geq 0,\ \ldots,\ \text{and}\ g_l(\mathbf{x}) \geq 0\}$. The following conditions are sufficient for a polynomial $B(\mathbf{x})$ to be a barrier function Def. 2.3:

$$\begin{array}{l} \mathbf{C0} : B(\mathbf{x}) - \epsilon = \sigma_0 +\ \sigma_1 g_1 + \cdots + \sigma_l g_l,\ \text{s.t. } \sigma_0, \sigma_1, \ldots, \sigma_l \text{ are } \textit{SOS} \\ \mathbf{C1} :\ -\nabla B\ \cdot f(\mathbf{x}) - \lambda B(\mathbf{x}) = \xi\ \text{ s.t. } \xi \text{ is SOS}\,. \end{array}$$

Note that conditions **C0, C1** express the fact that for all $\mathbf{x} \in X_u$, $B(\mathbf{x}) \geq \epsilon$ and for all $\mathbf{x}$, $\nabla B \cdot f \leq -\lambda B$ (see Def. 2.3). To discover a function B, we proceed as follows:

(1) Fix degree bounds for $B, \sigma_0, \ldots, \sigma_l, \xi$, expressing each unknown polynomial as a summation over the monomials with unknown coefficients.
(2) Conditions **C0, C1** above yield linear equations on the unknown coefficients.
(3) Express the SOS condition on ξ_j, σ_i in terms of the positive semi-definiteness of a matrix involving its coefficients [30].
(4) This results in a semi-definite programming problem whose feasible solution yields B [5].

This process is automated by useful software tools such as SOSTOOLS [28] and the Julia SumOfSquares.jl package [19, 39].

Example 2.8. Consider again the running example (Ex. 2.2) wherein we fix the external inputs $u_1 = 0.05$. Let $X_u = \{(x, y) | x^2 + y^2 \leq 0.04\}$. By fixing a maximum degree of 4 for the barrier function, we obtain

$$B_1 = \left(\begin{array}{c} 10 - 10\theta + 1.95y - 5.5x + 10\theta^2 - 10y\theta + 1.7y^2 + 1.3x\theta - 2.4xy + x^2 \\ + \cdots + \\ 0.05x^2\theta^2 + 0.075x^2y\theta + 0.0024x^2y^2 - 0.017x^3\theta - 0.0027x^3y + 0.001x^4 \end{array}\right).$$

This barrier function and the function B_0 from Example 2.5 were synthesized using the `SumOfSquares.jl` package in Julia [19, 39].

Algorithm 1: Algorithm for multiple control barrier computation

```
Data: System $\Sigma$, unsafe set $X_u$, finite control set $U_{fin}$,
      constants $\epsilon > 0$, $\eta$, $\delta$ and $\lambda$, degree bound $D$, test set $S$.
Result: Set of barrier functions and associated controls $\mathcal{B}$.
1  $\mathcal{B} = \emptyset$ /* Initialize to empty set */
2  for $\mathbf{u}_t \in U_{fin}$ do
3      for $\mathbf{x}_t \in S$ do
4          find polynomial $B$ of degree $D$
5            s.t. $(\forall \mathbf{x} \in X_u)\ B(\mathbf{x}) \geq \epsilon$
6            $(\forall \mathbf{x})\ \nabla B \cdot (f(\mathbf{x}, \mathbf{u}_t)) \leq -\lambda B$
7            $B(\mathbf{x}_t) \leq 0$
8          if $B$ found then
9              $\mathcal{B} = \mathcal{B} \cup \{(B, \mathbf{u}_t)\}$
10             $S = S \setminus \{\mathbf{x} \in S \mid B(\mathbf{x}) \leq 0\}$ /* remove already
                 eliminated test point */
11 return $\mathcal{B}$
```

3 Successive Control Barrier Functions

In this section, we describe the overall successive control barrier function approach of this paper.

a. First, we show that fixing a finite set of control inputs can be used to synthesize barrier functions that can be combined into a control barrier function. Section 3.1 describes this approach.
b. Unfortunately, our approach produces a non-smooth control barrier function which requires theoretical ideas from non-smooth analysis and in practice can lead to zenoness/chattering. We provide an approach for analyzing "transit times" that will allow us to work with discrete time monitors.
c. The combination of using SOS programming and enforcing a transit time condition make the resulting control barrier functions too *conservative*; i.e., their controlled invariant region marks off many states as "uncontrollable". We will address this by presenting the idea of successive control barriers.

3.1 From Barriers to Control Barriers

Let $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ be the dynamical model with a compact set of control inputs $\mathbf{u} \in U$ and X_u be the set of unsafe states. Let U_{fin} be a finite subset of U. Our control barrier function will consider controls only from this set U_{fin}.

Remark 1. *If the dynamics $f(\mathbf{x}, \mathbf{u})$ are control affine; i.e., $f(\mathbf{x}, \mathbf{u}) = f_1(\mathbf{x}) + f_2(\mathbf{x})\mathbf{u}$ and U is represented by a (compact) convex polyhedron, then U_{fin} can be taken to be the vertices of U. This is because whenever for a state $\mathbf{x}$ there exists a $\mathbf{u} \in U$, such that $(\nabla B(\mathbf{x})) \cdot (f_1(\mathbf{x}) + f_2(\mathbf{x})\mathbf{u}) \leq -\lambda B(\mathbf{x})$, we can show that one of the vertices $\mathbf{u} \in U_{fin}$ will satisfy the decrease condition since it is affine in $\mathbf{u}$.*

Algorithm 1 computes a set of barrier functions and associated controls of the form $\mathcal{B} = \{(B_1, \mathbf{u}_1), \ldots, (B_k, \mathbf{u}_k)\}$, wherein each B_i is a barrier function for the dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}_i)$ with $\mathbf{u} = \mathbf{u}_i$. To facilitate multiple barrier computation, we fix a finite set S of "test" states such that $S \cap X_u = \emptyset$. Our goal is to attempt to find a barrier for each test state $\mathbf{x}_t \in S$ that shows that it belongs to the control invariant region when the control is fixed to some $\mathbf{u}_i \in U_{\text{fin}}$.

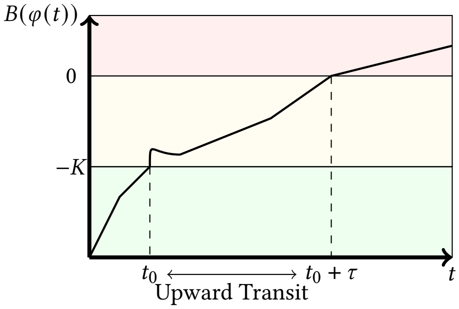


Figure 1: Illustration of a trajectory transiting from $B(\varphi(t_0)) \leq -K$ to $B(\varphi(t_0 + \tau)) \geq 0$.

Lines 4- 7 present the SOS formulation. In particular, line 7 adds a linear constraint in the coefficients of the unknown polynomial B that forces the polynomial to be non-positive at the test point $\mathbf{x}_t$. If a barrier is found for a test point $\mathbf{x}_t$ and some control $\mathbf{u}_j \in U_{\text{fin}}$, the test point is eliminated from the set S. In practice, S consists of randomly generated states that lie outside X_u.

