Learning Stabilizing Policies in Stochastic Control Systems

Đorđe Žikelić,* **Mathias Lechner,*** **Krishnendu Chatterjee, Thomas A. Henzinger**
IST Austria
Klosterneuburg, Austria
{djordje.zikelic, mathias.lechner, krishnendu.chatterjee, tah}@ist.ac.at

Abstract

In this work, we address the problem of learning provably stable neural network policies for stochastic control systems. While recent work has demonstrated the feasibility of certifying given policies using martingale theory, the problem of how to learn such policies is little explored. Here, we study the effectiveness of jointly learning a policy together with a martingale certificate that proves its stability using a single learning algorithm. We observe that the joint optimization problem becomes easily stuck in local minima when starting from a randomly initialized policy. Our results suggest that some form of pre-training of the policy is required for the joint optimization to repair and verify the policy successfully.

1 Introduction

Reinforcement learning (RL) has achieved impressive results in many domains in which the goal is to optimize expected reward (Sutton & Barto, 2018). This has fueled the desire to use RL in robotics and for control of non-linear systems. However, using RL in safety-critical domains such as autonomous driving requires formal safety guarantees as consequences of unsafe behavior could be disastrous (Amodei et al., 2016). Verification of systems with learned components and learning with safety guarantees have thus become active research topics (García & Fernández, 2015).

In this work, we consider the problem of learning stabilizing policies for stochastic control systems. Stability is one of the most basic and important safety properties of systems. It requires a system to be able to reach and stay within a region that is known to be safe, i.e., a stabilizing region, from any system state via the means of a stabilizing policy (Lyapunov, 1992). For instance, if a self-driving car were to drive at a speed that exceeds the allowed speed limit, then a stabilizing policy would stabilize the car back within the speed limit.

Our work builds on the recent method of (Lechner et al., 2021) for verifying that a *pre-learned* control policy ensures probability 1 stability of the system. Stability is verified by learning a *ranking supermartingale (RSM)*, which can be viewed as a stochastic extension of Lyapunov functions for deterministic systems (Khalil, 2002). The method of (Lechner et al., 2021) makes very mild assumptions on the system which make it applicable to a wide range of systems and control policies. However, the key drawback of the method is that it can only be used to *verify* probability 1 stability under a given policy. If the pre-learned policy does not ensure stability, the method in its current form provides no means to *learn* a new policy in a way that would provide stability guarantees. (Lechner et al., 2021) briefly discuss the possibility of extending the method to safe learning by jointly learning the policy and the RSM, however this idea is not empirically evaluated.

The goal of this work is to empirically evaluate this extension in order to understand the difficulties of learning stabilizing policies as opposed to the stability verification of pre-learned policies. In particular, our results show that a good initialization of the policy is *essential*, as otherwise the learning algorithm easily gets stuck in a local minimum at which the learned policy cannot be verified to be ensure stability. We propose a simple-form reward function that intuitively captures the stability task and use proximal policy optimization (PPO) (Schulman et al., 2017) with respect to the reward

---

*Equal Contribution.

function to initialize the policy. Our experimental results show that such an initialization leads to significant improvement in performance over a naive initialization of the policy.

Related work Learning methods present a promising approach to solving complex non-linear control tasks due to their generality. Several recent works propose learning stabilizing policies for *deterministic* control problems together with a Lyapunov function that certifies stability (Richards et al., 2018; Chang et al., 2019; Abate et al., 2021). In particular, (Chang et al., 2019; Abate et al., 2021) propose a learner-verifier framework similar to that of (Lechner et al., 2021) and the one presented in our work. There is a rich body of literature on theoretical stability analysis in stochastic dynamical systems and control problems, see (Kushner, 2014) for a survey, however very few works consider automated control with stability guarantees. Control with finite-time horizon reachability constraints has been considered in (Crespo & Sun, 2003; Soudjani et al., 2015; Lavaei et al., 2020; Cauchi & Abate, 2019; Vinod et al., 2019). With the exception of (Lechner et al., 2021) that was discussed above and that our work builds on, the work of (Vaidya, 2015) is to our best knowledge the only work that considers control with infinite-time horizon stability constraints and that provides formal guarantees, however the stability certificates are computed in a form that makes them piecewise constant and the work verifies a weaker notion of stability called "coarse stochastic stability".

