10-2023

Learning provably stabilizing neural controllers for discrete-time stochastic systems

Matin ANSARIPOUR

Krishnendu CHATTERJEE

A. Thomas HENZINGER

Mathias LECHNER

Dorde ZIKELIC
Singapore Management University, dzikelic@smu.edu.sg

Learning Provably Stabilizing Neural Controllers for Discrete-Time Stochastic Systems

Matin Ansaripour[1], Krishnendu Chatterjee[2], Thomas A. Henzinger[2], Mathias Lechner[3], and Đorđe Žikelić[2(✉)]

[1] Sharif University of Technology, Tehran, Iran
[2] Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
{krishnendu.chatterjee,tah,djordje.zikelic}@ist.ac.at
[3] Massachusetts Institute of Technology, Cambridge, MA, USA
mlechner@mit.edu

Abstract. We consider the problem of learning control policies in discrete-time stochastic systems which guarantee that the system stabilizes within some specified stabilization region with probability 1. Our approach is based on the novel notion of stabilizing ranking supermartingales (sRSMs) that we introduce in this work. Our sRSMs overcome the limitation of methods proposed in previous works whose applicability is restricted to systems in which the stabilizing region cannot be left once entered under any control policy. We present a learning procedure that learns a control policy together with an sRSM that formally certifies probability 1 stability, both learned as neural networks. We show that this procedure can also be adapted to formally verifying that, under a given Lipschitz continuous control policy, the stochastic system stabilizes within some stabilizing region with probability 1. Our experimental evaluation shows that our learning procedure can successfully learn provably stabilizing policies in practice.

Keywords: Learning-based control · Stochastic systems · Martingales · Formal verification · Stabilization

1 Introduction

Machine learning based methods and in particular reinforcement learning (RL) present a promising approach to solving highly non-linear control problems. This has sparked interest in the deployment of learning-based control methods in safety-critical autonomous systems such as self-driving cars or healthcare devices. However, the key challenge for their deployment in real-world scenarios is that they do not consider hard safety constraints. For instance, the main objective of RL is to maximize expected reward [46], but doing so provides no guarantees of the system's safety. A more recent paradigm in safe RL considers constrained Markov decision processes (cMDPs) [3,4,21,26,50], which are equiped with both

É. André and J. Sun (Eds.): ATVA 2023, LNCS 14215, pp. 357–379, 2023.
https://doi.org/10.1007/978-3-031-45329-8_17

a reward function and an auxiliary cost function. The goal of these works is then to maximize expected reward while keeping expected cost below some tolerable threshold. While these methods do enhance safety, they only ensure empirically that the expected cost function is below the threshold and do not provide any formal guarantees on constraint satisfaction.

This is particularly concerning for safety-critical applications, in which unsafe behavior of the system might have fatal consequences. Thus, a fundamental challenge for deploying learning-based methods in safety-critical autonomous systems applications is *formally certifying* safety of learned control policies [5,25].

Stability is a fundamental safety constraint in control theory, which requires the system to converge to and eventually stay within some specified stabilizing region with probability 1, a.k.a. almost-sure (a.s.) asymptotic stability [31,33]. Most existing research on learning policies for a control system with formal guarantees on stability considers *deterministic* systems and employs Lyapunov functions [31] for certifying the system's stability. In particular, a Lyapunov function is learned jointly with the control policy [1,8,15,42]. Informally, a Lyapunov function is a function that maps system states to nonnegative real numbers whose value decreases after every one-step evolution of the system until the stabilizing region is reached. Recently, [37] proposed a learning procedure for learning *ranking supermartingales (RSMs)* [11] for certifying a.s. asymptotic stability in discrete-time stochastic systems. RSMs generalize Lyapunov functions to supermartingale processes in probability theory [54] and decrease in value in *expectation* upon every one-step evolution of the system.

While these works present significant advances in learning control policies with formal stability guarantees as well as formal stability verification, they are either only applicable to deterministic systems or assume that the stabilizing set is *closed under system dynamics*, i.e., the agent cannot leave it once entered. In particular, the work of [37] reduces stability in stochastic systems to an *a.s. reachability* condition by assuming that the agent cannot leave the stabilization set. However, this assumption may not hold in real-world settings because the agent may be able to leave the stabilizing set with some positive probability due to the existence of stochastic disturbances, see Fig. 1. We illustrate the importance of relaxing this assumption on the classical example of balancing a pendulum in the upright position, which we also study in our experimental evaluation. The closedness under system dynamics assumption implies that, once the pendulum is in an upright position, it is ensured to stay upright and not move away. However, this is not a very realistic assumption due to possible existence of minor disturbances which the controller needs to balance out. The closedness under system dynamics assumption essentially assumes the existence of a balancing control policy which takes care of this problem. In contrast, our method does not assume such a balancing policy and learns a control policy which ensures that both (1) the pendulum reaches the upright position and (2) that the pendulum eventually stays upright with probability 1.

While the removal of the assumption that a stabilizing region cannot be left may appear to be a small improvement, in formal methods this is well-understood

to be a significant and difficult step. With the assumption, the desired controller has an a.s. reachability objective. Without the assumption, the desired controller has an a.s. persistence (or co-Büchi) objective, namely, to reach and stay in the stabilizing region with probability 1. Verification or synthesis for reachability conditions allow in general much simpler techniques than verification or synthesis for persistence conditions. For example, in non-stochastic systems, reachability can be expressed in alternation-free μ-calculus (i.e., fixpoint computation), whereas persistence requires alternation (i.e., nested fixpoint computation). Technically, reachability conditions are found on the first level of the Borel hierarchy, while persistence conditions are found on the second level [13]. It is, therefore, not surprising that also over continuous and stochastic state spaces, reachability techniques are insufficient for solving persistence problems.

In this work, we present the following three contributions.

1. **Theoretical Contributions.** In this work, we introduce *stabilizing ranking supermartingales (sRSMs)* and prove that they certify a.s. asymptotic stability in discrete-time stochastic systems even when the stabilizing set is not assumed to be closed under system dynamics. The key novelty of our sRSMs compared to RSMs is that they also impose an expected decrease condition within a part of the stabilizing region. The additional condition ensures that, once entered, the agent leaves the stabilizing region with probability at most $p < 1$. Thus, we show that the probability of the agent entering and leaving the stabilizing region N times is at most p^N, which by letting $N \to \infty$ implies that the agent eventually stabilizes within the region with probability 1. The key conceptual novelty is that we combine the convergence results of RSMs which were also exploited in [37] with a *concentration bound* on the supremum value of a supermartingale process. This combined reasoning allows us to formally guarantee a.s. asymptotic stability even for systems in which the stabilizing region is not closed under system dynamics. We remark that our proof that sRSMs certify a.s. asymptotic stability is not an immediate application of results from martingale theory, but that it introduces a novel method to reason about eventual stabilization within a set. We present this novel method in the proof of Theorem 1. Finally, we show that sRSMs not only present qualitative results to certify a.s. asymptotic stability but also present quantitative upper bounds on the number of time steps that the system may spend outside of the stabilization set prior to stabilization.
2. **Algorithmic Contributions.** Following our theoretical results on sRSMs, we present an algorithm for learning a control policy jointly with an sRSM that certifies a.s. asymptotic stability. The method parametrizes both the policy and the sRSM as neural networks and draws insight from established procedures for learning neural network Lyapunov functions [15] and RSMs [37]. It loops between a learner module that jointly trains a policy and an sRSM candidate and a verifier module that certifies the learned sRSM candidate by formally checking whether all sRSM conditions are satisfied. If the sRSM candidate violates some sRSM conditions, the verifier module produces counterexamples that are added to the learner module's training set to guide the

learner in the next loop iteration. Otherwise, if the verification is successful and the algorithm outputs a policy, then the policy guarantees a.s. asymptotic stability. By fixing the control policy and only learning and verifying the sRSM, our algorithm can also be used to verify that a given control policy guarantees a.s. asymptotic stability. This verification procedure only requires that the control policy is a Lipschitz continuous function.

3. **Experimental Contributions.** We experimentally evaluate our learning procedure on 2 stochastic RL tasks in which the stabilizing region is not closed under system dynamics and show that our learning procedure successfully learns control policies with a.s. asymptotic stability guarantees for both tasks.

Organization. The rest of this work is organized as follows. Section 2 contains preliminaries. In Sect. 3, we introduce our novel notion of stabilizing ranking supermartingales and prove that they provide a sound certificate for a.s. asymptotic stability, which is the main theoretical contribution of our work. In Sect. 4, we present the learner-verifier procedure for jointly learning a control policy together with an sRSM that formally certifies a.s. asymptotic stability. In Sect. 5, we experimentally evaluate our approach. We survey related work in Sect. 6. Finally, we conclude in Sect. 7.

