PRoTECT: Parallelized ConstRuction of SafeTy BarriEr Certificates for Nonlinear Polynomial SysTems

Ben Wooding(✉), Viacheslav Horbanov, and Abolfazl Lavaei

School of Computing, Newcastle University, Newcastle upon Tyne, UK
{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. In particular, PRoTECT is the first software tool that designs stochastic barrier certificates. PRoTECT is implemented in Python as an application programming interface (API), offering users the flexibility to interact either through its user-friendly graphical 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.

1 Introduction

Motivation for PRoTECT. [1] 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]. Barrier certificates (BCs) [19,20], 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 [15]. 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

[1] For Open Science, PRoTECT v1.3 (Python 3.10), the version available at the time of publication, is permanently preserved at: https://doi.org/10.5281/zenodo.11085376.

Z. Liu et al. (Eds.): ICTAC 2025, LNCS 16237, pp. 448–458, 2026.
https://doi.org/10.1007/978-3-032-11176-0_26

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.

Related Work. A comprehensive overview of barrier certificates can be found in [3,27]. BCs can verify systems with polynomial dynamics, designing polynomial BCs using sum-of-squares techniques, *e.g.,* via SOSTOOLS [21]. Some alternative approaches explore verifying nonpolynomial systems, such as counterexample guided inductive synthesis (CEGIS), leveraging satisfiability modulo theories (SMT) solvers (e.g., Z3 [8] or dReal [10]). Other techniques encompass neural barrier functions [17,29] and genetic programs [25]. While we focus on using BCs for *safety* specifications, they also hold significant value for addressing other temporal logic specifications [4,16].

The most significant tool dedicated to the construction of barrier certificates is FOSSIL [1,9]. FOSSIL designs 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 PRoTECT provides support for both discrete- and continuous-time *stochastic* systems. Recently, two new tools for constructing barrier certificates have been introduced. TRUST [11] is a *data-driven* tool that generates barrier certificates for deterministic systems with *unknown* polynomial dynamics, using only a single trajectory of collected data. In addition, CBFKIT [7] is a toolbox designed for safe robotic planning. It supports both deterministic and stochastic *continuous-time dynamics* but requires the user to provide a barrier function *a priori*, which it then verifies for correctness—unlike PRoTECT, which automatically synthesizes a barrier certificate to meet the required conditions.

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). In particular, PRoTECT is the first software tool that offers stochastic barrier certificates.
(ii) PRoTECT is implemented in Python using SumOfSquares [28], 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) PRoTECT offers advanced GUIs for all four classes of models, enhancing the tool's accessibility and user-friendliness.

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

https://github.com/Kiguli/PRoTECT

2 Problem Description

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: $X_{\mathcal{I}}$ and $X_{\mathcal{U}}$, 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 the 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 the 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.

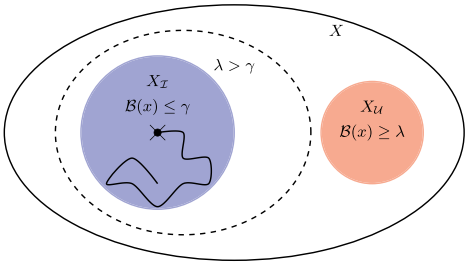

Fig. 1. A barrier certificate $\mathcal{B}(x)$ for a dynamical system. The dashed line denotes the initial level set $\mathcal{B}(x) = \gamma$.

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 horizon* $\mathcal{T} \in \mathbb{N} \cup \{\infty\}$. *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* $\phi \in [0, 1]$.

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

Algorithm 1: *Parallel* Construction of BCs

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

```
temp = [];
choose function func for Σ to identify the class of system;
forall the p ∈ {2, 4, ..., P} in parallel do
    barrier = func(p, K_req, K_opt);
    if barrier is SOS then
        temp.append(barrier);
        if Σ is deterministic then
            // terminate all parallel processes
        end
    end
end
// return element with highest confidence in temp
B(x) = max(temp);
```

Result: barrier certificate $\mathcal{B}(x)$, level sets γ, λ; confidence ϕ and constant c (for dt-SS and ct-SS)

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. Sect. 3).

