

PRoTECT: Parallelized Construction of Safety Barrier Certificates for Nonlinear Polynomial Systems

BEN WOODING, VIACHESLAV HORBANOV, AND ABOLFAZL LAVAEI

SCHOOL OF COMPUTING, NEWCASTLE UNIVERSITY, UNITED KINGDOM

{BEN.WOODING,V.HORBANOV2,ABOLFAZL.LAVAEI}@NEWCASTLE.AC.UK

ABSTRACT. We develop an open-source software tool, called PRoTECT, for the parallelized construction of safety barrier certificates (BCs) for nonlinear polynomial systems. This tool employs *sum-of-squares (SOS) optimization* programs to systematically search for polynomial-type BCs, while aiming to verify safety properties over *four classes of dynamical systems*: (i) discrete-time *stochastic* systems, (ii) discrete-time *deterministic* systems, (iii) *continuous-time* stochastic systems, and (iv) *continuous-time* deterministic systems. PRoTECT is implemented in Python as an application programming interface (API), offering users the flexibility to interact either through its user-friendly graphic user interface (GUI) or via function calls from other Python programs. PRoTECT leverages *parallelism* across different barrier degrees to efficiently search for a feasible BC.

Keywords: Barrier certificates, safety verification, sum-of-squares optimization program, nonlinear polynomial systems, stochastic systems

1. INTRODUCTION

1.0.1. *Motivation for PRoTECT.* Formal verification of dynamical systems has become a focal point over the past several years, primarily due to their widespread integration into safety-critical systems [13]. These systems, whose malfunction may lead to severe consequences such as loss of life, injuries, or substantial financial losses, are integral across a broad spectrum of domains including robotics, transportation systems, energy, and healthcare. When confronted with a property of interest for a dynamical model, formal verification aims to reliably assess whether the desired specification is fulfilled. If the model exhibits stochastic behavior, the objective shifts to formally quantifying the satisfaction probability of the desired property.

Barrier certificates (BCs), also known as *barrier functions*, have emerged as a fundamental solution approach, offering assurances regarding the safety behavior of diverse classes of systems. Specifically, BCs can be employed to directly assess the behavior of systems across *continuous-state spaces* with an uncountable number of states, without resorting to discretization, which contrasts with abstraction-based approaches [18]. This aspect is particularly noteworthy when considering *safety*, (*a.k.a. invariance*) properties, wherein the state transitions of the system remain within a region labeled as “safe”, ensuring no transitions occur to any region labeled “unsafe”. In particular, barrier certificates, akin to Lyapunov functions, are functions established over the system’s state space, fulfilling specific inequalities concerning both the function itself and the one-step transition (or the flow) of the system. A suitable level set of a BC can segregate an unsafe region from all system trajectories originating from a specified set of initial conditions. Hence, the presence of such a function offers a formal (probabilistic) certification for system safety.

1.0.2. *Related Literature on BCs.* Barrier certificates, initially introduced in [25, 26], have gained significant recognition for the formal safety analysis of dynamical systems over the past 20 years. These approaches can verify systems with polynomial dynamics, enabling the design of polynomial BCs using sum-of-squares techniques, *e.g.*, via SOSTOOLS [27]. Additionally, some alternative approaches explore verifying nonpolynomial systems, such as those discovered through counter-example guided inductive synthesis (CEGIS), leveraging satisfiability modulo theories (SMT) solvers including Z3 [9], dReal [11], or MathSat [8]. Other new techniques encompass neural barrier functions [35, 20] and genetic programs [32]. While we primarily focus on utilizing BCs for *safety* specifications in PROTECT, they also hold significant value for addressing other specifications, including reachability, reach-while-avoid, and other temporal logic specifications [5, 19]. A comprehensive overview of barrier certificates in both deterministic and stochastic settings can be found in the line of work [4, 33, 21, 22, 16, 12].

1.0.3. *Related Software Tools.* To the best of our knowledge, there exists only one other tool dedicated to the construction of barrier certificates, FOSSIL [1]. In the latest release [10], FOSSIL has expanded its capabilities to include finding barrier certificates for discrete- and continuous-time *deterministic* systems using the CEGIS approach, while facilitating verification and control synthesis for specifications including safety, reachability, and reach-while-avoid. However, FOSSIL lacks support for *stochastic* systems, whereas in PROTECT, we provide support for both discrete- and

continuous-time *stochastic* systems. Additionally, while FOSSIL can handle nonpolynomial BCs, it does not guarantee termination due to its use of CEGIS approach. In contrast, PRoTECT utilizes sum-of-squares optimization techniques, enabling the exploitation of *parallelism* to concurrently search for multiple candidate BCs with different degrees, aiming to construct them effectively.

There have also been some tools that concentrate on *finite abstraction* methods, an approach capable of providing safety guarantees but hampered by the curse of dimensionality, stemming from the necessity to grid the state/input space. Typically, these tools focus on only one or two cases of dynamical systems (*e.g.*, SCOTS [30] or AMYTISS [17]); however, in PRoTECT we consider *four* classes of dynamical systems.

1.0.4. *Original Contributions.* The primary contributions and noteworthy aspects of our tool paper are as follows:

- (i) We propose the first tool, employing SOS optimization techniques, that verifies the safe behavior of *four classes of dynamical systems*: (i) discrete-time *stochastic* systems (dt-SS), (ii) discrete-time *deterministic* systems (dt-DS), (iii) *continuous-time* stochastic systems (ct-SS), and (iv) *continuous-time* deterministic systems (ct-DS).
- (ii) PRoTECT is implemented in Python using SumOfSquares [34], and leverages *parallelization* to efficiently search for BCs of different degrees, aiming to satisfy the desired safety specifications.
- (iii) PRoTECT supports *normal*, *uniform*, and *exponential* noise distributions for dt-SS, as well as *Brownian motion* and *Poisson processes* for ct-SS.
- (iv) In stochastic settings, PRoTECT automatically optimizes the safety confidence level while designing different decision variables, including level sets of BCs.
- (v) PRoTECT offers advanced GUIs for all four classes of models, enhancing the tool’s accessibility and user-friendliness.
- (vi) We utilize PRoTECT across various real-world applications, including room temperature systems, a jet engine, a DC motor, a Van der Pol oscillator, a model of two fluid tanks, and an 8-dimensional system to showcase the scalability. This expansion broadens the applicability of formal method techniques to encompass safety-critical applications across different model classes. The results highlight significant improvements in computational efficiency.

The source code for PRoTECT, along with detailed guidelines on installation and usage, including tutorial videos, are available at:

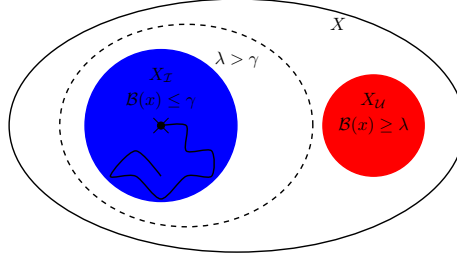


FIGURE 1. A barrier certificate $\mathcal{B}(x)$ for a dynamical system. The dashed line denotes the initial level set $\mathcal{B}(x) = \gamma$.

<https://github.com/Kiguli/PRoTECT>

2. PROBLEM DESCRIPTION

2.0.1. Preliminaries and Notation. The set of real numbers, non-negative and positive real numbers is denoted by $\mathbb{R}$, $\mathbb{R}_0^+$, and $\mathbb{R}^+$, respectively. The set of natural numbers including and excluding zero is represented by $\mathbb{N}$ and $\mathbb{N}^+$. The empty set is denoted by $\emptyset$. For any matrix $A \in \mathbb{R}^{n \times n}$, $\text{Tr}(A)$ represents the trace of A which is the sum of all its diagonal elements. For a system Σ and a property ϕ , the notation $\Sigma \models \phi$ signifies that Σ fulfills the property ϕ . We consider a probability space $(\Omega, \mathcal{F}_\Omega, \mathbb{P}_\Omega)$, where Ω represents the sample space, $\mathcal{F}_\Omega$ denotes the sigma-algebra of Ω comprising events, and $\mathbb{P}_\Omega$ signifies the probability measure assigning probabilities to each event. A topological space S is termed a Borel space when equipped with a metric that renders it a separable and complete metrizable space. Subsequently, S is furnished with a Borel sigma-algebra denoted as $\mathcal{B}(S)$. For continuous-time stochastic systems, we assume that triple $(\Omega, \mathcal{F}_\Omega, \mathbb{P}_\Omega)$ is equipped with a filtration $\mathbb{F} = (\mathcal{F}_s)_{s \geq 0}$ adhering to the standard conditions of completeness and right continuity. Consider $(\mathbb{W}_s)_{s \geq 0}$ as a b -dimensional $\mathbb{F}$ -Brownian motion, and $(\mathbb{P}_s)_{s \geq 0}$ as an r -dimensional $\mathbb{F}$ -Poisson process. We posit independence between the Poisson process and Brownian motion. Poisson process $\mathbb{P}_s = [\mathbb{P}_s^1; \dots; \mathbb{P}_s^r]$ represents r events, each assumed to occur independently of the others.