2 Preliminaries

We consider a discrete-time stochastic dynamical system of the form

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi(\mathbf{x}_t), \omega_t),$$

where $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ is a dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ is a control policy and $\omega_t \in \mathcal{N}$ is a stochastic disturbance vector. Here, we use $\mathcal{X} \subseteq \mathbb{R}^n$ to denote the state space, $\mathcal{U} \subseteq \mathbb{R}^m$ the action space and $\mathcal{N} \subseteq \mathbb{R}^p$ the stochastic disturbance space of the system. In each time step, ω_t is sampled according to a probability distribution d over $\mathcal{N}$, independently from the previous samples.

A sequence $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$ of state-action-disturbance triples is a trajectory of the system, if $\mathbf{u}_t = \pi(\mathbf{x}_t)$, $\omega_t \in \mathsf{support}(d)$ and $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$ hold for each $t \in \mathbb{N}_0$. For each state $\mathbf{x}_0 \in \mathcal{X}$, the system induces a Markov process and defines a probability space over the set of all trajectories that start in $\mathbf{x}_0$ [41], with the probability measure and the expectation operators $\mathbb{P}_{\mathbf{x}_0}$ and $\mathbb{E}_{\mathbf{x}_0}$.

Assumptions. The state space $\mathcal{X} \subseteq \mathbb{R}^n$, the action space $\mathcal{U} \subseteq \mathbb{R}^m$ and the stochastic disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ are all assumed to be Borel-measurable. Furthermore, we assume that the system has a *bounded maximal step size* under any policy π, i.e. that there exists $\Delta > 0$ such that for every $\mathbf{x} \in \mathcal{X}$, $\omega \in \mathcal{N}$ and policy π we have $||\mathbf{x} - f(\mathbf{x}, \pi(\mathbf{x}), \omega)||_1 \leq \Delta$. Note that this is a realistic assumption that is satisfied in many real-world scenarios, e.g. a self-driving car can only traverse a certain maximal distance within each time step whose bounds depend on the maximal speed that the car can develop.

For our learning procedure in Sect. 4, we assume that $\mathcal{X} \subseteq \mathbb{R}^n$ is compact and that f is Lipschitz continuous, which are common assumptions in control theory. Given two metric spaces (X, d_X) and (Y, d_Y), a function $g : X \to Y$ is said to be *Lipschitz continuous* if there exists a constant $L > 0$ such that for every $x_1, x_2 \in X$ we have $d_Y(g(x_1), g(x_2)) \leq L \cdot d_X(x_1, x_2)$. We say that L is a Lipschitz constant of g. For the verification procedure when the control policy π is given, we also assume that π is Lipschitz continuous. This is also a common assumption in control theory and RL that allows for a rich class of policies including neural network policies, as all standard activation functions such as ReLU, sigmoid or tanh are Lipschitz continuous [47]. Finally, in Sect. 4 we assume that the stochastic disturbance space $\mathcal{N}$ is bounded or that d is a product of independent univariate distributions, which is needed for efficient sampling and expected value computation.

Almost-Sure Asymptotic Stability. There are several notions of stability in stochastic systems. In this work, we consider the notion of almost-sure asymptotic stability [33], which requires the system to eventually *converge and stay within* the stabilizing set. In order to define this formally, for each $\mathbf{x} \in \mathcal{X}$ let $d(\mathbf{x}, \mathcal{X}_s) = \inf_{\mathbf{x_s} \in \mathcal{X}_s} ||\mathbf{x} - \mathbf{x}_s||_1$, where $||\cdot||_1$ is the l_1-norm on $\mathbb{R}^m$.

Definition 1. *A Borel-measurable set $\mathcal{X}_s \subseteq \mathcal{X}$ is* almost-surely (a.s.) asymptotically stable*, if for each initial state $\mathbf{x}_0 \in \mathcal{X}$ we have*

$$\mathbb{P}_{\mathbf{x}_0}\Big[\lim_{t \to \infty} d(\mathbf{x}_t, \mathcal{X}_s) = 0\Big] = 1.$$

The above definition slightly differs from that of [33] which considers the special case of a singleton $\mathcal{X}_s = \{\mathbf{0}\}$. The reason for this difference is that, analogously to [37] and to the existing works on learning stabilizing policies in deterministic systems [8,15,42], we need to consider stability with respect to an open neighborhood of the origin for learning to be numerically stable.

3 Theoretical Results

We now introduce our novel notion of stabilizing ranking supermartingales (sRSMs). We then show that sRSMs can be used to formally certify a.s. asymptotic stability with respect to a fixed policy π *without* requiring that the stabilizing set is closed under system dynamics. To that end, in this section we assume that the policy π is fixed. In the next section, we will then present our learning procedure.

Prior Work – Ranking Supermartingales (RSMs). In order to motivate our sRSMs and to explain their novelty, we first recall ranking supermartingales (RSMs) [11] that were used in [37] for certifying a.s. asymptotic stability under a given policy π, when the stabilizing set is assumed to be closed under system dynamics. If the stabilizing set is assumed to be closed under system dynamics, then a.s. asymptotic stability of $\mathcal{X}_s$ is equivalent to *a.s. reachability* since the agent cannot leave $\mathcal{X}_s$ once entered.

Intuitively, an RSM is a non-negative continuous function $V : \mathcal{X} \to \mathbb{R}$ whose value at each state in $\mathcal{X}\backslash\mathcal{X}_s$ strictly decreases in expected value by some $\epsilon > 0$ upon every one-step evolution of the system under the policy π.

Definition 2 (Ranking supermartingales [11,37]). *A continuous function $V : \mathcal{X} \to \mathbb{R}$ is a* ranking supermartingale (RSM) for $\mathcal{X}_s$ *if $V(\mathbf{x}) \geq 0$ for each $\mathbf{x} \in \mathcal{X}$ and if there exists $\epsilon > 0$ such that for each $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_s$ we have*

$$\mathbb{E}_{\omega\sim d}\Big[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))\Big] \leq V(\mathbf{x}) - \epsilon.$$

It was shown that, if a system under policy π admits an RSM *and* the stabilizing set $\mathcal{X}_s$ is assumed to be closed under system dynamics, then $\mathcal{X}_s$ is a.s. asymptotically stable. The intuition behind this result is that V needs to strictly decrease in expected value until $\mathcal{X}_s$ is reached while remaining bounded from below by 0. Results from martingale theory can then be used to prove that the agent must eventually converge and reach $\mathcal{X}_s$ with probability 1, due to a strict decrease in expected value by $\epsilon > 0$ outside of $\mathcal{X}_s$ which prevents convergence to any other state. However, apart from nonnegativity, the defining conditions on RSMs do not impose any conditions on the RSM once the agent reaches $\mathcal{X}_s$. In particular, if the stabilizing set $\mathcal{X}_s$ is *not* closed under system dynamics, then the defining conditions of RSMs do not prevent the agent from leaving and reentering $\mathcal{X}_s$ infinitely many times and thus never stabilizing. In order to formally ensure stability, the defining conditions of RSMs need to be strengthened and in the rest of this section we solve this problem.

Our New Certificate – Stabilizing Ranking Supermartingales (sRSMs). We now define our sRSMs, which certify a.s. asymptotic stability even when the stabilizing set is not assumed to be closed under system dynamics and thus overcome the limitation of RSMs of [37] that was discussed above. Recall, we use Δ to denote the maximal step size of the system.

Definition 3 (Stabilizing ranking supermartingales). *Let $\epsilon, M, \delta > 0$. A Lipschitz continuous function $V : \mathcal{X} \to \mathbb{R}$ is said to be an* (ϵ, M, δ)-stabilizing ranking supermartingale ((ϵ, M, δ)-sRSM) for $\mathcal{X}_s$ *if the following conditions hold:*

1. Nonnegativity. *$V(\mathbf{x}) \geq 0$ holds for each $\mathbf{x} \in \mathcal{X}$.*
2. Strict expected decrease if $V \geq M$. *For each $\mathbf{x} \in \mathcal{X}$, if $V(\mathbf{x}) \geq M$ then*

$$\mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \epsilon.$$

3. Lower bound outside $\mathcal{X}_s$. *$V(\mathbf{x}) \geq M + L_V \cdot \Delta + \delta$ holds for each $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_s$, where L_V is a Lipschitz constant of V.*

An example of an sRSM for a 1-dimensional stochastic dynamical system is shown in Fig. 1. The intuition behind our new conditions is as follows. Condition 2 in Definition 3 requires that, at each state in which $V \geq M$, the value of V decreases in expectation by $\epsilon > 0$ upon one-step evolution of the system.

