Proceedings of Machine Learning Research vol 144:1–24, 2021

Neural Lyapunov Redesign

Arash Mehrjou AMEHRJOU@ETHZ.CH
Max Planck Institute for Intelligent Systems & ETH Zürich

Mohammad Ghavamzadeh GHAVAMZA@GOOGLE.COM
Google Research

Bernhard Schölkopf BS@TUE.MPG.DE
Max Planck Institute for Intelligent Systems

Abstract

Learning controllers merely based on a performance metric has been proven effective in many physical and non-physical tasks in both control theory and reinforcement learning. However, in practice, the controller must guarantee some notion of safety to ensure that it does not harm either the agent or the environment. Stability is a crucial notion of safety, whose violation can certainly cause unsafe behaviors. Lyapunov functions are effective tools to assess stability in nonlinear dynamical systems. In this paper, we combine an improving Lyapunov function with automatic controller synthesis in an iterative fashion to obtain control policies with large safe regions. We propose a two-player collaborative algorithm that alternates between estimating a Lyapunov function and deriving a controller that gradually enlarges the stability region of the closed-loop system. We provide theoretical results on the class of systems that can be treated with the proposed algorithm and empirically evaluate the effectiveness of our method using an exemplary dynamical system.

Keywords: Lyapunov function, Controller Synthesis, Actor-Critic, Neural Networks

1. Introduction

Studying the stability region of autonomous systems and designing controllers (policies) to drive a non-autonomous system towards a target behavior are of fundamental importance in many disciplines, such as aviations (Liao and Wang, 2004), autonomous driving (Wen-Xing and Li-Dong, 2018), and robotics (Pierson and Gashler, 2017). An indisputable goal for a controller is to stabilize the system. Unlike the global nature of linear systems, stability is a local property in nonlinear systems. Knowledge of the stability region is essential in many applications, such as stability of power systems (Xin et al., 2007), design of associative memory in artificial neural networks (Hopfield, 1994), robotics (Westervelt et al., 2003), and biology (Baer et al., 2006).

Controllers that enhance the stability region of a system, also known as Region of Attraction (RoA), are highly desired as they make more clever use of the inherent nonlinear structure of the system. For example, an autonomous driving system will remain safe under more diverse and potentially harsh conditions (Imani Masouleh and Limebeer, 2018).

A great number of methods for designing a controller for nonlinear systems and determining their RoA (Isidori, 2014; Khalil and Grizzle, 2002) have been proposed in the literature mainly based on Lyapunov's theory of stability (Liapounoff, 1907). However, the design of a stabilizing controller has been ad-hoc for every class of nonlinear systems. Our work is inspired by a classic method called *Lyapunov Redesign* in control theory that uses a given Lyapunov function to design a controller such that the closed-loop system becomes stable when assessed by that Lyapunov function. Here, we relax the necessity to know the Lyapunov function by learning it together with the controller.

With the recent interest in data-driven control, machine learning tools have been employed to learn Lyapunov function for nonlinear systems (Richards et al., 2018). A given Lyapunov function is used in (Berkenkamp et al., 2016, 2017) to derive a controller with a safety guarantee. Chang et al. (2019) solve an optimization problem at every stage to find the states that violate the Lyapunov condition. Our work proposes improvements in various directions. We improve the Lyapunov learning algorithm of Richards et al. (2018) by a theoretically motivated method and show that *hyperbolicity* is a needed feature for nonlinear systems whose controllers are learned iteratively. Moreover, Berkenkamp et al. (2016, 2017) assume the Lyapunov function is given while our work co-learns the Lyapunov function and the controller. Chang et al. (2019)'s algorithm needs to be solved until the end that can be too costly as a global optimization problem has to be solved multiple times. However, our work is a growing method that improves the controller while the system is in action.

We bring together tools from control theory and machine learning to build an iterative algorithm that redesigns both controller and the Lyapunov function to enlarge the stability region. Our contributions are as follows: **1)** Improving the learning of the Lyapunov function, **2)** Interlacing the Lyapunov learning with the policy update to enlarge the RoA iteratively, and **3)** Providing theoretical results for the tractable class of systems and analysing the learning signal. Section 2 goes over the preliminary materials and Section 3 describe the problem. The proposed algorithm and its theoretical discussions are presented in Section 4. Finally, empirical evaluation comes in Section 5, followed by related work and conclusions in Section 6. To remain within the page limit, some proofs, theoretical discussions, and extended experimental results are provided in the Appendix, which can be found in the extended version of the paper `https://arxiv.org/pdf/2006.03947.pdf`. The developed software is open-sourced and is available at `https://github.com/amehrjou/neural_lyapunov_redesign`.

2. Preliminaries

Here we present the definitions, notations, and theoretical results that are used in later sections.
System: A discrete-time[1] time-invariant disturbance-free nonlinear dynamical system is described by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{1}$$

where $k \in \mathbb{Z}$ is the discrete time index, $\mathbf{x}_k \in \mathcal{X} \subseteq \mathbb{R}^d$ and $\mathbf{u}_k \in \mathcal{U} \subseteq \mathbb{R}^p$ are the state and control signals. We consider a fully observable regime, where the states are available to a feedback controller, i.e., $\mathbf{u}_k = \pi(\mathbf{x}_k)$, and π is the feedback law or policy. Hence, (1) can be written as a time-invariant autonomous (TIA) system $\mathbf{x}_{k+1} = f_\pi(\mathbf{x}_k)$ where f_π is the time-independent *dynamics* function. By assuming Lipschitz continuity for f and π, a unique solution to this system for every initial state exists that is captured by the so-called *flow* function $\Phi(\mathbf{x}, \cdot) : \mathbb{Z} \to \mathcal{X}$, with $\Phi(\mathbf{x}, 0) = \mathbf{x}$.
Sets: For a TIA system with dynamics function f, a state vector $\bar{\mathbf{x}}$ is called an *equilibrium point* if it is a fixed-point for f, i.e., $\bar{x} = f(\bar{x})$. A state vector is called a *regular point* if it is not an *equilibrium point*. Let $J_f(\mathbf{x})$ be the Jacobian of f at $\mathbf{x}$. If $J_f(\mathbf{x})$ has no eigenvalue with modulus one, $\mathbf{x}$ is called a *hyperbolic* equilibrium point. A hyperbolic equilibrium point is asymptotically stable when the eigenvalues of its corresponding Jacobian have modulus less than one; otherwise, it is an *unstable*

1. Including discretized continuous-time systems

equilibrium point. A system whose all equilibrium points are hyperbolic is called a hyperbolic system. A set $M \subseteq \mathcal{X}$ is called an *invariant* set, if $f(M) = M$, i.e., every trajectory starting in M remains in M, for $k \in \mathbb{Z}$. A point $\mathbf{p} \in \mathbb{R}^d$ is said to be in the ω-limit set (or α-limit set) of $\mathbf{x}$, if for every $\epsilon > 0$ and $N > 0$ ($N < 0$), there exists a $k > N$ ($k < N$) such that $\|\mathbf{x}_k - \mathbf{p}\| < \epsilon$. The stable and unstable manifolds of $\bar{\mathbf{x}}$ are defined as the set of points whose $\omega-$limit (α-limit) set is $\bar{\mathbf{x}}$ and denoted by $W^s(W^u)$. Both W^s and W^u are proved to be invariant sets (Palis and De Melo, 2012).
Stability: A fixed-point $\bar{\mathbf{x}}$ is said to be an asymptotically stable equilibrium, if $\lim_{k\to\infty} \mathbf{x}_k = \bar{\mathbf{x}}$. Nonlinear systems often have a local stability region (RoA) that is defined for the stable equilibrium $\bar{\mathbf{x}}$ as $\mathcal{R}^{\bar{\mathbf{x}}} = \{\mathbf{x} \in \mathcal{X} : \lim_{k\to\infty} \Phi(\mathbf{x}, k) = \bar{\mathbf{x}}\}$. Topologically speaking, when f is continuous, $\mathcal{R}^{\bar{\mathbf{x}}}$ is an open, invariant set (see Chiang and Alberto, 2015 for exact definitions). The stability boundary $\partial\mathcal{R}^{\bar{\mathbf{x}}}$ is a closed positively invariant set and is of dimension $n - 1$, if $\mathcal{R}^{\bar{\mathbf{x}}}$ is not dense in $\mathbb{R}^n$.

Lyapunov stability. Let f be a locally Lipschitz continuous dynamics function with an equilibrium point at the origin $\bar{\mathbf{x}} = \mathbf{0}$. Suppose there exists a locally Lipschitz continuous function $V = \mathcal{X} \to \mathbb{R}$ and a domain $\mathcal{D} \subseteq \mathcal{X}$, such that

$$V(\mathbf{0}) = 0 \quad \text{and} \quad V(\mathbf{x}) > 0 \qquad \forall \mathbf{x} \in \mathcal{D}\backslash\{\mathbf{0}\} \tag{2}$$

$$\Delta V(\mathbf{x}) := V(f(\mathbf{x})) - V(\mathbf{x}) < 0 \qquad \forall \mathbf{x} \in \mathcal{D}\backslash\{\mathbf{0}\} \tag{3}$$

Then, $\bar{\mathbf{x}}$ is asymptotically stable and V is a *Lyapunov function (lf)*. The domain $\mathcal{D}$ in which (3) is satisfied is called the *Lyapunov decrease region*. It is easy to show that every level set $\mathcal{S}_c(V) = \{\mathbf{x} \in \mathcal{X} : V(\mathbf{x}) < c\}$, for $c \in \mathbb{R}_+$, that is contained within $\mathcal{D}$ is invariant under the dynamics f.

3. Problem Statement

We consider a discrete-time TIA system as in (1), where the control signal is produced by a feedback-controller $\pi(\cdot;\psi) : \mathcal{X} \to \mathcal{U}$ parameterized by ψ. Therefore, the closed-loop dynamics denoted by f_π is a functional of the controller and is consequently parameterized by ψ as $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \pi(\mathbf{x}_k;\psi)) = f_\pi(\mathbf{x}_k;\psi)$. Without loss of generality, we assume that the equilibrium point of interest is located at the origin $\bar{\mathbf{x}} = \mathbf{0}$. The policy π induces a RoA around the equilibrium point denoted by $\mathcal{R}_\pi$.[2]

Each control task can be broken down into two subtasks: **1)** Controller synthesis and **2)** Closed-loop response evaluation. The controller is designed to optimize some measure of performance. In this work, the performance measure is the size of the stability region.