We wish to combine the set of barrier functions obtained from Algorithm 1 into a single *control barrier function*. For instance, we can consider the piecewise minimum: $B(\mathbf{x}) = \min_{(B_i, \mathbf{u}_i) \in \mathcal{B}} B_i(\mathbf{x})$, as the minimum over all the barriers considered. However, such a function is *non-smooth*. Although the theory of nonsmooth barrier functions [15, 16] have been studied using techniques from nonsmooth analysis [11], the results of Glotfelter et al are not directly applicable in our case, where the barrier functions $B_1, \ldots, B_m$ are obtained by altering the control inputs and thus technically, *pertain to different dynamics*. We now study a simpler but conservative scheme that imposes an additional transit time condition; imposing the transit time condition on the barriers computed using Algorithm 1 allows us to effectively combine them into a single control barrier function and enforce control invariance.

3.2 Transit Time Bounds for a Barrier Function

Let $B(\mathbf{x})$ be a barrier function for a fixed control input $\mathbf{u}_r \in U$. The set $\{\mathbf{x} \mid B(\mathbf{x}) \leq 0\}$ is the controlled invariant (CI) corresponding to $B(\mathbf{x})$ using the control $\mathbf{u} = \mathbf{u}_r$. Let $K > 0$ be a fixed threshold. Let $\mathbf{u}(t) \in U$ be an arbitrary control signal. Let $\varphi(t)$ denote the resulting trajectory. We say that φ "transits" from a state $\varphi(t_0)$ wherein $B(\varphi(t_0)) = -K$ to a state $B(\varphi(t_0 + \tau)) = 0$ for transit time $\tau > 0$ if for all time $t \in (t_0, t_0 + \tau)$, $B(\varphi(t)) \in (-K, 0)$ (See Fig. 1). Since we have assumed that the dynamics are time invariant, we can set $t_0 = 0$ without loss of generality.

Definition 3.1 (Minimum "Upward" Transit Time). The minimum (upward) transit time given a barrier function $B(\mathbf{x})$ and threshold $K > 0$ is defined as the infimum over the transit times of all possible trajectories $\varphi : [0, T) \to X$ of the system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ corresponding to a continuous control signal $\mathbf{u} : [0, T) \to U$.

$$\tau^* = \inf \left\{ \tau \;\middle|\; \begin{array}{c} \varphi \text{ is a trajectory for continuous } \mathbf{u} : [0, T) \to U, \\ 0 < \tau \leq T,\ B(\varphi(0)) = -K, B(\varphi(\tau)) = 0, \text{and} \\ \forall t \in (0, \tau)\ B(\varphi(t)) \in (-K, 0) \end{array} \right\}.$$

We will now augment the requirements for a barrier function to be able to provide a bound on its transit time given a threshold K.

Definition 3.2 (Upward Slow Transit Property). We say that a control barrier function has the upward *slow transit property* (STP) iff there exist $\delta, \eta \in \mathbb{R}$ such that

$$\forall\, \mathbf{x} \in X, \forall \mathbf{u} \in U,\ B(\mathbf{x}) \in [-K, 0] \implies \nabla B \cdot f(\mathbf{x}, \mathbf{u}) \le \eta B(\mathbf{x}) + \delta\,. \tag{1}$$

Theorem 3.3. *If a barrier function $B(\mathbf{x})$ satisfies* (1) *then (a) if* $\max(\delta, -\eta K + \delta) > 0$*, then* $\tau^* \ge \frac{K}{\max(\delta, -\eta K + \delta)}$*; and (b) if* $\max(\delta, -\eta K + \delta) \le 0$*, then no transits are possible: i.e,* $\tau^* = \infty$*.*

Proof. Let $\varphi : [0, T) \to X$ be any trajectory corresponding to some continuous control signal $\mathbf{u} : [0, T) \to U$ and $\tau > 0$ be a time such that the conditions for transit hold: $B(\varphi(0)) = -K, B(\varphi(\tau)) = 0$, and $\forall t \in (0, \tau)\ -K \le B(\varphi(t)) \le 0$. Since $\mathbf{u}$ is continuous, φ is continuous and differentiable. Applying the Lagrange mean value theorem, we obtain,

$$B(\varphi(\tau)) = B(\varphi(0)) + \tau \frac{d}{dt} B(\varphi(t))\,, \text{where } t \in (0, \tau)\,.$$

Next, we observe that $\frac{d}{dt} B(\varphi(t)) = \nabla B(\varphi(t)) \cdot f(\varphi(t)) \le \eta B(\varphi(t)) + \delta$, since $B(\varphi(t)) \in [-K, 0]$. We have

$$0 = B(\varphi(\tau)) = B(\varphi(0)) + \tau \frac{d}{dt} B(\varphi(t)) \le -K + \tau(\delta + \eta B(\varphi(t)))\,.$$

Since $B(\varphi(t)) \in [-K, 0]$, $\delta + \eta B(\varphi(t)) \le \begin{cases} \delta & \eta \ge 0 \\ \delta - \eta K & \eta < 0 \end{cases}.$

Therefore, $\delta + \eta B(\varphi(t)) \le \max(\delta, \delta - \eta K)$. Suppose $\max(\delta, \delta - \eta K) > 0$, then $\tau^* \ge \frac{K}{\max(\delta, -\eta K + \delta)}$. This proves (a).

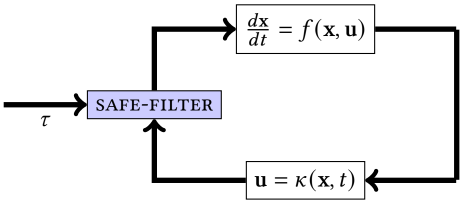
Otherwise, if $\max(\delta, \delta - \eta K) \le 0$, we have $\delta - \eta K \le 0$ and $\delta \le 0$. As a result, we conclude that whenever $B(\mathbf{x}) \in [-K, 0]$, $\frac{d}{dt} B(\mathbf{x}) = \nabla B(\mathbf{x}) \cdot f(\mathbf{x}, \mathbf{u}) \le \eta B(\mathbf{x}) + \delta \le 0$. Therefore, applying mean value theorem, $B(\varphi(\tau)) = B(\varphi(0)) + \tau \frac{d}{dt} B(\varphi(t)) \le B(\varphi(0)) \le -K$. Hence, no transit is possible. □

Example 3.4. Analyzing the barrier function in our running example 2.8 using SOS programming while fixing $K = 1$ yields a bound on the minimum transit time of $\tau^* \ge 1.86$.

3.3 Safety Enforcement using Multiple Barriers

Let $B_1, \ldots, B_m$ be barrier functions for unsafe set X_u obtained by fixing the control inputs to $\mathbf{u}_1, \ldots, \mathbf{u}_m$, respectively. We assume that all B_i satisfy the upward STP (Defs. 3.2). This is ensured by modifying Algorithm 1 to test that each barrier function discovered by the algorithm satisfies (1) and discarding ones that do not. Let $\tau_1, \ldots, \tau_m$ be lower bounds on the transit time for these barriers using the approach described in the previous section for a fixed threshold $K > 0$. Let $\tau = \min(\tau_1, \ldots, \tau_m)$. Let $\mathbf{u} = \kappa(\mathbf{x}, t) \in U$ be a nominal control law that is continuous in $\mathbf{x}$ and t [1].