2.0.2. Safety Barrier Certificates. Consider a state set X in an n -dimensional space, denoted as $X \subseteq \mathbb{R}^n$. Within this set, we identify two specific subsets: X_I and X_U , which represent the *initial* and *unsafe* sets, respectively. The primary objective is to construct a function $\mathcal{B}(x)$, termed the *barrier certificate*, along with constants γ and λ as the *initial* and *unsafe* level sets of $\mathcal{B}(x)$, as

illustrated in Fig. 1. Specifically, the design of BC incorporates two conditions concerning these level sets, in conjunction with a third criterion that captures the *state evolution* of the system. Collectively, satisfaction of conditions provides a (probabilistic) guarantee that the system's trajectories, originating from any initial condition $x_0 \in X_{\mathcal{I}}$, will not transition into the unsafe region $X_{\mathcal{U}}$.

We now formally introduce the safety specification that we aim to investigate in this work.

Definition 1 (Safety). *A safety specification is defined as $\varphi = (X_{\mathcal{I}}, X_{\mathcal{U}}, \mathcal{T})$, where $X_{\mathcal{I}}, X_{\mathcal{U}} \subseteq X$ with $X_{\mathcal{I}} \cap X_{\mathcal{U}} = \emptyset$, and $\mathcal{T} \in \mathbb{N} \cup \{\infty\}$. A dynamical system Σ is considered safe over an (in)finite time horizon $\mathcal{T}$, denoted as $\Sigma \models_{\mathcal{T}} \varphi$ if all trajectories of Σ starting from the initial set $X_{\mathcal{I}}$ never reach the unsafe set $X_{\mathcal{U}}$. If trajectories are probabilistic, the primary goal is to compute $\mathbb{P}\{\Sigma \models_{\mathcal{T}} \varphi\} \geq \phi$, with $\phi \in [0, 1]$.*

2.0.3. Overview of PROTECT. PROTECT offers functionalities that automatically generate BCs and verify the safety property across *four distinct classes of systems*. The description of the system serves as an input to the tool, triggering the appropriate function. These functions are named as **dt-SS**, **dt-DS**, **ct-SS** and **ct-DS**. Additionally, the geometric characteristics for sets of interest, which define the safety specification according to Definition 1, constitute another input to the tool. While a GUI is designed to enhance the user-friendliness of PROTECT, the BCs may also be verified via configuration files executed through the command line. As the output, PROTECT returns BC $\mathcal{B}(x)$, level sets γ and λ , and, for stochastic systems, the value c and the confidence level ϕ (cf. Section 3).

Utilizing methodologies from the *sum-of-squares* (SOS) domain, facilitated by the **SumOfSquares** Python toolbox [34], PROTECT adopts polynomial structures for BCs expressed as $\mathcal{B}(x) = \sum_{j=1}^z q_j p_j(x)$, with basis functions $p_j(x)$ that are monomials over x , and unknown coefficients $q = [q_0, \dots, q_z] \in \mathbb{R}^z$ that need to be designed. PROTECT leverages parallelization techniques to facilitate the *simultaneous* verification of multiple BCs, differentiating them based on their polynomial degrees, where degrees must be even [29]. In deterministic systems, upon finding a feasible BC, the parallel processing is terminated and the valid BC is returned to the user. Conversely, for stochastic systems, PROTECT awaits until all potential solutions are fully processed, subsequently selecting and returning the BC that offers the highest probabilistic confidence. This process is detailed in the provided pseudo-code, illustrated in Algorithm 1.

Algorithm 1: *Parallel Construction of BCs*

Input: system Σ , maximum polynomial degree P , *required* parameters K_{req} , *optional* parameters K_{opt}

```

1 temp = [];
2 choose function func for  $\Sigma$  to identify the class of system;
3 forall  $p \in \{2, 4, \dots, P\}$  in parallel do
4     barrier = func( $p, K_{req}, K_{opt}$ );
5     if barrier is SOS then
6         temp.append(barrier);
7         if  $\Sigma$  is deterministic then
8             // terminate all parallel processes
9         end
10    end
11 end

// return element with highest confidence in temp
11  $\mathcal{B}(x) = \max(\text{temp})$ ;

Output: barrier certificate  $\mathcal{B}(x)$ , level sets  $\gamma$ ,  $\lambda$ ; confidence  $\phi$  and constant  $c$  (for dt-SS and ct-SS)
    
```

3. DISCRETE-TIME STOCHASTIC SYSTEMS

In this section, we define the notion of barrier certificates for discrete-time stochastic systems (dt-SS). A dt-SS is a tuple $\Sigma_d^\varsigma = (X, \varsigma, f)$, where: $X \subseteq \mathbb{R}^n$ is a Borel space as the state set, ς is a sequence of independent and identically distributed (i.i.d.) random variables from a sample space Ω to a measurable set $\mathcal{V}_\varsigma$, i.e., $\varsigma := \{\varsigma(k) : \Omega \rightarrow \mathcal{V}_\varsigma, k \in \mathbb{N}\}$, and $f : X \times \mathcal{V}_\varsigma \rightarrow X$ is a measurable function characterizing the *state evolution* of the system. For a given initial state $x(0) \in X$, the state evolution of Σ_d^ς is characterized by

$$\Sigma_d^\varsigma: x(k+1) = f(x(k), \varsigma(k)), \quad k \in \mathbb{N}. \quad (1)$$

The stochastic process $x_{x_0} : \Omega \times \mathbb{N} \rightarrow X$, which fulfills (1) for any initial state $x_0 \in X$ is referred to as the *solution process* of dt-SS at time $k \in \mathbb{N}$. PROTECT accommodates *additive noise* types across

a range of distributions, including *uniform*, *normal*, and *exponential* distributions. The notion of barrier certificates for dt-SS is provided by the subsequent definition [26].

Definition 2 (BC for dt-SS). *Consider the dt-SS $\Sigma_d^\varsigma = (X, \varsigma, f)$ and $X_{\mathcal{I}}, X_{\mathcal{U}} \subseteq X$. A function $\mathcal{B} : X \rightarrow \mathbb{R}_0^+$ is known as the barrier certificate (BC), if there exists constants $\lambda, \gamma, c \in \mathbb{R}_0^+$, with $\lambda > \gamma$, such that*

$$\mathcal{B}(x) \leq \gamma, \quad \forall x \in X_{\mathcal{I}}, \quad (2)$$

$$\mathcal{B}(x) \geq \lambda, \quad \forall x \in X_{\mathcal{U}}, \quad (3)$$

$$\mathbb{E}[\mathcal{B}(f(x, \varsigma)) \mid x] \leq \mathcal{B}(x) + c, \quad \forall x \in X, \quad (4)$$

where $\mathbb{E}$ denotes the expected value of the system's one-step transition, taken with respect to ς .

We now leverage the BC in Definition 2 and quantify a lower bound confidence over the safety of dt-SS [15, 26].