We endow the state space $\mathcal{X}$ with a measure μ to obtain the measure space $(\mathbb{R}, \mathcal{B}(\mathbb{R}^d), \mu)$ with Borel sigma-algebra $\mathcal{B}$. To prevent pathological cases, we assume $\mathcal{X}$ to be compact with $\mu(\mathcal{X}) < \mu_\infty < \infty$. Let $\Pi = \{\pi : \mathcal{X} \to \mathcal{U} : \pi \in C^1 \text{ and bounded}\}$ be the set of all functions from which the policy is chosen. The goal is to find a member π^* of the equivalence class of optimal policies $\Pi^* \subseteq \Pi$, where Π^* is defined as $\Pi^* = \{\pi^* \in \Pi : \mu(\mathcal{R}_{\pi^*}) = \max_{\pi\in\Pi} \mu(\mathcal{R}_\pi)\}$.

The main challenge in this optimization problem is the fact that there is no analytical or straightforward way to infer how changing π changes $\mu(\mathcal{R}_\pi)$. If there exists a differentiable map from π to $\mu(\mathcal{R}_\pi)$, one could locally increase the stability region by perturbing the policy in the direction of $\partial\mu(\mathcal{R}_\pi)/\partial\pi$. However, unless for extremely simple systems, such a map cannot be derived analytically. In this work, we construct a bridge between these two objects by an auxiliary function, which is an evolving Lyapunov function that is learned alongside the policy.

2. We drop $\bar{\mathbf{x}}$ from the superscript of $\mathcal{R}_\pi^{\bar{\mathbf{x}}}$, since we always assume $\bar{\mathbf{x}} = \mathbf{0}$, unless otherwise stated.

Our goal is to construct a sequence of policies $(\pi_1, \pi_2, \ldots)$ that gives a sequence of RoAs $(\mathcal{R}_{\pi_1}, \mathcal{R}_{\pi_2}, \ldots)$, such that $\mathcal{R}_{\pi_n} \xrightarrow{\mu} \bar{\mathcal{R}}$ as $n \to \infty$, where $\bar{\mathcal{R}} \subseteq \mathcal{D}$ is the largest achievable RoA that is constrained by the physical limitations of the system (see Appendix A for the characterization of $\bar{\mathcal{R}}$ using the concept of control Lyapunov function). To achieve this goal, we need to address two challenges: **1)** Approximating $\mathcal{R}_{\pi_n}$ for a fixed π_n and **2)** Using $(f, \mathcal{R}_{\pi_n})$ to find π_{n+1}. Next section, explains our proposed method to address these two challenges.

4. Proposed Method

For a function g, let's define its sublevel set with level value a as $\mathcal{S}_a(g) := \{\mathbf{x} \in \mathbb{R}^d : g(\mathbf{x}) < a\}$. The index $n \in \mathbb{N} \cup \{0\}$ refers to a phase of the algorithm. At phase n, let $\mathcal{R}_{\pi_n}$ be the RoA of the closed-loop system (1) that is induced by the state-feedback policy π_n. It can be shown that there exists an optimal Lyapunov function V_{π_n} for this system with a level value c_n, such that $\mathcal{R}_{\pi_n} = S_{c_n}(V_{\pi_n})$ (Vannelli and Vidyasagar, 1985). Therefore, the information of $\mathcal{R}_{\pi_n}$ is encoded in (V_{π_n}, c_n). Starting with a conservative controller (e.g., a quadratic controller for the linearized system), our method inductively constructs a sequence of policies $(\pi_1, \pi_2, \ldots)$ that eventually converges to a policy with maximal achievable RoA.

Each step of this inductive process is called a *phase* of the algorithm. Each phase consists of two sub-phases: **1)** Learning the Lyapunov function and the RoA corresponding to the policy of that phase **2)** Updating the policy to enlarge the RoA. The RoA estimation sub-phase finds a Lyapunov function V_{π_n} and level value c_n, such that $\mathcal{S}_{c_n}(V_{\pi_n}) = \mathcal{R}_{\pi_n}$. Then, the policy update phase learns a new policy π_{n+1}, such that $\mathcal{S}_{c_n}(V_{\pi_n}) \subseteq \mathcal{R}_{\pi_{n+1}}$. These conditions need to be satisfied for every n that consequently limits the class of treatable systems. Before delving into the algorithmic implementation, Section 4.1 provides necessary theoretical insights into this favorable class of systems along with other theoretical considerations for the proposed multi-phase growing algorithm.

4.1. Theoretical Discussion

We start with the assumptions that are necessary for the practical applicability of the method. Then, we show for which class of systems these assumptions are satisfied.

Assumption 1 *Let $R : \Pi \to 2^{\mathcal{X}}$ be a set-valued function defined as $R(\pi) = \mathcal{R}_\pi$. Let $d_\Pi : \Pi \times \Pi \to \mathbb{R}_+$ and $d_{\mathcal{X}} : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to \mathbb{R}_+$ be some specified metrics in the space of policies and the space of all subsets of the state space, respectively. Then, the map R is assumed to be continuous with respect to the topologies induced by the metrics d_π and $d_{\mathcal{X}}$.*

Intuitively, this assumption indicates that a small change in the policy leads to a small change in the RoA that it induces. This property that we refer to as the *persistence* of RoA is formalized as follows: let $(\mathcal{R}_{\pi_{n-1}}, \pi_n)$ comprise the RoA of the previous step and the policy of the current step for which we want to find the RoA. Given Assumption 1, for every $\epsilon > 0$, one can choose small enough $\delta > 0$ such that $d_\Pi(\pi_n, \pi_{n-1}) < \delta$ results in $\mu(\mathcal{R}_{\pi_n} \triangle \mathcal{R}_{\pi_{n-1}}) < \epsilon$. We now show in the following theorem that this assumption is in fact valid for hyperbolic systems.

Theorem 2 (Persistence of the stability boundary with variations in the policy) *Consider the closed-loop hyperbolic system $\mathbf{x}_{k+1} = f_\pi(\mathbf{x}_k)$ with policy π. Suppose f_π is a diffeomorphism and all equilibrium points on $\partial\mathcal{R}_{\tilde{\pi}}$ are hyperbolic. Moreover, let the stable and unstable manifolds of* *the equilibrium points on $\partial\mathcal{R}_{\tilde{\pi}}$ intersect transversally*[3]*. Finally, assume that every trajectory on $\partial\mathcal{R}_{\tilde{\pi}}$ approaches one of the equilibrium points. For a certain policy $\pi = \tilde{\pi}$, let $(\bar{\mathbf{x}}_{\tilde{\pi}}, \mathcal{R}_{\tilde{\pi}})$ be an asymptotically stable equilibrium point and its corresponding RoA. Then, for every $\epsilon > 0$, there exists $\delta > 0$ such that for every π' with $d_{\Pi}(\tilde{\pi}, \pi') < \delta$, we have $\mu(\mathcal{R}_{\tilde{\pi}} \triangle \mathcal{R}_{\pi'}) < \epsilon$.*

Proof The first step to prove this result is to characterize the RoA in terms of the properties of the dynamics function f_π. As $\mathcal{R}_{\tilde{\pi}}$ is characterized by its boundary, we focus our attention on the stability boundary $\partial\mathcal{R}_{\tilde{\pi}}$. The critical elements[4] of the dynamical system determine the structure of the stability boundary. If $\{\mathbf{x}_1, \mathbf{x}_2, \ldots\}$ are the hyperbolic equilibrium points on $\partial\mathcal{R}_{\tilde{\pi}}$, $\partial\mathcal{R}_{\tilde{\pi}}$ is completely characterized by

$$\partial\mathcal{R}_{\tilde{\pi}} = \cup_i W^s(\mathbf{x}_i). \tag{4}$$

See Theorems 9-11 in Chiang and Alberto (2015) for the detailed proof. With this characterisation, to show the persistence of $\partial\mathcal{R}_{\tilde{\pi}}$, it is enough to show the persistence of the equilibrium points that live on $\partial\mathcal{R}_{\tilde{\pi}}$ and the persistence of their stability condition. As a result of the continuity of $f(\mathbf{x}, \pi(\mathbf{x}))$ and $\pi(\mathbf{x})$ w.r.t. their arguments, implicit function theorem guarantees that small perturbations to π cause small changes in the hyperbolic equilibrium points (Krantz and Parks, 2012). If $\mathbf{x}^*_{\tilde{\pi}}$ is a hyperbolic equilibrium point of $\mathbf{x}_{k+1} = f_\pi(\mathbf{x}_k)$ for the policy $\pi = \tilde{\pi}$, there exists a $\delta > 0$ and a neighborhood U of $\mathbf{x}^*_{\tilde{\pi}}$ that contains a unique hyperbolic equilibrium point $\mathbf{x}^*_{\pi'}$ for every $\pi' \in \{\pi \in \Pi : d_\Pi(\pi, \tilde{\pi}) < \delta\}$. Similarly, the continuity of the eigenvalues of $J_{f_\pi}(\mathbf{x}_\pi)$ w.r.t. π affirms that the perturbed equilibrium point $\mathbf{x}^*_{\pi'}$ has the same stability condition as $\mathbf{x}^*_{\tilde{\pi}}$, i.e., there exists a homeomorphism $h : \mathbb{R}^n \to \mathbb{R}^n$ from $\Phi_{\tilde{\pi}}(\mathbf{x}, t)$ to $\Phi_{\pi'}(\mathbf{x}, t)$. As stated above and given the results of Chiang and Alberto (2015), $\partial\mathcal{R}_{\tilde{\pi}}$ is characterized by the stable manifolds of unstable equilibrium points living on the boundary[5]. It was also shown that the hyperbolic equilibrium points on $\partial\mathcal{R}_{\tilde{\pi}}$ together with their stable and unstable manifolds change continuously with $\tilde{\pi}$. This results in a continuous change of $\partial\mathcal{R}_{\tilde{\pi}}$, and consequently $\mathcal{R}_{\tilde{\pi}}$, w.r.t. small variations in $\tilde{\pi}$. ■

This result suggests restricting the change of the policy at the policy update sub-phase of each growing phase of the algorithm to bound the change of its induced RoA. For parametric policies such as neural networks, this is achieved by cropping parameter values after training.