Figure 2 shows the setup of the safety filter corresponding to multiple barriers $B_1, \ldots, B_m$. The safe-filter module is invoked every τ time units and can issue one of two decrees: (a) Pass: allow the feedback law κ to operate for the next τ time units; or (b) Override($\mathbf{u}_j$): keep a constant control input $\mathbf{u} = \mathbf{u}_j$ for the next τ time units.



$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u})$
τ
SAFE-FILTER
$\mathbf{u} = \kappa(\mathbf{x}, t)$

Figure 2: Safety enforcement for multiple barrier functions through a safety filter.

Let $B_{min}(\mathbf{x}) = \min(B_1(\mathbf{x}), \ldots, B_m(\mathbf{x}))$ be the minimum over all the barrier functions. We will let $\mathrm{ind}(\mathbf{x}) = \min\{j \mid B_j(\mathbf{x}) = B_{min}(\mathbf{x})\}$ be the least index j for which $B_j(\mathbf{x}) = B_{min}(\mathbf{x})$ and $\mathbf{u}_{min}(\mathbf{x}) = \mathbf{u}_{\mathrm{ind}(\mathbf{x})}$ be the control input corresponding to the minimum index. The safety filter is defined as follows:

$$\textsc{safe-filter}(\mathbf{x}; B_{min}) = \begin{cases} \textsc{Pass} & \text{if } B_{min}(\mathbf{x}) < -K \\ \textsc{Override}(\mathbf{u}_{min}(\mathbf{x})) & \text{otherwise} \end{cases}$$

In other words, the safety filter allows the feedback κ to be applied for the next τ time units if at least one of the functions $B_1, \ldots, B_m$ is below the threshold $-K$. Otherwise, it applies the control $\mathbf{u}_j$ corresponding to that barrier function B_j whose value is minimal among $B_1, \ldots, B_m$.

Let $\varphi : [0, T) \to X$ be a trajectory of the closed loop system with the safety filter applied (see Fig. 2). If $B_{min}(\varphi(0)) < 0$ then $B_{min}(\varphi(t)) < 0$ for all $t \in [0, T)$. We will assume that the safety filter is evaluated at times $t = 0, \tau, 2\tau, \ldots$. Furthermore, we assume that the execution of κ and the safety filter are instantaneous. First, we establish a lemma for the safety filter. Let $\mathbf{x}_k = \varphi(k\tau)$ for $k \in \mathbb{N}$.

Lemma 3.5. *If $B_{min}(\mathbf{x}_k) \in [-K, 0)$, then applying the overriding feedback $\mathbf{u}_j$ provided by* safe-filter$(\varphi(k\tau); B_{min})$ *ensures that $B_{min}(\varphi(t)) < 0$, for all $t \in [k\tau, (k+1)\tau]$.*

Proof. Let $j = \mathrm{ind}(\mathbf{x}_k)$. Applying the control $\mathbf{u}_j$ ensures that for $t \in [k\tau, (k+1)\tau]$, we have $\frac{d}{dt} B_j(\varphi(t))|_{\mathbf{u}=\mathbf{u}_j} \le -\lambda B_j(\varphi(t))$. Since, $B_{min}(\mathbf{x}_k) = B_j(\mathbf{x}_k) < 0$. Applying theorem 2.4, yields $B_{min}(\varphi(t)) \le B_j(\varphi(t)) < 0$ for $t \in [k\tau, (k+1)\tau]$. □

Lemma 3.6. *If $B_{min}(\mathbf{x}_k) < -K$ then $B_{min}(\varphi(t)) < 0\ \forall\ t \in [k\tau, (k+1)\tau]$.*

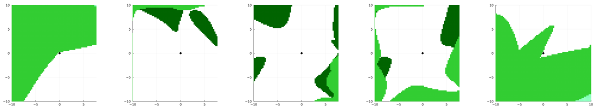
Proof. Suppose we have $B_{min}(\varphi(t)) \ge 0$ for some $t \in [k\tau, (k+1)\tau]$, by the continuity of $B(\varphi(t))$, there exist time intervals t_1, t_2 such that (a) $B(\varphi(t_1)) = -K$, (b) $B(\varphi(t_2)) = 0$ and $B(\varphi(s)) \in (-K, 0)$ for $s \in (t_1, t_2)$. The trajectory transits during the time interval $[t_1, t_2]$. By definition of τ, we have $\tau \le t_2 - t_1$. Therefore, t_1, t_2 cannot both be in the time interval $[k\tau, (k+1)\tau]$. This yields a contradiction. Thus $B_{min}(\varphi(t)) < 0$ for all $t \in [k\tau, (k+1)\tau]$. □

Theorem 3.7. *Let $\varphi : [0, T) \to X$ be any closed-loop trajectory with the safety filter applied at times $t_k = k\tau$ for $k \in \mathbb{N}$. If $B_{min}(\varphi(0)) < 0$ then $B_{min}(\varphi(t)) < 0$ for all times $t \in [0, T)$.*

Proof. The proof follows directly from Lemmas 3.5 and 3.6. □

Thus, the time-triggered application of the safe-filter rule ensures that the set $\{\mathbf{x} \in X \mid B_1(\mathbf{x}) \le 0 \text{ or } \cdots \text{or } B_m(\mathbf{x}) \le 0\}$ can

[1] The dependence on time allows κ to be a feed-forward control input: $\mathbf{u} = \kappa(t)$.



$\theta = -\pi$ $\theta = -\frac{\pi}{2}$ $\theta = 0$ $\theta = \frac{\pi}{2}$ $\theta = \pi$

Figure 3: Successive barrier functions extend the overall control invariant region. The earlier approximations of the control invariant set are shown in light green. The minimum transit time was computed to be 0.02.

be maintained as a controlled invariant set. Our goal, however, is to find as large a control invariant set as possible so that we can apply the nominal controller without overriding too often. We will present the idea of a successive barrier function to add more states that can be added to the control invariant region.

3.4 Successive Barriers

Suppose we have synthesized multiple barrier functions $B_1, \ldots, B_m$ with associated controls $\mathbf{u}_1, \ldots, \mathbf{u}_k$. We can establish control invariance using the function $B^{(0)}(\mathbf{x}) = \min_{j=1}^{m} B_j(\mathbf{x})$ for states $\mathbf{x}$ wherein $B^{(0)}(\mathbf{x}) \leq 0$, using the upward slow transit property. We will focus on synthesizing *successive* barrier functions that "work" only when $B^{(0)}(\mathbf{x}) \geq 0$. The goal is that these functions together add to the set of states from which we can apply controls to successfully avoid reaching the unsafe states. Let $U_{\text{fin}} = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\} \subseteq U$ be a fixed finite set of control inputs.

Definition 3.8. A function $B_i(\mathbf{x})$ is a successive barrier function w.r.t a previously established multiple control barrier function $B^{(0)}(\mathbf{x})$ for a fixed control input $\mathbf{u}_i \in U_{fin}$ iff

(1) $B_i(\mathbf{x}) \geq \epsilon$ for all $\mathbf{x} \in X_u$,

(2) for all $\mathbf{x}$ such that $B^{(0)}(\mathbf{x}) \geq -K$, $\nabla B_i \cdot f(\mathbf{x}, \mathbf{u}_i) \leq -\lambda B_i(\mathbf{x})$. In other words, B_i satisfies the derivative condition only for those states wherein $B^{(0)}(\mathbf{x}) \geq -K$.

