Stability Verification in Stochastic Control Systems via Neural Network Supermartingales

Mathias Lechner[*], Đorđe Žikelić[*], Krishnendu Chatterjee, Thomas A. Henzinger

IST Austria
Klosterneuburg, Austria
{mlechner, dzikelic,krishnendu.chatterjee,tah}@ist.ac.at

Abstract

We consider the problem of formally verifying almost-sure (a.s.) asymptotic stability in discrete-time nonlinear stochastic control systems. While verifying stability in deterministic control systems is extensively studied in the literature, verifying stability in stochastic control systems is an open problem. The few existing works on this topic either consider only specialized forms of stochasticity or make restrictive assumptions on the system, rendering them inapplicable to learning algorithms with neural network policies. In this work, we present an approach for general nonlinear stochastic control problems with two novel aspects: (a) instead of classical stochastic extensions of Lyapunov functions, we use ranking supermartingales (RSMs) to certify a.s. asymptotic stability, and (b) we present a method for learning neural network RSMs. We prove that our approach guarantees a.s. asymptotic stability of the system and provides the first method to obtain bounds on the stabilization time, which stochastic Lyapunov functions do not. Finally, we validate our approach experimentally on a set of nonlinear stochastic reinforcement learning environments with neural network policies.

Introduction

Reinforcement learning (RL) presents a promising approach to learning high-performing control policies in nonlinear control problems. However, most RL algorithms focus on learning a policy that maximizes the expected reward (Sutton and Barto 2018), and do not take safety constraints into account. This raises concerns about their suitability for safety-critical applications such as autonomous vehicles or healthcare. Thus, a fundamental challenge for the deployment of policies learned via RL algorithms in safety-critical applications is certifying their safety (Amodei et al. 2016).

Stability is one of the most important properties that a control policy needs to ensure for the system to be safe (Lyapunov 1992). In their training phase, RL algorithms explore unknown environments through randomized actions while optimizing the learned policy's expected reward. Without a control mechanism to ensure that the system safely recovers from such exploratory actions and goes back to some known safe region, this might lead to catastrophic events. For example (Berkenkamp et al. 2017), if a self-driving car ends up driving outside its lane, a safe control policy needs to be able to stabilize the car back within the lane. Stability analysis is concerned with providing formal guarantees that the system can, with probability 1, recover back to this safe region from any system state and stay there indefinitely.

Formal verification of stability in *deterministic* control problems is well-studied. In particular, Lyapunov functions are an established method for stability analysis of deterministic systems (Khalil 2002). A more recent line of research focuses on automatically learning a Lyapunov function in the form of a neural network (see Related Work). While there are several theoretical results on extending Lyapunov functions to stochastic systems (Kushner 1965, 2014), only a few works consider the problem of automated stability analysis for control problems where the stochasticity originates from the environment (Crespo and Sun 2003; Vaidya 2015). Moreover, these works rely on restrictive assumptions on the system, making them inapplicable to learning algorithms with neural network policies, and they only verify strictly weaker notions of stability. Since uncertainty is a crucial component of RL systems, through exploration and to bridge the simulation-to-real gap (Tobin et al. 2017; James, Davison, and Johns 2017), methods for stability verification in stochastic control problems are needed. These methods need to support neural network policies and truly certify stability.

In this work, we present a method for formally verifying stability in discrete-time stochastic control problems. Our method is based on *ranking supermartingales (RSMs)*, which were originally introduced in the programming languages literature for termination analysis of probabilistic programs (Chakarov and Sankaranarayanan 2013). Intuitively, RSMs are nonnegative functions that decrease in expectation by at least $\epsilon > 0$ after every one-step evolution of the system and in each state that is not in the target region. We prove, for the first time, that RSMs can also be used to define stability certificates for stochastic control problems.

There are two key advantages of using RSMs instead of existing stochastic extensions of Lyapunov functions. First, we show that the defining properties of RSMs are much easier to encode within a learning framework. Second, we show that RSMs provide the first method to obtain bounds on the *stabilization time*, which stochastic Lyapunov functions do not. Ensuring that stabilization happens within some tolerable time limit is another practical concern about system

[*]Equal Contribution.

safety. For instance, given a stabilizing policy for a self-driving car that drives at a very high speed, it is not sufficient to *only* ensure that the speed eventually stabilizes within the allowed speed limit. A good stabilizing policy in such scenarios *additionally* needs to provide plausible guarantees on the stabilization time. One of the key benefits of using RSMs is that they provide such guarantees.

We then proceed to presenting an algorithmic framework for learning RSMs in the form of neural networks. Our algorithm draws insight from existing methods for learning Lyapunov functions in deterministic control problems (Chang, Roohi, and Gao 2019), and consists of two modules: the *learner* which learns an RSM candidate in the form of a neural network, and the *verifier* which then verifies the learned candidate. Whenever the verification step fails, a set of counterexamples showing that the candidate is not an RSM is computed, which are then used by the learner to fine-tune the candidate. This loop is repeated until a learned RSM candidate is successfully verified.

One of the key algorithmic challenges in designing the verifier module, compared to the case of deterministic systems, is that we need to verify an expected decrease condition which requires being able to compute the *expected value* of a neural network function over a probability distribution. Note, sampling cannot be used for this task since it only allows computing statistical bounds. To solve this challenge, we propose a method for efficiently computing formal and tight bounds on the expected value of an arbitrary neural network function over a probability distribution. We also demonstrate experimentally that our method computes tight bounds in practice. Our algorithmic contribution on computing expected value bounds for neural networks might on its own open various research directions on analyzing neural networks in probabilistic settings.

Finally, we evaluate our approach on two stochastic RL tasks with neural network control policies. It successfully learns RSMs proving that the policies stabilize the systems.

Contributions Our contributions are as follows:

1. We show that ranking supermartingales (RSMs) provide a stability certificate for stochastic control problems, as well as guarantees on the stabilization time.
2. We present a framework for learning neural network RSMs which also formally verifies the learned RSM.
3. As a part of our verification framework, we present a method for efficiently computing formal bounds on the expected value of a neural network function over a probability distribution. We are not aware of any existing works that tackle this problem.
4. We empirically validate that our approach can prove stability of stochastic systems with neural network policies.