4.2. Algorithm

The implementation of the algorithm that was outlined in Section 4 comes in the following. Notice that the algorithm is a multi-phase inductive method. Hence, we only present two sub-phases of a single phase. The entire algorithm is iteration over this phase multiple times.

RoA Estimation Sub-Phase: Learn $\mathcal{R}_{\pi_n}$ from $(\mathcal{R}_{\pi_{n-1}}, \pi_{n-1})$. This sub-phase takes the previous policy π_{n-1} and its associated RoA estimate $\mathcal{S}_{c_{n-1}}(V_{\pi_{n-1}})$ and outputs $\mathcal{S}_{c_n}(V_{\pi_n})$ that approximates $\mathcal{R}_{\pi_n}$. For this sub-phase, we improve the growing algorithm of (Richards et al., 2018) to learn an inner estimate of RoA. Our RoA estimation algorithm takes advantage of the theoretical results of (Chiang and Thorp, 1989) to improve the stability of training. The proposed loss function is more robust against local minima observed by Richards et al. (2018) as is empirically shown in Section 5.

3. Roughly speaking, the manifolds intersect in a generic way.
4. For theoretical discussion, we focus on critical elements that are equilibrium points. Similar results exist for other types of critical elements such as limit cycles but are left out of this work for brevity
5. Knowing the stable manifolds of the equilibriums that live on the boundary is enough to determine the boundary.

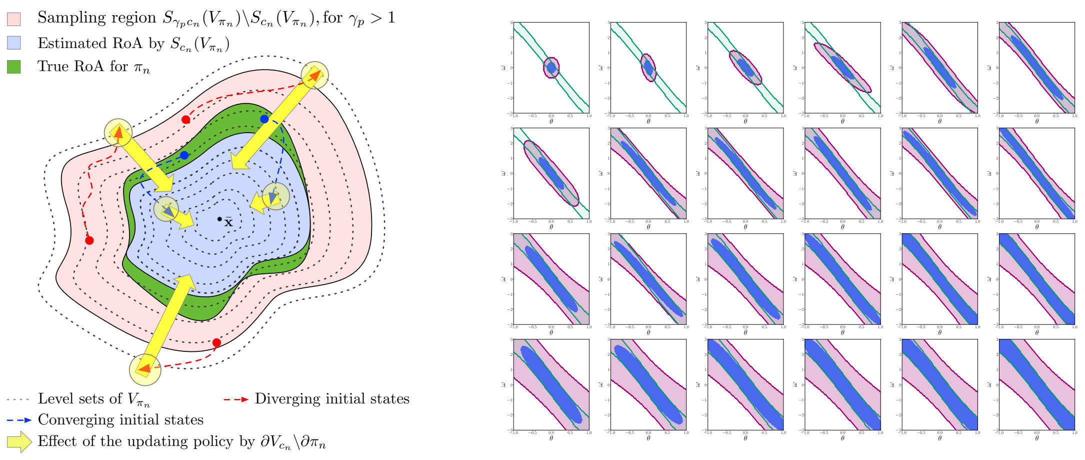

Sampling region $S_{\gamma_p c_n}(V_{\pi_n})\backslash S_{c_n}(V_{\pi_n})$, for $\gamma_p > 1$
Estimated RoA by $S_{c_n}(V_{\pi_n})$
True RoA for π_n
$\mathbf{x}$
Level sets of V_{π_n}
Diverging initial states
Converging initial states
Effect of the updating policy by $\partial V_{c_n}\backslash\partial\pi_n$
θ

Figure 1: (Left) Illustration of the policy update sub-phase. Given the estimated RoA, the policy update sub-phase (yellow arrows) tries to pull the diverging trajectories towards the level sets of the estimated Lyapunov function that reside inside the RoA. (Right) Visualizing the true RoA which is enlarged by the improved policy and is chased by a learned Lyapunov function. Green boundary: True RoA, Blue: $S_{c_n}(V_{\pi_n})$, Pink: $S_{\gamma c_n}(V_{\pi_n})$ for $\gamma = 4$. The pink area shows the region from which the samples outside the estimated RoA are taken for both estimating the RoA and updating the policy.

The learning algorithm is verbally described here and the pseudo-code can be seen in Algorithm 1 in the Appendix. It starts with an initial conservative estimate of RoA and grows it during multiple iterations (phases). A neural network, as a universal function approximator, is employed to estimate a Lyapunov function for the system. At each iteration, the Lyapunov function is improved such that one of its level sets gives a better estimate of the RoA. For this purpose, a fixed number of initial states are taken from a gap surrounding the current estimate of the RoA. The initial states are integrated forward in time by the closed-loop dynamics to produce the solution $\Phi(\mathbf{x}, k)$ for each initial state $\mathbf{x}$. The final state of each trajectory determines the stability of its corresponding initial state. We train the Lyapunov function as a classifier. This allows the method to enjoy the well-developed supervised learning tools that have been exceptionally successful in scaling to higher dimensions especially when implemented as neural networks. The current estimate of the RoA produced by the current Lyapunov function is called the largest stable level set. The initial states are labelled *stable* if their trajectory enters the largest stable level set and are labelled *unstable* otherwise. We denote these two sets of initial states as $\mathbb{X}^{\text{IN}}$ and $\mathbb{X}^{\text{OUT}}$ respectively and minimize the loss function

$$\mathcal{L}(V) = \sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}}[V(\mathbf{x}) - \bar{c}] - \sum_{\mathbf{x}\in\mathbb{X}^{\text{OUT}}}[V(\mathbf{x}) - \bar{c}] + \lambda_{\text{RoA}}\sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}}\Delta V(\mathbf{x}) + \lambda_{\text{monot}}\sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}}[V(\mathbf{x}) - V_{\pi_{n-1}}(f_{\pi_{n-1}}(\mathbf{x}))]^2 \quad (5)$$

to update the Lyapunov function where $\bar{c}$ is fixed to a constant value (1 throughout this work). The idea is to absorb this degree of freedom in V to ease training. The loss function is minimized by automatic differentiation and Stochastic Gradient Descent (SGD) with respect to the parameters of the neural network that realizes V. Once it is minimized, a line search is carried out on the level value to obtain c_n such that $\mathcal{S}_{c_n}(V_{\pi_n})$ does not exceed the true RoA. The same process repeats for a certain number of iterations until a good inner estimate of the RoA is achieved. An important component in this process is the size of the gap around the largest stable level set of each iteration of the algorithm. This gap is produced by $\mathcal{G}_n = \mathcal{S}_{\gamma_r c_n}(V_{\pi_n})\backslash\mathcal{S}_{c_n}(V_{\pi_n})$ with $\gamma_r > 1$. The size of this gap is controlled by γ_r. Larger values of γ_r gives a faster convergence but less stable training.

To give an intuitive idea of the terms in Equation (5), the first two terms encourage the Lyapunov function to change in a way that its level set $S_{\bar{c}}(V)$ includes the stable initial states $\mathbb{X}^{\text{IN}}$ and excludes the unstable initial states $\mathbb{X}^{\text{OUT}}$. The third term weighted by λ_{RoA} encourages the negative definiteness of ΔV on the RoA. The last term is inspired by the constructive method of (Chiang and Thorp, 1989) that accelerates capturing the entire RoA (see Appendix C). To satisfy Lyapunov conditions, V needs to be positive definite on its domain. Rather than treating this condition in the loss function, we encode it in the architecture of the neural network using the construction proposed by Richards et al. (2018). Check Appendix G.1 for the detailed description of the architecture.

Policy Update Sub-Phase: Learn π_{n+1} from $\mathcal{R}_{\pi_n}$: This sub-phase of the algorithm uses the estimated Lyapunov function V_{π_n} to update the policy so that the new policy gives rise to a larger RoA. The idea is to change the policy in a way that the unstable trajectories starting from around the current RoA enter the RoA and become stable. Let $\mathcal{D}$ be the domain of f around the equilibrium $\bar{\mathbf{x}}$. Given a hypothesis class of feasible policies Π, only a subset of the domain $\mathcal{D}$ is stabilizable. Assume $\bar{\mathcal{B}} \subseteq \mathcal{D}$ is the maximum stabilizable subset of the domain with $\mu(\bar{\mathcal{B}}) = \bar{\mu}$. Therefore, an attempt to improve π_n amounts to appending points from $\bar{\mathcal{B}}\backslash\mathcal{R}_{\pi_n}$ to $\mathcal{R}_{\pi_n}$. The set $\bar{\mathcal{B}}$ is not fully known in advance but some of its properties can be derived. Especially, for system (1), if $f, \pi \in C^{\infty}$, the maximum stabilizable set $\bar{\mathcal{B}}$ whose measure materializes as $\bar{\mu}$ is compact and connected. Using this theoretical result, if $\mathcal{R}_{\pi_n} \subsetneq \bar{\mathcal{B}}$, the stabilizable states can be chosen in a gap around $\mathcal{R}_{\pi_n}$.