(3) B_i has its upward transit time bounded from below by τ_i.

Let $B_1(\mathbf{x}), \ldots, B_M(\mathbf{x})$ be successive barrier functions for control inputs $\mathbf{u}_1, \ldots, \mathbf{u}_M$, respectively and previously established control barrier function $B^{(0)}(\mathbf{x})$. Let $B^{(1)}(\mathbf{x}) = \min(B_1(\mathbf{x}), \ldots, B_M(\mathbf{x}))$ and τ be the minimum transit time among all barriers that have been synthesized thus far.

Note that the concept of successive barriers can be iterated. Let $B^{(0)}$ be our initial control barrier function. We can compute a successive barrier $B^{(1)}$ which satisfies the derivative conditions only if $B^{(0)}(\mathbf{x}) \geq -K$. In turn, we can use this to derive another function $B^{(2)}$ as the minimum over multiple polynomials each of whose derivative conditions holds only if $B^{(0)}(\mathbf{x}) \geq -K$ and $B^{(1)}(\mathbf{x}) \geq -K$ and so on. This yields a sequence of successive barrier functions. Figure 3 shows how successive barriers can add to the set of control invariant states.

We will now outline a scheme to synthesize successive barrier functions iteratively. Algorithm 2 shows the overall algorithm. The approach maintains a finite set S of sample *test states* and a family of sets of successive barrier functions: $\mathcal{B} = \left\langle B^{(0)}, B^{(1)}, \ldots, B^{(k)} \right\rangle$, wherein each $B^{(i)}$ is of the form $\min(B_{i,1}, \ldots, B_{i,j})$ for some polynomial functions $B_{i,1}, \ldots, B_{i,j}$. Note that the constraint $B^{(i)} \geq 0$ is equivalent to $\bigwedge_{k=1}^{j} B_{i,k} \geq 0$. We initialize $\mathcal{B}$ to be the "ancestors" (set of previous successive barriers) and the test states to be a randomly sampled set of some fixed size N.

We iterate over the control inputs $\mathbf{u}_i \in U_{fin}$ and test states $\mathbf{x}_t \in S$. We attempt to discover a barrier function B of degree at most D such that (a) $\forall \mathbf{x} \in X_u, B(\mathbf{x}) \geq \epsilon$; (b) the derivative condition; $\bigwedge_{l=1}^{k} B^{(l)} \geq 0 \implies \nabla B \cdot f(\mathbf{x}, \mathbf{u}) \leq -\lambda B$, and finally (c) we insist that $B(\mathbf{x}_t) \leq 0$, i.e, the barrier ensures safety at the test point $\mathbf{x}_t$. We use sum of squares (SOS) programming to find such a barrier function (Lines 4-7). If it succeeds, then we add the new barrier to our list (Line 9), eliminating $\mathbf{x}_t$ and other test points already excluded from the unsafe region (Line 10).

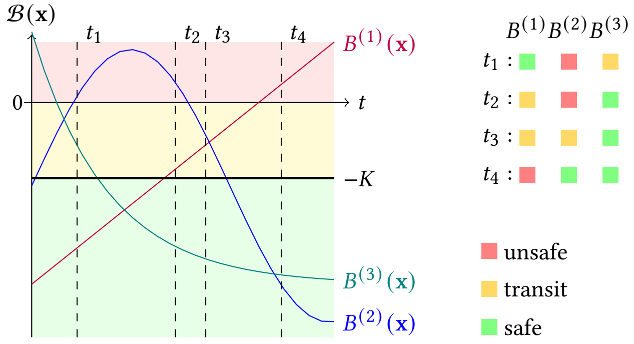
Algorithm 2: Algorithm for successive barrier computation

```
Data: System $\Sigma$, unsafe set $X_u$, finite control set $U_{fin}$,
      previous successive barriers
      $\mathcal{B} = \left\langle B^{(0)}, B^{(1)}, \ldots, B^{(k)} \right\rangle$, constants $\epsilon > 0$, $\eta$, $\delta$ and
      $\lambda$, degree bound $D$, test set $S$.
Result: $k+1$ level successive barrier function $B^{(k+1)}$.
1  $B^{(k+1)} \longleftarrow \infty$ /* Initialize to trivial function */
2  for $\mathbf{u} \in U_{fin}$ do
3      for $\mathbf{x}_t \in S$ do
4          find polynomial $B$ of degree $D$
5            s.t. $\mathbf{x} \in X_u \implies B(\mathbf{x}) \geq \epsilon$
6          $\bigwedge_{l=1}^{k} B^{(l)} \geq 0 \implies \nabla B \cdot f(\mathbf{x}, \mathbf{u}) \leq -\lambda B$
7            $B(\mathbf{x}_t) \leq 0$
8          if B found and B has bounded transit time $\tau$ then
9              $B^{(k+1)} = \min(B^{(k+1)}, B)$
10             $S = S \setminus \{\mathbf{x} \in S \mid B(\mathbf{x}) \leq 0\}$ /* remove already
                eliminated test points from $S$ */
11 return $B^{(k+1)}$
```

4 Runtime Safety using Successive Barriers

Suppose we have synthesized a sequence of successive barrier functions $B^{(0)}, B^{(1)}, \ldots, B^{(k)}$, wherein each $B^{(i)} = \min(B_{i,1}, \ldots, B_{i,n(i)})$ is obtained as a combination of multiple barrier functions with associated inputs $\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,n(i)}$. Let $\tau_1, \ldots, \tau_k > 0$ be bounds on the upward transit times for $B^{(1)}, \ldots, B^{(k)}$, respectively. We define $\tau = \min(\tau_1, \ldots, \tau_k)$. Our goal is to combine the safety monitors for each of the individual control barriers, as explained in Section 3.3 to yield a composite safety monitor for the overall sequence of successive barrier functions.

Let $\mathbf{x}$ be the state at any time t. We say that the control barrier function $B^{(i)}$ is *green* if $B^{(i)} < -K$, *yellow* if $B^{(i)} \in [-K, 0)$ and *red* if $B^{(i)} \geq 0$. Note that if the control barrier is green or yellow, then the state $\mathbf{x}$ cannot be in the unsafe set X_u by construction. A color indicator function for a given state $\mathbf{x}$ and control barrier $B^{(i)}$ can



	$B^{(1)}$	$B^{(2)}$	$B^{(3)}$
t_1 :	🟩	🟥	🟨
t_2 :	🟨	🟥	🟩
t_3 :	🟨	🟨	🟩
t_4 :	🟥	🟩	🟩

🟥 unsafe
🟨 transit
🟩 safe

Figure 4: Barrier function evolution and the corresponding states of the runtime monitor.

be defined as follows:

$$C_i(\mathbf{x}) = \begin{cases} G & \text{if } B^{(i)}(\mathbf{x}) < -K \\ Y & \text{if } -K \leq B^{(i)}(\mathbf{x}) < 0 \\ R & \text{otherwise} \end{cases}$$

The concept is illustrated in Figure 4. Now we can synthesize the runtime safety shield for the successive barriers. Assume that a nominal control law $\mathbf{u} = \kappa(t, \mathbf{x})$ is used to decide on the control input for a given state $\mathbf{x}$. For each of the barriers $B^{(i)}$, we will run the safety filter defined in section 3.3 using τ as the time period. If $C_i(\mathbf{x}) = G$, then the judgement is PASS, whereas if $C_i(\mathbf{x}) = Y$, the judgement is OVERRIDE($\mathbf{u}_i$) for some $\mathbf{u}_i \in U_{\text{fin}}$. Let $\varphi(t)$ be some trajectory and $\mathbf{x}_l = \varphi(l\tau)$ for some natural number l. We will now define the safety filter function.