Utilizing methodologies from the *sum-of-squares* (SOS) domain, facilitated by the SumOfSquares Python toolbox [28], 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 $q = [q_0, \ldots, q_z] \in \mathbb{R}^z$ 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 [23]. 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.

Remark 1 Due to space limitations, this work focuses on the presentation of dt-SS, given its complexity when handled by our tool. However, the other classes, Python code snippets, and a comparison between FOSSIL and PRoTECT for the deterministic classes can be found in the extended version of this work [26].

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.,* $\varsigma := \{\varsigma(k)\colon \Omega \to \mathcal{V}_\varsigma,\ k \in \mathbb{N}\}$, and $f\colon X \times \mathcal{V}_\varsigma \to X$ 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

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

The stochastic process $x_{x_0}\colon \Omega \times \mathbb{N} \to X$, 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 [20].

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 $\mathcal{B} : X \to \mathbb{R}_0^+$ is known as the barrier certificate (BC), if there exists constants $\lambda, \gamma, c \in \mathbb{R}_0^+$, with $\lambda > \gamma$, such that*

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

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

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

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 [12,14,20]. This lemma, commonly found in the literature (e.g. [24]), provides the safety confidence for stochastic systems. The same confidence formula applies to *continuous-time* stochastic systems (see extended version [26, Section 6]).

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*

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

Under the assumption that f is a polynomial function of state x and sets $X_\mathcal{I}, X_\mathcal{U}, X$ are semi-algebraic—*i.e.,* representable by polynomial inequalities—the extended version [26] provides a reformulation of (2)(4) as an SOS optimization program for designing a polynomial-type BC.

Remark 2 PRoTECT is equipped to accommodate any *arbitrary number* of unsafe regions $X_{\mathcal{U}_i}$, where $i \in \{1, \ldots, m\}$. In such scenarios, condition (3) should be reiterated and enforced for each distinct unsafe region.

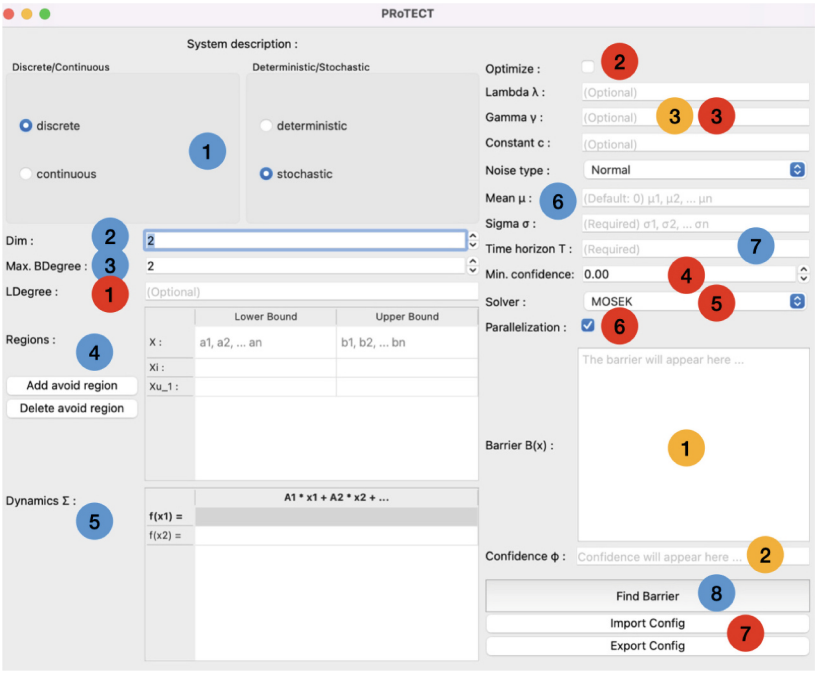

Fig. 2. PRoTECT GUI for dt-SS, where required parameters, optional parameters, and outputs are marked with blue, red, and yellow circles, respectively.

Graphical 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, colors and numbers are used to denote labels. 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`.