Because the RoA estimation sub-phase estimates it as a level set of a Lyapunov function, i.e. $\mathcal{R}_{\pi_n} \approx S_{c_n}(V_{\pi_n})$), the sampling gap is constructed as $\mathcal{G}_n = S_{\gamma_p c_n}(V_{\pi_n})\backslash S_{c_n}(V_{\pi_n})$ for a $\gamma_p > 1$. To make sure the policy does not destabilize the regions that are already stabilized in the previous phases, algorithm also samples initial states from $\mathcal{S}_{c_n}(V_{\pi_n})$. All sampled initial states are integrated forward for L_p steps and the policy is updated via minimizing the loss function

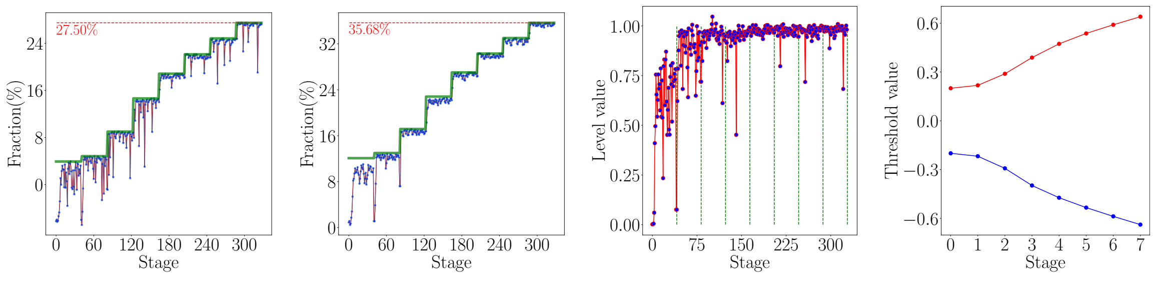
$$\mathcal{L}(\pi) = \sum_{\mathbf{x}\in\mathcal{G}_n\cup S_{c_n}(V_{\pi_n})} [1_{[V_{\pi_n}(\Phi_\pi(\mathbf{x},L_p))<c_n]} + \lambda_{\text{u}} 1_{[V_{\pi_n}(\Phi_\pi(\mathbf{x},L_p))>c_n]}] V_{\pi_n}(\Phi_\pi(\mathbf{x}, L_p)). \tag{6}$$

We implement the policy as a differentiable function such as a neural network and use automatic differentiation and SGD for minimization. The parameters of the policy appear in (6) via the end state $\Phi_\pi(\mathbf{x}, L_p)$ of the closed-loop trajectories. It is clear in (6) that minimizing $\mathcal{L}(\pi)$ with respect to π while the Lyapunov function is fixed, pushes the trajectories towards the areas where the Lyapunov function assumes smaller values. This affects both stable and unstable trajectories while its influence on unstable trajectory can be magnified by choosing $\lambda_u > 1$. The detailed pseudo-code of this phase can be found in Algorithm 2 in the Appendix. The quality of the training signal is analyzed next.

Theoretical analysis of the training signal— The effect of the policy π on $\mathcal{L}(\pi)$ passes through the Lyapunov function V as can be seen in (6). Unlike the conventional Lyapunov redesign method in control theory where the Lyapunov function is fixed, here the Lyapunov function itself is learned by the RoA estimation sub-phase of the algorithm. Hence, an ill-conditioned V can harm the policy update phase. To take a closer look at this issue, we expand the training signal analytically. Let the policy π be a function of states parameterized by ψ. Let $T = L_p$ be the time step of the final state of the trajectory. The training signal to update the policy is proportional to $\partial\mathcal{L}/\partial\psi$ expanded as

$$\frac{\partial\mathcal{L}}{\partial\psi} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}_T} \sum_{1\leq k\leq T} \left(\frac{\partial\mathbf{x}_T}{\partial\mathbf{x}_k}\frac{\partial^+\mathbf{x}_k}{\partial\psi}\right). \tag{7}$$

by applying the chain rule for differentiation. The term $\partial^+\mathbf{x}_k/\partial\psi$ is the single-step effect of ψ on $\mathbf{x}_k$ when $\mathbf{x}_{k-1}$ is fixed. We discuss every term in this equation in the following. Observe that $\partial\mathcal{L}/\partial\mathbf{x}_T$ is multiplied the summation, i.e., its small value diminish the entire signal. It can be expanded as



(a) Unimproved process (b) Improved process (c) Level values (d) Policy parameters

Figure 2: (a), (b) The size of the RoA against the iterative stages (phases) of the Algorithms where (a) uses the RoA estimation method of Richards et al. (2018) while (b) uses ours. The fraction is computed with respect to a rectangular domain around the equilibrium point that is large enough to enclose the RoA. Green: size of the true RoA. Each jump corresponds to a policy update sub-phase that increases the size of the true RoA. Red: The size of level set that the RoA estimation sub-phase learns to approximate the RoA. After each policy update, the RoA estimation sub-phase takes multiple growth iterations to capture the true RoA as close as possible. (c) The trace of the level values corresponding to every iteration of the RoA estimation sub-phase. (d) Red (Blue): The trace of the value of the upper (lower) threshold parameter of the policy during training. Each point corresponds to a policy update iteration.

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_T} = \frac{\partial \mathcal{L}(V)}{\partial V}|_{V=V(\mathbf{x}_T)} \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_T}. \tag{8}$$

The first term on the r.h.s. does not vanish as it is 1 or λ_u for the loss function defined by Equation (6). The second term of the r.h.s depends on the slope of V evaluated at the final state of the trajectory. One potential pathological condition occurs for stable and long trajectories. The reason is that $\nabla_{\mathbf{x}} V(\mathbf{x})$ continuously vanishes at the equilibrium (see Lemma 4 in the Appendix). This means that the derivative of the Lyapunov functions gradually becomes smaller as we get closer to the equilibrium. Therefore, for long stable trajectories that $\mathbf{x}_T$ is too close to the equilibrium, the learning signal to update the policy will be too small. Too long trajectories are detrimental for unstable trajectories as well because the states may grow exponentially and cause damages to the system. Hence, the length of the trajectory is an important design parameter that needs special attention when applying our proposed method in practice. The terms inside the sum depend on the properties of the dynamics. Specifically, the first term shows how long the system keeps the memory of the past states and the second term shows how sensitive the system is with respect to the controller parameters. A more detailed discussion is deferred to Appendix F.

5. Experiments

We consider an inverted pendulum system defined as $\dot{\theta} = \omega$ and $\dot{\omega} = \frac{g}{l}\sin(\theta) + \frac{u}{I} - \mu_f \frac{\omega}{I}$ where the state vector $\mathbf{x} = (\theta, \omega)$ consists of the angle and angular velocity. Moreover $(g = 0.81, l = 0.5, I = \text{mass} \times l^2 = 0.25, \mu_f = 0)$ are the acceleration of gravity, length, inertia, and friction coefficient. The scalar u is the input force. The open-loop system (with $u = 0$) has equilibrium points at $(\theta, \omega) = (k\pi, 0)$ with $k \in \mathbb{Z}$. We focus on the equilibrium point $(0, 0)$ in the frictionless setting where the system shows oscillatory behavior and consequently is not asymptotically stable (see Appendix G.2 for system and modeling details). First, an LQR controller K is designed for the linearized system around the origin (see the vector field and the initial RoA

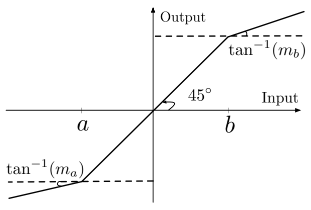
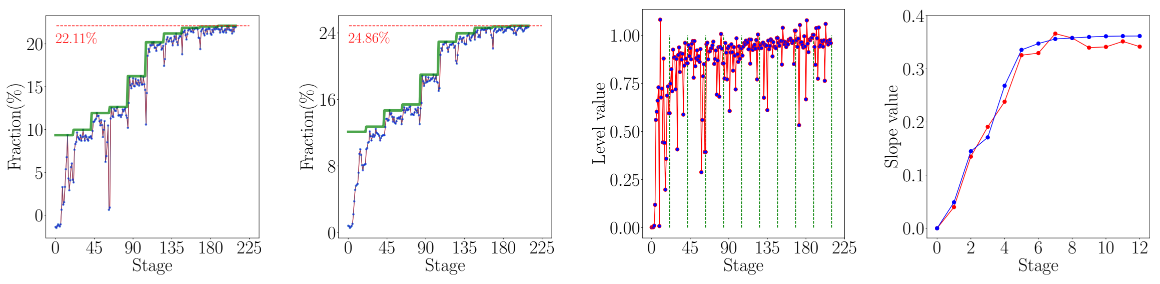


Figure 4: Loose saturation



(a) Unimproved process (b) Improved process (c) Level values (d) Policy parameters

Figure 3: The same plots as Figure 2 with the exception that, here the threshold parameters are kept fixed at $a = 0.2$ and $b = -0.2$ while the slopes m_a and m_b are trainable parameters.

in Figure 5 in the appendix). The control signal passes through a loose saturation function as $u = \pi_0(\mathbf{x};\psi) = \text{SAT}_\psi([\theta,\omega]^\mathsf{T} K)$. The function SAT is parameterized by $\psi = (a, b, m_a, m_b)$ as illustrated in Figure 4. In the first experiment, the slopes $m_a = 0$ and $m_b = 0$ are kept fixed where a and b are trainable parameters of the policy, i.e., $\psi = (a, b)$. The Lyapunov function $V(\cdot;\theta)$ is realized by a $3-$layer neural network parameterized by θ. Each layer has 64 neurons with a special architecture (see Appendix G.1) inspired by (Richards et al., 2018) followed by tanh activation function that imposes the positive definiteness of the entire network as is required by the Lyapunov conditions in (2). The chosen hyper-parameters for the SGD training can be found in Appendix G.3.

The initial controller gives a small RoA since it is designed for the locally linearized system. As the initial policy is LQR designed for the locally linearized model, $V(\cdot;\theta)$ is pre-trained by the quadratic function $0.1\theta^2 + 0.1\omega^2$. After pre-training, sub-phases of the algorithm of Section 4.2 are run alternately to capture the RoA and improve the policy. The green step-like plot in Figure 2a and Figure 2b shows the true size of the RoA. Each jump in the green plots shows one iteration of the policy update algorithm resulting in an increased RoA. The fluctuating red plot in Figure 2a shows the size of the estimated RoA without our improvement over the RoA estimation algorithm of Richards et al. (2018) while Figure 2b shows the outcome of the presence of our proposed additional term in the loss function (5). It shows that the added term results in a less fluctuating estimate of the RoA, and when combined with the policy update phase, gives a faster convergence to a larger RoA (35.68% vs 27.50% fraction of the domain volume after 7 policy updates).