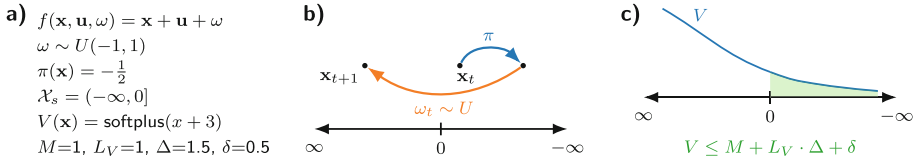

Fig. 1. Example of a 1-dimensional stochastic dynamical system for which the stabilizing set $\mathcal{X}_s$ is not closed under system dynamics since from every system state any other state is reachable with positive probability. **a)** System definition and an sRSM that it admits. **b)** Illustration of a single time step evolution of the system. **c)** Visualization of the sRSM and the corresponding level set used to bound the probability of leaving the stabilizing region.

As we show below, this ensures probability 1 convergence to the set of states $S=\{\mathbf{x}\in\mathcal{X}\mid V(\mathbf{x})\leq M\}$ from any other state of the system. On the other hand, condition 3 in Definition 3 requires that $V\geq M+L_V\cdot\Delta+\delta$ outside of the stabilizing set $\mathcal{X}_s$, thus $S\subseteq\mathcal{X}_s$. Moreover, if the agent is in a state where $V\leq M$, the value of V in the next state has to be $\leq M+L_V\cdot\Delta$ due to Lipschitz continuity of V and Δ being the maximal step size of the system. Therefore, even if the agent leaves S, for the agent to actually leave $\mathcal{X}_s$ the value of V has to *increase* from a value $\leq M+L_V\cdot\Delta$ to a value $\geq M+L_V\cdot\Delta+\delta$ while satisfying the strict expected *decrease* condition imposed by condition 2 in Definition 3 at every intermediate state that is not contained in S. The following theorem is the main result of this section.

Theorem 1. *If there exist $\epsilon,M,\delta>0$ and an (ϵ,M,δ)-sRSM for $\mathcal{X}_s$, then $\mathcal{X}_s$ is a.s. asymptotically stable.*

Proof sketch, full proof in the extended version [6]. In order to prove Theorem 1, we need to show that $\mathbb{P}_{\mathbf{x}_0}[\lim_{t\to\infty}d(\mathbf{x}_t,\mathcal{X}_s)=0]=1$ for every $\mathbf{x}_0\in\mathcal{X}$. We show this by proving the following two claims. First, we show that, from each initial state $\mathbf{x}_0\in\mathcal{X}$, the agent converges to and reaches $S=\{\mathbf{x}\in\mathcal{X}\mid V(\mathbf{x})\leq M\}$ with probability 1. The set S is a subset of $\mathcal{X}_s$ by condition 3 in Definition 3 of sRSMs. Second, we show that once the agent is in S it may leave $\mathcal{X}_s$ with probability at most $p=\frac{M+L_V\cdot\Delta}{M+L_V\cdot\Delta+\delta}<1$. We then prove that the two claims imply Theorem 1.

Claim 1. For each initial state $\mathbf{x}_0\in\mathcal{X}$, the agent converges to and reaches $S=\{\mathbf{x}\in\mathcal{X}\mid V(\mathbf{x})\leq M\}$ with probability 1.

To prove Claim 1, let $\mathbf{x}_0\in\mathcal{X}$. If $\mathbf{x}_0\in S$, then the claim trivially holds. So suppose w.l.o.g. that $\mathbf{x}_0\notin S$. We consider the probability space $(\Omega_{\mathbf{x}_0},\mathcal{F}_{\mathbf{x}_0},\mathbb{P}_{\mathbf{x}_0})$ of all system trajectories that start in $\mathbf{x}_0$, and define a *stopping time* $T_S:\Omega_{\mathbf{x}_0}\to\mathbb{N}_0\cup\{\infty\}$ which to each trajectory assigns the first hitting time of the set S and is equal to ∞ if the trajectory does not reach S. Furthermore, for each $i\in\mathbb{N}_0$, we define a random variable X_i in this probability space via

$$X_i(\rho)=\begin{cases}V(\mathbf{x}_i), & \text{if } i<T_S(\rho)\\ V(\mathbf{x}_{T_S(\rho)}), & \text{otherwise}\end{cases} \tag{1}$$

for each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0} \in \Omega_{\mathbf{x}_0}$. In words, X_i is equal to the value of V at the i-th state along the trajectory until S is reached, upon which it becomes constant and equal to the value of V upon first entry into S. We prove that $(X_i)_{i=0}^{\infty}$ is an instance of the mathematical notion of *ϵ-ranking supermartingales (ϵ-RSMs)* [11] for the stopping time T_S. Intuitively, an ϵ-RSM for T_S is a stochastic process which is non-negative, decreases in expected value upon every one-step evolution of the system and furthermore the decrease is strict and by $\epsilon > 0$ until the stopping time T_S is exceeded. If ϵ is allowed to be 0 as well, then the process is simply said to be a *supermartingale* [54]. It is a known result in martingale theory that, if an ϵ-RSM exists for T_S, then $\mathbb{P}_{\mathbf{x}_0}[T_S < \infty] = \mathbb{P}_{\mathbf{x}_0}[\text{Reach}(S)] = 1$. Thus, by proving that $(X_i)_{i=0}^{\infty}$ defined above is an ϵ-RSM for T_S, we also prove Claim 1. We provide an overview of martingale theory results used in this proof in the extended version of the paper [6].

Claim 2. $\mathbb{P}_{\mathbf{x}_0}[\exists\, t \in \mathbb{N}_0 \text{ s.t. } \mathbf{x}_t \notin \mathcal{X}_s] = p < 1$ where $p = \frac{M+L_V\cdot\Delta}{M+L_V\cdot\Delta+\delta}$, for each $\mathbf{x}_0 \in S$.

To prove Claim 2, recall that $S = \{\mathbf{x} \in \mathcal{X} \mid V(\mathbf{x}) \leq M\}$. Thus, as V is Lipschitz continuous with Lipschitz constant L_V and Δ is the maximal step size of the system, it follows that the value of V immediately upon the agent leaving the set S is $\leq M + L_V \cdot \Delta$. Hence, for the agent to leave $\mathcal{X}_s$ from $\mathbf{x}_0 \in S$, it first has to reach a state $\mathbf{x}_1$ with $M < V(\mathbf{x}_1) \leq M + L_V \cdot \Delta$ and then to also reach a state $\mathbf{x}_2 \notin \mathcal{X}_s$ from $\mathbf{x}_1$ without reentering S. By condition 3 in Definition 3 of sRSMs, we have $V(\mathbf{x}_2) \geq M + L_V \cdot \Delta + \delta$. We claim that this happens with probability at most $p = \frac{M+L_V\cdot\Delta}{M+L_V\cdot\Delta+\delta}$. To prove this, we use another result from martingale theory which says that, if $(Z_i)_{i=0}^{\infty}$ is a nonnegative supermartingale and $\lambda > 0$, then $\mathbb{P}[\sup_{i\geq 0} Z_i \geq \lambda] \leq \frac{\mathbb{E}[Z_0]}{\lambda}$ (see the extended version for full proof [6]). We apply this theorem to the process $(X'_i)_{i=0}^{\infty}$ defined analogously as in Eq. 1, but in the probability space of trajectories that start in $\mathbf{x}_1$. Then, since in this probability space we have that X_0 is equal to $V(\mathbf{x}_1) \leq M + L_V \cdot \Delta$, by plugging in $\lambda = M + L_V \cdot \Delta + \delta$ we conclude that the probability of the process ever leaving $\mathcal{X}_s$ and thus reaching a state in which $V \geq M + L_V \cdot \Delta + \delta$ is

$$
\begin{aligned}
&\mathbb{P}_{\mathbf{x}_0}[\exists\, t \in \mathbb{N}_0 \text{ s.t. } \mathbf{x}_t \notin \mathcal{X}_s] \\
\leq &\mathbb{P}_{\mathbf{x}_0}[\sup_{i\geq 0} X_i \geq M + L_V \cdot \Delta + \delta] \\
\leq &\mathbb{P}_{\mathbf{x}_1}[\sup_{i\geq 0} X'_i \geq M + L_V \cdot \Delta + \delta] \\
\leq &\frac{M + L_V \cdot \Delta}{M + L_V \cdot \Delta + \delta} = p < 1,
\end{aligned}
$$

so Claim 2 follows. The above inequality is formally proved in the extended version [6].