Related Work

Stability verification via Lyapunov functions Stability verification in *deterministic* dynamical systems has received a lot of attention in recent works. For systems with polynomial dynamics and Lyapunov functions restricted to the sum-of-squares (SOS) form, a Lyapunov function can be computed via semi-definite programming (Henrion and Garulli 2005; Parrilo 2000; Jarvis-Wloszek et al. 2003). A learner-verifier framework similar to ours but for computing polynomial Lyapunov functions has been proposed in (Ravanbakhsh and Sankaranarayanan 2019). However, these methods require polynomial approximations and may not be efficient for systems with general nonlinearities. Moreover, it is known that even some simple dynamical systems that are asymptotically stable do not admit polynomial Lyapunov functions (Ahmadi, Krstic, and Parrilo 2011).

Learning Lyapunov functions in the form of a neural network has been considered in (Richards, Berkenkamp, and Krause 2018; Chang, Roohi, and Gao 2019; Abate et al. 2021), and it is an approach that is better suited to dynamical systems with general nonlinearities. In particular, (Richards, Berkenkamp, and Krause 2018) learn a Lyapunov function together with a region in which the system is stable by first discretizing the state space of the system, then learning a Lyapunov function candidate which tries to maximize the number of the discrete states at which the Lyapunov condition holds, and finally verifying that the candidate is indeed a Lyapunov function. The works (Chang, Roohi, and Gao 2019; Abate et al. 2021) propose a learner-verifier framework which uses counterexamples found by the verifier to improve the loss function and thus learn a new candidate. This loop is repeated until the verifier certifies that the Lyapunov function is correct. Our method for stability verification combines and extends ideas from these works.

Stability for stochastic control problems All of the above methods consider deterministic dynamical systems. While there are several theoretical results on the stability of stochastic dynamical systems (see (Kushner 2014) for a comprehensive survey), to our best knowledge there are very few works that consider their automated stability verification (Vaidya 2015; Crespo and Sun 2003). Both of these are numerical approaches that first partition the system's state space into finitely many regions and then over-approximate the system's continuous dynamics via a discrete finite-state abstraction. Thus, the computed stability certificates are piecewise-constant. Furthermore, (Vaidya 2015) verifies a weaker notion of stability called "coarse stochastic stability" that depends on the partition of the state space, and (Crespo and Sun 2003) imposes stability by requiring the system to reach the stabilizing region within some pre-specified finite time and deterministically (i.e. for each sample path).

Reachability for deterministic control problems Reachability is a property that is naturally related to stability since stability requires reachability of the stabilization set. There are several approaches and tools that analyze reachability in *deterministic* continuous-time feedback loop systems controlled by neural network policies. Some notable examples are Sherlock (Dutta, Chen, and Sankaranarayanan 2019) and ReachNN/ReachNN* (Huang et al. 2019; Fan et al. 2020) which use polynomial approximations to efficiently over-approximate the reachable set over some given time horizon, NNV (Tran et al. 2020) which is based on abstract interpretation, LRT-NG (Gruenbacher et al. 2020) which overapproximates the reachable set as sequence of hyperspheres, or Verisig (Ivanov et al. 2019) which reduces the problem to reachability analysis in hybrid systems. Furthermore, GoTube (Gruenbacher et al. 2021) constructs the reachable set of a deterministic continuous-time system with statistical guarantees about the constructed set overapproximating the true reachable states.

Note, however, that the goal of reachability analysis is to compute a set of states that are visited by *some* trajecotry of the system. In contrast, the goal of stability analysis is to show that *all* trajectories stabilize within the stabilization set (or with probability 1 in the case of stochastic systems). Furthermore, the above tools consider reachability over some finite time horizon and in deterministic systems, whereas in this work we do not impose any time limit and consider stochastic systems. Thus, these tools are not applicable to the stability verification problem in stochastic control systems.

Safe exploration RL algorithms need to explore the environment via randomized actions to learn which actions lead to a high future reward. However, in safety-critical environments, random actions may lead to catastrophic results. Safe exploration RL aims to restrict the exploratory actions to those that ensure safety of the environment. The most dominant approach to addressing this problem is learning the system dynamics' uncertainty bounds and limiting the exploratory actions within a high probability safety region. In the literature, Gaussian Processes (Koller et al. 2018; Turchetta, Berkenkamp, and Krause 2019; Berkenkamp 2019), linearized models (Dalal et al. 2018) , deep robust regression (Liu et al. 2020), and Bayesian neural networks (Lechner et al. 2021) are used for learning the uncertainty bounds.

Learning stable dynamics Learning dynamics from observation data is the first step in many control methods as well as model-based RL. Recent works considered learning deterministic system dynamics with guarantees on stability of some specified region (Kolter and Manek 2019). Learning stochastic dynamics from observation data has been studied in (Umlauft and Hirche 2017; Lawrence et al. 2020).

RSMs for probabilistic programs Ranking supermartingales (RSMs) were first introduced in the programming languages community in order to reason about termination of probabilistic programs (PPs) (Chakarov and Sankaranarayanan 2013). They are a stochastic extension of the classical notion of ranking functions in programs (Floyd 1967), and in addition to ensuring probability 1 termination they also provide guarantees on termination time (Fioriti and Hermanns 2015; Chatterjee et al. 2016). Our theoretical guarantees on the stabilization time draw insight from these results.

While some of our theoretical results are motivated by the works on PPs, our approach to stability verification differs significantly from the existing methods for RSM computation in PPs (Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2016; Chatterjee, Fu, and Goharshady 2016). In particular, these methods compute linear/polynomial RSMs via linear/semi-definite programming, and are more similar to the early methods for the computation of polynomial Lyapunov functions that we discussed above. On the contrary, our method learns an RSM in the form of a neural network. The only method for learning RSMs in PPs has been presented in the recent work of (Abate, Giacobbe, and Roy 2021), but this work computes only neural network RSMs with a single hidden layer and for a restricted class of PPs. In contrast, one of the main algorithmic novelties of our work is that we propose a general framework for computing the expected value of a neural network function over some probability distribution, which allows us to learn multi-layer neural network RSMs for general nonlinear systems.

Preliminaries

We consider a discrete-time stochastic dynamical system

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t),\ t \in \mathbb{N}_0.$$

The dynamics of the system are defined by the dynamics function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^m$ is the state space, $\mathcal{U} \subseteq \mathbb{R}^n$ is the control action space and $\mathcal{N} \subseteq \mathbb{R}^p$ is the disturbance space. The system starts in some initial state $\mathbf{x}_0 \in \mathcal{X}$ and at each time step t, given a state $\mathbf{x}_t$, the action $\mathbf{u}_t = \pi(\mathbf{x}_t)$ is chosen according to a control policy $\pi : \mathcal{X} \to \mathcal{U}$. The action $\mathbf{u}_t$, the state $\mathbf{x}_t$ and a randomly sampled disturbance vector $\omega_t \sim d$ then give rise to the subsequent state $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$. Here, we use d to denote the probability distribution over $\mathcal{N}$ from which the disturbance vector is sampled. Thus, the dynamics function f, the policy π and the probability distribution d together form a stochastic feedback loop system (or a closed-loop system).