2 RANKING SUPERMARTINGALES FOR STABILITY VERIFICATION

We consider a discrete-time stochastic dynamical system defined by the equation
$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \pi(\mathbf{x}_t), \omega_t),$$
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 policy and ω_t is a stochastic disturbance vector sampled according to a probability distribution d over $\mathcal{N}$. Here, $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space of the system, $\mathcal{U} \subseteq \mathbb{R}^m$ is the action space and $\mathcal{N} \subseteq \mathbb{R}^p$ is the stochastic disturbance space, all of which are required to be Borel-measurable. We assume that $\mathcal{X} \subseteq \mathbb{R}^n$ is compact and that f and π are Lipschitz continuous, which are common assumptions in control theory.

Almost-sure asymptotic stability There are several notions of stability in stochastic systems, and we consider almost-sure asymptotic stability (Kushner, 1965). Let $\mathcal{X}_s \subseteq \mathcal{X}$ is a non-empty Borel-measurable subset that is closed under system dynamics so that the system cannot leave it once reached, i.e., for every $\mathbf{x} \in \mathcal{X}_s$ we have that $f(\mathbf{x}, \pi(\mathbf{x}), \omega) \in \mathcal{X}_s$ for any $\omega \in \mathcal{N}$. We say that $\mathcal{X}_s$ is *almost-surely (a.s.) asymptotically stable*, if for each initial state $\mathbf{x}_0 \in \mathcal{X}$ we have that
$$\mathbb{P}_{\mathbf{x}_0}\Big[\lim_{t \to \infty} \Big(\inf_{\mathbf{y} \in \mathcal{X}_s} ||\mathbf{x}_t - \mathbf{y}||_1 \Big) = 0 \Big] = 1.$$

Here, $\mathbb{P}_{\mathbf{x}_0}$ is the probability measure over the set of all system trajectories that start in the initial state $\mathbf{x}_0$ defined by the MDP semantics of the system (Puterman, 1994). Our definition slightly differs from that of (Kushner, 1965) which considers the special case $\mathcal{X}_s = \{\mathbf{0}\}$. The reason is that, analogously to (Lechner et al., 2021) and to the existing works on learning stabilizing policies in deterministic systems (Berkenkamp et al., 2017; Richards et al., 2018; Chang et al., 2019), we need to assume the existence of an open neighbourhood of the origin that is known to be stable for learning to be stable, which we ensure by assuming that $\mathcal{X}_s$ is closed under system dynamics.

Ranking supermartingales Intuitively, a ranking supermartingale for the set $\mathcal{X}_s$ is a nonnegative continuous function $V : \mathcal{X} \to \mathbb{R}$ which strictly decreases in expected value upon every one-step execution of the system, until the set $\mathcal{X}_s$ is reached. The name comes from its connection to supermartingale processes in probability theory (Williams, 1991).

Definition 1 *A ranking supermartingale (RSM) for $\mathcal{X}_s$ is a nonnegative continuous function $V : \mathcal{X} \to \mathbb{R}$ for which there exists $\epsilon > 0$ such that, for every $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_s$, we have that*
$$\mathbb{E}_{\omega \sim d}[V(\mathbf{x}, \pi(\mathbf{x}), \omega)] \leq V(\mathbf{x}) - \epsilon.$$

The following theorem is the key result on the use of RSMs for a.s. asymptotic stability analysis.