Lemma 1 (Confidence ϕ). *For dt-SS Σ_d^ς , let there exist a BC as in Definition 2. Then the probability that trajectories of dt-SS starting from any initial condition $x_0 \in X_{\mathcal{I}}$ will not reach the unsafe region $X_{\mathcal{U}}$ within a finite time horizon $k \in [0, \mathcal{T}]$ is quantified as*

$$\phi = \mathbb{P}\left\{x_{x_0}(k) \notin X_{\mathcal{U}} \text{ for all } k \in [0, \mathcal{T}] \mid x_0 = x(0)\right\} \geq 1 - \frac{\gamma + c\mathcal{T}}{\lambda}. \quad (5)$$

Under the assumption that f is a polynomial function of state x and sets $X_{\mathcal{I}}, X_{\mathcal{U}}, X$ are semi-algebraic sets, *i.e.* they can be represented by polynomial inequalities, we now describe how to reformulate (2)-(4) as an SOS optimization program to search for a polynomial-type BC [26, Prop. 18].

Lemma 2. *Let sets $X_{\mathcal{I}}, X_{\mathcal{U}}, X$ be defined element-wise by vectors of polynomial inequalities $X_{\mathcal{I}} = \{x \in \mathbb{R}^n \mid g_i(x) \geq 0\}$, $X_{\mathcal{U}} = \{x \in \mathbb{R}^n \mid g_u(x) \geq 0\}$, and $X = \{x \in \mathbb{R}^n \mid g(x) \geq 0\}$. Suppose there exists an SOS polynomial $\mathcal{B}(x)$, constants $\lambda, \gamma, c \in \mathbb{R}_0^+$, with $\lambda > \gamma$, and vectors of SOS polynomials $l_i(x)$, $l_u(x)$, and $l(x)$ such that the following expressions are SOS polynomials:*

$$-\mathcal{B}(x) - l_i(x)^\top g_i(x) + \gamma, \quad (6)$$

$$\mathcal{B}(x) - l_u(x)^\top g_u(x) - \lambda, \quad (7)$$

$$-\mathbb{E}[\mathcal{B}(f(x, \varsigma)) \mid x] + \mathcal{B}(x) - l(x)^\top g(x) + c. \quad (8)$$

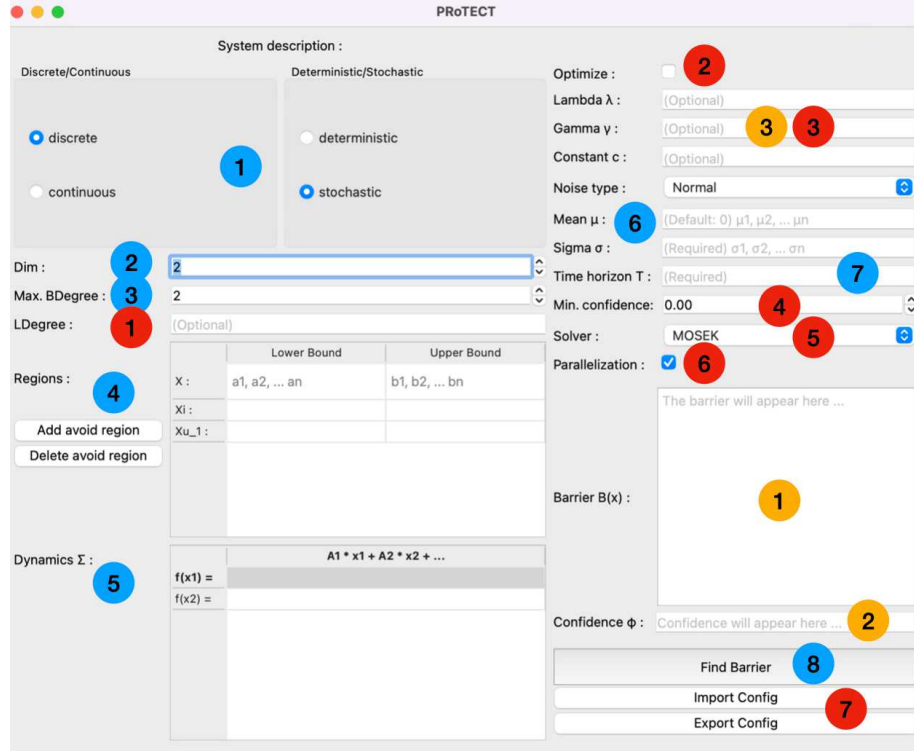


FIGURE 2. PRoTECT GUI for dt-SS, where required parameters, optional parameters, and outputs are marked with blue, red, and yellow circles, respectively (see Section 3.1).

Then $\mathcal{B}(x)$ satisfies conditions (2)-(4) in Definition 2.

Remark 1. *PRoTECT is equipped to accommodate any arbitrary number of unsafe regions $X_{\mathcal{U}_i}$, where $i \in \{1, \dots, m\}$. In such scenarios, condition (7) should be reiterated and enforced for each distinct unsafe region.*

3.1. PRoTECT Implementation for dt-SS.

3.1.1. *Graphic User Interface (GUI).* To enhance accessibility and user-friendliness of the tool, PRoTECT offers the Model-View-Presenter architecture incorporating a GUI. Specifically, a GUI strengthens user-friendliness by abstracting away implementation details for the code, allowing for a push-button method to construct barrier certificates. In Fig. 2, color notation is utilized to represent labels by their corresponding color and number. While PRoTECT provides GUIs for all

four classes of systems (see Fig. 2 (blue-1)), we only depict it for dt-SS due to space constraints. Our tool offers two implementations, either serial or parallel (red-6). The tool processes the information entered into the GUI before executing the desired function upon pressing the *Find Barrier* button (blue-8). Outputs of barrier certificate $\mathcal{B}(x)$, confidence ϕ , level sets γ and λ , and constant c are displayed at (yellow-1), (yellow-2), and (yellow-3), respectively. Optionally, the GUI allows for the import and export of configuration parameters in JSON format using the *Import Config* and *Export Config* buttons (red-7), with examples available in the folder `/ex/GUI_config_files`.

3.1.2. Application Programming Interface (API). In general, the backend of PROTECT behaves as an API, with functions that can be called and used in any python program. We provide some generic configuration files in `/ex/benchmarks-stochastic` and `/ex/benchmarks-deterministic`, which demonstrate how to use the functions in a standard python program. The user is expected to provide the following *required* parameters: dimension of the state set $X \subseteq \mathbb{R}^n$ (blue-2), indicated by `dim`, and the degree of the barrier certificate (blue-3), denoted by `b_degree`. The lower and upper bounds of the initial region $X_{\mathcal{I}}$, labeled as `L_initial` and `U_initial`; lower and upper bounds of the unsafe region $X_{\mathcal{U}}$, referred to as `L_unsafe` and `U_unsafe`; lower and upper bounds for the state set X , denoted as `L_space` and `U_space`; where the value of each dimension is separated with a comma (blue-4). Due to possible scenarios with multiple unsafe regions, the unsafe region is passed to the functions as a numpy array of numpy arrays describing each individual unsafe region. The transition map f , represented by `f`, written as a SymPy expression¹ for each dimension using states `x1,x2,...` and noise parameters `varsigma1,varsigma2,...` (blue-5). The time horizon $\mathcal{T}$, noted as `t` (blue-7). The distribution of the noise, `NoiseType`, can be specified as either `"normal"`, `"exponential"`, or `"uniform"` (blue-6).

Users may also specify *optional* parameters, with default values provided in Listing 1. These include the degree of the Lagrangian multipliers $l_i(x), l_u(x), l(x)$: `l_degree` (red-1), which, if not specified (i.e., set to `None`), will default to the same value as `b_degree`; the type of solver: `solver` (red-5), that can be either set to `"mosek"` [7] or `"cvxopt"` [6]. The confidence level ϕ in (5) can be optimized using `optimize` (red-2), if set to `True`. In this case, due to having a bilinearity between γ and λ in (5), the user is required to provide one λ : `lam`, e.g., select $\lambda = 1$ (red-3). The tool will then optimize for the other decision variables including γ and c to provide the highest confidence

¹https://docs.sympy.org/latest/tutorials/intro-tutorial/basic_operations.html

level ϕ . Alternatively, the user can select a minimum confidence level ϕ (red-4) using **confidence** they desire, so that PROTECT attempts to search for a BC satisfying that confidence level. The parameters for the distributions should be specified as follows (blue-6): for normal distributions, the mean μ can be set using **mean**, and the diagonal covariance matrix σ can be provided using **sigma**. For exponential distributions, the rate parameter for each dimension can be set using **rate**. For uniform distributions, the boundaries for each dimension can be set using **a** and **b**. We provide two functions for dt-SS (red-6): the first **dt_SS** finds a barrier for a single degree, and the second **parallel_dt_SS** runs the first function in parallel for all barrier degrees up to the maximum barrier degree specified (also called **b_degree**).

```

1  dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x
    , varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False, solver="
    mosek", confidence=None, gam=None, lam=None, c_val=None, mean=None, sigma=None,
    rate=None, a=None, b=None)
2  parallel_dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space,
    U_space, x, varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False,
    solver="mosek", confidence=None, gam=None, lam=None, c_val=None, mean=None,
    sigma=None, rate=None, a=None, b=None)
    
```

LISTING 1. dt-SS functions.