Claim 1 and Claim 2 Imply Theorem 1. Finally, we show that these two claims imply the theorem statement. By Claim 1, the agent with probability 1 converges to and reaches $S \subseteq \mathcal{X}_s$ from any initial state. On the other hand, by Claim 2,

upon reaching a state in S the probability of leaving $\mathcal{X}_s$ is at most $p < 1$. Furthermore, even if $\mathcal{X}_s$ is left, by Claim 1 the agent is guaranteed to again converge to and reach S. Hence, due to the system dynamics under a fixed policy satisfying Markov property, the probability of the agent leaving and reentering S more than N times is bounded from above by p^N. By letting $N \to \infty$, we conclude that the probability of the agent leaving $\mathcal{X}_s$ and reentering infinitely many times is 0, so the agent with probability 1 eventually enters and S and does not leave $\mathcal{X}_s$ after that. This implies that $\mathcal{X}_s$ is a.s. asymptotically stable. □

Bounds on Stabilization Time. We conclude this section by showing that our sRSMs not only certify a.s. asymptotic stability of $\mathcal{X}_s$, but also provide bounds on the number of time steps that the agent may spend outside of $\mathcal{X}_s$. This is particularly relevant for safety-critical applications in which the goal is not only to ensure stabilization but also to ensure that the agent spends as little time outside the stabilization set as possible. For each trajectory $\rho = (\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t \in \mathbb{N}_0}$, let $\mathsf{Out}_{\mathcal{X}_s}(\rho) = |\{t \in \mathbb{N}_0 \mid \mathbf{x}_t \notin \mathcal{X}_s\}| \in \mathbb{N}_0 \cup \{\infty\}$.

Theorem 2 (Proof in the extended version [6]). *Let $\epsilon, M, \delta > 0$ and suppose that $V : \mathcal{X} \to \mathbb{R}$ is an (ϵ, M, δ)-sRSM for $\mathcal{X}_s$. Let $\Gamma = \sup_{\mathbf{x} \in \mathcal{X}_s} V(\mathbf{x})$ be the supremum of all possible values that V can attain over the stabilizing set $\mathcal{X}_s$. Then, for each initial state $\mathbf{x}_0 \in \mathcal{X}$, we have that*

1. $\mathbb{E}_{\mathbf{x}_0}[\mathsf{Out}_{\mathcal{X}_s}] \leq \frac{V(\mathbf{x}_0)}{\epsilon} + \frac{(M + L_V \cdot \Delta) \cdot (\Gamma + L_V \cdot \Delta)}{\delta \cdot \epsilon}$.
2. $\mathbb{P}_{\mathbf{x}_0}[\mathsf{Out}_{\mathcal{X}_s} \geq t] \leq \frac{V(\mathbf{x}_0)}{t \cdot \epsilon} + \frac{(M + L_V \cdot \Delta) \cdot (\Gamma + L_V \cdot \Delta)}{\delta \cdot \epsilon \cdot t}$, *for any time $t \in \mathbb{N}$.*

4 Learning Stabilizing Policies and sRSMs on Compact State Spaces

In this section, we present our method for learning a stabilizing policy together with an sRSM that formally certifies a.s. asymptotic stability. As stated in Sect. 2, our method assumes that the state space $\mathcal{X} \subseteq \mathbb{R}^n$ is compact and that f is Lipschitz continuous with Lipschitz constant L_f. We prove that, if the method outputs a policy, then it guarantees a.s. asymptotic stability. After presenting the method for learning control policies, we show that it can also be adapted to a formal verification procedure that learns an sRSM for a given Lipschitz continuous control policy π.

Outline of the Method. We parameterize the policy and the sRSM via two neural networks $\pi_\theta : \mathcal{X} \to \mathcal{U}$ and $V_\nu : \mathcal{X} \to \mathbb{R}$, where θ and ν are vectors of neural network parameters. To enforce condition 1 in Definition 3, which requires the sRSM to be a nonnegative function, our method applies the softplus activation function $x \mapsto \log(\exp(x) + 1)$ to the output of V_ν. The remaining layers of π_θ and V_ν apply ReLU activation functions, therefore π_θ and V_ν are also Lipschitz continuous [47]. Our method draws insight from the algorithms of [15,55] for learning policies together with Lyapunov functions or RSMs and it comprises

Algorithm 1. Learner-verifier procedure

1: **Input** Dynamics function f, stochastic disturbance distribution d, stabilizing region $\mathcal{X}_s \subseteq \mathcal{X}$, Lipschitz constant L_f
2: **Parameters** $\tau > 0$, $N_{\text{cond 2}} \in \mathbb{N}$, $N_{\text{cond 3}} \in \mathbb{N}$, ϵ_{train}, δ_{train}
3: $\widetilde{\mathcal{X}} \leftarrow$ centers of cells of a discretization rectangular grid in $\mathcal{X}$ with mesh τ
4: $B \leftarrow$ centers of grid cells of a subgrid of $\widetilde{\mathcal{X}}$
5: $\pi_\theta \leftarrow$ policy trained by using PPO [44]
6: $M \leftarrow 1$
7: **while** timeout not reached **do**
8: $\quad \pi_\theta, V_\nu \leftarrow$ jointly trained by minimizing the loss in (2) on dataset B
9: $\quad \widetilde{\mathcal{X}}_{\geq M} \leftarrow$ centers of cells over which $V_\nu(\mathbf{x}) \geq M$
10: $\quad L_\pi, L_V \leftarrow$ Lipschitz constants of π_θ, V_ν
11: $\quad K \leftarrow L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$
12: $\quad \widetilde{\mathcal{X}}_{ce} \leftarrow$ counterexamples to condition 2 on $\widetilde{\mathcal{X}}_{\geq M}$
13: $\quad$ **if** $\widetilde{\mathcal{X}}_{ce} = \{\}$ **then**
14: $\quad\quad \text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s} \leftarrow$ grid cells that intersect $\mathcal{X}\setminus\mathcal{X}_s$
15: $\quad\quad \Delta_\theta \leftarrow$ max. step size of the system with policy π
16: $\quad\quad$ **if** $\underline{V}_\nu(\text{cell}) > M + L_V \cdot \Delta_\theta$ for all cell $\in \text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s}$ **then**
17: $\quad\quad\quad$ **return** π_θ, V_ν, "$\mathcal{X}_s$ is a.s. asymptotically stable under π_θ"
18: $\quad\quad$ **end if**
19: $\quad$ **else**
20: $\quad\quad B \leftarrow (B \setminus \{\mathbf{x} \in B | V_\nu(\mathbf{x}) < M\}) \cup \widetilde{\mathcal{X}}_{ce}$
21: $\quad$ **end if**
22: **end while**
23: **Return** Unknown

of a *learner* and a *verifier* module that are composed into a loop. In each loop iteration, the learner module first trains both π_θ and V_ν on a training objective in the form of a differentiable approximation of the sRSM conditions 2 and 3 in Definition 3. Once the training has converged, the verifier module formally checks whether the learned sRSM candidate satisfies conditions 2 and 3 in Definition 3. If both conditions are fulfilled, our method terminates and returns a policy together with an sRSM that formally certifies a.s. asymptotic stability. If at least one sRSM condition is violated, the verifier module enlarges the training set of the learner by counterexample states that violate the condition in order to guide the learner towards fixing the policy and the sRSM in the next learner iteration. This loop is repeated until either the verifier successfully verifies the learned sRSM and outputs the control policy and the sRSM, or until some specified timeout is reached in which case no control policy is returned by the method. The pseudocode of the algorithm is shown in Algorithm 1. In what follows, we provide details on algorithm initialization (lines 3–6, Algorithm 1) and on the learner and the verifier modules (lines 7–22, Algorithm 1).

4.1 Initialization

State Space Discretization. The key challenge in verifying an sRSM candidate is to check the expected decrease condition imposed by condition 2 in Definition 3. To check this condition, following the idea of [8] and [37] our method first computes a discretization of the state space $\mathcal{X}$. A *discretization* $\widetilde{\mathcal{X}}$ of $\mathcal{X}$ with *mesh* $\tau > 0$ is a finite subset $\widetilde{\mathcal{X}} \subseteq \mathcal{X}$ such that for every $\mathbf{x} \in \mathcal{X}$ there exists $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}$ with $||\widetilde{\mathbf{x}} - \mathbf{x}||_1 < \tau$. Our method computes the discretization by considering *centers of cells of a rectangular grid* of sufficiently small cell size (line 3, Algorithm 1). The discretization will later be used by the verifier in order to reduce verification of condition 2 to checking a slightly stricter condition at discretization vertices, due to all involved functions being Lipschitz continuous (more details Sect. 4.3).