As stated in Section 4.2, $\bar{c} = 1$ is not necessarily equal to c_n. After learning V_n with $\bar{c} = 1$ in (5), the algorithm searches for a value of c_n such that the Lyapunov decrease condition is met for all states within the level set $\mathcal{S}_{c_n}(V_n)$. It can be seen in Figure 2c that these values converge to $\bar{c} = 1$ as a sign of stable training. The trace of the parameters of the policy is shown in Figure 2d. As these policy's trainable parameters represent the upper and lower limits of the loose threshold function, the policy learning algorithm updates them in the directions that decrease their suppressing effect that is expected from the physics of the system. Graphical visualization of the policy update and RoA estimation phase is shown in Figure 1(Right). Each row shows one phase of the algorithm. The policies are updated along the columns from top to bottom. Within one row, the policy is fixed and RoA is estimated from left to right (See Figure 7 in the appendix for a larger visualization).

In the second experiment, the threshold limits $a = -0.2$ and $b = 0.2$ are kept fixed while the slopes $\psi = (m_a, m_b)$ are trainable parameters. The rest of the training setting remains the same as the previous experiment. Figures 3a and 3b shows that the policy update phase enlarges the RoA (green plot) of the system while the RoA estimation phase (red plot) manages to follow the new RoA after each policy update. Our improved RoA estimation algorithm results in a more monotonic convergence of the estimated RoA that ultimately learns a controller that induces a larger RoA (24.86% vs 22.11%). Similar to Figure 2c, convergence of c_n values to $\bar{c} = 1$ can be seen in Figure 3c. The trace of the upper and lower slopes are shown in Figure 3d. Almost equal learned values for upper and lower slopes are expected due to the structural symmetry of the saturation function (Figure 4) that appears in the closed-loop system.

Scalability. At each growth step of the algorithm, samples are taken from around the RoA that is an $n-1$ dimensional surface. As these are the most informative trajectories for RoA maximization, it can be seen as a clever sampling that is hoped to scale better compared with policy update methods with uniform sampling. An empirical investigation of this hypothesis is deferred to future work.

The following remark discusses to what extent the model of the system is needed and how can this requirement be relaxed in the future.

Remark 3 *The RoA estimation phase does not need the model of the system. The system can be launched from sampled initial states and the generated trajectories are all we need in* (5)*. The policy update phase of the algorithm requires a local estimate of the system to be able to compute $\partial V/\partial \pi$. The locality of the model is inversely proportional to the length of the trajectory L_p in* (6)*. A detailed theoretical discussion on this point is deferred to Appendix E.*

6. Related Work and Conclusions

We have proposed a two-player collaborative and iterative algorithm that iterates over two sub-phases that learn the Lyapunov function and use it to learn a controller to enlarge the RoA of the system. The existing approaches that are close to the purpose of this paper are those that simultaneously synthesize a controller and maximize the stability region. Unlike the large body of literature that consider tractable polynomial systems (e.g. Jarvis-Wloszek et al. (2003); Majumdar et al. (2013, 2014)) our work does not put any limit on the class of considered systems except the generic condition of hyperbolicity. Jin et al. (2020) suggest a joint training of the policy and neural certificate functions that model unsafe regions (Barrier function) and reach a target set using Lyapunov-like functions. There are two major differences with our work: Our method proposed a safe sampling using RoA information instead of uniform sampling in the state space that could be harmful to the system in practice. In addition, their objective function does not necessarily enlarge the RoA which is the main concern of our work. Boffi et al. (2020) use methods from statistical learning theory to assess the generalizability of the learned certificate functions such as Lyapunov functions. Robey et al. (2020) learn a control barrier function from safely sampled trajectories with known dynamics. These two methods are related to our work in the sense that they learn certificate functions but, unlike our work, they do not include a specialized sampling (from around the RoA) and also do not learn a policy.

In the context of reinforcement learning, our method can be seen as an actor-critic approach Grondman et al. (2012); Bhasin et al. (2013); Lillicrap et al. (2015) where the actor tries to stabilize the system while the critic estimates the size of the RoA induced by the controller. In control theory, as the title of our work suggests, the proposed algorithm can be seen as an automatic version of the known Lyapunov redesign method where the Lyapunov function and the closed-loop system are being re-designed together iteratively.

In this work, we assume the model of the system is given, even though this condition can be relaxed as suggested by Remark 3. Investigating this relaxation could be a promising future direction.

References

Steven M Baer, Bingtuan Li, and Hal L Smith. Multiple limit cycles in the standard model of three species competition for three essential resources. *Journal of mathematical biology*, 52(6):745–760, 2006.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pages 908–918, 2017.

Shubhendu Bhasin, Rushikesh Kamalapurkar, Marcus Johnson, Kyriakos G Vamvoudakis, Frank L Lewis, and Warren E Dixon. A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems. *Automatica*, 49(1):82–92, 2013.

Nicholas M Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques E Slotine, and Vikas Sindhwani. Learning stability certificates from data. *arXiv preprint arXiv:2008.05952*, 2020.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, pages 3245–3254, 2019.

H-D Chiang and James S Thorp. Stability regions of nonlinear dynamical systems: A constructive methodology. *IEEE Transactions on Automatic Control*, 34(12):1229–1241, 1989.

Hsiao-Dong Chiang and Luís FC Alberto. *Stability regions of nonlinear dynamical systems: theory, estimation, and applications*. Cambridge University Press, 2015.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

John J Hopfield. Neurons, dynamics and computation. *Physics Today*, 47(2):40–47, 1994.

Mehdi Imani Masouleh and David JN Limebeer. Region of attraction analysis for nonlinear vehicle lateral dynamics using sum-of-squares programming. *Vehicle System Dynamics*, 56(7):1118–1138, 2018.

Alberto Isidori. *Nonlinear Control Systems Design 1989: Selected Papers from the IFAC Symposium, Capri, Italy, 14-16 June 1989*. Elsevier, 2014.

Zachary Jarvis-Wloszek, Ryan Feeley, Weehong Tan, Kunpeng Sun, and Andrew Packard. Some controls applications of sum of squares programming. In *42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475)*, volume 5, pages 4676–4681. IEEE, 2003.

Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

F Liao and JL Wang. Analysis and synthesis of reliable flight control systems via parameter-dependent lyapunov functions. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 218(6):433–450, 2004.

Alexandre Liapounoff. Problème général de la stabilité du mouvement. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 9, pages 203–474, 1907.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.

Anirudha Majumdar, Ram Vasudevan, Mark M Tobenkin, and Russ Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *The International Journal of Robotics Research*, 33(9):1209–1230, 2014.

J Jr Palis and Welington De Melo. *Geometric theory of dynamical systems: an introduction*. Springer Science & Business Media, 2012.

Harry A Pierson and Michael S Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems. *arXiv preprint arXiv:1808.00924*, 2018.

Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

Anthony Vannelli and Mathukumalli Vidyasagar. Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

Zhu Wen-Xing and Zhang Li-Dong. A new car-following model for autonomous vehicles flow with mean expected velocity field. *Physica A: Statistical Mechanics and its Applications*, 492: 2154–2165, 2018.

Eric R Westervelt, Jessy W Grizzle, and C Canudas De Wit. Switching and pi control of walking motions of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(2):308–312, 2003.

H Xin, D Gan, J Qiu, and Z Qu. Methods for estimating stability regions with applications to power systems. *European transactions on electrical power*, 17(2):113–133, 2007.

Appendix A. Control Lyapunov Function and Maximal Stabilizable Set

The use of the Lyapunov theory to guide designing the input of a system has been made precise with the introduction of *control Lyapunov function (clf)*. A clf for a system of the form $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ is a C^1, radially unbounded function $V : \mathcal{X} \to \mathbb{R}_+$ if

$$V(\mathbf{0}) = 0 \quad \text{and} \quad V(\mathbf{x}) > 0 \qquad \forall \mathbf{x} \in \mathcal{D}\backslash\{\mathbf{0}\} \tag{9}$$

$$\inf_{\mathbf{u}\in U}[V(\mathbf{x}) - V(f(\mathbf{x}, \mathbf{u}))] \leq 0 \qquad \forall \mathbf{x} \in \mathcal{D}\backslash\{\mathbf{0}\} \tag{10}$$

Just as the existence of a Lyapunov function is necessary and sufficient conditions for the stability of an autonomous system, the existence of a clf is a necessary and sufficient condition for *stabilizability* of a system with a control input. In other words, the existence of clf guarantees the existence of a controller that stabilizes the system for all initial states within a neighborhood around the equilibrium point.

According to the definition of Lyapunov function in Section 2 and clf above, Lyapunov function assess the stability of a closed-loop system for a fixed controller while clf investigates the *existence* of a control signal that stabilizes the system in a domain $\mathcal{D}$. In the definition of clf, no functional limitation is assumed for the control signal $\mathbf{u}_k$. In practice, $\mathbf{u}_k$ is produced by a state-feedback controller via the policy function $\pi \in \Pi$. Moreover, due to the implementation constraints, only a subset $\tilde{\Pi} \subseteq \Pi$ of these functions can be realized. Therefore, it is quite likely that $\mu(\mathcal{R}_{\pi^*}) < \mu(\mathcal{D})$ with μ be the Lebesgue measure, i.e., the best feasible controller cannot expand the RoA of the system to the entire $\mathcal{D}$. Let $U(\mathbf{x})$ be the values that the control signal can take at state $\mathbf{x}$. Then, we define

$$\bar{\mathcal{R}} := \sup_{\mathcal{B}\subseteq\mathcal{D}} \mathcal{B} \quad \text{such that} \quad \exists \text{ clf } V \text{ on } \mathcal{B} \text{ and } \exists \pi \in \tilde{\Pi} \text{ that materializes } \bar{\mathcal{R}}. \tag{11}$$

Here $\bar{\mathcal{R}}$ is the maximal stabilizable set that can be attained using the policies of $\tilde{\Pi}$.

A closed-loop system with a fixed controller can be seen as an autonomous system parameterized by the policy denoted by $\mathbf{x}_{k+1} = f_\pi(\mathbf{x}_k)$. Therefore, the policy update (Algorithm 2) can be seen as perturbing the vector field f_π. The qualitative changes in the behavior of dynamical systems due to variations in the parameters of the vector field are studied under the title *bifurcation theory*. Bifurcation in RoA due to the perturbation caused by the policy update can significantly impact its shape and size which is a fundamental concern in our algorithm and many other applications that rely on a smooth change of the RoA with respect to the vector field.

Appendix B. Proofs

In this section, a more detailed theoretical exposition of some of the material that is dropped from the main text due to space limitation is presented.