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. In the project folders `/ex/benchmarks-stochastic` and `/ex/benchmarks-deterministic`, we provide some generic configuration files 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 is written as a SymPy expression[2] 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, 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'' ``` [6] or ``` ``cvxopt'' ``` [5]. 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 resolve this, *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 level ϕ. Alternatively, the user can select a minimum confidence level ϕ (red-4) they require using `confidence`, 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`).

[2] https://docs.sympy.org/latest/tutorials/intro-tutorial/basic_operations.html.

Table 1. Efficiency evaluation in BC construction for *stochastic systems* via PRoTECT. We present case studies for the barrier degrees 4, optimizing for the barrier with the highest confidence. Parameters γ and c are designed via SOS optimization and λ is fixed a priori. VDP denotes the stochastic Van der Pol oscillator. All experiments were conducted on a desktop computer (Intel i9-12900).

				One Shot Stochastic Systems					
	n	system	$\mathcal{T}$	b_degree	γ	λ	c	ϕ	time (sec)
RoomTemp [18]	1	ct-SS	5	4	$4.8e^{-5}$	10	$9.6e^{-6}$	0.99	0.16
RoomTemp [18]	1	dt-SS	5	4	$4.4e^{-5}$	10	$8.9e^{-6}$	0.99	0.25
VDP [2]	2	dt-SS	5	4	N/A	1000	N/A	N/A	1.65
ex_lin_1 [19]	2	ct-SS	5	4	1.39	10	0.26	0.73	0.61
ex_nonlin_1 [19]	2	ct-SS	5	4	3.34	10	0.53	0.40	0.61
TwoTanks [22]	2	dt-SS	5	4	$1.0e^{-6}$	10	$3.4e^{-8}$	0.99	1.25
RoomTemp [24]	3	dt-SS	3	4	$1.1e^{-8}$	10	$7.5e^{-9}$	0.99	29.1
hi-ord_4 [1]	4	ct-SS	3	4	0.02	10	0.02	0.99	68.1

4 Benchmarking

Table 1 and Table 2 employ PRoTECT for stochastic benchmarks. In the 'One Shot' setting with a fixed barrier degree of 4, as shown in Table 1, PRoTECT optimizes all parameters, γ and c, during the SOS formulation, while λ is fixed a priori. Notably, the VDP example cannot find a barrier certificate with degree 4. Alternatively, the user can set a maximum degree and run computations in parallel up to this degree, returning the barrier certificate with the highest confidence. We call this the "Parallel" setting, as shown in Table 2, where λ is once again fixed a priori. Consequently, stochastic case studies typically require longer completion times compared to the "One Shot" setting or parallelism for deterministic systems (see extended version [26]), but at the gain of offering a higher confidence, *e.g.* example ex_nonlin_1 has a confidence improvement of 32%. This is a trade-off the user should navigate based on their particular setting. The parallelism introduces minimal overhead, and running the degrees in parallel is 19–27% faster than computing degrees (2, 4, 6) sequentially.

Remark 3 In Table 1, the VDP case does not yield an optimized solution for $\lambda = 1000$. However, by *setting the minimum confidence level* to $\phi = 0.8$ (as shown in red-4 in Fig. 2), feasible values for λ, γ, and c can be identified with confidence exceeding the specified threshold. These results are omitted from Table 1 to maintain consistency in the comparison.

Table 2. Efficiency evaluation in BC construction for *stochastic systems* via PRoTECT. We present case studies across three barrier degrees (2, 4, and 6), returning the barrier with the highest confidence. Parameter λ is fixed to a certain value and then γ and c are designed via SOS optimization. VDP denotes the stochastic Van der Pol oscillator. All experiments were conducted on a desktop computer (Intel i9-12900).