The algorithm also collects the set B of grid cell centers of a subgrid of $\widetilde{\mathcal{X}}$ of larger mesh (line 4, Algorithm 1). This set will be used as the initial training set for the learner, and will then be gradually expanded by counterexamples computed by the verifier.

Policy Initialization. We initialize parameters of the neural network policy π_θ by running several iterations of the proximal policy optimization (PPO) [44] RL algorithm (line 5, Algorithm 1). In particular, we induce a Markov decision process (MDP) from the given system by using the reward function $r : \mathcal{X} \to \mathbb{R}$ defined via

$$r(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in \mathcal{X}_s \\ 0, & \text{otherwise} \end{cases}$$

in order to learn an initial policy that drives the system toward the stabilizing set. The practical importance of initialization for learning stabilizing policies in deterministic systems was observed in [15].

Fix the Value $M = 1$. As the last initialization step, we observe that one may always rescale the value of an sRSM by a strictly positive constant factor while preserving all conditions in Definition 3. Therefore, without loss of generality, we fix the value $M = 1$ in Definition 3 for our sRSM (line 6, Algorithm 1).

4.2 Learner

The policy and the sRSM candidate are learned by minimizing the loss

$$\mathcal{L}(\theta, \nu) = \mathcal{L}_{\text{cond }2}(\theta, \nu) + \mathcal{L}_{\text{cond }3}(\theta, \nu) \tag{2}$$

(line 8, Algorithm 1). The two loss terms guide the learner towards an sRSM candidate that satisfies conditions 2 and 3 in Definition 3.

We define the loss term for condition 2 via

$$\mathcal{L}_{\text{cond }2}(\theta, \nu) = \frac{1}{|B|} \sum_{\mathbf{x} \in B} \Big(\max \Big\{ \sum_{\omega_1, \ldots, \omega_{N_{\text{cond }2}} \sim d} \frac{V_\nu\big(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega_i)\big)}{N_{\text{cond }2}} - V_\nu(\mathbf{x}) + \epsilon_{\text{train}}, 0 \Big\} \Big).$$

Intuitively, for each element $\mathbf{x} \in B$ of the training set, the corresponding term in the sum incurs a loss whenever condition 2 is violated at $\mathbf{x}$. Since the expected value of V_ν at a successor state of $\mathbf{x}$ does not admit a closed form expression due to V_ν being a neural network, we approximate it as the mean of values of V_ν at $N_{\text{cond 2}}$ independently sampled successor states of $\mathbf{x}$, with $N_{\text{cond 2}}$ being an algorithm parameter.

For condition 3, the loss term samples $N_{\text{cond 3}}$ system states from $\mathcal{X} \backslash \mathcal{X}_s$ with $N_{\text{cond 3}}$ an algorithm parameter and incurs a loss whenever condition 3 is not satisfied at some sampled state:

$$\mathcal{L}_{\text{cond3}}(\theta, \nu) = \max\Big\{M + L_{V_\nu} + \Delta_\theta + \delta_{\text{train}} - \min_{x_1,\ldots x_{N_{\text{cond 3}}} \sim \mathcal{X} \backslash \mathcal{X}_s} V_\nu(x_i), 0\Big\}.$$

Regularization Terms in the Implementation. In our implementation, we also add two regularization terms to the loss function used by the learner. The first term favors learning an sRSM candidate whose global minimum is within the stabilizing set. The second term penalizes large Lipschitz bounds of the networks π_θ and V_ν by adding a regularization term. While these two loss terms do not directly enforce any particular condition in Definition 3, we observe that they help the learning and the verification process and decrease the number of needed learner-verifier iterations. See the extended version [6] for details on regularization terms.

4.3 Verifier

The verifier formally checks whether the learned sRSM candidate satisfies conditions 2 and 3 in Definition 3. Recall, condition 1 is satisfied due to the softplus activation function applied to the output of V_ν.

Formal Verification of Condition 2. The key challenge in checking the expected decrease condition in condition 2 is that the expected value of a neural network function does not admit a closed-form expression, so we cannot evaluate it directly. Instead, we check condition 2 by first showing that it suffices to check a slightly stricter condition at vertices of the discretization $\widetilde{\mathcal{X}}$, due to all involved functions being Lipschitz continuous. We then show how this stricter condition is checked at each discretization vertex.

To verify condition 2, the verifier first collects the set $\widetilde{\mathcal{X}}_{\geq M}$ of centers of all grid cells that contain a state $\mathbf{x}$ with $V_\nu(\mathbf{x}) \geq M$ (line 9, Algorithm 1). This set is computed via interval arithmetic abstract interpretation (IA-AI) [22,27], which for each grid cell propagates interval bounds across neural network layers in order to bound from below the minimal value that V_ν attains over that cell. The center of the grid cell is added to $\widetilde{\mathcal{X}}_{\geq M}$ whenever this lower bound is smaller than M. We use the method of [27] to perform IA-AI with respect to a neural network function V_ν so we refer the reader to [27] for details on this step.

Once $\widetilde{\mathcal{X}}_{\geq M}$ is computed, the verifier uses the method of [47, Section 4.3] to compute the Lipschitz constants L_π and L_V of neural networks π_θ and V_ν,

respectively (line 10, Algorithm 1). It then sets $K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$ (line 11, Algorithm 1). Finally, for each $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$ the verifier checks the following stricter inequality

$$\mathbb{E}_{\omega \sim d}\Big[V_\nu\Big(f(\widetilde{\mathbf{x}}, \pi_\theta(\widetilde{\mathbf{x}}), \omega)\Big)\Big] < V_\nu(\widetilde{\mathbf{x}}) - \tau \cdot K, \tag{3}$$

and collects the set $\widetilde{\mathcal{X}}_{ce} \subseteq \widetilde{\mathcal{X}}_{\geq M}$ of counterexamples at which this inequality is violated (line 12, Algorithm 1). The reason behind checking this stronger constraint is that, due to Lipschitz continuity of all involved functions and due to τ being the mesh of the discretization, we can show (formally done in the proof of Theorem 3) that this condition being satisfied for each $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$ implies that the expected decrease condition $\mathbb{E}_{\omega \sim d}[V_\nu(f(\mathbf{x}, \pi_\theta(\widetilde{\mathbf{x}}), \omega))] < V_\nu(\mathbf{x})$ is satisfied for all $\mathbf{x} \in \mathcal{X}$ with $V(\mathbf{x}) \geq M$. Then, due to both sides of the inequality being continuous functions and $\{\mathbf{x} \in \mathcal{X} \mid V_\nu(\mathbf{x}) \geq M\}$ being a compact set, their difference admits a strictly positive global minimum $\epsilon > 0$ so that $\mathbb{E}_{\omega \sim d}[V_\nu(f(\mathbf{x}, \pi_\theta(\widetilde{\mathbf{x}}), \omega))] \leq V_\nu(\mathbf{x}) - \epsilon$ is satisfied for all $\mathbf{x} \in \mathcal{X}$ with $V(\mathbf{x}) \geq M$. We show in the paragraph below how our method formally checks whether the inequality in (3) is satisfied at some $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$.

If (3) is satisfied for each $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$ and so $\widetilde{\mathcal{X}}_{ce} = \emptyset$, the verifier concludes that V_ν satisfies condition 2 in Definition 3 and proceeds to checking condition 3 in Definition 3 (lines 14–18, Algorithm 1). Otherwise, any computed counterexample to this constraint is added to B to help the learner fine-tune an sRSM candidate (line 20, Algorithm 1) and the algorithm proceeds to the start of the next learner-verifier iteration (line 7, Algorithm 1).

Checking Inequality (3) *and Expected Value Computation.* To check (3) at some $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$, we need to compute the expected value $\mathbb{E}_{\omega \sim d}[V_\nu(f(\widetilde{\mathbf{x}}, \pi_\theta(\widetilde{\mathbf{x}}), \omega))]$. Note that this expected value does not admit a closed form expression due to V_ν being a neural network function, so we cannot evaluate it directly. Instead, we use the method of [37] in order to compute an upper bound on this expected value and use this upper bound to formally check whether (3) is satisfied at $\widetilde{\mathbf{x}}$. For completeness of our presentation, we briefly describe this expected value bound computation below. Recall, in our assumptions in Sect. 2, we said that our algorithm assumes that the stochastic disturbance space $\mathcal{N}$ is bounded or that d is a product of independent univariate distributions.