The following lemma shows that the derivative of a Lyapunov function vanishes at the equilibrium point. One can see the Lyapunov redesign method as an actor-critic algorithm where the Lyapunov function plays the role of the critic. As the learning signal for updating the actor (policy) passes through the derivative of the critic (Lyapunov function) due to the chain rule, the following lemma implies that getting closer to the equilibrium will weaken the information content of the signal for learning the policy.

Lemma 4 *Derivative of a Lyapunov function at the origin: Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a $d-$dimensional vector space and $V : \mathcal{X} \to \mathbb{R}$ be a continuous positive definite function, i.e., $V(\mathbf{x}) > 0$ for $\mathbf{x} \neq \mathbf{0}$ and $V(\mathbf{0}) = 0$. Then, $\nabla_{\mathbf{x}} V(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} = 0$.*

Proof We use the technique of proof by contradiction. Let $\mathbf{g} = [g_1, g_2, \ldots, g_d]^{\mathsf{T}} = \nabla_{\mathbf{x}} V(\mathbf{x})|_{x=\mathbf{0}} \neq 0$. Suppose there exists an index $i \in \{1, 2, \ldots, d\}$ such that $g_i \neq 0$. Due to the continuity of V, we expand $V(\mathbf{x})$ at $\mathbf{x} = \mathbf{0}$ in the direction of g_i as

$$V(0, \ldots, x_i = c, \ldots, 0) = V(\mathbf{0}) + c\frac{\partial V(\mathbf{x})}{\partial x_i}|_{\mathbf{x}=\mathbf{0}} + o(x_i)$$

for an arbitrary value of c close to 0. Since c is arbitrary, we choose $c = -\epsilon \frac{\partial V(\mathbf{x})}{\partial x_i}|_{\mathbf{x}=\mathbf{0}} = -\epsilon g_i$. As we assumed $V(\mathbf{0}) = 0$, for ϵ sufficiently close to 0, we can write

$$V(0, \ldots, x_i = c, \ldots, 0) = -\epsilon g_i^2 < 0 \;\text{ for }\; g_i \neq 0$$

which is in contrast with the positive definiteness of V. Therefore, g_i cannot be nonzero. As i is chosen arbitrarily from $\{1, 2, \ldots, d\}$, the derivative of V with respect to any of its arguments is zero at the origin, meaning that, $\nabla_{\mathbf{x}} V(\mathbf{x})|_{\mathbf{x}=\mathbf{0}} = 0$. ■

Each stage of the RoA estimation algorithm expands the level set of the estimated Lyapunov function to sample from the gap $\mathcal{G} = S_{\alpha c}(V) \backslash S_c(V)$ surrounding the current estimate of the RoA for a $\alpha > 1$. Both too small and too large gaps are harmful to the stability of the growing RoA estimation algorithm. A too small gap results in too few samples and prolongs the number of growth stages. Moreover, if the gap is too small, it is more likely that all initial states taken from the gap either converges to the equilibrium or diverges. Therefore, the dataset for the optimization problem (5) will be highly skewed that slows down the learning process even further. A too large gap is also harmful as it may advance far beyond the true RoA of the system and many sampled initial states can diverge to unknown and potentially dangerous regions of the state space. As a result, investigating the growth rate of the gap $\mathcal{G}$ as a function of the properties of V is desirable to regularize or prevent harmful sampling behaviors. The following theorem sheds light on this matter.

Theorem 5 *Growth rate of level sets: Assume $V : \mathcal{X} \to \mathbb{R}$ is a positive definite Lipschitz continuous function on $\mathcal{X} \subseteq \mathbb{R}^d$. Let $\mathcal{S}_c(V) = \{\mathbf{x} \in \mathcal{X} : V(\mathbf{x}) \leq c\}$ be the region enclosed by the level set $\partial\mathcal{S}_c(V) = \{\mathbf{x} \in \mathcal{X} : V(\mathbf{x}) = c\}$ at the level value c. If $G \leq \|\nabla_{\mathbf{x}} V(\mathbf{x})\|$ for $G \in \mathbb{R}^{>0}$ and $\mathbf{x} \in \mathcal{S}_c(V)$, then $\partial\mu(\mathcal{S}_c(V))\backslash\partial c \propto G^{-1}$.*

Proof Let $\mathbf{z} = z\nabla_{\mathbf{x}} V(\mathbf{x})/\|\nabla_{\mathbf{x}} V(\mathbf{x})\|$ be a tiny perturbation in the direction of the normal to the level set. V is expanded around $\mathbf{x} \in \partial\mathcal{S}_c(V)$ as

$$\begin{aligned} V(\mathbf{x} + \mathbf{z}) &= V(\mathbf{x}) + \nabla_{\mathbf{x}} V(\mathbf{x})^{\mathsf{T}}\mathbf{z} + O(z^2) \\ &= V(\mathbf{x}) + z\|\nabla_{\mathbf{x}} V(\mathbf{x})\| + O(z^2) \end{aligned}$$

where $O(z^2)$ can be ignored for sufficiently small $z = \|\mathbf{z}\|$. For $\alpha \in \mathbb{R}^{>1}$ and sufficiently close to 1,

$$\begin{aligned} \mathcal{S}_{\alpha c}(V) &= \{\mathbf{x} + \mathbf{z} : \mathbf{x} \in \partial\mathcal{S}_c(V) \text{ and } V(\mathbf{x}) + \nabla_{\mathbf{x}} V(\mathbf{x})^{\mathsf{T}}\mathbf{z} \leq \alpha c\} \\ &= \{\mathbf{x} + \mathbf{z} : \mathbf{x} \in \partial\mathcal{S}_c(V) \text{ and } z\|\nabla_{\mathbf{x}} V(\mathbf{x})\| \leq (\alpha - 1)c\}. \end{aligned}$$

As G is assumed to be a lower bound of $\|\nabla_{\mathbf{x}} V(\mathbf{x})\|$, $z\|\nabla_{\mathbf{x}} V(\mathbf{x})\| \leq (\alpha - 1)c$ implies $z \leq c(\alpha-1)\|\nabla_{\mathbf{x}} V(\mathbf{x})\|^{-1} \leq c(\alpha-1)G^{-1}$.

Notice that $\partial\mathcal{S}_c(V)$ is a $(d-1)-$dimensional surface that encloses $\mathcal{S}_c(V)$ an $d-$dimensional volume that are both embedded in a $d-$dimensional embedding space $\mathcal{X}$. We are interested in the volume of $\mathcal{G} := \mathcal{S}_{\alpha c}(V) \backslash \mathcal{S}_c(V)$. Assume $\mathrm{d}\boldsymbol{\omega}$ is the differential form for $\mathcal{G}$. We can write $\mathrm{d}\boldsymbol{\omega} = \mathrm{ds}\|\mathbf{z}\| = z\,\mathrm{ds}$ where ds is the surface differential form for $\partial\mathcal{S}_c(V)$. Hence,

$$\mu(\mathcal{G}) = \mathrm{d}\mu(\mathcal{S}_c(V)) = \int_{\mathcal{G}} \mathrm{d}\boldsymbol{\omega} = \int_{\partial\mathcal{S}_c(V)} z\,\mathrm{ds} \leq c(\alpha-1)G^{-1}\int_{\partial\mathcal{S}_c(V)} \mathrm{ds}. \tag{12}$$

where $\int_{\partial\mathcal{S}_c(V)} \mathrm{ds}$ does not depend on α or G. Hence, by pushing $\alpha \to 0$, $\partial\mu(\mathcal{S}_c(V)) \backslash \partial\alpha = c\mu(\partial\mathcal{S}_c(V))G^{-1} \propto G^{-1}$ that completes the proof. ■

As a result of this theorem, in some applications, one may need to control $\|\nabla_{\mathbf{x}} V(\mathbf{x})\|$ for $\mathbf{x} \in \partial\mathcal{S}_c(V)$ to prevent sampling from a too large or too small gap.

Appendix C. Theoretical Motivation of $[V(\mathbf{x}) - V_{\pi_{n-1}}(f_\pi(\mathbf{x}))]^2$ in Equation (5)

In this section we discuss the theory behind the term $[V(\mathbf{x}) - V_{\pi_{n-1}}(f_\pi(\mathbf{x}))]^2$ in Equation (5) that we added as an improvement to (Richards et al., 2018) to facilitate learning the RoA. The objective function for the RoA estimation phase is restated here:

$$\mathcal{L}(V) = \sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}} [V(\mathbf{x}) - \bar{c}] - \sum_{\mathbf{x}\in\mathbb{X}^{\text{OUT}}} [V(\mathbf{x}) - \bar{c}] + \lambda_{\text{RoA}} \sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}} \Delta V(\mathbf{x}) + \lambda_{\text{monot}} \sum_{\mathbf{x}\in\mathbb{X}^{\text{IN}}} [V(\mathbf{x}) - V_{\pi_{n-1}}(f_{\pi_{n-1}}(\mathbf{x}))]^2.$$

The first two terms construct a classifier objective. The third term is added to conform with the Lyapunov decrease conditions. Here, we focus on the last term, i.e., $[V(\mathbf{x}) - V_{\pi_{n-1}}(f_\pi(\mathbf{x}))]^2$. The motivation behind adding this term comes from the constructive method proposed by Chiang and Thorp (1989).

The idea of the constructive methodology of Chiang and Thorp (1989) is to construct a sequence of functions $V_0, V_1, \ldots$ for the autonomous dynamical system $\mathbf{x}_{k+1} = f(\mathbf{x}_k)$ in order to use the level sets of the accumulating function of this sequence for estimating the RoA of the vector field f. The theory is developed for a class of functions more general than Lyapunov functions that are called energy-like functions. An energy-like function decreases over the trajectories of the system (see Chiang and Alberto (2015) for a precise definition). Given an energy-like function V_0 for the vector field f, the following sequence of functions is constructed

$$\begin{aligned} V_1(\mathbf{x}) &= V_0(\mathbf{x} + \epsilon_1 f(\mathbf{x})) \\ V_2(\mathbf{x}) &= V_1(\mathbf{x} + \epsilon_2 f(\mathbf{x})) \\ &\cdots \\ V_n(\mathbf{x}) &= V_{n-1}(\mathbf{x} + \epsilon_n f(\mathbf{x})) \end{aligned} \tag{13}$$

where $d_i, i = 1, 2, \cdots, n$ are positive numbers. Two facts about this sequence must be proved: **1)** All functions produced in this sequence are energy-liked functions. **2)** For a fixed positive c, $S_c(V_i) \subset S_c(V_{i+1})$. The following two theorems guarantee these points.