Definition 4.1. Given successive barrier functions $B^{(1)}, \ldots, B^{(k)}$, the function SAFE-FILTER($\mathbf{x}_l; B^{(1)}, \ldots, B^{(k)}$) operates as follows:

rule-1: Suppose there is an index $j \in [1, k]$ such that $C_j(\mathbf{x}_l) = G$, then SAFE-FILTER($\mathbf{x}_l; B^{(1)}, \ldots, B^{(k)}$) is set to PASS, or equivalently, allows the nominal control law to be applied for the next τ time units.

rule-2: Otherwise, let $j \in [1, k]$ be the least yellow index: i.e, (a) $C_j(\mathbf{x}_l) = Y$, (b) for all $i < j$, $C_i(\mathbf{x}_l) = R$ and (c) for all $i > j$, $C_i(\mathbf{x}_l) \in \{R, Y\}$. We define SAFE-FILTER($\mathbf{x}_l; B^{(1)}, \ldots, B^{(k)}$) = SAFE-FILTER($\mathbf{x}_l; B^{(j)}$) = OVERRIDE($\mathbf{u}_j$), for some $\mathbf{u}_j \in U_{\text{fin}}$.

rule-3: Otherwise, all of the control barriers are in the red state. This state should never be reached since the trajectory may be potentially unsafe. SAFE-FILTER is undefined for this case.

Let $\varphi(t)$ be a trajectory obtained by composing nominal controller $\mathbf{u} = \kappa(t, \mathbf{x})$ and the SAFETY-FILTER($\mathbf{x}; B^{(1)}, \ldots, B^{(k)}$) executed with time period τ. We will first prove that if we are in a state $\mathbf{x}_l$ at time $t = l\tau$ such that $C_j(\mathbf{x}_l) \in \{G, Y\}$ for at least one index $j \in [1, k]$ (i.e, not all control barriers are red) then $C_i(\mathbf{x}_{l+1}) \in \{G, Y\}$ for some index i.

LEMMA 4.2. *Let $j \in [1, k]$ such that $C_j(\mathbf{x}_l) = G$. It follows that $C_j(\varphi(t)) \in \{G, Y\}$ for all $t \in [l\tau, (l+1)\tau]$.*

PROOF. This follows directly from Lemma 3.6. □

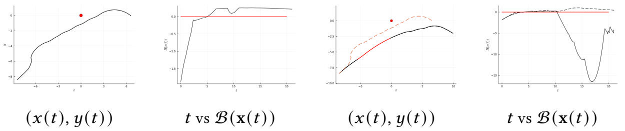

$(x(t), y(t))$ t vs $\mathcal{B}(\mathbf{x}(t))$ $(x(t), y(t))$ t vs $\mathcal{B}(\mathbf{x}(t))$

Figure 5: (Left) Nominal trajectory for a fixed (feedforward) control signal $u(t)$ for dynamics in Ex. 2.2. ((Right) Effect of implementing the safety filter from Figure 2. Dashed lines show the original behavior.

LEMMA 4.3. *If $C_j(\mathbf{x}_l) = Y$ and $C_i(\mathbf{x}_l) = R$ for all $i \in [1, j)$ then for all $t \in [l\tau, (l+1)\tau]$, there exists $i \leq j$ such that $C_i(\varphi(t)) \in \{G, Y\}$.*

PROOF. Consider two cases: (**case-1**) all lower indices remain "red" or "yellow" for the entire interval, i.e, $\forall\, t \in [l\tau, (l+1)\tau]$ and for all $1 \leq i < j$, we have $C_i(\varphi(t)) \in \{R, Y\}$, or (**case-2**) there exists a time instant $t \in [l\tau, (l+1)\tau]$ such that $C_i(\varphi(t)) = G$ for some $1 \leq i < j$. Note that if $j = 1$, then **case-1** must hold trivially since there are no lower indices.

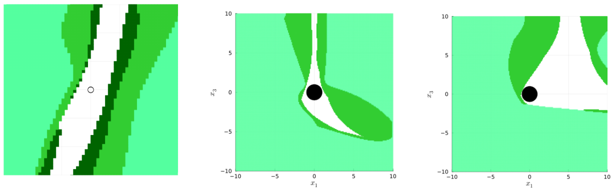
(**case-1**:) Let SAFETY-FILTER($\mathbf{x}_l, B^{(j)}$) = OVERRIDE($\mathbf{u}_j$) for $\mathbf{u}_j \in U_{\text{fin}}$. Suppose **case-1** holds, then we have that for some barrier $B_{i,j}$ satisfies the derivative condition $\frac{dB}{dt} \leq -\lambda B$ with control inputs fixed to $\mathbf{u}_j$ throughout the time interval $[l\tau, (l+1)\tau]$. Therefore, since $B_{i,j}(\mathbf{x}_l) < 0$, we have $B_{i,j}(\varphi(l\tau + t)) \leq e^{-\lambda t} B_{i,j}(\mathbf{x}_l) < 0$. Hence, $C_j(\varphi(t)) = Y$ for all $t \in [l\tau, (l+1)\tau]$ This establishes the result for **case**-1.

(**case-2**:) Let $B^{(i)}$ be such that $C_i(\varphi(t)) = G$ for some t in the time interval. There must exist a time instant $t^* \in [l\tau, (l+1)\tau]$ such that $B^{(i)}(\varphi(t^*)) = -K$ for some $i < j$ and $B^{(i)}((\varphi(t^* + \delta)) < -K$ for some $\delta > 0$. Let t^* be the infimum among all such time instants satisfying the property. Using the upward transit time bound, we note that $C_i(\varphi(t^* + \delta)) = G$ and the control input during the time interval $[l\tau, (l+1)\tau]$ is fixed to $\mathbf{u} = \mathbf{u}_j$. Therefore, the control input is continuous. Hence, we conclude that $C_i(\varphi(t)) \in \{G, Y\}$ for $t \in [t^*, (l+1)\tau]$ because $(l+1)\tau - t^* < \tau$ and $C_j(\varphi(t)) \in \{G, Y\}$ for $t \in [l\tau, t^*)$. □

THEOREM 4.4. *Starting from a state $\mathbf{x}_0$, wherein, $C_j(\mathbf{x}_0) \neq R$ for all $j \in [1, k]$ the system will never reach an unsafe state $x \in X_u$.*

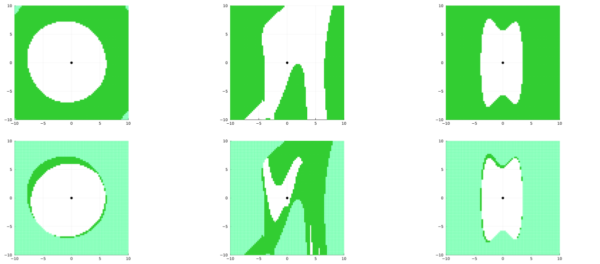
PROOF. Using Lemma 4.2 and 4.3 we conclude that the state where all $C_j(\varphi(t)) \neq R$ for all $j \in [1, k]$ cannot be reached in any trajectory φ starting from $\mathbf{x}_0$. Since for some j, $C_j(\varphi(t)) \in \{G, Y\}$, it follows that $\varphi(t) \notin X_u$. □

Example 4.5. Figure 5 shows a nominal trajectory $(x(t), y(t))$ for a fixed control input signal $u(t)$ along with a plot of the function $\mathcal{B}(\mathbf{x}(t))$ over time, using the dynamics from Ex. 2.2. The portion of the trajectory that enters the unsafe region overapproximation is shaded red. This also corresponds to the part where $\mathcal{B}(\mathbf{x}) \geq -K$. On the other hand, we implement the safety filter. We periodically sampled states with a period $\tau = 0.02$. Notice that the override happens with $\mathcal{B}(\mathbf{x}) \geq -K$ where K is set to 1 in our implementation. After the override, the value of $\mathcal{B}(\mathbf{x})$ falls under the action of the control input corresponding to the control barrier function.