First, consider the case when $\mathcal{N}$ is bounded. We partition the stochastic disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ into finitely many cells $\text{cell}(\mathcal{N}) = \{\mathcal{N}_1, \ldots, \mathcal{N}_k\}$. We denote by $\text{maxvol} = \max_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \mathsf{vol}(\mathcal{N}_i)$ the maximal volume of any cell in the partition with respect to the Lebesgue measure over $\mathbb{R}^p$. The expected value can then be bounded from above via

$$\mathbb{E}_{\omega \sim d}\Big[V_\nu\Big(f(\widetilde{\mathbf{x}}, \pi_\theta(\widetilde{\mathbf{x}}), \omega)\Big)\Big] \leq \sum_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \text{maxvol} \cdot \sup_{\omega \in \mathcal{N}_i} F(\omega)$$

where $F(\omega) = V_\nu(f(\widetilde{\mathbf{x}}, \pi_\theta(\widetilde{\mathbf{x}}), \omega)$. Each supremum on the right-hand-side is then bounded from above by using the IA-AI-based method of [27].

Second, consider the case when $\mathcal{N}$ is unbounded but d is a product of independent univariate distributions. Note that in this case we cannot directly follow the above approach since maxvol $= \max_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \mathsf{vol}(\mathcal{N}_i)$ would be infinite. However, since d is a product of independent univariate distributions, we may first apply the Probability Integral Transform [39] to each univariate distribution in d to transform it into a finite support distribution and then proceed as above.

Formal Verification of Condition 3. To formally verify condition 3 in Definition 3, the verifier collects the set $\text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s}$ of all grid cells that intersect $\mathcal{X}\setminus\mathcal{X}_s$ (line 14, Algorithm 1). Then, for each $\text{cell} \in \text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s}$, it uses IA-AI to check

$$\underline{V}_\nu(\text{cell}) > M + L_V \cdot \Delta_\theta, \tag{4}$$

with $\underline{V}_\nu(\text{cell})$ denoting the lower bound on V_ν over the cell computed by IA-AI (lines 15–16, Algorithm 1). If this holds, then the verifier concludes that V_ν satisfies condition 3 in Definition 3 with $\delta = \min_{\text{cell}\in\text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s}} \{\underline{V}_\nu(\text{cell}) - M - L_V \cdot \Delta_\theta\}$. Hence, as conditions 2 and 3 have both been formally verified to be satisfied, the method returns the policy π_θ and the sRSM V_ν which formally proves that $\mathcal{X}_s$ is a.s. asymptotically stable under π_θ (line 17, Algorithm 1). Otherwise, the method proceeds to the next learner-verifier loop iteration (line 7, Algorithm 1).

Algorithm Correctness. The following theorem establishes the correctness of Algorithm 1. In particular, it shows that if the verifier confirms that conditions 2 and 3 in Definition 3 are satisfied and therefore Algorithm 1 returns a control policy π_θ and an sRSM V_ν, then it holds that V_ν is indeed an sRSM and that $\mathcal{X}_s$ is a.s. asymptotically stable under π_θ.

Theorem 3 (Algorithm correctness, proof in the extended version [6]). *Suppose that the verifier shows that V_ν satisfies (3) for each $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}_{\geq M}$ and (4) for each cell $\in$ $\text{Cells}_{\mathcal{X}\setminus\mathcal{X}_s}$, so Algorithm 1 returns π_θ and V_ν. Then V_ν is an sRSM and $\mathcal{X}_s$ is a.s. asymptotically stable under π_θ.*

4.4 Adaptation into a Formal Verification Procedure

To conclude this section, we show that Algorithm 1 can be easily adapted into a formal verification procedure for showing that $\mathcal{X}_s$ is a.s. asymptotically stable under some given control policy π. This adaptation only assumes that π is Lipschitz continuous with a given Lipschitz constant L_π, or alternatively that it is a neural network policy with Lipschitz continuous activation functions in which case we use the method of [47] to compute its Lipschitz constant L_π.

Instead of jointly learning the control policy and the sRSM, the formal verification procedure now only learns a neural network sRSM V_ν. This is done by executing the analogous learner-verifier loop described in Algorithm 1. The only difference happens in the learner module, where now only the parameters ν of the sRSM neural network are learned. Hence, the loss function in (2) that is used

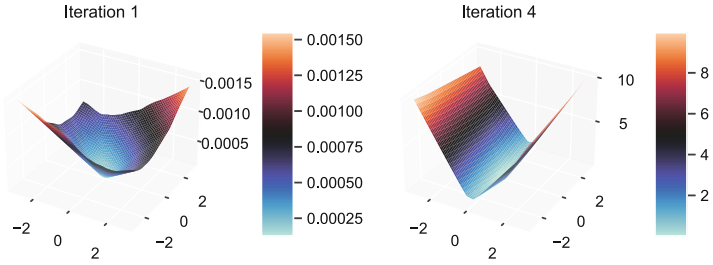

Fig. 2. Visualization of the sRSM candidate after 1 and 4 iterations of our algorithm for the inverted pendulum task. The candidate after 1 iteration does not satisfy all sRSM conditions, while the candidate after 4 iterations is an sRSM.

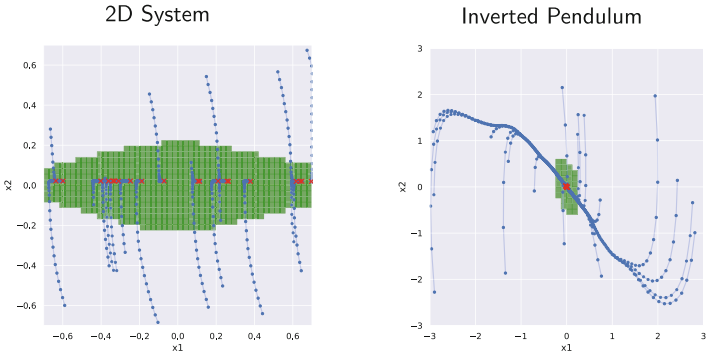

Fig. 3. Visualization of the learned stabilizing sets in green, in which the system will remain with probability 1. (Color figure online)

in (line 8, Algorithm 1) has the same form as in Sect. 4.2, but now it only takes parameters ν as input:

$$\mathcal{L}(\nu) = \mathcal{L}_{\text{cond }2}(\nu) + \mathcal{L}_{\text{cond }3}(\nu).$$

Additionally, the control policy initialization in (line 5, Algorithm 1) becomes redundant because the control policy π is given. Apart from these two changes, the formal verification procedure remains identical to Algorithm 1 and its correctness follows from Theorem 3.

5 Experimental Results

In this section, we experimentally evaluate the effectiveness of our method[1]. We consider the same experimental setting and the two benchmarks studied in [37]. However, in contrast to [37], we do not assume that the stabilization sets are

[1] Our implementation is available at https://github.com/mlech26l/neural_martingales/tree/ATVA2023.

Table 1. Results of our experimental evaluation. The first column shows benchmark names. The second column shows the numer of learner-verifier loop iterations needed to successfully learn and verify a control policy and an sRSM. The third column shows the mesh of the used discretization grid. The fourth column shows runtime in seconds.

Benchmark	Iters.	Mesh (τ)	Runtime
2D system	5	0.0007	3660 s
Pendulum	4	0.003	2619 s

closed under system dynamics and that the system stabilizes immediately upon reaching the stabilization set. In our evaluation, we modify both environments so that this assumption is violated. The goal of our evaluation is to confirm that our method based on sRSMs can in practice learn policies that formally guarantee a.s. asymptotic stability even when the stabilization set is not closed under system dynamics.

We parameterize both π_θ and V_ν by two fully-connected neural networks with 2 hidden ReLU layers, each with 128 neurons. Below we describe both benchmarks considered in our evaluation, and refer the reader to the extended version of the paper [6] for further details and formal definitions of environment dynamics.