4. DISCRETE-TIME DETERMINISTIC SYSTEMS

Discrete-time deterministic systems (dt-DS) are characterized via $\Sigma_d : x(k+1) = f(x(k))$, where $f : X \rightarrow X$ is the transition map describing the state evolution of dt-DS. A BC $\mathcal{B} : X \rightarrow \mathbb{R}$ for dt-DS can be formulated analogously to Definition 2, adhering to the same conditions (2)-(3). However, modification is applied to condition (4) while removing the expected value and setting $c = 0$ as

$$\mathcal{B}(f(x)) \leq \mathcal{B}(x), \quad \forall x \in X. \quad (9)$$

by applying Lemma 2, we recast the BC's conditions as SOS polynomials (6) and (7) as before, and introduce a third SOS polynomial as

$$-\mathcal{B}(f(x)) + \mathcal{B}(x) - l(x)^\top g(x). \quad (10)$$

Given that the underlying system is deterministic, finding a BC offers a safety guarantee over an *infinite* time horizon [25].

4.1. PRoTECT Implementation for dt-DS. The user is required to input necessary (and optional) parameters as outlined in Subsection 3.1, excluding those parameters relevant to stochasticity (*e.g.*, time horizon, constant c , noise distribution, and confidence level). Optionally, the user can specify the level sets γ using `gam` or λ using `lam`. It is important to note that optimization for the level sets γ and λ is not performed, as any feasible solution with $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Similarly, we provide two functions `dt_DS` and `parallel_dt_DS` for the serial and parallel execution.

```

1  dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    f, l_degree=None, solver="mosek",gam=None,lam=None)
2  parallel_dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space,
    U_space, x, f, l_degree=None, solver="mosek",gam=None,lam=None)

```

LISTING 2. dt-DS functions.

5. CONTINUOUS-TIME STOCHASTIC SYSTEM

Here, we define the notion of barrier certificates for continuous-time stochastic systems (ct-SS). In particular, a ct-SS is defined by a tuple $\Sigma_c^\delta = (X, f, \delta, \rho)$, where $X \subseteq \mathbb{R}^n$ is the state set, $f: X \rightarrow \mathbb{R}^n$ is the drift term, $\delta: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times b}$ is the diffusion term, and $\rho: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times r}$ is the reset term. A ct-SS Σ_c^δ satisfies

$$\Sigma_c^\delta: \mathbf{d}x(t) = f(x(t))\mathbf{d}t + \delta(x(t))\mathbf{d}\mathbb{W}_t + \rho(x(t))\mathbf{d}\mathbb{P}_t,$$

where $\mathbb{W}_t$ and $\mathbb{P}_t$ are *Brownian motions* and *Poisson processes*. We consider the rate of Poisson processes $\mathbb{P}_t^z$ as ω_z for any $z \in [1, \dots, r]$. We assume that all drift, diffusion, and reset terms are *globally Lipschitz continuous* to ensure existence, uniqueness, and strong Markov property of the solution process [24].

A *twice differentiable* BC $\mathcal{B}: X \rightarrow \mathbb{R}_0^+$ for ct-SS is defined similarly to Definition 2, following the same conditions (2)-(3), with a modification to condition (4) as

$$\mathcal{LB}(x) \leq c, \quad \forall x \in X, \quad (11)$$

where $\mathcal{LB}$ is the *infinitesimal generator* of the stochastic process acting on the function $\mathcal{B}: X \rightarrow \mathbb{R}_0^+$, defined as

$$\mathcal{LB}(x) = \partial_x \mathcal{B}(x) f(x) + \frac{1}{2} \mathbf{Tr}(\delta(x) \delta(x)^\top \partial_{x,x} \mathcal{B}(x)) + \sum_{j=1}^r \omega_j (\mathcal{B}(x + \rho(x) \mathbf{e}_j^r) - \mathcal{B}(x)),$$

where $\mathbf{e}_j^r$ denotes an r -dimensional vector with 1 on the j -th entry and 0 elsewhere.

Applying Lemma 2, we recast the BC conditions as SOS polynomials (6) and (7) as before, and introduce a third SOS polynomial:

$$-\mathcal{LB}(x) - l(x)^\top g(x) + c. \quad (12)$$

One can apply Lemma 1 to offer the confidence level ϕ in (5) using the constructed level sets γ, λ and the constant c in (12).

5.1. PRoTECT Implementation for ct-SS. The user is asked to enter necessary (and optional) parameters as detailed in Subsection 3.1. Additionally, via the corresponding GUI, users must provide the diffusion term δ using **delta** for Brownian motion, the reset term ρ using **rho** for Poisson process, and the Poisson process rate ω using **p_rate**. For cases lacking either Brownian motion or Poisson processes, the corresponding parameter should be set to zero. The confidence level ϕ can also be optimized if **optimize** is set to **True**. We provide functions **ct_SS** and **parallel_ct_SS** for the serial and parallel execution.

```

1  ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x
    , f, t, l_degree=None, delta=None, rho=None, p_rate=None, optimize=False, solver
    ="mosek", confidence=None, gam=None, lam=None, c_val=None)
2  parallel_ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space,
    U_space, x, f, t, l_degree=None, delta=None, rho=None, p_rate=None, optimize=
    False, solver="mosek", confidence=None, gam=None, lam=None, c_val=None)
    
```

LISTING 3. ct-SS functions.

6. CONTINUOUS-TIME DETERMINISTIC SYSTEMS

Continuous-time deterministic systems (ct-DS), as the last class of models, can be described by $\Sigma_c : \dot{x}(t) = f(x(t))$, $t \in \mathbb{R}_0^+$ where $f : X \rightarrow X$ is the vector field. A *differentiable* BC $\mathcal{B} : X \rightarrow \mathbb{R}$ for ct-DS is defined akin to Definition 2, maintaining conditions (2)-(3), with an adjustment to condition (4) with $c = 0$ as

$$L_f \mathcal{B}(x) \leq 0, \quad \forall x \in X, \quad (13)$$

where $L_f \mathcal{B}$ is the *Lie derivative* of $\mathcal{B} : X \rightarrow \mathbb{R}$ with respect to vector field f , defined as $L_f \mathcal{B}(x) = \partial_x \mathcal{B}(x) f(x)$. Utilizing Lemma 2, we reformulate the BC into SOS polynomials (6),(7), while introducing a third SOS polynomial:

$$-L_f \mathcal{B}(x) - l(x)^\top g(x). \quad (14)$$

Since the underlying system is deterministic, finding a BC provides a safety assurance over an *infinite* time horizon [25].

6.1. PROTECT Implementation for ct-DS. The user is expected to input necessary (and optional) parameters as detailed in Subsection 3.1, omitting parameters pertinent to stochasticity (*e.g.*, time horizon, constant c , and confidence level). Optionally, users can define the level sets γ using `gam` or λ using `lam`. Optimization for level sets γ and λ is not conducted, *i.e.*, any feasible solution where $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Functions `ct_DS` and `parallel_ct_DS` are employed for the serial and parallel execution.

```

1  ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x
    , f, l_degree=None, solver="mosek", gam=None, lam=None)
2  parallel_ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space,
    U_space, x, f, l_degree=None, solver="mosek", gam=None, lam=None)

```

LISTING 4. ct-DS functions.