Model assumptions Stability analysis of stochastic dynamical systems would be impossible without additional assumptions on the system, so that the model is sufficiently well-behaved. To that end, we assume that $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{N}$ are all Borel-measurable for the system semantics to be well-defined, and that $\mathcal{X}$ is compact in the Euclidean topology of $\mathbb{R}^m$. The dynamics function f and the control policy π are assumed to be Lipschitz continuous, which is a common assumption in control theory and allows a rich class of control policies including various types of neural networks (Szegedy et al. 2014). Moreover, assuming Lipschitz continuity is standard in existing works on stability analysis (Richards, Berkenkamp, and Krause 2018; Chang, Roohi, and Gao 2019). Finally, we assume that d is a product of independent univariate distributions, which is needed for efficient sampling and expected value computation.

Probability space of trajectories A sequence of state-action-disturbance triples $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$ is said to be a trajectory of the system, if for each $t \in \mathbb{N}_0$ we have $\mathbf{u}_t = \pi(\mathbf{x}_t)$, $\omega_t \in \mathsf{support}(d)$ and $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \omega_t)$. For each initial state $\mathbf{x}_0 \in \mathcal{X}$, the system dynamics induces a Markov process which gives rise to the probability space over the set of all trajectories that start in $\mathbf{x}_0$ (Puterman 1994, Section 2). We use $\mathbb{P}_{\mathbf{x}_0}$ and $\mathbb{E}_{\mathbf{x}_0}$ to denote the probability measure and the expectation operator in this probability space.

Almost-sure (a.s.) asymptotic stability There are several notions of stochastic stability, so we formally define the one that we consider in this work (Kushner 1965). Consider a stochastic feedback loop system defined as above, and let $\mathcal{X}_s \subseteq \mathcal{X}$ be Borel-measurable. We say that $\mathcal{X}_s$ is *closed under system dynamics* if, for every $\mathbf{x} \in \mathcal{X}_s$ and $\omega \in \mathsf{support}(d)$, we have that $f(\mathbf{x}, \pi(\mathbf{x}), \omega) \in \mathcal{X}_s$.

For $\mathcal{X}_s \subseteq \mathcal{X}$ that is closed under system dynamics, we say that it is almost-surely asymptotically stable if from any initial state the system almost-surely converges to $\mathcal{X}_s$ (and therefore stays in $\mathcal{X}_s$ due to closedness under system dynamics). 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 the Euclidean space $\mathbb{R}^m$.

Definition 1. *A non-empty set of states $\mathcal{X}_s \subseteq \mathcal{X}$ that is closed under system dynamics is said to be* almost-surely (a.s.) asymptotically stable *if, for each $\mathbf{x}_0 \in \mathcal{X}$, we have*

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

Our definition slightly differs from that of (Kushner 1965) which considers the special case of the stabilization set being the singleton equilibrium point at the origin, i.e. $\mathcal{X}_s = \{\mathbf{0}\}$. The reason for this discrepancy is that many practical approaches to the stability analysis of nonlinear systems need to make additional assumptions on the system's behavior around the origin, as otherwise they would suffer from numerical error issues. For instance, (Berkenkamp et al. 2017; Richards, Berkenkamp, and Krause 2018) study stability of deterministic dynamical systems and both assume that some open neighbourhood of the origin is *a priori* known to be stable, whereas (Chang, Roohi, and Gao 2019) only check stability conditions away from some neighbourhood around the origin. In order to avoid making such assumptions and to ensure that our method truly certifies stability, we assume that the region $\mathcal{X}_s$ has *non-empty interior* (i.e. it contains an open ball around a point in $\mathcal{X}$). By making either of the assumptions from the aforementioned works, our method naturally extends to the case where $\mathcal{X}_s = \{\mathbf{0}\}$.

Relation to a.s. reachability verification We remark that our method can also formally verify a.s. reachability of a specified target set, i.e. that for any initial state the system reaches a state in the target set with probability 1. In fact, due to the assumption that the stabilization set $\mathcal{X}_s$ is closed under system dynamics, the problem of verifying a.s. asymptotic stability reduces to the a.s. reachability verification problem for the stabilization set.

Assuming the closedness under system dynamics of the stabilization set is a reasonable and a realistic choice, due to dynamical systems typically expressing weak dynamics around the systems' stable points. As discussed above, many works on stability of deterministic dynamical systems also make a similar assumption, i.e. that an open neighbourhood of the origin $\mathbf{0}$ is closed under system dynamics (Berkenkamp et al. 2017; Richards, Berkenkamp, and Krause 2018; Chang, Roohi, and Gao 2019).

Theoretical Results

We now present a theoretical framework for formally certifying stability of a region in a discrete-time stochastic dynamical system. Our framework is based on *ranking supermartingales* which we introduce below.

Ranking supermartingales Consider a discrete-time stochastic dynamical system defined by a dynamics function f, a policy π and a probability distribution d with model assumptions as in the previous section, and let $\mathcal{X}_s \subseteq \mathcal{X}$ be closed under system dynamics and have non-empty interior.

Intuitively, a ranking supermartingale (RSM) is a non-negative continuous function whose value at each state in $\mathcal{X}\backslash\mathcal{X}_s$ decreases in expectation by at least $\epsilon > 0$ (is ϵ-*ranked*) after a one-step evolution of the system under the policy π, where the expected value is taken with respect to the probability distribution d over disturbance vectors. The name comes from the connection to supermartingales, a class of discrete-time stochastic processes in probability theory whose value decreases in expectation after each time step (Williams 1991). RSMs were first introduced in (Chakarov and Sankaranarayanan 2013) for the termination analysis of probabilistic programs, and we adapt them to the setting of stochastic dynamical systems.

Definition 2. *A continuous function $V : \mathcal{X} \to \mathbb{R}$ is said to be a* ranking supermartingale (RSM) *for $\mathcal{X}_s$, if $V(\mathbf{x}) \geq 0$ holds for any $\mathbf{x} \in \mathcal{X}$ and if there exists $\epsilon > 0$ such that*

$$\mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \epsilon \tag{1}$$

holds for every $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_s$.