(a) **(b)** **(c)**

Figure 6: Improvement in approximation of the control invariant set through successive barrier functions. (a) approximation for the 2D example from section 5.1 with three iterations of successive barrier functions; (b) two iterations of successive barrier functions for 4D example from Section 5.2 with $x_2 = x_4 = 0$; (c) $x_2 = x_4 = -5$.



(a) **poly1** **(b)** **van der Pol** **(c)** **inv pendulum**

Figure 7: Extension of the controlled invariant set through successive barrier functions for different dynamical systems.

5 Empirical Evaluation

In this section, we will illustrate our approach with a few numerical examples. We demonstrate that the approach of computing successive barrier functions yields increasingly tighter approximations of the unsafe region.[2]

5.1 2D Nonlinear Dynamics

Consider a nonlinear system with two state variables (x_1, x_2) and two control inputs $(u_1, u_2) \in [-0.05, 0.05]^2$.

$$\begin{aligned} \dot{x}_1 &= \frac{1}{2}x_1 - \frac{1}{5}x_2 - \frac{1}{100}x_1x_2 - \frac{1}{2}u_1 + \frac{1}{2}u_2, \\ \dot{x}_2 &= x_1 - \frac{2}{5}x_2 - \frac{1}{20}x_2^2 - \frac{7}{10}u_2 + \frac{1}{10}u_1, \end{aligned}$$

We take $U_{fin} = \{(u_1, u_2) \mid u_1 = \pm 0.05, u_2 = \pm 0.05\}$ and generate 500 randomly generated test points. The unsafe set is taken to be $X_u = \{(x_1, x_2) | x_1^2 + x_2^2 \leq 0.1\}$. We computed three levels of successive barrier functions by applying Algorithm 2. The approach managed to eliminate 490 test points from the unsafe region. $B^{(1)}$ is obtained as the minimum of two polynomial barrier functions, $B^{(2)}$ is obtained as a minimum of 9 functions and $B^{(3)}$ consists of 5 functions. Figure 6(a) shows the overall control invariant region obtained by our approach. Synthesizing the first level barriers required 160 seconds of CPU time, the second level barrier functions required 256 seconds of CPU time and the third level required 344 seconds. All timings were measured on a MAC OSX laptop running 3.1 GHz Quad-Core Intel Core i7 and 16GB RAM.

5.2 4D Nonlinear Dynamics

We consider an example of a polynomial system with 4 state variables and 2 control inputs $u_1, u_2 \in [-0.1, 0.1]^2$.

$$\begin{aligned} \dot{x}_1 &= x_2, \quad \dot{x}_2 = \frac{1}{2}x_1 - \frac{1}{5}x_2 + \frac{1}{20}x_3x_1 - \frac{1}{100}x_1x_2 - \frac{1}{2}u_1, \\ \dot{x}_3 &= x_4, \quad \dot{x}_4 = -\frac{2}{5}x_4 + \frac{1}{5}x_1 - \frac{1}{20}x_3^2 - \frac{7}{10}u_2, \end{aligned}$$

The unsafe set is taken to be $X_u = \{(x_1, x_2) | x_1^2 + x_2^2 \leq 1.0\}$. The set U_{fin} was set to the vertices of $U : [-0.1, 0.1]^2$ and we chose 200 randomly selected test states. We computed two levels of successive barrier functions by applying Algorithm 2. The barrier functions synthesized eliminated 190 test points from the unsafe set. $B^{(1)}$ is obtained as the minimum of 4, and $B^{(2)}$ as a minimum of 34 polynomial functions. The minimum transit time was 1.76. Figure 6(b,c) show the control invariant set thus obtained. Synthesizing the first level barriers required 170 seconds of CPU time, and the second level barrier functions required 1010 seconds of CPU time.

5.3 5D Dynamics (Coordinated Turn Model)

We now consider a coordinated turn model with 5 state variables and 2 control inputs $u_1, u_2 \in [-5, 5]^2$.

$$\begin{aligned} \dot{x}_1 &= x_3 \cos x_4, & \dot{x}_4 &= x_5, \\ \dot{x}_2 &= x_3 \sin x_4, & \dot{x}_5 &= u_2, \\ \dot{x}_3 &= u_1, \end{aligned}$$

The unsafe set is taken to be $X_u = \{(x_1, x_2) | x_1^2 + x_2^2 \leq 0.1\}$. The set U_{fin} was set to the vertices of $U : [-5, 5]^2$ and we chose 500 randomly selected test states. We computed three levels of successive barrier functions by applying Algorithm 2. $B^{(1)}$ is obtained as the minimum of four polynomial barrier functions, $B^{(2)}$ is obtained as a minimum of 15, and $B^{(3)}$ is obtained as a minimum of 12 polynomial functions. Figure 8(a,b,c) show the resulting control invariant sets obtained by our approach.

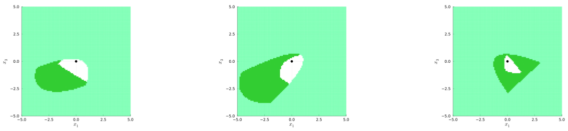
We use a polynomial approximation of trigonometric functions. We also model the error between the piecewise affine approximation and the trigonometric function as a disturbance input whose values are appropriately bounded. Details of the polynomial functions and error bounds are available as part of the supplementary material.

5.4 6D Nonlinear Dynamics (Planar Multirotor)

We now consider a planar multirotor model with 6 state variables and 2 control inputs $u_1, u_2 \in [0, 2]^2$.

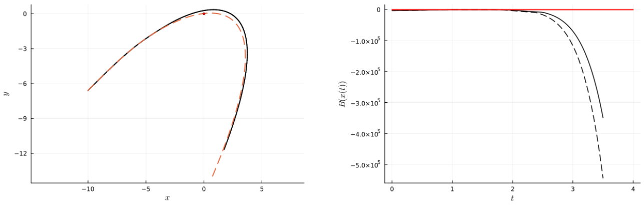
$$\begin{aligned} \dot{x}_1 &= x_3, & \dot{x}_4 &= (u_1 + u_2)\cos x_5 - 2, \\ \dot{x}_2 &= x_4, & \dot{x}_5 &= x_6, \\ \dot{x}_3 &= (u_1 + u_2)\sin x_5, & \dot{x}_6 &= u_1 - u_2, \end{aligned}$$

[2] The implementation and resulting barrier functions are available at https://github.com/rameezw/SuccessiveBarriers.