Theorem 1 (Lechner et al. (2021)) *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.*

3 ALGORITHM FOR LEARNING RANKING SUPERMARTINGALES

We now present the details behind our learner-verifier framework for simultaneously learning a stabilizing policy π_μ and an RSM V_θ for $\mathcal{X}_s$, both of which are parametrized as neural networks with parameters μ and θ. The verifier module is analogous to that of (Lechner et al., 2021) for verifying a learned RSM candidate under the fixed policy, so we only provide a high level presentation of it. The novelty of our algorithm lies in modifying the learner in a way which allows learning π_μ and V_θ simultaneously as briefly discussed in (Lechner et al., 2021), and in observing the importance of algorithm initialization. The algorithm pseudocode is provided in Appendix A.1.

Initialization Our algorithm first initializes the policy π_μ by using proximal policy optimization (PPO) (Schulman et al., 2017). In particular, the algorithm first induces an MDP from the system and defines a reward function $r : \mathcal{X} \to [0, 1]$ via $r(\mathbf{x}) := \mathbb{I}[\mathcal{X}_s](\mathbf{x})$. The first learner-verifier iteration is then started by a call to the verifier, followed by a call to the learner. This contrasts the verification algorithm of (Lechner et al., 2021), which starts with the call to the learner. Our experimental results will show a naive initialization often does not allow learning a stabilizing policy. The importance of initialization was also observed in (Chang et al., 2019) which showed that a proper initialization of the policy is necessary for a similar method for stability verification of deterministic neural network-controlled systems. Specifically, (Chang et al., 2019) initialized their linear policy with the LQR solution of the system linearized at the origin. However, it is not clear how one could linearize stochastic dynamics with non-additive kinds of stochastic disturbance or systems that are highly non-linear, whereas our initialization is applicable to general classes of stochastic disturbances and system dynamics.

Verifier Due to the continuity of V_θ and the compactness of $\mathcal{X}$, V_θ is bounded from below and therefore may be increased by a constant value in order to make it nonnegative. Hence, the verifier only needs to check whether the RSM candidate satisfies the expected decrease condition. To do this, the algorithm computes a discretization the part of the state space $\mathcal{X} \backslash \mathcal{X}_s$. A *discretization* of $\mathcal{X} \backslash \mathcal{X}_s$ with *mesh* $\tau > 0$ is a set $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ such that, for every $\mathbf{x} \in \mathcal{X} \backslash \mathcal{X}_s$, there exists $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ such that $||\mathbf{x} - \tilde{\mathbf{x}}||_1 < \tau$. As f is assumed to be Lipschitz continuous and as π_μ and V_θ are neural networks with continuous activation functions thus also Lipschitz continuous (Szegedy et al., 2014), we can show that the expected decrease condition may be checked by checking a slightly stricter condition at the discretization points. In particular, let L_V, L_f and L_π being Lipschitz constants of V_θ, f and π_μ. We assume that L_f is provided, and use the method of (Szegedy et al., 2014) to compute L_V and L_π. Let $K = L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$. Then, if we show that for every $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ we have

$$\mathbb{E}_{\omega \sim d}\Big[V_\theta\Big(f(\tilde{\mathbf{x}}, \pi_\mu(\tilde{\mathbf{x}}), \omega)\Big)\Big] < V_\theta(\tilde{\mathbf{x}}) - \tau \cdot K, \tag{1}$$

then there exists $\epsilon > 0$ for which V_θ satisfied the expected decrease condition (Lechner et al., 2021, Theorem 3). Hence, the verifier may check that V_θ is an RSM by simply checking whether the condition in eq. (1) is satisfied at each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$. The expected value on the left hand side of the above inequality is computed by using interval arithmetic abstract interpretation, see (Lechner et al., 2021) for details. If the verifier concludes that eq. (1) is satisfied at each $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$, the algorithm concludes a.s. asymptotic stability of $\mathcal{X}_s$. Otherwise, for each state $\tilde{\mathbf{x}} \in \tilde{\mathcal{X}}$ at which the condition is violated, the verifier samples N successor states of $\tilde{\mathbf{x}}$ where N is an algorithm parameter, and adds them to a set $\mathcal{D}_{\tilde{\mathbf{x}}}$ that will be used by the learner to approximate the expected value $\mathbb{E}_{\omega \sim d}[V_\theta(f(\tilde{\mathbf{x}}, \pi_\mu(\tilde{\mathbf{x}}), \omega))]$.