We note that RSMs differ from the commonly considered stochastic Lyapunov functions for discrete-time stochastic systems (Kushner 2014), which require V to be continuous and to satisfy the following conditions:

- $\mathbb{E}_{\omega\sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] < V(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_s$,
- $V(\mathbf{x}) > 0$ for $\mathbf{x} \in \mathcal{X}\backslash\mathcal{X}_s$, and
- $V(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}_s$.

The third condition would be quite restrictive if we tried to learn V in the form of a neural network (which will be the goal of our novel approach to stability verification in the next section). Thus, one of the key benefits of considering RSMs instead of stochastic Lyapunov functions is that we may replace the $V(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{X}_s$ condition by a slightly stricter expected decrease condition that requires the decrease by at least some $\epsilon > 0$. Theorem 1 establishes that RSMs are indeed sufficient to prove a.s. asymptotic stability.

Theorem 1. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a Lipschitz continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a Lipschitz continuous policy and d a distribution over $\mathcal{N}$. Suppose that $\mathcal{X}$ is compact and let $\mathcal{X}_s \subseteq \mathcal{X}$ be closed under system dynamics and have a non-empty interior. Suppose that there exists an RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_s$. Then $\mathcal{X}_s$ is a.s. asymptotically stable.*

The main idea behind the proof of Theorem 1 is as follows. For each state $\mathbf{x}_0 \in \mathcal{X}$, we consider the probability space of all trajectories that start in $\mathbf{x}_0$. We then show that the RSM V for $\mathcal{X}_s$ gives rise to an instance of the mathematical notion of RSMs in this probability space, and use results from probability theory on the convergence of RSMs to conclude that $\mathcal{X}_s$ is a.s. asymptotically stable. The overview of the results from probability and martingale theory that we use in our proof as well as the formal proof of the theorem can be found in the Supplementary Material.

Bounds on the convergence time While formally verifying that a control policy stabilizes the system with probability 1 is very important for safety critical applications, another practical concern is to ensure that stabilization happens within some tolerable time limit.

Another important caveat of using RSMs for stability analysis of stochastic systems is that they provide formal guarantees on the stabilization time. For a system trajectory $(\mathbf{x}_t, \mathbf{u}_t, \omega_t)_{t\in\mathbb{N}_0}$, we define its stabilization time $T_{\mathcal{X}_s} = \inf\{t \in \mathbb{N}_0 \mid \mathbf{x}_t \in \mathcal{X}_s\}$ to be the first hitting time of the region $\mathcal{X}_s$ (with $T_{\mathcal{X}_s} = \infty$ if trajectory never reaches $\mathcal{X}_s$). Given $c > 0$, the system has *c-bounded differences* if the distance between any two consecutive system states with respect to the l_1-norm does not exceed c, i.e. for any $\mathbf{x} \in \mathcal{X}$ and $\omega \in \mathsf{support}(d)$ we have $||f(\mathbf{x}, \pi(\mathbf{x}), \omega) - \mathbf{x}||_1 \leq c$.

Theorem 2. *Let $f : \mathcal{X} \times \mathcal{U} \times \mathcal{N} \to \mathcal{X}$ be a Lipschitz continuous dynamics function, $\pi : \mathcal{X} \to \mathcal{U}$ a Lipschitz continuous policy and d a distribution over $\mathcal{N}$. Suppose that $\mathcal{X}$ is compact and let $\mathcal{X}_s \subseteq \mathcal{X}$ be closed under system dynamics and have a non-empty interior. Suppose that there exists an ϵ-RSM $V : \mathcal{X} \to \mathbb{R}$ for $\mathcal{X}_s$. Then, for any initial state $\mathbf{x}_0 \in \mathcal{X}$,*

1. $\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_s}] \leq \frac{V(\mathbf{x}_0)}{\epsilon}$.
2. $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_s} \geq t] \leq \frac{V(\mathbf{x}_0)}{\epsilon \cdot t}$, *for any time $t \in \mathbb{N}$.*
3. *If the system has c-*bounded differences *for $c > 0$, then $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_s} \geq t] \leq A \cdot e^{-t\cdot\epsilon^2/(2\cdot(c+\epsilon)^2)}$ for any time $t \in \mathbb{N}$ and $A = e^{\epsilon\cdot V(\mathbf{x}_0)/(c+\epsilon)^2}$.*

The proof of Theorem 2 can be found in the Supplementary Material and here we present the key ideas. The first part shows that the expected stabilization time is bounded from above by the initial value of V divided by ϵ. To prove it, we show that the stabilization time gives rise to a stopping time in the probability space of all trajectories that start in $\mathbf{x}_0$. We then observe that the RSM V satisfies the expected decrease condition until $T_{\mathcal{X}_s}$ is exceeded and use the results from probability theory on the convergence of RSMs to conclude the bound on the expected value of this stopping time.

The second part shows a bound on the probability that the stabilization time exceeds a threshold t, and it follows immediately from the first part by an application of Markov's inequality. Note that this bound decays linearly in t, as $t \to \infty$.

Finally, the third part shows an asymptotically tighter bound with the decay in t being exponential, for systems that have c-bounded differences. The proof follows by an application of Azuma's inequality (Azuma 1967) which is a classical result from martingale theory and which we also include in the Supplementary Material.

Method for Stability Verification

In this section, we present our method for verifying a.s. asymptotic stability of a given region via RSM computation. Our method consists of two modules which alternate within a loop: the *learner* and the *verifier*. In each loop iteration, the learner first learns an RSM candidate in the form of a neural network. The candidate is then passed to the verifier, which checks whether the learned candidate is indeed an RSM. If the answer is positive, the verifier terminates the loop and concludes the system's a.s. asymptotic stability. Otherwise, the verifier computes a set of counterexamples which show that the candidate is not an RSM and passes it to the learner, which then proceeds with the next learning iteration. This process is repeated until either a learned candidate is verified or a given timeout is reached.

We consider a discrete-time stochastic dynamical system defined by a dynamics function f, a policy π and a probability distribution d with model assumptions as in the previous sections, and $\mathcal{X}_s \subseteq \mathcal{X}$ which is closed under system dynamics and has non-empty interior. The rest of this section describes the details behind our method for stability verification. The algorithm is presented in Algorithm 1.