7. BENCHMARKING AND CASE STUDIES

Table 1 presents a comparison of the efficiency in finding barrier certificates for deterministic systems, dt-DS and ct-DS, using PROTECT against the tool FOSSIL [1]. Unless referenced, the case studies are taken from FOSSIL’s own benchmarks. Descriptions of all case studies, along with their corresponding simulation results, are provided in the appendix. All examples in this table, including the FOSSIL configuration files, can be located in the folder `/ex/benchmarks-deterministic`. Notably, PROTECT demonstrates higher efficiency for case studies up to four dimensions, while FOSSIL excels for those with six and eight dimensions. This discrepancy can be attributed to the increase in Lagrangian multipliers in PROTECT, which also necessitates designing more decision variables when searching for higher-degree BCs. Table 2 employs PROTECT for stochastic benchmarks. We remind the reader that, unlike the deterministic setting, PROTECT awaits barriers of all degrees to terminate, returning the one with the *highest confidence*. Consequently, stochastic

TABLE 1. Efficiency comparison of BC construction for deterministic systems: P_{Ro}TECT vs. FOSSIL. Models marked with “*” were adjusted to switch domains from spheres to hypercubes. All case studies were run on the same desktop computer (Intel i9-12900).

					P _{Ro} TECT		FOSSIL
	n	system	γ	λ	serial (sec)	parallel (sec)	(sec)
1D system	1	dt-DS	3.90	4.04	0.09	1.16	0.20
DC Motor [3]	2	dt-DS	1.19	1.20	0.20	1.31	0.22
barr ₂ room _{DT}	2	dt-DS	18.0	19.5	0.17	1.32	0.44
Jet Engine [14]	2	ct-DS	1.41	1.53	0.18	0.22	0.28
*hi-ord ₄	4	ct-DS	14.5	15.0	1.30	1.42	27.5
*hi-ord ₆	6	ct-DS	32.5	33.9	17.0	17.6	11.6
hi-ord ₈	8	ct-DS	45.7	57.2	172	172	21.3

case studies typically require notably longer completion times compared to deterministic ones, but at the gain of offering the highest confidence.

8. CONCLUSION

This work introduced P_{Ro}TECT, a pioneer software tool utilizing *SOS optimization* to explore polynomial-type BCs for verifying safety properties across *four classes of dynamical systems*: dt-SS, dt-DS, ct-SS, and ct-DS. The tool is developed in Python and incorporates a user-friendly GUI to enhance its usability. Additionally, P_{Ro}TECT offers *parallelization* to concurrently search for BCs of different degrees, ensuring an efficient construction. In the future, P_{Ro}TECT will be expanded to incorporate reachability and reach-while-avoid specifications, along with designing controllers using SOS optimization techniques.

REFERENCES

- [1] Abate, A., Ahmed, D., Edwards, A., Giacobbe, M., Peruffo, A.: FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In: Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control. pp. 1–11 (2021)
- [2] Abate, A., Blom, H., Cauchi, N., Delicaris, J., Hartmanns, A., Khaled, M., Lavaei, A., Pilch, C., Remke, A., Schupp, S., Shmarov, F., Soudjani, S., Vinod, A., Wooding, B., Zamani, M., Zuliani, P.: ARCH-COMP20 Category Report: Stochastic Models (2020)
- [3] Adewuyi, P.A.: DC motor speed control: A case between PID controller and fuzzy logic controller. international journal of multidisciplinary sciences and engineering **4**(4), 36–40 (2013)
- [4] Ames, A.D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., Tabuada, P.: Control Barrier Functions: Theory and Applications. In: 2019 18th European Control Conference (ECC). pp. 3420–3431. IEEE (2019)
- [5] Anand, M., Lavaei, A., Zamani, M.: Compositional synthesis of control barrier certificates for networks of stochastic systems against ω -regular specifications. Nonlinear Analysis: Hybrid Systems **51** (2024)
- [6] Andersen, M.S., Dahl, J., Vandenberghe, L., et al.: Cvxopt: A python package for convex optimization. Available at cvxopt.org **54** (2013)
- [7] ApS, M.: MOSEK Optimizer API for Python (2022)
- [8] Cimatti, A., Griggio, A., Schaafsma, B. J. and Sebastiani, R.: The MathSAT5 SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 93–107. Lecture Notes in Computer Science (2013)
- [9] De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proceedings of the International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340 (2008)
- [10] Edwards, A., Peruffo, A., Abate, A.: Fossil 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models. arXiv:2311.09793 (2023)
- [11] Gao, S., Avigad, J., Clarke, E.M.: δ -complete decision procedures for satisfiability over the reals. In: Automated Reasoning. pp. 286–300. Lecture Notes in Computer Science (2012)
- [12] Jahanshahi, N., Lavaei, A. and Zamani, M.: Compositional construction of safety controllers for networks of continuous-space pomdps. IEEE Transactions on Control of Network Systems **10**(1), 87–99 (2022)
- [13] Knight, J.C.: Safety critical systems: challenges and directions. In: Proceedings of the 24th international conference on software engineering. pp. 547–550 (2002)
- [14] Krstic, M., Kokotovic, P.V.: Lean backstepping design for a jet engine compressor model. In: Proceedings of International Conference on Control Applications. pp. 1047–1052. IEEE (1995)
- [15] Kushner, H.J.: On the stability of stochastic dynamical systems. Proceedings of the National Academy of Sciences **53**(1), 8–12 (1965)
- [16] Lavaei, A., Frazzoli, E.: Scalable synthesis of safety barrier certificates for networks of stochastic switched systems. IEEE Transactions on Automatic Control (2024)

- [17] Lavaei, A., Khaled, M., Soudjani, S., Zamani, M.: AMYTISS: Parallelized automated controller synthesis for large-scale stochastic systems. In: International conference on computer aided verification. pp. 461–474. Springer (2020)
- [18] Lavaei, A., Soudjani, S., Abate, A., Zamani, M.: Automated verification and synthesis of stochastic hybrid systems: A survey. *Automatica* **146** (2022)
- [19] Lindemann, L., Dimarogonas, D.V.: Barrier function based collaborative control of multiple robots under signal temporal logic tasks. *IEEE Transactions on Control of Network Systems* **7**(4), 1916–1928 (2020)
- [20] Mathiesen, F.B., Calvert, S.C., Laurenti, L.: Safety certification for stochastic systems via neural barrier functions. *IEEE Control Systems Letters* **7**, 973–978 (2022)
- [21] Nejati, A., Zamani, M.: From dissipativity theory to compositional construction of control barrier certificates. Schloss-Dagstuhl-Leibniz Zentrum für Informatik (2022)
- [22] Nejati, A., Zhong, B., Caccamo, M., Zamani, M.: Data-driven controller synthesis of unknown nonlinear polynomial systems via control barrier certificates. In: Learning for Dynamics and Control Conference. pp. 763–776. PMLR (2022)
- [23] Nejati, A., Soudjani, S., Zamani, M.: Compositional abstraction-based synthesis for continuous-time stochastic hybrid systems. *European Journal of Control* **57**, 82–94 (2021)
- [24] Øksendal, B.K., Sulem, A.: Applied stochastic control of jump diffusions, vol. 498. Springer (2005)
- [25] Prajna, S., Jadbabaie, A., Pappas, G.J.: Stochastic safety verification using barrier certificates. In: 2004 43rd IEEE conference on decision and control (CDC). vol. 1, pp. 929–934. IEEE (2004)
- [26] Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control* **52**(8), 1415–1428 (2007)
- [27] Prajna, S., Papachristodoulou, A., Parrilo, P.A.: Introducing sostools: A general purpose sum of squares programming solver. In: Proceedings of the 41st IEEE Conference on Decision and Control, 2002. vol. 1, pp. 741–746. IEEE (2002)
- [28] Ramos, J.A., Dos Santos, P.L.: Mathematical modeling, system identification, and controller design of a two tank system. In: 2007 46th IEEE Conference on Decision and Control. pp. 2838–2843. IEEE (2007)
- [29] Reznick, B.: Some concrete aspects of Hilbert’s 17th problem. *Contemporary mathematics* **253**, 251–272 (2000)
- [30] Rungger, M., Zamani, M.: SCOTS: A tool for the synthesis of symbolic controllers. In: Proceedings of the 19th international conference on hybrid systems: Computation and control. pp. 99–104 (2016)
- [31] Salamati, A., Lavaei, A., Soudjani, S., Zamani, M.: Data-driven verification and synthesis of stochastic systems via barrier certificates. *Automatica* **159**, 111323 (2024)
- [32] Verdier, C.F., Mazo, M.: Formal synthesis of analytic controllers for sampled-data systems via genetic programming. In: 2018 IEEE Conference on Decision and Control (CDC). pp. 4896–4901. IEEE (2018)
- [33] Xiao, W., Cassandras, C.G., Belta, C.: Safe Autonomy with Control Barrier Functions: Theory and Applications. Springer (2023)
- [34] Yuan, C.: SumOfSquares.py, <https://github.com/yuanchenyang/SumOfSquares.py>