(a) (b) (c)

Figure 8: Control invariant set improvement through three iterations of successive barrier functions for 5D coordinated turn model from Section 5.3 with (a)$x_3 = 4, x_4 = 0, x_5 = 3$; (b)$x_3 = 4, x_4 = 1, x_5 = 3$; (c) $x_2 = x_4 = -5$.



$(x(t), y(t))$ t vs $\mathcal{B}(\mathbf{x}(t))$

Figure 9: Effect of implementing the safety filter from Figure 2 for a fixed (feedforward) control signal $u(t)$ for dynamics in Ex. 5.4. Dashed lines show the original behavior.

The unsafe set is taken to be $X_u = \{(x_1, x_2) | x_1^2 + x_2^2 \leq 0.1\}$. The set U_{fin} was set to the vertices of $U : [0, 2]^2$ and we chose 900 randomly selected test states. We computed three levels of successive barrier functions by applying Algorithm 2. $B^{(1)}$ is obtained as the minimum of four polynomial barrier functions, $B^{(2)}$ is obtained as a minimum of 19 polynomial functions, and $B^{(3)}$ is obtained as a minimum of 33 polynomial functions.

5.5 Certifying Sum-Of-Square (SOS) Programs

We have, thus far, relied exclusively on using SOS programming to synthesize successive barrier functions. SOS programming uses semi-definite programming solvers to synthesize functions $B(\mathbf{x})$. In order to certify that B satisfies the conditions for being a barrier/successive barrier, we need to verify that the results are not invalidated due to numerical issues [34]. Note that each barrier consists of multiple entailment relations of the form:

$$p_1(\mathbf{x}) \geq 0, \ldots, p_m(\mathbf{x}) \geq 0 \ \models \ p \geq 0 \,.$$

SOS programming certifies this through a Putinar positivstellensatz proof that states that

$$\exists\, \sigma_1, \ldots, \sigma_m \in \mathrm{SOS}_d[\mathbf{x}]\ p - \sigma_1 p_1 - \cdots - \sigma_m p_m \in \mathrm{SOS}_d[\mathbf{x}] \,,$$

wherein $\mathrm{SOS}_d[\mathbf{x}]$ represents the set of all SOS polynomials over $\mathbf{x}$ of degree at most d, and d is a parameter chosen by the user/SOS modeling package [19].

We verified the results of the SOS programming package used (SumOfSquares.jl) by outputting the polynomials $\sigma_1, \ldots, \sigma_m$ and computing the "residue" $p - \sigma_1 p_1 - \cdots - \sigma_m p_m$. In each case, we obtain a representation $\sigma_i = m(\mathbf{x})^\top Q_i m(\mathbf{x})$, wherein $m(\mathbf{x})$ is a monomial basis whose entries consist of monomials of degree $\leq \frac{d}{2}$ and Q_i is a $|m(\mathbf{x})| \times |m(\mathbf{x})|$ matrix. We verify that Q_i is positive semi-definite by computing its Cholesky decomposition. The C++ library Eigen was used for this purpose: we used 512 bit floating point representation to carry out the Cholesky decomposition.

Using this process, we verified that all the SOS programming results obtained in the process of generating the successive barrier function were verified using high precision floating point arithmetic. In the future, we will improve this process by enabling verification through rational arithmetic: a process that cannot work for Cholesky decomposition since it involves taking square roots. We will also explore the use of interval arithmetic approaches to certify positive semi-definiteness along the lines of Roux et al [34].

5.6 Comparison with FOSSIL

FOSSIL[13] is a tool for computing verified neural certificates. A comparison showing the effectiveness of our approach for barrier synthesis was conducted for multiple dynamical systems. The results are displayed in Table 1. The neural certificate was trained on a 2-layer network having 10 neurons each, with sigmoid and square activations, respectively; and was verified using dReal. In these evaluations, a timeout limit was set at 10^4 seconds, beyond which the experiment was considered a failure. The comparisons show that although FOSSIL swiftly computes barriers for low-dimensional systems, it does not terminate for high dimensions. On the contrary, our SOS-based successive barriers approach, although takes some time is successful in computing the barrier functions for multiple levels of the hierarchy, even for higher dimensions.

Another comparison was the size of the controlled invariant set computed using the two approaches. The results are shown in Table 2. The comparison for done over 1000 test points sampled using a fixed seed over the same domains. It turns out that the successive barriers approach improves the size of the safe sets defined by the neural CBFs computed using FOSSIL. Hence, using successive barrier functions results in less conservative approximations of the controlled invariant set.

6 Conclusion

We develop the concept of successive control barrier functions for tight over-approximations of unsafe sets for nonlinear systems. A runtime monitor is also presented which utilizes a shield based on the successive barriers to ensure collision avoidance. The switching strategy between the successive barriers respects minimum transit time constraints. For future work, we plan on improving the algorithm to include multiple fixed and moving obstacles, along with more challenging system dynamics.

Acknowledgments

We thank the anonymous reviewer for detailed comments on this submission. This work was funded, in part, by the US National Science Foundation (NSF) under award number CNS-1836900. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table 1: Barrier synthesis comparison

			FOSSIL		Ours					
system	dim	inputs	success	time(s)	success	time(s):$B^{(1)}$	# barriers	max degree	time(s):$B^{(2)}$	# barriers
poly1	2	1	✓	3.2	✓	**0.6**	2	2	34.1	1
poly2	2	1	✓	1.9	✓	**1.0**	2	2	36.9	3
van der Pol	2	1	✓	4.8	✓	**2.0**	2	2	247.4	7
inv pendulum	2	1	✓	3.2	✓	**2.0**	2	2	207.3	1
poly3	2	2	✓	**1.3**	✓	7.6	4	4	194.1	12
poly4	3	2	✓	74.8	✓	**6.8**	4	4	281.2	8
poly5	4	2	✗	-	✓	**36.4**	4	4	587.2	27
coord turn	5	2	✗	-	✓	**108.4**	4	4	2290.8	15
planar multirotor	6	2	✗	-	✓	**131.6**	4	4	3305.5	9

Table 2: Controlled invariant set comparison for 1000 test points. S denotes points in the safe region(inside the control invariant set) and U denotes points in the unsafe region(outside the control invariant set)

system	dimensions	inputs	FOSSIL(S)	Ours(S)	FOSSIL(U) & Ours(S)	Ours(U) & FOSSIL(S)
poly1	2	1	417	**649**	**375**	143
poly2	2	1	716	**922**	**215**	9
van der Pol	2	1	507	**649**	**322**	180
inv pendulum	2	1	683	**798**	**193**	78
poly3	2	2	556	**977**	**421**	0

References

[1] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*. IEEE, 3420–3431.

[2] Kostas E Bekris. 2010. Avoiding inevitable collision states: Safety and computational efficiency in replanning with sampling-based algorithms. In *Workshop on Guaranteeing Safe Navigation in Dynamic Environments. In: International Conference on Robotics and Automation (ICRA-10)*.

[3] Kostas E Bekris, Konstantinos I Tsianos, and Lydia E Kavraki. 2009. Safe and distributed kinodynamic replanning for vehicular networks. *Mobile Networks and Applications* 14 (2009), 292–308.