Discretization and initial sampling

Recall, an RSM V needs to satisfy the expected decrease condition in eq. (1) at each point in $\mathcal{X}\backslash\mathcal{X}_s$. However, one of the main difficulties in verifying this condition when V has a neural network form is that it is not clear how to compute a closed form for the expected value of V at a successor system state. In order to be able to verify neural network RSM candidates, our method discretizes the state space and then verifies the expected decrease condition only at the states in the discretization (which we will show to be possible due to f, π and V all being Lipschitz continuous and $\mathcal{X}$ being compact). The *discretization* $\tilde{\mathcal{X}}$ of $\mathcal{X}\backslash\mathcal{X}_s$ satisfies the property that, for each $\mathbf{x} \in \mathcal{X}$, there is $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ with $||\mathbf{x}-\tilde{\mathbf{x}}||_1 < \tau$, with τ an algorithm parameter that we call the *mesh* of $\tilde{X}$. Since $\mathcal{X}$ is compact and so $\mathcal{X}\backslash\mathcal{X}_s$ is bounded, the discretization consists of finitely many states.

The method also initializes the collection of pairs $\mathcal{D} = \{(\mathbf{x}, \mathcal{D}_\mathbf{x}) \mid \mathbf{x} \in \tilde{\mathcal{X}}\}$, where each $\mathcal{D}_\mathbf{x}$ consists of N successor states of $\mathbf{x}$ obtained by independent sampling. Here, $N \in \mathbb{N}$ is an algorithm parameter. The collection $\mathcal{D}$ will be used to approximate expected values at successor states for each $\mathbf{x}$ in $\tilde{\mathcal{X}}$ in the loss function used by the learner.

Verifier

In order to motivate the form of the loss function used by the learner, we first describe the verifier module of our algorithm. For a neural network V to be an RSM as in Definition 2, it needs to be (1) continuous, (2) nonnegative at each state, and (3) to satisfy the expected decrease condition in eq. (1) for each state in $\mathcal{X}\backslash\mathcal{X}_s$. Since V is a neural network we already know that it is a continuous function. Moreover, since $\mathcal{X}$ is compact and V is continuous, the function V admits a finite global lower bound $-m \in \mathbb{R}$. Hence, if we verify that V satisfies the expected decrease condition, we may consider the function $V'(\mathbf{x}) = V(\mathbf{x}) + m$ which is in addition nonnegative and thus an RSM to conclude a.s. asymptotic stability of $\mathcal{X}_s$. Therefore, the verifier only needs to check that V satisfies the expected decrease condition in eq. (1) for each state in $\mathcal{X}\backslash\mathcal{X}_s$, from which it immediately follows that V' is an RSM.

As explained above, checking this for each state in $\mathcal{X}\backslash\mathcal{X}_s$ is not feasible since we cannot compute a closed form for the expected value of V at a successor system state. Instead, we show that it is sufficient to check a slightly stricter condition

Algorithm 1: Verification of a.s. asymptotic stability

```
Input Dynamics function $f$, policy $\pi$, disturbance distribution $d$, region $\mathcal{X}_s \subseteq \mathcal{X}$, Lipschitz constants $L_f$, $L_\pi$ parameters $\tau > 0$, $N \in \mathbb{N}$, $\lambda > 0$
$\tilde{\mathcal{X}} \leftarrow$ discretization of $\mathcal{X}\backslash\mathcal{X}_s$ with mesh $\tau$
for $\mathbf{x}$ in $\tilde{\mathcal{X}}$ do
  $\mathcal{D}_\mathbf{x} \leftarrow N$ sampled successor states of $\mathbf{x}$
end for
while timeout not reached do
  $V \leftarrow$ trained candidate by minimizing the loss in eq. (4)
  $L_V \leftarrow$ Lipschitz constant of $V$
  $K \leftarrow L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$
  if $\exists \mathbf{x} \in \tilde{\mathcal{X}}$ s.t. $\mathbb{E}_{\omega\sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))] \geq V(\mathbf{x}) - \tau \cdot K$ then
    $\mathcal{D}_\mathbf{x} \leftarrow$ add $N$ sampled successor states of $\mathbf{x}$
  else
    Return A.s. asymptotically stable
  end if
end while
Return Unknown
```

on states in the discretization $\tilde{\mathcal{X}}$. Let L_f, L_π and L_V be the Lipschitz constants of f, π and the candidate function V, respectively. We assume that the Lipschitz constant for the dynamics function f and the policy π are provided, and use the method of (Szegedy et al. 2014) to compute the Lipschitz constant of the neural network candidate V (and also of π, in cases when π is a neural network policy). Then define

$$K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1). \tag{2}$$

In order to verify that V satisfies the expected decrease condition in eq. (1) for each state in $\mathcal{X}\backslash\mathcal{X}_s$, the verifier checks for each $\mathbf{x}$ in the discretization $\tilde{\mathcal{X}}$ that

$$\mathbb{E}_{\omega\sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] < V(\mathbf{x}) - \tau \cdot K \tag{3}$$

If eq. (3) holds for each $\mathbf{x} \in \tilde{\mathcal{X}}$, the verifier concludes a.s. asymptotic stability of $\mathcal{X}_s$. Otherwise, if $\mathbf{x} \in \tilde{\mathcal{X}}$ for which eq. (3) does not hold is found, it is passed to the learner by independently sampling N successor states of $\mathbf{x}$ which are then added to the set $\mathcal{D}_\mathbf{x}$.

Theorem 3 establishes the correctness of Algorithm 1 by showing that it indeed suffices to check eq. (3) for states in the discretization. The proof uses the fact that f and π are Lipschitz continuous and that $\mathcal{X}$ is compact, and is provided in the Supplementary Material.

Theorem 3. *Suppose that the verifier in Algorithm 1 verifies that V satisfies eq. (3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$. Let $-m \in \mathbb{R}$ be such that $V(\mathbf{x}) \geq -m$ for each $\mathbf{x} \in \mathcal{X}$. Then, the function $V'(\mathbf{x}) = V(\mathbf{x}) + m$ is an RSM for $\mathcal{X}_s$. Hence, $\mathcal{X}_s$ is a.s. asymptotically stable.*

We remark that the cardinality of the discretization $\tilde{\mathcal{X}}$ grows exponentially in the dimension of the state space, which in turn implies an exponential complexity for each verification step in our algorithm. This limitation is also present in related works on stability analysis in deterministic dynamical systems (Berkenkamp et al. 2017). A potential approach to overcome the complexity bottleneck would be to discretize different dimensions and regions of the state space with a heterogeneous instead of a uniform granularity.