				Parallel Stochastic Systems					
	n	system	$\mathcal{T}$	b_degree	γ	λ	c	ϕ	time (sec)
RoomTemp [18]	1	ct-SS	5	6	$1.1e^{-6}$	10	$2.3e^{-7}$	0.99	0.33
RoomTemp [18]	1	dt-SS	5	6	$4.4e^{-7}$	10	$1.0e^{-7}$	0.99	0.53
VDP [2]	2	dt-SS	5	6	97.5	1000	3.53	0.88	14.3
ex_lin_1 [19]	2	ct-SS	5	6	0.34	10	0.04	0.95	1.73
ex_nonlin_1 [19]	2	ct-SS	5	6	1.84	10	0.2	0.72	1.81
TwoTanks [22]	2	dt-SS	5	4	$1.0e^{-6}$	10	$3.4e^{-8}$	0.99	5.14
RoomTemp [24]	3	dt-SS	3	4	$1.1e^{-8}$	10	$7.5e^{-9}$	0.99	1501
hi-ord_4 [1]	4	ct-SS	3	6	$1.8e^{-3}$	10	$1.2e^{-3}$	0.99	1308

5 Conclusion

This work introduced PRoTECT, 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. In particular, PRoTECT is the first software tool that designs stochastic barrier certificates. The tool is developed in Python and incorporates a user-friendly GUI to enhance its usability. Additionally, PRoTECT offers *parallelization* to concurrently search for BCs of different degrees, ensuring an efficient construction. In the future, PRoTECT 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., et al.: ARCH-COMP20 Category Report: Stochastic Models (2020)
3. 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)
4. 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)
5. Andersen, M.S., Dahl, J., Vandenberghe, L., et al.: CVXOPT: A Python package for convex optimization. Available at cvxopt.org **54** (2013)
6. ApS, M.: MOSEK Optimizer API for Python (2022)

7. Black, M., Fainekos, G., Hoxha, B., Okamoto, H., Prokhorov, D.: CBFKIT: A Control Barrier Function Toolbox for Robotics Applications. arXiv preprint arXiv:2404.07158 (2024)
8. 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)
9. Edwards, A., Peruffo, A., Abate, A.: Fossil 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models. arXiv:2311.09793 (2023)
10. 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)
11. Gardner, J., Wooding, B., Nejati, A., Lavaei, A.: TRUST: StabiliTy and Safety ContRoller Synthesis for Unknown Dynamical Models Using a Single Trajectory. In: Proceedings of the 28th ACM International Conference on Hybrid Systems: Computation and Control, pp. 1–16 (2025)
12. Jagtap, P., Soudjani, S., Zamani, M.: Formal synthesis of stochastic systems via control barrier certificates. IEEE Trans. Autom. Control **66**(7), 3097–3110 (2020)
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. Kushner, H.J.: On the stability of stochastic dynamical systems. Proc. Natl. Acad. Sci. **53**(1), 8–12 (1965)
15. Lavaei, A., Soudjani, S., Abate, A., Zamani, M.: Automated verification and synthesis of stochastic hybrid systems: A survey. Automatica **146** (2022)
16. Lindemann, L., Dimarogonas, D.V.: Barrier function based collaborative control of multiple robots under signal temporal logic tasks. IEEE Trans. Control Netw. Syst. **7**(4), 1916–1928 (2020)
17. Mathiesen, F.B., Calvert, S.C., Laurenti, L.: Safety certification for stochastic systems via neural barrier functions. IEEE Control Syst. Lett. **7**, 973–978 (2022)
18. Nejati, A., Soudjani, S., Zamani, M.: Compositional abstraction-based synthesis for continuous-time stochastic hybrid systems. Eur. J. Control. **57**, 82–94 (2021)
19. 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)
20. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Trans. Autom. Control **52**(8), 1415–1428 (2007)
21. 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)
22. 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)
23. Reznick, B.: Some concrete aspects of Hilbert's 17th problem. Contemp. Math. **253**, 251–272 (2000)
24. Salamati, A., Lavaei, A., Soudjani, S., Zamani, M.: Data-driven verification and synthesis of stochastic systems via barrier certificates. Automatica **159**, 111323 (2024)
25. 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)

26. Wooding, B., Horbanov, V., Lavaei, A.: PRoTECT: Parallelized Construction of Safety Barrier Certificates for Nonlinear Polynomial Systems. arXiv pp. 1–27 (2024). https://doi.org/10.48550/arXiv.2404.14804
27. Xiao, W., Cassandras, C.G., Belta, C.: Safe Autonomy with Control Barrier Functions: Theory and Applications. Springer (2023)
28. Yuan, C.: SumOfSquares.py. https://github.com/yuanchenyang/SumOfSquares.py
29. 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)