Learner A policy π_μ and an RSM candidate V_θ are learned by minimizing the loss function

$$\mathcal{L}(\mu, \theta) = \mathcal{L}_{\text{RSM}}(\mu, \theta) + \lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\mu, \theta). \tag{2}$$

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

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

Intuitively, each term in the sum is used to guide the learner towards learning an RSM candidate that satisfies the expected decrease condition at the state $\tilde{\mathbf{x}} \in \mathcal{X}$. Since we cannot compute the closed form expression for the expected value of V_θ over the successor states of $\tilde{\mathbf{x}}$, we use sampled successor states that were computed by the verifier at counterexample states. Such an on-demand

Table 1: Number of certified stable instances with different number of PPO pre-training iterations.

Number of PPO iterations	μ fixed	μ learnable
0	0/0	0/0
20	0/10	1/10
30	6/10	7/10
40	9/10	10/10
50	10/10	10/10

(a) 20 PPO pre-training iterations (failed)   (b) 30 PPO pre-training iterations (success)
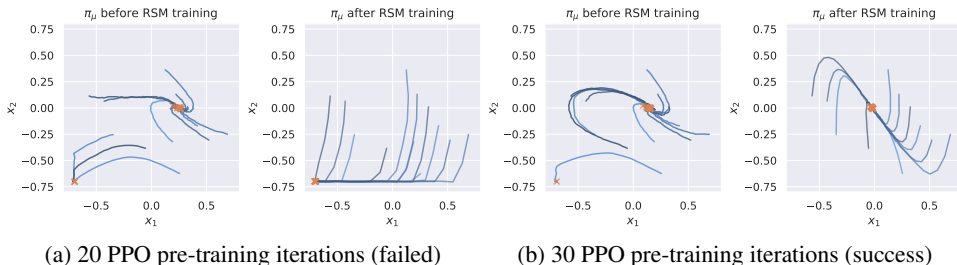
Figure 1: System trajectories before and after RSM learning with Algorithm 1. The orange markers indicate the trajectories' terminal states after 200 steps. The policy is pre-trained with 20 (1a) and 30 (1b) PPO iterations. The RSM learning process in 1a is stuck in a local minimum where the RSM decrease condition is satisfied in all except one state (left bottom). The RSM learning process in 1b terminated successfully, proving that policy stabilizes the system to a set around the origin.

sampling of successor states improves the scalability of our algorithm. The second loss term $\lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\mu, \theta)$ in the sum is a regularization term used to guide the learner towards learning π_μ and V_θ with small Lipschitz constant, so as to make the term $\tau \cdot K$ in the condition checked by the verifier as small as possible. We defer the details on the regularization term to Appendix A.2.

4 EXPERIMENTAL EVALUATION

This section investigates the importance of policy initialization.

First, we consider the performance of both the algorithm of (Lechner et al., 2021) which keeps the policy's parameters μ frozen and its extension to an algorithm that learns a stabilizing policy by making μ trainable. We repeat this experiment for a different number of policy pre-training iterations with PPO and random seeds (n=10). We then report the algorithm's success, i.e., whether a valid RSM is found. We use 2D system benchmark environment of Lechner et al. (2021) for our experimental study. Details on the experiment setup can be found in Appendix A.3 and A.4.

The results in Table 1 show that, without a proper initialization, the algorithm cannot learn a safe policy and a valid RSM regardless of whether the policy's parameters μ are fixed or trainable. After a short pre-training of 20 PPO iterations, the algorithm could only learn a stable policy in 1 out of 10 trials. Conversely, if the initially provided policy is well-trained, the algorithm can prove its stability even if the policies' parameters are frozen for all ten tested runs. Aggregated over all 40 runs, only in 3 cases (7.5%) the joint training of the policy and the RSM helped over just training an RSM with a fixed pre-trained policy.