Expected value computation What is left to be described is how our algorithm computes the expected value in eq. (3) for a given state $\mathbf{x} \in \tilde{\mathcal{X}}$. This is *not* trivial, since V is a neural network and so we do not have a closed form for the expected value. However, we can bound the expected value via interval arithmetic. In particular, let $\mathbf{x} \in \tilde{\mathcal{X}}$ be a throughout fixed state for which we want to bound the expected value $\mathbb{E}_{\omega\sim d}[V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))]$. Our algorithm partitions the disturbance space $\mathcal{N} \subseteq \mathbb{R}^p$ into finitely many cells $\text{cell}(\mathcal{N}) = \{\mathcal{N}_1, \dots, \mathcal{N}_k\}$, with k being the number of cells. We use $\text{maxvol} = \max_{\mathcal{N}_i\in\text{cell}(\mathcal{N})} \text{vol}(\mathcal{N}_i)$ to denote the maximal volume with respect to the Lebesgue measure over $\mathbb{R}^p$ of any cell in the partition. The algorithm then bounds the expected value via

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

where $F(\omega) = V(f(\mathbf{x}, \pi(\mathbf{x}), \omega))$. Each supremum is then bounded from above via interval arithmetic by using the method of (Gowal et al. 2018). In our experimental evaluation, we observed that this method computes very tight bounds when the number of cells is sufficiently large.

Note that maxvol is not finite in cases when $\mathcal{N}$ is unbounded. In order to allow expected value computation for an unbounded $\mathcal{N}$, we first apply the probability integral transform (Murphy 2012) to each univariate probability distribution in d. Recall, in our model assumptions we assumed that d is a product of univariate distributions and our dynamics function f takes the most general form.

Learner

We now describe the learner module of our algorithm. The learner constructs an RSM candidate function as a multilayer neural network V_θ, where θ is the vector of neural network parameters. A candidate neural network is learned by minimizing the following loss function

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{RSM}}(\theta) + \lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\theta). \tag{4}$$

The first loss term $\mathcal{L}_{\text{RSM}}(\theta)$ is defined via

$$\mathcal{L}_{\text{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|}\sum_{\mathbf{x}\in\tilde{\mathcal{X}}}\Big(\max\Big\{\sum_{\mathbf{x}'\in\mathcal{D}_\mathbf{x}}\frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_\mathbf{x}|} - V_\theta(\mathbf{x}) + \tau\cdot K, 0\Big\}\Big).$$

Intuitively, for $\mathbf{x} \in \tilde{\mathcal{X}}$, the corresponding term in the sum incurs a loss whenever the condition in eq. (3) is violated. Since the closed form for the expected value in eq. (3) in terms of parameters θ cannot be computed, for each $\mathbf{x} \in \tilde{\mathcal{X}}$ we approximate it as the mean of the values of V at sampled successor states of $\mathbf{x}$ that the algorithm stores in the set $\mathcal{D}_\mathbf{x}$.

The second loss term $\lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\theta)$ is the regularization term used to incentivize that the Lipschitz constant L_{V_θ} of V_θ does not exceed some tolerable threshold, and hence to

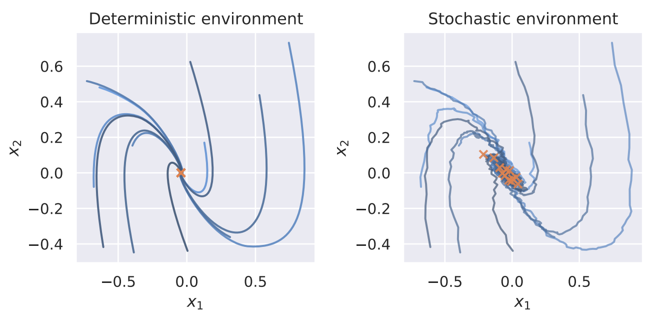

Figure 1: Example of a deterministic and a stochastic system with the same dynamics function, illustrating the difficulties of proving stability in stochastic systems. The orange markers indicate the system state after 200 time steps.

enforce that $\tau \cdot K$ in eq. (3) is sufficiently small. The constant λ is an algorithm parameter balancing the two loss terms, and we define

$$\mathcal{L}_{\text{Lipschitz}}(\theta) = \max\Big\{\frac{\delta}{\tau \cdot (L_f \cdot (L_\pi + 1) + 1)} - L_{V_\theta}, 0\Big\}.$$

Here, δ is a parameter that specifies the threshold, and L_{V_θ} in terms of θ is computed as in (Szegedy et al. 2014).

To conclude this section, we note that the loss function $\mathcal{L}(\theta)$ is nonnegative but is not necessarily equal to 0 even if V_θ satisfies eq. (3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and its Lipschitz constant is below the allowed threshold. This is because $\mathcal{L}(\theta)$ depends on samples in $\mathcal{D}$ which are used to *approximate* the expected values in eq. (3). However, in Theorem 4 we show that the loss $\mathcal{L}(\theta) \to 0$ almost-surely as we add samples to the set $\mathcal{D}_{\mathbf{x}}$ for each $\mathbf{x} \in \tilde{\mathcal{X}}$, whenever V_θ satisfies eq. (3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and its Lipschitz constant is below the allowed threshold. The claim follows from the Strong Law of Large Numbers (Williams 1991, Section 12.10) and the proof can be found in the Supplementary Material.

Theorem 4. *Let $M = \min_{\mathbf{x}\in\tilde{\mathcal{X}}} |\mathcal{D}_{\mathbf{x}}|$. If V_θ satisfies eq. (3) for each $\mathbf{x} \in \tilde{\mathcal{X}}$ and if $L_{V_\theta} \leq \delta/(\tau \cdot (L_f \cdot (L_\pi + 1) + 1))$, then $\lim_{M\to\infty} \mathcal{L}(\theta) = 0$ holds almost-surely.*

Learning stable policies

While in this work we focus on the stability verification problem for a given control policy, our approach can also be adapted to the setting in which we want to *learn* a stable neural network policy for the region $\mathcal{X}_s$ together with a formal certificate for the a.s. asymptotic stability of $\mathcal{X}_s$. This can be done by replacing the loss function in eq. (4) with

$$\mathcal{L}(\theta, \mathbf{u}) = \mathcal{L}_{\text{RSM}}(\theta, \mathbf{u}) + \lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\theta, \mathbf{u})$$

where $\mathbf{u}$ is now a vector of policy parameters while θ is again a vector of neural network parameters for the RSM candidate. The correctness of our algorithm proved in Theorem 3 then ensures that any learned and verified control policy is indeed stable. Note that this modified algorithm does not try to optimize the expected reward obtained by the learned

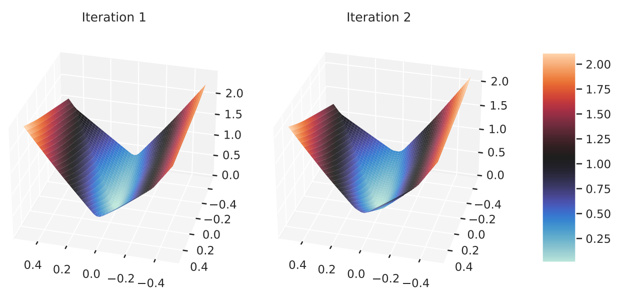

Figure 2: Learned RSM candidates after 1 and 2 iterations of our algorithm for the stochastic inverted pendulum task. The candidate on the left violates the expected decrease condition while the function of the right is a verified RSM.

policy, but only ensures stability. Exploring ways to learn a stable policy while simultaneously maximizing the expected reward is an interesting direction of future work.