The first benchmark is a two-dimensional linear dynamical system with non-linear control bounds and is of the form $x_{t+1} = Ax_t + Bg(u_t) + \omega$, where ω is a stochastic disturbance vector sampled from a zero-mean triangular distribution. The function g clips the action to stay within the interval [1, -1]. The state space is $\mathcal{X} = \{x \mid |x_1| \leq 0.7, |x_2| \leq 0.7\}$ and we want to learn a policy for the stabilizing set

$$\mathcal{X}_s = \mathcal{X} \setminus \Big(\{x \mid -0.7 \leq x_1 \leq -0.6, -0.7 \leq x_2 \leq -0.4\} \bigcup \{x \mid 0.6 \leq x_1 \leq 0.7, 0.4 \leq x_2 \leq 0.7\} \Big).$$

The second benchmark is a modified version of the inverted pendulum problem adapted from the OpenAI gym [9]. Note that this benchmark has non-polynomial dynamics, as its dynamics function involves a sine function (see the extended version [6]). The system is expressed by two state variables that represent the angle and the angular velocity of the pendulum. Contrary to the original task, the problem considered here introduces triangular-shaped random noise to the state after each update step. The state space is define as $\mathcal{X} = \{x \mid |x_1| \leq 3, |x_2| \leq 3\}$, and objective of the agent is to stabilize the pendulum within the stabilizing set

$$\mathcal{X}_s = \mathcal{X} \setminus \Big(\{x \mid -3 \leq x_1 \leq -2.9, -3 \leq x_2 \leq 0\} \bigcup \{x \mid 2.9 \leq x_1 \leq 3, 0 \leq x_2 \leq 3\} \Big).$$

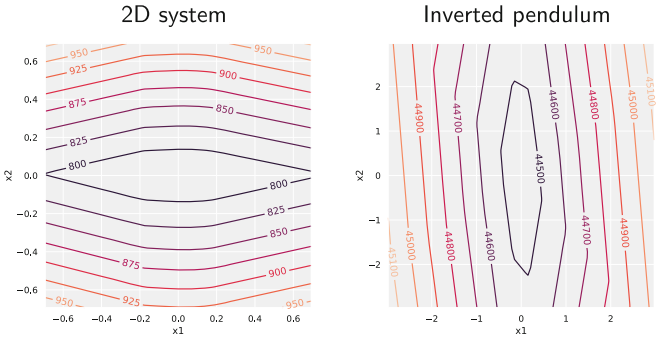

Fig. 4. Contour lines of the expected stabilization time implied by Theorem 2 for the 2D system task on the left and the inverted pendulum task on the right.

For both tasks, our algorithm could find valid sRSMs and prove stability. The runtime characteristics, such as the number of iterations and total runtime, is shown in Table 1. In Fig. 2 we plot the sRSM found by our algorithm for the inverted pendulum task. We also visualize for both tasks in Fig. 3 in green the subset of $\mathcal{X}_s$ implied by the learned sRSM in which the system stabilizes. Finally, in Fig. 4 we show the contour lines of the expected stabilization time bounds that are obtained by applying Theorem 2 to the learned sRSMs.

Limitations. We conclude by discussing limitations of our approach. Verification of neural networks is inherently a computationally difficult problem [8,30,43]. Our method is subject to this barrier as well. In particular, the complexity of the grid decomposition routine for checking the expected decrease condition is exponential in the dimension of the system state space. Consideration of different grid decomposition strategies and in particular non-uniform grids that incorporate properties of the state space is an interesting direction of future work towards improving the scalability of our method. However, a key advantage of our approach is that the complexity is only linear in the size of the neural network policy. Consequently, our approach allows learning and verifying networks that are of the size of typical networks used in reinforcement learning [44]. Moreover, our grid decomposition procedure runs entirely on accelerator devices, including CPUs, GPUs, and TPUs, thus leveraging future advances in these computing devices. A technical limitation of our learning procedure is that it is restricted to compact state spaces. Our theoretical results are applicable to arbitrary (potentially unbounded) state spaces, as shown in Fig. 1.

6 Related Work

Stability for Deterministic Systems. Most early works on control with stability constraints rely either on hand-designed certificates or their computation via sum-of-squares (SOS) programming [28,40]. Automation via SOS programming

is restricted to problems with polynomial dynamics and does not scale well with dimension. Learning-based methods present a promising approach to overcome these limitations [14,29,42]. In particular, the methods of [1,15] also learn a control policy and a Lyapunov function as neural networks by using a learner-verifier framework that our method builds on and extends to stochastic systems.

Stability for Stochastic Systems. While the theory behind stochastic system stability is well studied [33,34], only a few works consider automated controller synthesis with formal stability guarantees for stochastic systems with continuous dynamics. The methods of [23,51] are numerical and certify weaker notions of stability. Recently, [37,55] used RSMs and learn a stabilizing policy together with an RSM that certifies a.s. asymptotic stability. However, this method assumes closedness under system dynamics and essentially considers the stability problem as a reachability problem. In contrast, our proof in Sect. 3 introduces a new type of reasoning about supermartingales which allows us to handle stabilization without prior knowledge of a set that is closed under the system dynamics.

Reachability and Safety for Stochastic Systems. Comparatively more works have studied controller synthesis in stochastic systems with formal reachability and safety guarantees. A number of methods abstract the system as a finite-state Markov decision process (MDP) and synthesize a controller for the MDP to provide formal reachability or safety guarantees over finite time horizon [10,35,45,53]. An abstraction based method for obtaining infinite time horizon PAC-style guarantees on the probability of reach-avoidance in linear stochastic systems was proposed in [7]. A method for formal controller synthesis in infinite time horizon non-linear stochastic systems with guarantees on the probability of co-safety properties was proposed in [52]. A learning-based approach for learning a control policy that provides formal reachability and avoidance infinite time horizon guarantees was proposed in [56].

Safe Exploration RL. Safe exploration RL restricts exploration of RL algorithms in a way that a given safety constraint is satisfied. This is typically ensured by learning the system dynamics' uncertainty and limiting exploratory actions within a high probability safe region via Gaussian Processes [32,49], linearized models [24], deep robust regression [38] and Bayesian neural networks [36].

Probabilistic Program Analysis. Ranking supermartingales were originally proposed for proving a.s. termination in probabilistic programs (PPs) [11]. Since then, martingale-based methods have been used for termination [2,16,17,19] safety [18,20,48] and recurrence and persistence [12] analysis in PPs, with the latter being equivalent to stability. However, the persistence certificate of [12] is substantially different from ours. In particular, the certificate of [12] requires strict expected decrease outside the stabilizing set and non-strict expected decrease within the stabilizing set. In contrast, our sRSMs require strict expected decrease outside and only within a small part of the stabilizing set (see Definition 3). We also note that the certificate of [12] cannot be combined with our learner-verifier

procedure. Indeed, since our verifier module discretizes the state space and verifies a stricter condition at discretization vertices, if we tried to verify an instance of the certificate of [12] then we would be verifying the strict expected decrease condition over the whole state space. But this condition is not satisfiable over compact state spaces, as any continuous function must admit a global minimum.

7 Conclusion

In this work, we developed a method for learning control policies for stochastic systems with formal guarantees about the systems' a.s. asymptotic stability. Compared to the existing literature, which assumes that the stabilizing set is closed under system dynamics and cannot be left once entered, our approach does not impose this assumption. Our method is based on the novel notion of stabilizing ranking supermartingales (sRSMs) that serve as a formal certificate of a.s. asymptotic stability. We experimentally showed that our learning procedure is able to learn stabilizing policies and stability certificates in practice.

Acknowledgement. This work was supported in part by the ERC-2020-AdG 101020093, ERC CoG 863818 (FoRM-SMArt) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385.

References

1. Abate, A., Ahmed, D., Giacobbe, M., Peruffo, A.: Formal synthesis of Lyapunov neural networks. IEEE Control. Syst. Lett. **5**(3), 773–778 (2021). https://doi.org/10.1109/LCSYS.2020.3005328
2. Abate, A., Giacobbe, M., Roy, D.: Learning probabilistic termination proofs. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12760, pp. 3–26. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81688-9_1
3. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International Conference on Machine Learning, pp. 22–31. PMLR (2017)
4. Altman, E.: Constrained Markov Decision Processes, vol. 7. CRC Press (1999)
5. Amodei, D., Olah, C., Steinhardt, J., Christiano, P.F., Schulman, J., Mané, D.: Concrete problems in AI safety. CoRR abs/1606.06565 (2016). https://arxiv.org/abs/1606.06565
6. Ansaripour, M., Chatterjee, K., Henzinger, T.A., Lechner, M., Zikelic, D.: Learning provably stabilizing neural controllers for discrete-time stochastic systems. CoRR abs/2210.05304 (2022). https://doi.org/10.48550/arXiv.2210.05304
7. Badings, T.S., et al.: Robust control for dynamical systems with non-gaussian noise via formal abstractions. J. Artif. Intell. Res. **76**, 341–391 (2023). https://doi.org/10.1613/jair.1.14253
8. Berkenkamp, F., Turchetta, M., Schoellig, A.P., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017, pp. 908–918 (2017). https://proceedings.neurips.cc/paper/2017/hash/766ebcd59621e305170616ba3d3dac32-Abstract.html