In Figure 1, we analyze the difficulties of the joint optimization problem. Figure 1a shows that the training algorithm gets stuck in a local minimum without a good initialization. The expected decrease condition is fulfilled everywhere on the state space except in a single point (bottom left). Consequently, the system in Figure 1a experiences a low training loss but our algorithm learns an invalid RSM. Escaping this sub-optimal minimum requires the RSM and the policy to change their behavior on the entire state space, making it improbable using stochastic gradient descent. Figure 1b visualizes an example when the pre-trained policy is unsafe but good enough for our algorithm to repair it to a certifiable safe policy.

REFERENCES

Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control. Syst. Lett.*, 5(3):773–778, 2021. doi: 10.1109/LCSYS. 2020.3005328. URL https://doi.org/10.1109/LCSYS.2020.3005328.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 908–918, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/766ebcd59621e305170616ba3d3dac32-Abstract.html.

Nathalie Cauchi and Alessandro Abate. 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.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (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*, pp. 3240–3249, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2647c1dba23bc0e0f9cdf75339e120d2-Abstract.html.

Luis G. Crespo and Jian-Qiao Sun. Stochastic optimal control via bellman's principle. *Autom.*, 39(12):2109–2114, 2003. doi: 10.1016/S0005-1098(03)00238-3. URL https://doi.org/10.1016/S0005-1098(03)00238-3.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015. URL http://dl.acm.org/citation.cfm?id=2886795.

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

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

Harold J. Kushner. A partial history of the early development of continuous-time nonlinear stochastic systems theory. *Autom.*, 50(2):303–334, 2014. doi: 10.1016/j.automatica.2013.10.013. URL https://doi.org/10.1016/j.automatica.2013.10.013.

Abolfazl Lavaei, Mahmoud Khaled, Sadegh Soudjani, and Majid Zamani. AMYTISS: parallelized automated controller synthesis for large-scale stochastic systems. In Shuvendu K. Lahiri and Chao Wang (eds.), *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pp. 461–474. Springer, 2020. doi: 10.1007/978-3-030-53291-8_24. URL https://doi.org/10.1007/978-3-030-53291-8_24.

Mathias Lechner, Ðorđe Žikelić, Krishnendu Chatterjee, and Thomas A. Henzinger. Stability verification in stochastic control systems via neural network supermartingales, 2021.

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

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887.

Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. 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*, pp. 466–476. PMLR, 2018. URL `http://proceedings.mlr.press/v87/richards18a.html`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sadegh Esmaeil Zadeh Soudjani, Caspar Gevaerts, and Alessandro Abate. FAUST ^{2}: Formal abstractions of uncountable-state stochastic processes. In Christel Baier and Cesare Tinelli (eds.), *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pp. 272–286. Springer, 2015. doi: 10.1007/978-3-662-46681-0_23. URL `https://doi.org/10.1007/978-3-662-46681-0_23`.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Umesh Vaidya. Stochastic stability analysis of discrete-time system using lyapunov measure. In *American Control Conference, ACC 2015, Chicago, IL, USA, July 1-3, 2015*, pp. 4646–4651. IEEE, 2015. doi: 10.1109/ACC.2015.7172061. URL `https://doi.org/10.1109/ACC.2015.7172061`.

Abraham P. Vinod, Joseph D. Gleason, and Meeko M. K. Oishi. Sreachtools: a MATLAB stochastic reachability toolbox. In Necmiye Ozay and Pavithra Prabhakar (eds.), *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019*, pp. 33–38. ACM, 2019. doi: 10.1145/3302504.3311809. URL `https://doi.org/10.1145/3302504.3311809`.