Experiments

We validate our algorithm empirically on two RL benchmark environments. Our first benchmark is a two-dimensional dynamical system of the form $\mathbf{x}_{t+1} = A\mathbf{x}_t + Bg(\mathbf{u}_t) + \omega$, where ω is a disturbance vector sampled from a zero-mean triangular distribution. The function g clips the control action to stay within the interval $[-1, 1]$. The matrices A and B are provided in the Supplementary Material.

Our second benchmark is the inverted pendulum problem (Brockman et al. 2016). Contrarily to the standard inverted pendulum task, which has deterministic dynamics, we consider a more difficult stochastic variant. The system has two state variables x_1 and x_2 which represent the angle and the angular velocity of the pendulum. The objective of this task is to balance the pendulum in an upright position through control actions in the form of a torque that is applied to the pendulum. Our stochastic variant of the task applies a zero-mean triangular noise to both state variables.

For each RL task, we consider the state space $\mathcal{X} = \{\mathbf{x} \mid ||\mathbf{x}||_1 \leq 0.5\}$ and train a control policy comprised of two hidden layers with 128 ReLU units each by using proximal policy optimization (Schulman et al. 2017), while applying our Lipschitz regularization to keep the Lipschitz constant of the policy within a reasonable bound. We then run our algorithm to verify that the region $\mathcal{X}_s = \{\mathbf{x} \mid ||\mathbf{x}||_1 \leq 0.2\}$ is a.s. asymptotically stable. Our RSM neural networks consist of one hidden layer with 128 ReLU units.

Example trajectories of a policy trained for the first benchmark with the deterministic ($\omega = 0$) and stochastic dynamics are shown in Figure 1. The policy stabilizes the deterministic system in a single point, however this is not the case for the stochastic system. This illustrates the intricacies of verifying stability in stochastic systems, and justifies our choice to consider stabilizing regions with non-empty interiors.

Our method could successfully learn and verify RSMs for both systems within a reasonable time frame. The runtime statistics are shown in Table 1. The final RSM neural network for the inverted pendulum task is shown in Figure 2.

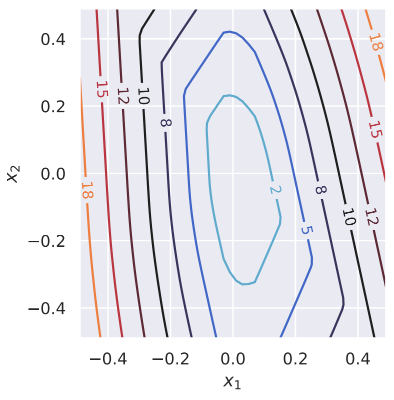

Figure 3: Contour lines of the convergence time bounds obtained from the RSM on the inverted pendulum task.

Environment	Iters.	Mesh (τ)	Runtime
2D system	4	0.002	559
Inverted pendulum	2	0.01	176

Table 1: Number of learner-verifier loop iterations, mesh of the discretization used by the verifier, and the total algorithm runtime (in seconds).

We further computed the ϵ of the RSM network according to Definition 2 for the inverted pendulum task to obtain the convergence time bounds as outlined in Theorem 2. The resulting convergence time bounds are shown in Figure 3.

We perform an additional experiment to study the effectiveness of our method for computing bounds on the expected value a neural network. In particular, we sample 100 random states of the inverted pendulum environment. For each sampled state, we use our method to compute the bound on the expected value of the final RSM neural network (shown in Figure 2) in a successor system state, with different sizes of the cell partition. We then compute the ground-truth of the expected value by averaging the RSM value at 1000 independently sampled successor states (Strong Law of Large Numbers). The results shown in Figure 4 indicate that, even with a modest size of the cell partition, a tight bound can be obtained. As the partition is further refined, the expected value bound converges to the ground-truth.

Conclusion

In this work, we study the stability verification problem for nonlinear stochastic control systems. We show, for the first time, that ranking supermartingales (RSMs) provide a formal certificate for a.s. asymptotic stability as well as guarantees on the stabilization time. We then present a method for a.s. asymptotic stability verification which learns and verifies an RSM in the form of a neural network. In order to design the verifier module of our algorithm, we propose a method for efficiently computing tight bounds on the expected value of a neural network function over a probability distribution. Finally, we validate our approach experimentally on a set of nonlinear stochastic RL environments with neural network policies. There are several interesting venues for future work. While we showed how our verification algorithm can be adapted to also learn a stabilizing policy, this adaptation does not try to optimize the learned policy. Exploring ways to learn high performing stabilizing policies is an interesting direction. A limiting factor of our algorithm for computing RSMs is that the complexity of the verification step grows exponentially with the dimension of the state space. In order to overcome this limitation and improve scalability, future work may consider different ways to discretize the state space, such as an on-demand discretization that does not use the same granularity at all parts of the state space. Another future research direction is to integrate our approach to safe exploration RL in systems with stochastic environments.

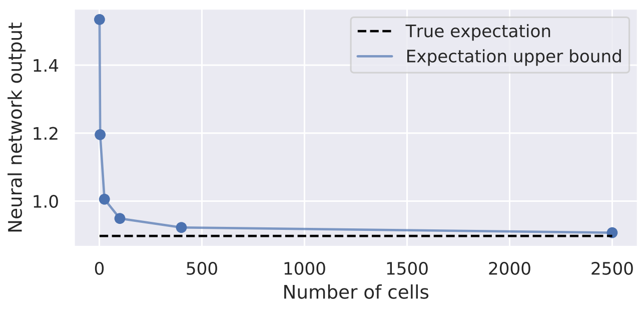

Figure 4: Comparison of our method for bounding the expected value of an RSM neural network with the ground-truth expected value on 100 randomly sampled states of the inverted pendulum environment.

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

Abate, A.; Ahmed, D.; Giacobbe, M.; and Peruffo, A. 2021. Formal Synthesis of Lyapunov Neural Networks. *IEEE Control. Syst. Lett.*, 5(3): 773–778.