[4] Guillaume Berger, Masoumeh Ghanbarpour, and Sriram Sankaranarayanan. 2024. Cone-Based Abstract Interpretation for Nonlinear Positive Invariant Synthesis. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 1–16.

[5] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[6] Joseph Breeden and Dimitra Panagou. 2023. Compositions of multiple control barrier functions under input constraints. In *2023 American Control Conference (ACC)*. IEEE, 3688–3695.

[7] Joseph Breeden and Dimitra Panagou. 2023. Safety-critical control for systems with impulsive actuators and dwell time constraints. *IEEE Control Systems Letters* 7 (2023), 2119–2124.

[8] Joseph Breeden, Luca Zaccarian, and Dimitra Panagou. 2024. Robust safety-critical control for systems with sporadic measurements and dwell time constraints. *IEEE Control Systems Letters* (2024).

[9] Mo Chen, Sylvia L Herbert, Haimin Hu, Ye Pu, Jaime Fernandez Fisac, Somil Bansal, SooJean Han, and Claire J Tomlin. 2021. Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking. *IEEE Trans. Automat. Control* 66, 12 (2021), 5861–5876.

[10] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. 2021. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 6814–6821.

[11] Frank H. Clarke. 1983. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York.

[12] Jyotirmoy V Deshmukh and Sriram Sankaranarayanan. 2019. Formal techniques for verification and testing of cyber-physical systems. *Design Automation of Cyber-Physical Systems* (2019), 69–105.

[13] Alec Edwards, Andrea Peruffo, and Alessandro Abate. 2024. Fossil 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*. 1–10.

[14] Thierry Fraichard and Hajime Asama. 2004. Inevitable collision states—a step towards safer robots? *Advanced Robotics* 18, 10 (2004), 1001–1024.

[15] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. 2017. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters* 1, 2 (2017), 310–315.

[16] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. 2018. Boolean composability of constraints and control synthesis for multi-robot systems via nonsmooth control barrier functions. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 897–902.

[17] Hui Kong, Fei He, Xiaoyu Song, William NN Hung, and Ming Gu. 2013. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 242–257.

[18] Jean B Lasserre. 2001. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization* 11, 3 (2001), 796–817.

[19] Benoît Legat, Chris Coey, Robin Deits, Joey Huchette, and Amelia Perry. 2017. Sum-of-squares optimization in Julia. In *The First Annual JuMP-dev Workshop*.

[20] Jaeyoung Lim, Florian Achermann, Rik Girod, Nicholas Lawrance, and Roland Siegwart. 2024. Safe Low-Altitude Navigation in Steep Terrain with Fixed-Wing Aerial Vehicles. *IEEE Robotics and Automation Letters* (2024).

[21] Lars Lindemann and Dimos V Dimarogonas. 2018. Control barrier functions for signal temporal logic tasks. *IEEE control systems letters* 3, 1 (2018), 96–101.

[22] Simin Liu, Changliu Liu, and John Dolan. 2023. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*. PMLR, 1970–1980.

[23] Aniketh Manjunath and Quan Nguyen. 2021. Safe and robust motion planning for dynamic robotics via control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2122–2128.

[24] Luis Martinez-Gomez. 2010. *Safe navigation for autonomous vehicles in dynamic environments: an Inevitable Collision State (ICS) perspective.* Ph. D. Dissertation. Université de Grenoble.

[25] Luis Martinez-Gomez and Thierry Fraichard. 2008. An efficient and generic 2d inevitable collision state-checker. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 234–241.

[26] Luis Martinez-Gomez and Thierry Fraichard. 2009. Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, 100–105.

[27] Quan Nguyen and Koushil Sreenath. 2016. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*. IEEE, 322–328.

[28] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew M Peet, and Declan Jagt. 2013. SOSTOOLS version 4.00 sum of squares optimization toolbox for MATLAB. *arXiv preprint arXiv:1310.4716* (2013).

[29] Pablo A Parrilo. 2003. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming* 96 (2003), 293–320.

[30] Pablo A. Parrilo. 2012. *Polynomial Optimization, Sums of Squares, and Applications*. SIAM, Chapter 3, 47–157. doi:10.1137/1.9781611972290.ch3

[31] Christian Pek and Matthias Althoff. 2018. Efficient computation of invariably safe states for motion planning of self-driving vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3523–3530.

[32] Stephen Prajna and Ali Jadbabaie. 2004. Safety Verification of Hybrid Systems Using Barrier Certificates. In *Hybrid Systems: Computation and Control*, Rajeev Alur and George J. Pappas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 477–492.

[33] Manuel Rauscher, Melanie Kimmel, and Sandra Hirche. 2016. Constrained robot control using control barrier functions. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 279–285.

[34] Pierre Roux, Yuen-Lam Voronin, and Sriram Sankaranarayanan. 2018. Validating numerical semidefinite programming solvers for polynomial invariants. *Formal Methods Syst. Des.* 53, 2 (2018), 286–312. doi:10.1007/S10703-017-0302-Y

[35] Naum Z. Shor. 1987. An Approach to Obtaining Global Extrema in Polynomial Problems of Mathematical Programming. *Kibernetika (Kiev)* 5 (1987), 102–6. Issue 136.

[36] W. Tan and A. Packard. 2007. Stability Region Analysis using Sum of Squares Programming. In *Proc. ACC*.

[37] Xiao Tan, Wenceslao Shaw Cortez, and Dimos V Dimarogonas. 2021. High-order barrier functions: Robustness, safety, and performance-critical control. *IEEE Trans. Automat. Control* 67, 6 (2021), 3021–3028.

[38] Li Wang, Dongkun Han, and Magnus Egerstedt. 2018. Permissive barrier certificates for safe stabilization using sum-of-squares. In *2018 Annual American Control Conference (ACC)*. IEEE, 585–590.

[39] Tillmann Weisser, Benoît Legat, Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. 2019. Polynomial and Moment Optimization in Julia and JuMP. In *JuliaCon*. https://pretalx.com/juliacon2019/talk/QZBKAU/

[40] Peter Wieland and Frank Allgöwer. 2007. Constructive safety using control barrier functions. *IFAC Proceedings Volumes* 40, 12 (2007), 462–467.

[41] Wei Xiao and Calin Belta. 2021. High-order control barrier functions. *IEEE Trans. Automat. Control* 67, 7 (2021), 3655–3662.

[42] Wei Xiao, Calin A Belta, and Christos G Cassandras. 2020. Feasibility-guided learning for constrained optimal control problems. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 1896–1901.

[43] Xiangru Xu. 2018. Constrained control of input–output linearizable systems using control sharing barrier functions. *Automatica* 87 (2018), 195–201.

[44] Xiangru Xu, Paulo Tabuada, Jessy W Grizzle, and Aaron D Ames. 2015. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine* 48, 27 (2015), 54–61.

[45] Xia Zeng, Wang Lin, Zhengfeng Yang, Xin Chen, and Lilei Wang. 2016. Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In *Proceedings of the 13th International Conference on Embedded Software*. 1–10.

[46] Songyuan Zhang, Kunal Garg, and Chuchu Fan. 2023. Neural graph control barrier functions guided distributed collision-avoidance multi-agent control. In *Conference on Robot Learning*. PMLR, 2373–2392.