-
- [35] Zhao, H., Zeng, X., Chen, T., Liu, Z.: Synthesizing barrier certificates using neural networks. In: Proceedings of the 23rd international conference on hybrid systems: Computation and control. pp. 1–11 (2020)

APPENDIX A. CASE STUDY DESCRIPTIONS AND SIMULATION RESULTS

A.1. Discrete-time Stochastic Systems (dt-SS).

A.1.1. *Room Temperature.* We consider a basic 1-dimensional room temperature system, based on [23], with dynamics

$$\Sigma_d^\varsigma: x(k+1) = (1 - \beta - \theta\nu)x(k) + \theta T_h \nu(k) + \beta T_e + R\varsigma(k),$$

where $x(k)$ is the temperature of the room, $T_h = 45^\circ\text{C}$ is the heater temperature, $T_e = -15^\circ\text{C}$ is the ambient temperature of the room, $\nu = -0.0120155x + 0.8$, $R = 0.1$, and $\beta = 0.6$ and $\theta = 0.145$ are conduction factors. The exponential noise ς has a rate of 1. Regions of interest are $X = [1, 50]$, $X_{\mathcal{I}} = [19.5, 20]$, $X_{\mathcal{U}_1} = [1, 17]$, $X_{\mathcal{U}_2} = [23, 50]$.

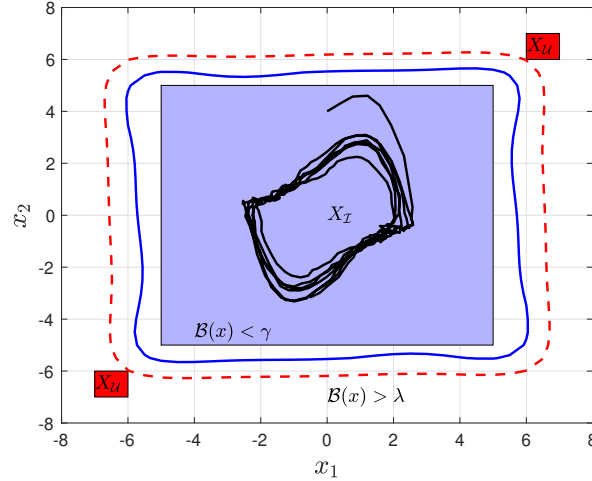


FIGURE 3. 2D Van der Pol oscillator system, with initial and unsafe regions $X_{\mathcal{I}}, X_{\mathcal{U}}$ marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (0, 4)$. The uniform noise has bound $[-0.1, 0.1]$ in both dimensions.

A.1.2. *2D Van der Pol oscillator.* We consider the stochastic Van der Pol oscillator benchmark from the ARCH competition for stochastic models [2], with the following dynamics:

$$\Sigma_d^s: \begin{cases} x_1(k+1) = x_1(k) + \tau x_2(k) + \varsigma_1(k), \\ x_2(k+1) = x_2(k) + \tau(-x_1(k) + (1 - x_1(k)^2)x_2(k)) + \varsigma_2(k), \end{cases}$$

where $\tau = 0.1$ is the sampling time, and ς_1, ς_2 are additive noises with uniform distributions with compact supports $[-0.02, 0.02] \times [-0.02, 0.02]$. We consider $X = [-7, 7]^2$, $X_I = [-5, 5]^2$ and $X_U = [-6, -7] \cup [6, 7]$. Fig. 3 shows level sets of $\gamma = 400$ and $\lambda = 1000$ for a constructed BC.

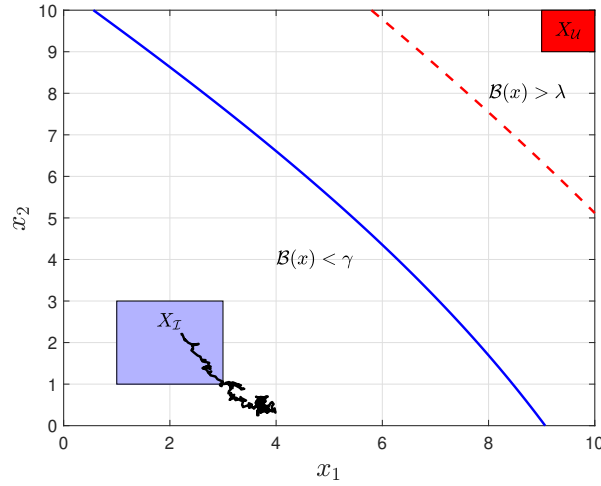


FIGURE 4. 2D Two-Tank system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (2.2, 2.2)$.

A.1.3. *Two-Tank System.* Consider a two-tank system [28], characterized by the following difference equations:

$$\Sigma_d^s: \begin{cases} h_1(k+1) = (1 - \tau \frac{\alpha_1}{A_1})h_1(k) + \tau \frac{q_1(k)}{A_1} + \varsigma_1(k), \\ h_2(k+1) = \tau \frac{\alpha_1}{A_2}h_1(k) + (1 - \tau \frac{\alpha_2}{A_2})h_2(k) + \tau \frac{q_2(k)}{A_2} + \varsigma_2(k), \end{cases}$$

where h_1, h_2 are heights of the fluid in two tanks, and ς_1, ς_2 are Gaussian noises with zero mean and standard deviations of 0.01. In addition, α_i and A_i are the valve coefficient and area of tank i ,

and q_1 and q_o are the inflow and outflow rate of tank 1 and 2, respectively. Furthermore, $\tau = 0.1$, $\frac{\alpha_1}{A_1} = 1$, $\frac{q_1}{A_1} = 4.5$, $\frac{\alpha_1}{A_2} = 1$, $\frac{\alpha_2}{A_2} = 1$ and $\frac{q_o}{A_2} = -3$. Regions of interest are $X = [1, 10] \times [1, 10]$, $X_{\mathcal{I}} = [1.75, 2.25] \times [1.75, 2.25]$, $X_{\mathcal{U}} = [9, 10] \times [9, 10]$. Fig. 4 shows level sets $\gamma = 1$ and $\lambda = 10$ of a constructed BC.

A.1.4. 3D Room Temperature. We consider the 3-dimensional room temperature case study [17], with dynamics

$$\Sigma_d^s: \begin{cases} x_1(k+1) = (1 - \tau(\alpha + \alpha_e))x_1(k) + \tau\alpha x_2(k) + \tau\alpha_e T_e + \varsigma_1(k), \\ x_2(k+1) = (1 - \tau(2\alpha + \alpha_e))x_1(k) + \tau\alpha(x_1(k) + x_3(k)) + \tau\alpha_e T_e + \varsigma_2(k), \\ x_3(k+1) = (1 - \tau(\alpha + \alpha_e))x_3(k) + \tau\alpha x_2(k) + \tau\alpha_e T_e + \varsigma_3(k), \end{cases}$$

where x_1, x_2, x_3 are temperatures of the three rooms, $\varsigma_1, \varsigma_2, \varsigma_3$ are zero-mean Gaussian noises with standard deviation 0.01, $T_e = 10^\circ\text{C}$ is the ambient temperature, $\alpha_e = 8 \times 10^{-3}$ and $\alpha = 6.2 \times 10^{-3}$ are heat exchange coefficients, and $\tau = 5$ minutes is the sampling time. Regions of interest are $X = [17, 30]^3$, $X_{\mathcal{I}} = [17, 18]^3$ and $X_{\mathcal{U}} = [29, 30]^3$.