Theorem 6 (Energy-like functions, lemma 4.2 in Chiang and Thorp (1989)) *Let $V : \mathbb{R}^d \to \mathbb{R}$ be an energy-like function for the nonlinear autonomous system $\mathbf{x}_{k+1} = f(\mathbf{x}_k)$ with the equilibrium point $\bar{\mathbf{x}} = \mathbf{0}$. Let $\mathcal{D}$ be a compact set around $\bar{\mathbf{x}}$ that contains no other equilibrium points. Then, there exists an $\tilde{\epsilon} > 0$ such that for $\epsilon < \hat{\epsilon}$, the function $V_1 = V(\mathbf{x} + \epsilon f(\mathbf{x}))$ is also an energy-like function on the compact set $\mathcal{D}$ for the vector field f.*

This theorem guarantees that all functions in the constructive process (13) are energy-like functions.

Theorem 7 (Monotonic level sets, lemma 4.1 in Chiang and Thorp (1989)) *Let $V : \mathbb{R}^d \to \mathbb{R}$ be an energy-like function for the nonlinear autonomous system $\mathbf{x}_{k+1} = f(\mathbf{x}_k)$ with the equilibrium point $\bar{x} = \mathbf{0}$. Let $\mathcal{D}$ be a compact set around $\bar{\mathbf{x}}$ that contains no other equilibrium points. Assume the set $S_c(V) := \{\mathbf{x} : V(\mathbf{x}) \leq c \text{ and } \mathbf{x} \in \mathcal{D}\}$ is non-empty for some constant c. Then there exists an $\tilde{\epsilon} > 0$ such that for the set characterized by $S_c(V_1) := \{\mathbf{x} : V_1(\mathbf{x}) \leq c \text{ and } \mathbf{x} \in \mathcal{D}\}$ where $V_1(\mathbf{x}) = V(\mathbf{x} + \epsilon f(\mathbf{x}))$ and $\epsilon < \tilde{\epsilon}$, the following holds*

$$S_c(V) \subset S_c(V_1). \tag{14}$$

This theorem guarantees that the sequence built by the constructive process (13) gives a monotonically increasing sequence of level sets.

Now, we get back to the added term to the objective function, i.e., $[V(\mathbf{x}) - V_{\pi_{n-1}}(f_{\pi_{n-1}}(\mathbf{x}))]^2$. Notice that $V_{\pi_{n-1}}(\cdot)$ is fixed and the minimization is performed with respect to $V(\cdot)$. Hence, minimizing the above term at each phase of the algorithm emulates the above constructive process and encourages the learned Lyapunov functions to be monotonic in the sense that their sublevel sets $\mathcal{S}_c(V_n)$ tend to become a monotonically increasing sequence that covers more and more space of the true RoA.

Appendix D. Algorithms

The proposed algorithms in this work are verbally described in Section 4.2. To facilitate implementation, the detailed pseudo-code of the algorithms comes here. The RoA estimation phase is realized as Algorithm 1 and the policy update phase is realized as Algorithm 2.

Appendix E. Model-Based Assumption

The model of the system is used to produce the trajectories of the system which is needed in evaluating both objective functions in (6) and (5). However, the information needed to compute (5) can also be obtained by experimentation with the physical system without requiring the dynamics function of the system. In the following, we show that the knowledge of the model of the system can be relaxed in both sub-phases of the algorithm.

E.1. RoA Estimation Sub-Phase

As can be seen in Equation (5) and Algorithm 1, in this sub-phase, the model is used to produce trajectories starting from the sampled initial states from the gap around the current estimate of the RoA or from within the RoA. In both cases, as also suggested by Richards et al. (2018), instead of the model, the real system can be used to produce the trajectories. Assume γ_r in Algorithm 1 satisfies the conditions of Theorem 5. Therefore, the gap from which the initial states are sampled is

Algorithm 1 RoA estimation: Learn $\mathcal{R}_{\pi_n}$ from $(\mathcal{R}_{\pi_{n-1}}, \pi_{n-1})$

```
input  : $(V_{\pi_{n-1}}, c_{n-1})$: The Lyapunov function and level value of stage $n$ where | $\mathcal{R}_{\pi_{n-1}} = \mathcal{S}_{c_{n-1}}(V_{\pi_{n-1}})$ | $f_{\pi_{n-1}}$: Closed-loop system vector field | $\gamma_r > 1$: Level value multiplicative factor | $N \in \mathbb{N}$: Number of sampled states | $M \in \mathbb{N}$: Number of stages | $0 \leq \beta_r \leq 1$: Mixture parameter | $L_r \in \mathbb{N}$: Trajectory length | $\mathcal{D}$: Domain | $\lambda_{\text{RoA}}$: Negative definiteness weighting factor | $\lambda_{\text{monot}}$: Monotonicity weighting factor
output : $(V_{\pi_n}, c_n)$
Init $\hat{V}$ to $V_{\pi_{n-1}}$ and $\hat{c}$ to $c_{n-1}$
 Init the sampling distribution $p_r$ to $U(\mathcal{D})$, i.e., uniform distribution over the domain $\mathcal{D}$
 for $m = 1, \dots, M$ do
    $\mathcal{G} \leftarrow \mathcal{S}_{\gamma_r \hat{c}}(\hat{V}) \backslash \mathcal{S}_{\hat{c}}(\hat{V})$
    $p_r \leftarrow \beta_r U(\mathcal{G}) + (1 - \beta_r) U(\mathcal{D})$
    $\mathbb{X}_0 \leftarrow$ Generate $N$ samples from $p_r$
    $\mathbb{X}_{L_r} \leftarrow$ Run $f_{\pi_{n-1}}$ on $\mathbb{X}_0$ for $L_r$ steps
    $\mathbb{X}_0^{\text{IN}} \leftarrow \{(\mathbf{x}, 1) : \mathbf{x} \in \mathbb{X}_0, \Phi(\mathbf{x}, L_r) \in \mathcal{S}_{\hat{c}}(\hat{V})\}$
    $\mathbb{X}_0^{\text{OUT}} \leftarrow \{(\mathbf{x}, 0) : \mathbf{x} \in \mathbb{X}_0, \Phi(\mathbf{x}, L_r) \notin \mathcal{S}_{\hat{c}}(\hat{V})\}$
    $V^* \leftarrow$ Optimize for $V$ in the objective function (5) using the dataset $\{\mathbb{X}_0^{\text{IN}}, \mathbb{X}_0^{\text{OUT}}\}$
    $\hat{c} \leftarrow \text{argmax}_c\{c \in \mathbb{R} : \Delta f_{\pi_{n-1}}(\mathbf{x}) < 0 \text{ for } \mathbf{x} \in \mathcal{S}_c(V^*)\}$
    $\hat{V} \leftarrow V^*$
end
$(V_{\pi_n}, c_n) \leftarrow (\hat{V}, \hat{c})$
```

Algorithm 2 Policy update: Learn π_{n+1} from $\mathcal{R}_{\pi_n}$

```
input  : $(V_{\pi_n}, c_n)$: The Lyapunov function and level value of stage $n$ where $\mathcal{R}_{\pi_n} = \mathcal{S}_{c_n}(V_{\pi_n})$ | Closed-loop system vector field $f_{\pi_n}$ | $\gamma_p > 1$: Level value multiplicative factor | $N \in \mathbb{N}$: Number of sampled states | $0 \leq \beta_p \leq 1$: Mixture parameter | $L_p \in \mathbb{N}$: Trajectory length | $\lambda_u$: Unstable states weighting factor
output : $\pi_{n+1}$
Init sampling distribution $p_p$ to $U(\mathcal{S}_{c_n}(V_{\pi_n}))$, i.e., uniform distribution over $\mathcal{S}_{c_n}(V_{\pi_n})$
 $\mathcal{G} \leftarrow \mathcal{S}_{\gamma_p c_n}(V_{\pi_n}) \backslash \mathcal{S}_{c_n}(V_{\pi_n})$
 $p_p \leftarrow \beta_p U(\mathcal{G}) + (1 - \beta_p) U(\mathcal{S}_{c_n}(V_{\pi_n}))$
 $\mathbb{X}_0 \leftarrow$ Generate $N$ samples from $p_p$
 $\pi_{n+1} \leftarrow$ Optimize for $\pi$ in the objective function (6) using the dataset $\{\mathbb{X}_0\}$
```

not too large and a hyperbolic physical system does not exhibit drastically different and potentially dangerous behavior when it is launched from initial states sampled from this surrounding gap. Hence, in an approach similar to active learning, the real system can be used to produce the trajectories and label them based on whether they enter the current target level set of the estimated Lyapunov function or not. In conclusion, in this sub-phase, there will be no need to know the model of the system or estimate it.

E.2. Policy Update Sub-Phase

The model requirement is a bit different in this sub-phase compared with the RoA estimation sub-phase. As we need to update the policy, information on the way the behavior of the system changes with respect to a change in the policy is required. However, looking at (6), it is observed that the behavior of the system influences the objective function only via the Lyapunov function $V(\Phi(\mathbf{x}, L_p))$. Suppose the policy is parameterized as $\pi(\mathbf{x}; \phi)$. Then, the closed-loop system becomes $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \pi(\mathbf{x}_k))$. The required gradient to update the policy parameters contains the term $\partial V(\mathbf{x}_k)/\partial\psi$ where the information of ψ is encoded in $\mathbf{x}_k = \Phi(\mathbf{x}_0, k)$. The derivative decomposes as

$$\frac{\partial V(\mathbf{x}_k)}{\partial \psi} = \frac{\partial V(\mathbf{x}_k)}{\partial \mathbf{x}_k}\frac{\partial \mathbf{x}_k}{\partial \psi}. \tag{15}$$

where the knowledge of the model is required to compute $\partial\mathbf{x}_k/\partial\psi$ that will be a matrix of size $\dim(\mathcal{X}) \times \dim(\psi)$. When the model is not given, one must instead try to estimate the model of the system $\hat{f} : \mathbb{R}^d \to \mathbb{R}^d$. However, this vector-valued function is difficult to estimate unless in very limited cases. Instead, one can try to directly estimate V rather than f as a function of ψ. As V is a scalar-valued function, it takes fewer trajectories to obtain a decent estimate of $\partial V/\partial\psi$. Hence, even though the model of the system is needed for this phase, there are two factors that relax this requirement: **1)** As the trajectories are integrated forward only for L_p steps in Algorithm 2, a local estimation would be sufficient. **2)** Even in the local estimation regime, one does not need to estimate the nonlinear vector field as an $\mathbb{R}^d$ to $\mathbb{R}^d$ function. What matters is how the vector field looks like through the lens of the Lyapunov function that is a scalar-valued function. Hence, an $\mathbb{R}^d \to \mathbb{R}^d$ estimation problem can be replaced by an $\mathbb{R}^d \to \mathbb{R}$ estimation problem.