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

Network	Hidden size	Learning rate	Lipschitz threshold	Output activation
Policy π_μ	128, 128	$5 \cdot 10^{-5}$	3.0	-
RSM V_θ	128, 128	$5 \cdot 10^{-4}$	8.0	Softplus
Value function (for PPO)	128, 128	$5 \cdot 10^{-4}$	-	-

Table 2: Hyperparameters of the networks used in our experiments. All networks use a ReLU activation function in their hidden layers. The regularization factor λ was set to 10^{-3}.

A   APPENDIX

A.1   ALGORITHM PSEUDOCODE

Algorithm 1 Learning a stabilizing policy

1: **Input** Dynamics function f, disturbance distribution d, region $\mathcal{X}_s \subseteq \mathcal{X}$
2:      Lipschitz constant L_f, parameters $\tau > 0$, $N \in \mathbb{N}$, $\lambda > 0$
3: $\pi_\mu \leftarrow$ policy initialized via PPO
4: $\tilde{\mathcal{X}} \leftarrow$ discretization of $\mathcal{X} \backslash \mathcal{X}_s$ with mesh τ
5: **for** $\mathbf{x}$ in $\tilde{\mathcal{X}}$ **do**
6:      $\mathcal{D}_\mathbf{x} \leftarrow N$ sampled successor states of $\mathbf{x}$
7: **end for**
8: $V_\theta \leftarrow$ trained RSM candidate by minimizing the loss in eq. (2) with parameters μ fixed
9: **while** timeout not reached **do**
10:      $L_\pi, L_V \leftarrow$ Lipschitz constants of π_μ, V_θ
11:      $K \leftarrow L_V \cdot (L_f \cdot (L_\pi + 1) + 1)$
12:      **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**
13:           $\mathcal{D}_\mathbf{x} \leftarrow$ add N sampled successor states of $\mathbf{x}$
14:           $\pi_\mu, V_\theta \leftarrow$ trained policy and RSM candidate by minimizing the loss in eq. (2)
15:      **else**
16:           **Return** A.s. asymptotically stable
17:      **end if**
18: **end while**
19: **Return** Unknown

A.2   LIPSCHITZ LOSS TERM

The loss term $\lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\mu, \theta)$ is the regularization term used to guide the learner towards learning a policy π_μ and an RSM candidate V_θ such that the Lipschitz constant L_θ does not exceed a tolerable threshold $\delta > 0$. This is done in order to make the term $\tau \cdot K$ in the condition verified by the verifier sufficiently small. The constant $\lambda > 0$ is an algorithm parameter balancing the two loss terms, and we define

$$\mathcal{L}_{\text{Lipschitz}}(\mu, \theta) = \max\left\{L_V - \frac{\delta}{\tau \cdot (L_f \cdot (L_\pi + 1) + 1)}, 0\right\},$$

where the Lipschitz constants L_π and L_θ are computed as in (Szegedy et al., 2014).

A.3   TRAINING DETAILS

The networks size and training hyperparameters for all network involved in our experiments are listed in Table 2.

A.4   PPO DETAILS

Here, we list the settings used for the PPO pre-training process of the policy networks Schulman et al. (2017). In every PPO iteration we collect 30 episodes of the environment as training data in the experience buffer. The policy π_μ is made stochastic using a Gaussian distributed random variable

that is added to the policy's output, i.e., the policy predicts the mean of the Gaussian. The standard deviation of the Gaussian is annealed during the policy training process, starting from 0.5 at first PPO iteration to 0.05 at PPO iteration 50. We normalize the advantage values, i.e., the difference between the observed discounted returns and the predicted return by the value function, by subtracting the mean and dividing by the standard deviation of the advantage values of the experience buffer. The PPO clipping value ε is set to 0.2 and the discount factor γ to 0.99. In every PPO iteration, the policy is trained for 10 epochs, except for the first iteration where the network is trained for 30 epochs. An epoch corresponds to a pass over the entire data in the experience buffer, i.e., the data from the the 30 episodes. The value network is trained for 5 epochs, expect in the first PPO iteration, where the training is performed for 10 epochs. We apply the Lipschitz regularization on the policy parameters already during the PPO pre-training of the policy.