Abate, A.; Giacobbe, M.; and Roy, D. 2021. Learning Probabilistic Termination Proofs. In Silva, A.; and Leino, K. R. M., eds., *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II*, volume 12760 of *Lecture Notes in Computer Science*, 3–26. Springer.

Ahmadi, A. A.; Krstic, M.; and Parrilo, P. A. 2011. A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In *50th IEEE Conference on Decision and Control and European Control Conference, 11th European Control Conference, CDC/ECC 2011, Orlando, FL, USA, December 12-15, 2011*, 7579–7580. IEEE.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565.

Azuma, K. 1967. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3): 357–367.

Berkenkamp, F. 2019. Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics.

Berkenkamp, F.; Turchetta, M.; Schoellig, A. P.; and Krause, A. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 908–918.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Chakarov, A.; and Sankaranarayanan, S. 2013. Probabilistic Program Analysis with Martingales. In Sharygina, N.; and Veith, H., eds., *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, 511–526. Springer.

Chang, Y.; Roohi, N.; and Gao, S. 2019. Neural Lyapunov Control. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 3240–3249.

Chatterjee, K.; Fu, H.; and Goharshady, A. K. 2016. Termination Analysis of Probabilistic Programs Through Positivstellensatz's. In Chaudhuri, S.; and Farzan, A., eds., *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, volume 9779 of *Lecture Notes in Computer Science*, 3–22. Springer.

Chatterjee, K.; Fu, H.; Novotný, P.; and Hasheminezhad, R. 2016. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In Bodík, R.; and Majumdar, R., eds., *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, 327–342. ACM.

Crespo, L. G.; and Sun, J. 2003. Stochastic optimal control via Bellman's principle. *Autom.*, 39(12): 2109–2114.

Dalal, G.; Dvijotham, K.; Vecerík, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe Exploration in Continuous Action Spaces. *ArXiv*, abs/1801.08757.

Dutta, S.; Chen, X.; and Sankaranarayanan, S. 2019. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In Ozay, N.; and Prabhakar, P., eds., *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, 157–168. ACM.

Fan, J.; Huang, C.; Chen, X.; Li, W.; and Zhu, Q. 2020. ReachNN*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. In Hung, D. V.; and Sokolsky, O., eds., *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, volume 12302 of *Lecture Notes in Computer Science*, 537–542. Springer.

Fioriti, L. M. F.; and Hermanns, H. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. In Rajamani, S. K.; and Walker, D., eds., *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, 489–501. ACM.

Floyd, R. W. 1967. Assigning Meanings to Programs. *Proceedings of Symposium on Applied Mathematics*, 19: 19–32.

Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T. A.; and Kohli, P. 2018. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *CoRR*, abs/1810.12715.

Gruenbacher, S.; Cyranka, J.; Lechner, M.; Islam, M. A.; Smolka, S. A.; and Grosu, R. 2020. Lagrangian Reachtubes: The Next Generation. In *CDC*, 1556–1563. IEEE.

Gruenbacher, S.; Lechner, M.; Hasani, R.; Rus, D.; Henzinger, T. A.; Smolka, S.; and Grosu, R. 2021. Gotube: Scalable stochastic verification of continuous-depth models. *arXiv preprint arXiv:2107.08467*.

Henrion, D.; and Garulli, A. 2005. *Positive polynomials in control*, volume 312. Springer Science & Business Media.

Huang, C.; Fan, J.; Li, W.; Chen, X.; and Zhu, Q. 2019. ReachNN: Reachability Analysis of Neural-Network Controlled Systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s): 106:1–106:22.

Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G. J.; and Lee, I. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In Ozay, N.; and Prabhakar, P., eds., *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, 169–178. ACM.

James, S.; Davison, A. J.; and Johns, E. 2017. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*, 334–343. PMLR.

Jarvis-Wloszek, Z.; Feeley, R.; Tan, W.; Sun, K.; and Packard, A. 2003. Some controls applications of sum of squares programming. In *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475)*, volume 5, 4676–4681. IEEE.

Khalil, H. 2002. *Nonlinear Systems*. Pearson Education. Prentice Hall.

Koller, T.; Berkenkamp, F.; Turchetta, M.; and Krause, A. 2018. Learning-Based Model Predictive Control for Safe Exploration. *2018 IEEE Conference on Decision and Control (CDC)*, 6059–6066.

Kolter, J. Z.; and Manek, G. 2019. Learning Stable Deep Dynamics Models. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 11126–11134.

Kushner, H. J. 1965. On the stability of stochastic dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 53(1): 8.

Kushner, H. J. 2014. A partial history of the early development of continuous-time nonlinear stochastic systems theory. *Autom.*, 50(2): 303–334.

Lawrence, N. P.; Loewen, P. D.; Forbes, M. G.; Backström, J. U.; and Gopaluni, R. B. 2020. Almost Surely Stable Deep Dynamics. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Lechner, M.; Žikelić, Đ.; Chatterjee, K.; and Henzinger, T. 2021. Infinite Time Horizon Safety of Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2021.*

Liu, A.; Shi, G.; Chung, S.-J.; Anandkumar, A.; and Yue, Y. 2020. Robust Regression for Safe Exploration in Control. In *L4DC*.

Lyapunov, A. M. 1992. The general problem of the stability of motion. *International journal of control*, 55(3): 531–534.

Murphy, K. P. 2012. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press. ISBN 0262018020.

Parrilo, P. A. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-47161977-2.

Ravanbakhsh, H.; and Sankaranarayanan, S. 2019. Learning control lyapunov functions from counterexamples and demonstrations. *Auton. Robots*, 43(2): 275–307.

Richards, S. M.; Berkenkamp, F.; and Krause, A. 2018. 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*, volume 87 of *Proceedings of Machine Learning Research*, 466–476. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30. IEEE.

Tran, H.; Yang, X.; Lopez, D. M.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In Lahiri, S. K.; and Wang, C., eds., *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, volume 12224 of *Lecture Notes in Computer Science*, 3–17. Springer.

Turchetta, M.; Berkenkamp, F.; and Krause, A. 2019. Safe Exploration for Interactive Machine Learning. In *NeurIPS*.

Umlauft, J.; and Hirche, S. 2017. Learning Stable Stochastic Nonlinear Dynamical Systems. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 3502–3510. PMLR.

Vaidya, U. 2015. Stochastic stability analysis of discrete-time system using Lyapunov measure. In *American Control Conference, ACC 2015, Chicago, IL, USA, July 1-3, 2015*, 4646–4651. IEEE.

Williams, D. 1991. *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press. ISBN 978-0-521-40605-5.