A.2. Discrete-time Deterministic Systems (dt-DS).

A.2.1. 1D Basic System. We consider a simple scalar test case, with dynamics

$$\Sigma_d: x(k+1) = x(k) + \tau(15 - x(k) + 0.1\alpha_h(55 - x(k))),$$

where $\tau = 5$, $\alpha_h = 3.6 \times 10^{-3}$ with regions of interest $X = [-6, 6]$, $X_{\mathcal{I}} = [-0.5, 0.5]$, and $X_{\mathcal{U}} = [-6, -5]$.

A.2.2. FOSSIL Benchmark - 2D Room System. We consider the two room system from the FOSSIL benchmarks [1], with dynamics:

$$\Sigma_d: \begin{cases} x_1(k+1) = (1 - \tau(\alpha + \alpha_{e1}))x_1(k) + \tau\alpha x_2(k) + \tau\alpha_{e1}T_e, \\ x_2(k+1) = (1 - \tau(\alpha + \alpha_{e2}))x_2(k) + \tau\alpha x_1(k) + \tau\alpha_{e2}T_e, \end{cases}$$

where the discretization parameter $\tau = 5$, heat exchange constants $\alpha = 5 \times 10^{-2}$, $\alpha_{e1} = 5 \times 10^{-3}$, $\alpha_{e2} = 8 \times 10^{-3}$, and external temperature $T_e = 15$. We consider regions of interest $X = [18, 23]^2$, $X_{\mathcal{I}} = [18, 19.75]^2$, and $X_{\mathcal{U}} = [22, 23]$. Fig. 5 demonstrates a feasible BC with $\gamma = 17.9$ and $\lambda = 19.2$.

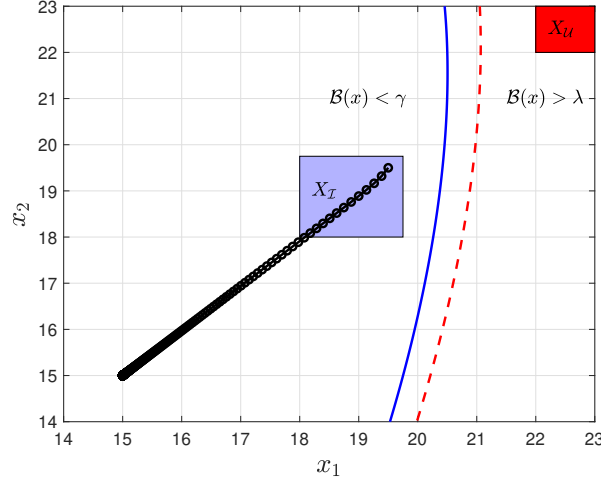


FIGURE 5. 2D Room system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (19.5, 19.5)$.

A.2.3. *2D DC Motor.* We consider the following DC motor case study [3]

$$\Sigma_d : \begin{cases} x_1(k+1) = x_1(k) + \tau(-\frac{R}{L}x_1(k) - \frac{k_{dc}}{L}x_2(k)), \\ x_2(k+1) = x_2(k) + \tau(\frac{k_{dc}}{J}x_1(k) - \frac{b}{J}x_2(k)), \end{cases}$$

where x_1 is the armature current, x_2 is the rotational speed of the shaft, τ is the sampling time 0.01. $R = 1, L = 0.01, J = 0.01, b = 1, k_{dc} = 0.01$ are, respectively, the electrical resistance, the electrical inductance, the moment of inertia of the rotor, the motor torque and the back electromotive force. We consider regions of interest $X = [0.1, 0.5] \times [0.1, 1]$, $X_I = [0.1, 0.4] \times [0.1, 1]$, and $X_U = [0.45, 0.5] \times [0.6, 1]$. Fig. 6 demonstrates a feasible BC with $\gamma = 1.19$ and $\lambda = 1.20$.

A.3. Continuous-time Stochastic Systems (ct-SS).

A.3.1. *Room Temperature.* We consider a room temperature system with dynamics [23]

$$\Sigma_c^\delta : dx(t) = ((-2\eta - \beta - \theta\nu)x(t) + \theta T_h\nu + \beta T_e)dt + \delta dW_t + \rho dP_t,$$

where x is the temperature of the room, $T_h = 48^\circ\text{C}$ is the heater temperature, $T_e = -15^\circ\text{C}$ is the outside temperature, $\nu = -0.0120155x + 0.7$, and $\eta = 0.005, \beta = 0.6$ and $\theta = 0.156$ are conduction

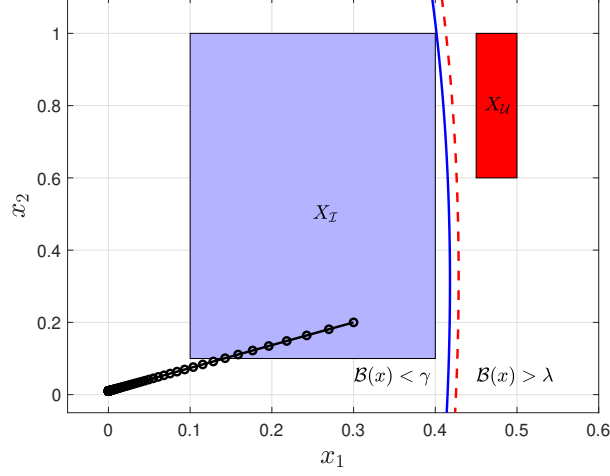


FIGURE 6. 2D DC Motor system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (0.3, 0.2)$.

factors. The system noise consists of both both Brownian with diffusion term $\delta = 0.1$ and Poisson process with reset term $\rho = 0.1$ and rate 0.1. Region of interest are $X = [1, 50]$, $X_I = [19.5, 20]$, and $X_{U_1} = [1, 17]$, $X_{U_2} = [23, 50]$.

A.3.2. *2D Linear System.* We consider the following linear example [25]

$$\Sigma_c^\delta : \begin{cases} dx_1(t) = (-5x_1(t) - 4x_2(t))dt, \\ dx_2(t) = (-x_1(t) - 2x_2(t))dt + \delta(x_2(t))dW_t, \end{cases}$$

where the diffusion term $\delta(x_2(t))$ is $0.5x_2(t)$. Regions of interest are $X = [-3, 3] \times [-1.5, 3.5]$, $X_I = [-0.25, 0.25] \times [2.75, 3.25]$, and $X_U = [-3, 3] \times [-1.5, -1]$. Fig. 7 demonstrates a feasible BC with $\gamma = 1.38$ and $\lambda = 10$.

A.3.3. *2D Nonlinear System.* We consider the following nonlinear system [25]

$$\Sigma_c^\delta : \begin{cases} dx_1(t) = x_2(t)dt, \\ dx_2(t) = (-x_1(t) - x_2(t) - 0.5x_1^3(t))dt + \delta dW_t, \end{cases}$$

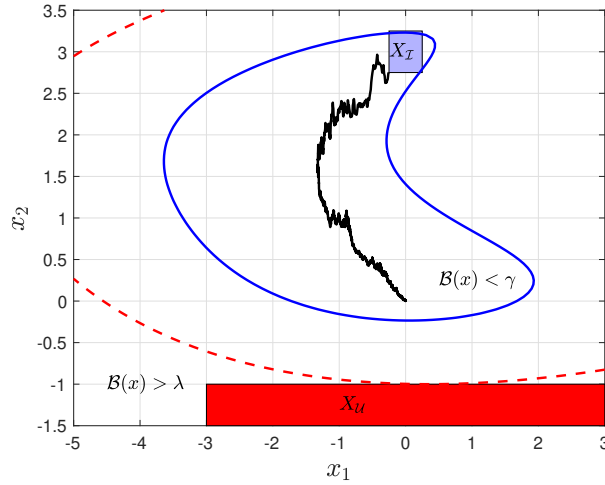


FIGURE 7. 2D linear system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (-0.25, 2.75)$.

where the diffusion term δ is 0.5. Regions of interest are $X = [-3, 3]^2$, $X_I = [-1, 1]^2$, and $X_U = [-3, 3] \times [2.25, 3]$. Fig. 8 demonstrates a feasible BC with $\gamma = 3.33$ and $\lambda = 10$.