9. Brockman, G., et al.: OpenAI gym. arXiv preprint arXiv:1606.01540 (2016)
10. Cauchi, N., Abate, A.: Stochy-automated verification and synthesis of stochastic processes. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp. 258–259 (2019)
11. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 511–526. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_34
12. Chakarov, A., Voronin, Y.-L., Sankaranarayanan, S.: Deductive proofs of almost sure persistence and recurrence properties. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 260–279. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_15
13. Chang, E., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 474–486. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55719-9_97
14. Chang, Y., Gao, S.: Stabilizing neural control using self-learned almost Lyapunov critics. In: IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, 30 May–5 June 2021, pp. 1803–1809. IEEE (2021). https://doi.org/10.1109/ICRA48506.2021.9560886
15. Chang, Y., Roohi, N., Gao, S.: Neural Lyapunov control. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019, pp. 3240–3249 (2019). https://proceedings.neurips.cc/paper/2019/hash/2647c1dba23bc0e0f9cdf75339e120d2-Abstract.html
16. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through Positivstellensatz's. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 3–22. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_1
17. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In: Bodík, R., Majumdar, R. (eds.) Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, 20–22 January 2016, pp. 327–342. ACM (2016). https://doi.org/10.1145/2837614.2837639
18. Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Zikelic, D.: Sound and complete certificates for quantitative termination analysis of probabilistic programs. In: Shoham, S., Vizel, Y. (eds.) CAV 2022. LNCS, vol. 13371, pp. 55–78. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_4
19. Chatterjee, K., Goharshady, E.K., Novotný, P., Zárevúcky, J., Žikelić, Đ: On lexicographic proof rules for probabilistic termination. In: Huisman, M., Păsăreanu, C., Zhan, N. (eds.) FM 2021. LNCS, vol. 13047, pp. 619–639. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90870-6_33
20. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: Castagna, G., Gordon, A.D. (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, 18–20 January 2017, pp. 145–160. ACM (2017). https://doi.org/10.1145/3009837.3009873
21. Chow, Y., Nachum, O., Duéñez-Guzmán, E.A., Ghavamzadeh, M.: A Lyapunov-based approach to safe reinforcement learning. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in

Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada, 3–8 December 2018, pp. 8103–8112 (2018). https://proceedings.neurips.cc/paper/2018/hash/4fe5149039b52765bde64beb9f674940-Abstract.html
22. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Graham, R.M., Harrison, M.A., Sethi, R. (eds.) Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977, pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973
23. Crespo, L.G., Sun, J.: Stochastic optimal control via Bellman's principle. Automatica **39**(12), 2109–2114 (2003). https://doi.org/10.1016/S0005-1098(03)00238-3
24. Dalal, G., Dvijotham, K., Vecerík, M., Hester, T., Paduraru, C., Tassa, Y.: Safe exploration in continuous action spaces. arXiv abs/1801.08757 (2018)
25. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. **16**, 1437–1480 (2015). https://dl.acm.org/citation.cfm?id=2886795
26. Geibel, P.: Reinforcement learning for MDPs with constraints. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 646–653. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_63
27. Gowal, S., et al.: On the effectiveness of interval bound propagation for training verifiably robust models. CoRR abs/1810.12715 (2018). https://arxiv.org/abs/1810.12715
28. Henrion, D., Garulli, A.: Positive Polynomials in Control, vol. 312. Springer, Heidelberg (2005)
29. Jin, W., Wang, Z., Yang, Z., Mou, S.: Neural certificates for safe control policies. CoRR abs/2006.08465 (2020). https://arxiv.org/abs/2006.08465
30. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
31. Khalil, H.: Nonlinear Systems. Pearson Education, Prentice Hall (2002)
32. Koller, T., Berkenkamp, F., Turchetta, M., Krause, A.: Learning-based model predictive control for safe exploration. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 6059–6066 (2018)
33. Kushner, H.J.: On the stability of stochastic dynamical systems. Proc. Natl. Acad. Sci. U.S.A. **53**(1), 8 (1965)
34. Kushner, H.J.: A partial history of the early development of continuous-time nonlinear stochastic systems theory. Automatica **50**(2), 303–334 (2014). https://doi.org/10.1016/j.automatica.2013.10.013
35. Lavaei, A., Khaled, M., Soudjani, S., Zamani, M.: AMYTISS: parallelized automated controller synthesis for large-scale stochastic systems. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12225, pp. 461–474. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_24
36. Lechner, M., Zikelic, D., Chatterjee, K., Henzinger, T.A.: Infinite time horizon safety of Bayesian neural networks. In: Ranzato, M., Beygelzimer, A., Dauphin, Y.N., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, 6–14 December 2021, Virtual, pp. 10171–10185 (2021). https://proceedings.neurips.cc/paper/2021/hash/544defa9fddff50c53b71c43e0da72be-Abstract.html

37. Lechner, M., Zikelic, D., Chatterjee, K., Henzinger, T.A.: Stability verification in stochastic control systems via neural network supermartingales. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, 22 February–1 March 2022, pp. 7326–7336. AAAI Press (2022). https://ojs.aaai.org/index.php/AAAI/article/view/20695
38. Liu, A., Shi, G., Chung, S.J., Anandkumar, A., Yue, Y.: Robust regression for safe exploration in control. In: L4DC (2020)
39. Murphy, K.P.: Machine Learning - A Probabilistic Perspective. Adaptive Computation and Machine Learning Series. MIT Press (2012)
40. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. California Institute of Technology (2000)
41. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics. Wiley (1994). https://doi.org/10.1002/9780470316887
42. Richards, S.M., Berkenkamp, F., Krause, A.: The Lyapunov neural network: adaptive stability certification for safe learning of dynamical systems. In: 2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29–31 October 2018, Proceedings. Proceedings of Machine Learning Research, vol. 87, pp. 466–476. PMLR (2018). https://proceedings.mlr.press/v87/richards18a.html
43. Sälzer, M., Lange, M.: Reachability is NP-complete even for the simplest neural networks. In: Bell, P.C., Totzke, P., Potapov, I. (eds.) RP 2021. LNCS, vol. 13035, pp. 149–164. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-89716-1_10
44. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
45. Soudjani, S.E.Z., Gevaerts, C., Abate, A.: FAUST^2: formal abstractions of uncountable-state stochastic processes. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 272–286. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_23
46. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
47. Szegedy, C., et al.: Intriguing properties of neural networks. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014, Conference Track Proceedings (2014). https://arxiv.org/abs/1312.6199
48. Takisaka, T., Oyabu, Y., Urabe, N., Hasuo, I.: Ranking and repulsing supermartingales for reachability in randomized programs. ACM Trans. Program. Lang. Syst. **43**(2), 5:1–5:46 (2021). https://doi.org/10.1145/3450967
49. Turchetta, M., Berkenkamp, F., Krause, A.: Safe exploration for interactive machine learning. In: NeurIPS (2019)
50. Uchibe, E., Doya, K.: Constrained reinforcement learning from intrinsic and extrinsic rewards. In: 2007 IEEE 6th International Conference on Development and Learning, pp. 163–168. IEEE (2007)
51. Vaidya, U.: Stochastic stability analysis of discrete-time system using Lyapunov measure. In: American Control Conference, ACC 2015, Chicago, IL, USA, 1–3 July 2015, pp. 4646–4651. IEEE (2015). https://doi.org/10.1109/ACC.2015.7172061
52. Van Huijgevoort, B., Schön, O., Soudjani, S., Haesaert, S.: SySCoRe: synthesis via stochastic coupling relations. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2023. Association for Computing Machinery (2023). https://doi.org/10.1145/3575870.3587123

53. Vinod, A.P., Gleason, J.D., Oishi, M.M.K.: SReachTools: a MATLAB stochastic reachability toolbox. In: Ozay, N., Prabhakar, P. (eds.) Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, 16–18 April 2019, pp. 33–38. ACM (2019). https://doi.org/10.1145/3302504.3311809
54. Williams, D.: Probability with Martingales. Cambridge Mathematical Textbooks. Cambridge University Press (1991)
55. Zikelic, D., Lechner, M., Chatterjee, K., Henzinger, T.A.: Learning stabilizing policies in stochastic control systems. CoRR abs/2205.11991 (2022). https://doi.org/10.48550/arXiv.2205.11991
56. Zikelic, D., Lechner, M., Henzinger, T.A., Chatterjee, K.: Learning control policies for stochastic systems with reach-avoid guarantees. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 10, pp. 11926–11935 (2023). https://doi.org/10.1609/aaai.v37i10.26407