A.3.4. *Benchmark - High Order 4.* We consider the following 4-dimensional dynamics

$$\Sigma_c^\delta : \begin{cases} dx_1(t) = x_2(t)dt + \delta_1 d\mathbb{W}_t, \\ dx_2(t) = x_3(t)dt + \delta_2 d\mathbb{W}_t, \\ dx_3(t) = x_4(t)dt + \delta_3 d\mathbb{W}_t, \\ dx_4(t) = (-3980x_4(t) - 4180x_3(t) - 2400x_2(t) - 576x_1(t))dt + \delta_4 d\mathbb{W}_t, \end{cases}$$

with diffusion terms $\delta_i = 0.1$, for all $i \in \{1, \dots, 4\}$. The regions of interest are $X = [-2, 2]^4$, $X_I = [0.5, 1.5]^4$, and $X_U = [-2.4, -1.6]^4$.

A.4. Continuous-time Deterministic Systems (ct-DS).

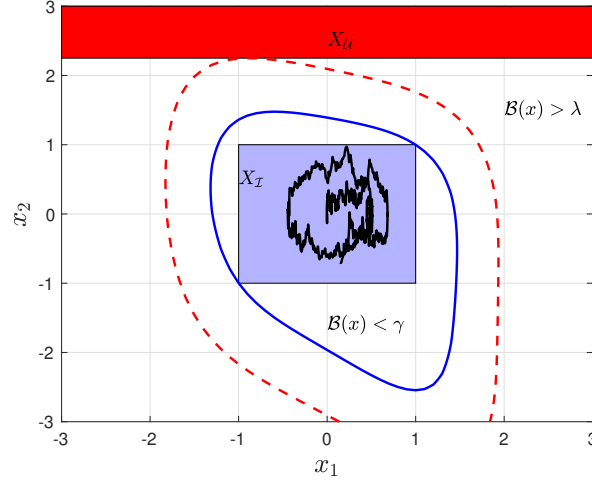


FIGURE 8. 2D nonlinear system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (0, 0)$.

A.4.1. *2D Jet Engine*. We consider the nonlinear Moore-Greitzer jet engine model [14]:

$$\Sigma_c : \begin{cases} \dot{x}_1(t) = -x_2(t) - 1.5x_1^2(t) - 0.5x_1^3(t), \\ \dot{x}_2(t) = x_1(t), \end{cases}$$

where $x_1 = \Phi - 1$, $x_2 = \Psi - \Lambda - 2$, with Φ, Ψ, Λ being, respectively, the mass flow, pressure rise, and a constant. We consider $X = [0.1, 1]^2$, $X_I = [0.1, 0.5]^2$, and $X_U = [0.7, 1]^2$. Fig. 9 shows level sets $\gamma = 1.41$ and $\lambda = 1.53$ of a feasible BC.

A.4.2. *FOSSIL Benchmark - High Order 4*. We consider the following 4-dimensional benchmark [1]

$$\Sigma_c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = -3980x_4(t) - 4180x_3(t) - 2400x_2(t) - 576x_1(t), \end{cases}$$

with the state space $X = [-2, 2]^4$, initial region $X_I = [0.5, 1.5]^4$, and unsafe region $X_U = [-2.4, -1.6]^4$.

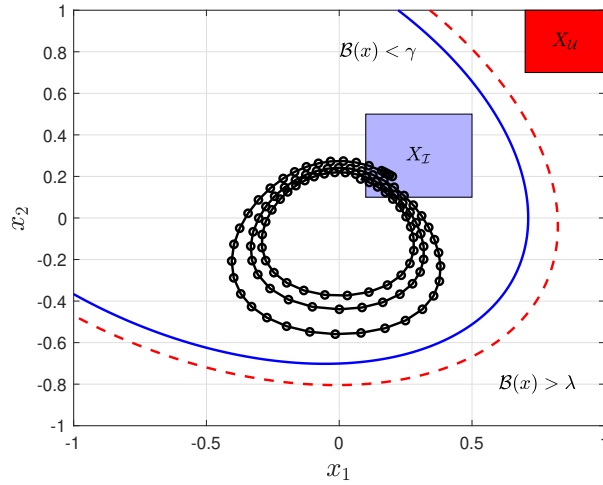


FIGURE 9. 2D Jet Engine system, with initial and unsafe regions X_I, X_U marked by blue and red boxes, respectively. Level sets of the barrier certificate are displayed in solid blue and red dashed for $\mathcal{B}(x) = \gamma$ and $\mathcal{B}(x) = \lambda$, respectively. A trajectory of the system is shown in black starting from an initial state $x_0 = (0.2, 0.2)$.

A.4.3. *FOSSIL benchmark - High Order 6.* We consider the following 6-dimensional benchmark [1]

$$\Sigma_c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = x_5(t), \\ \dot{x}_5(t) = x_6(t), \\ \dot{x}_6(t) = -800x_6(t) - 2273x_5(t) - 3980x_4(t) - 4180x_3(t) - 2400x_2(t) - 576x_1(t), \end{cases}$$

with the state space $X = [-2, 2]^6$, initial region $X_I = [0.5, 1.5]^6$, and unsafe region $X_U = [-2.4, -1.6]^6$.

A.4.4. *FOSSIL benchmark - High Order 8*. We consider the following 8-dimensional benchmark [1]

$$\Sigma_c : \begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = x_3(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = x_5(t), \\ \dot{x}_5(t) = x_6(t), \\ \dot{x}_6(t) = x_7(t), \\ \dot{x}_7(t) = x_8(t), \\ \dot{x}_8(t) = -20x_8(t) - 170x_7(t) - 800x_6(t) - 2273x_5(t) - 3980x_4(t) - 4180x_3(t) \\ \quad - 2400x_2(t) - 576x_1(t), \end{cases}$$

with the state space $X = [-2.2, 2.2]^8$, initial region $X_{\mathcal{I}} = [0.9, 1.1]^8$, and unsafe region $X_{\mathcal{U}} = [-2.2, -1.8]^8$.

TABLE 2. Efficiency evaluation in BC construction for *stochastic systems* via PROTECT. We compare case studies across three barrier degrees (2, 4, and 6), returning the barrier with the highest confidence. Additionally, we evaluate an optimized version using different fixed λ values. VDP denotes the stochastic Van der Pol oscillator. All experiments were conducted on a desktop computer (Intel i9-12900).

				Standard						Optimized					
	n	system	$\mathcal{T}$	b_degree	γ	λ	c	ϕ	time (sec)	b_degree	γ	λ	c	ϕ	time (sec)
RoomTemp [23]	1	ct-SS	5	6	0.74	3.55	0.41	0.23	0.33	6	$1.1e^{-6}$	10	$2.3e^{-7}$	0.99	0.33
RoomTemp [23]	1	dt-SS	5	6	1.12	6.16	0.72	0.23	0.52	6	$4.4e^{-7}$	10	$1.0e^{-7}$	0.99	0.53
VDP [2]	2	dt-SS	5	6	327	610	90.5	0.02	14.1	6	97.5	1000	3.53	0.88	14.3
ex_lin ₁ [25]	2	ct-SS	5	6	5.81	11.2	0.99	0.03	1.81	6	0.34	10	0.04	0.95	1.73
ex_nonlin ₁ [25]	2	ct-SS	5	6	12.4	42.8	5.73	0.04	1.63	6	1.84	10	0.2	0.72	1.81
TwoTanks [28]	2	dt-SS	5	4	6.23	265	8.06	0.82	5.13	4	$1.0e^{-6}$	10	$3.4e^{-8}$	0.99	5.14
RoomTemp [31]	3	dt-SS	3	6	2.55	6248	1.74	0.99	1509	4	$1.1e^{-8}$	10	$7.5e^{-9}$	0.99	1501
hi-ord ₄ [1]	4	ct-SS	3	2	5.12	30.4	7.78	0.05	1309	6	$1.8e^{-3}$	10	$1.2e^{-3}$	0.99	1308