Appendix F. Weak Learning Signal

In this section, we take a closer look at the occasions that the learning signal for the policy update stage of the algorithm (Section 4.2) is weak. We discuss the problem for a general setting and show that the setting of this paper is a special case.

Problem Statement. Assume the discrete-time time-invariant non-autonomous dynamical system $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$. The goal is to design $\mathbf{u}_k$ for $0 \leq k \leq T$ such that $\mathbf{x}_k$ meets some specified conditions for $0 \leq k \leq T$. In the Lyapunov stability analysis, these conditions are assessed by a function $V : \mathcal{X} \to \mathbb{R}^{\geq 0}$, i.e., after rolling out the starting state $\mathbf{x}_0$ for T steps by the dynamics f controlled by $\mathbf{u}_k$, $V(\mathbf{x}_{-T}) \in \mathcal{S}$ where $\mathcal{S}$ encapsulates the desired conditions for the trajectory $\mathbf{x}_{-T} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)$ of the system. Assume the deviation of $V(\mathbf{x}_{-T})$ from its desired set $\mathcal{S}$ is measured by a deviation metric $\mathcal{L}(V(\mathbf{x}_{-T}), \mathcal{S})$. In many control tasks such as *tracking*, the entire trajectory matters, and $\mathcal{L}$ will be a function of every state in the trajectory. However, in control tasks such as *reaching*, only the final state $\mathbf{x}_T$ matters; consequently the objective function $\mathcal{L}$ only depends on $\mathbf{x}_T$. Stability, in the presence of a Lyapunov function, can be seen as an example of the second class of tasks where the relative position of $\mathbf{x}_T$ compared to the level sets of the Lyapunov function is sufficient to decide the convergence of that trajectory. In practice, the control signal $\mathbf{u}_k$ is produced as a parametric function, i.e. $\mathbf{u}_k = \pi(\mathbf{x}_k; \psi)$. Therefore, the entire trajectory $\mathbf{x}_{-T}(\psi)$ is now parameterized by ψ and so is the loss function $\mathcal{L}(\psi)$. The class of algorithms known as *policy gradient* uses $\partial\mathcal{L}(\psi)/\partial\psi$ to learn the controller $\pi(\cdot; \psi)$. In this section, we study the condition of this gradient and show under which circumstances it vanishes and results in slow convergence.

In a general case where V in $\mathcal{L}(V(\mathbf{x}_{-T}), \mathcal{S})$ is a function of the entire trajectory $\mathbf{x}_{-T}$, the gradient w.r.t. the controller parameters is written as:

$$\frac{\partial \mathcal{L}}{\partial \psi} = \sum_{1\leq p\leq T} \frac{\partial \mathcal{L}_p}{\partial \psi} \tag{16}$$

$$\frac{\partial \mathcal{L}_k}{\partial \psi} = \sum_{1\leq p\leq k} \left(\frac{\partial \mathcal{L}}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_p} \frac{\partial^+ \mathbf{x}_k}{\partial \psi} \right) \tag{17}$$

$$\frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_p} = \prod_{p<i\leq k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{p<i\leq k} \left(\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_{i-1}} + \frac{\partial f}{\partial \mathbf{u}}|_{\mathbf{u}=\pi(\mathbf{x}_{i-1})} \frac{\partial \pi(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_{i-1}} \right) \tag{18}$$

The gradients are derived as sum-of-products and $\frac{\partial^+ \mathbf{x}_k}{\partial \psi}$ refers to the *immediate* partial derivative of state $\mathbf{x}_k$ with respect to ψ when $\mathbf{x}_{k-1}$ is considered as a constant.

In the special case where V is the Lyapunov function in $\mathcal{L}(V(\mathbf{x}_{-T}), \mathcal{S})$ and the concern is the stability of the system, $V(\mathbf{x}_{-T}) = V(\mathbf{x}_T)$, meaning that the sum on the r.h.s of (16) will only have one term $\partial \mathcal{L}_T/\partial \psi$ and (17) will transform to

$$\frac{\partial \mathcal{L}_T}{\partial \psi} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_T} \sum_{1\leq k\leq T} \left(\frac{\partial \mathbf{x}_T}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \psi} \right). \tag{19}$$

In the following, we analyze the role of each term that contributes to this gradient.

F.1. The Component $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_T}$

This term concerns the differential condition of $\mathcal{L}(\mathbf{x}) = \mathcal{L}(V(\mathbf{x}), \mathcal{S})$ at $\mathbf{x} = \mathbf{x}_T$ as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_T} = \frac{\partial \mathcal{L}(V)}{\partial V}|_{V=V(\mathbf{x}_T)} \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_T} \tag{20}$$

If any of these terms gets too small, the overall gradient gets small too resulting in a vanishing gradient issue. To avoid this, the algorithm must ensure that these terms remain sufficiently large. For the first term $\partial \mathcal{L}(V)/\partial V$, this condition is normally fulfilled if the deviation metric $\mathcal{L}(\cdot,\cdot)$ is designed properly. For example, one candidate function for $\mathcal{L}$ would be a signed distance function that measures how far $\mathbf{x}_T$ is from the maximal stable level set $\partial S_{c_{\max}}(V)$, e.g., $\mathcal{L}(V(\mathbf{x}_T), S_{c_{\max}}(V)) = V(\mathbf{x}_T) - c_{\max}$. Another candidate deviation metric could be $\mathcal{L}(V(\mathbf{x}_T), S_{c_{\max}}(V)) = \max((V(\mathbf{x}_T) - c_{\max}), 0)$ that does not care about the actual value of $V(\mathbf{x}_T)$ as long as $\mathbf{x}_T$ lives within the sublevel set $S_{c_{\max}}(V)$. For the former case, $\partial \mathcal{L}/\partial V = V$ meaning that the gradient does not vanish as long as $\mathbf{x}_T \neq \bar{\mathbf{x}}$.

The second term of the r.h.s of (20) depends on the slope of the function V evaluated at the final state of the trajectory. If V is a candidate Lyapunov function, $\nabla_{\mathbf{x}} V(\mathbf{x})$ vanishes at the equilibrium as we proved in Lemma 4. Therefore, if $\mathbf{x}_T$ is too close to the equilibrium (i.e., the system converges), the learning signal to update the policy will vanish. This point is formalized in the following remark.

Remark 8 *Consider the closed-loop system $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \pi(\mathbf{x}_k))$. Let $\mathbf{x}_0$ be the initial state of a trajectory that starts from within the RoA of the equilibrium point $\bar{\mathbf{x}}$ of the closed-loop system denoted by $\mathcal{R}_\pi^{\bar{\mathbf{x}}}$. Let V be the Lyapunov function and the objective function $\mathcal{L}(V_\pi(\mathbf{x}_{-T}), \mathcal{S})$ is optimized to update the policy π. One must be careful not to let the trajectories roll out for too long ($T \gg 1$). As $\mathbf{x}_T$ gets too close to the equilibrium, the information of the trajectory degenerates (see Theorem 9)* *and the gradient to update the controller vanishes (see Lemma 4). On the other hand, If $\mathbf{x}_0$ does not belong to the RoA of $\bar{\mathbf{x}}$, it can go too far from $\mathcal{R}_\pi^{\bar{\mathbf{x}}}$ and may escape the validity domain of the Lyapunov function V. Hence, in either case, the length T of the trajectory influences the information content of the trajectory for updating the policy. Both too large and too small values of T must be avoided when the trajectories pass through the Lyapunov function and the Lyapunov function acts as a critic in the algorithm for learning the policy. Too small values of T are non-informative due to the continuity of V. Too large values of T, on the other hand, is non-informative as the trajectory enters the flat regions of V or escapes the domain of validity of V.*

Theorem 9 *Consider the dynamical system $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ where the control signal is issued by the policy function $\mathbf{u}_k = \pi(\mathbf{x}_k; \psi)$ that is parameterized by ψ. Let $\bar{\mathbf{x}} = \mathbf{0}$ be an asymptotic equilibrium point of this system. If $\mathbf{x}_0 \in \mathcal{R}_\pi^{\mathbf{0}}$ and $V : \mathcal{X} \to \mathbb{R}^{0\geq}$ is a C^r Lyapunov function with $r \geq 1$, then*

$$\forall \epsilon > 0, \exists T > 0 \;\; \text{such that} \;\; \|\frac{\partial V(\mathbf{x})}{\partial \psi}|_{\mathbf{x}=\mathbf{x}_T}\| < \epsilon \tag{21}$$

Proof *It can be seen in (19) that $\nabla_{\mathbf{x}} V$ appears multiplicatively in $\nabla_\psi V$. As stated in Lemma 4, $\nabla_{\mathbf{x}} V(\mathbf{x}) = 0$ at $\bar{\mathbf{x}} = \mathbf{0}$ that makes $\nabla_\psi V$ vanish as well at the equilibrium point. Now suppose $V \in C^r(r \geq 1)$ with respect to both its arguments $\mathbf{x}$ and $\mathbf{u}$. Moreover, $\pi(\cdot, \psi)$ is assumed to be smooth with respect to its parameters ψ. Therefore, the map $\psi \to \nabla_\psi V$ is continuous. Furthermore, $\mathbf{x}_k \to \mathbf{0}$ as $k \to \infty$ because $\mathbf{x}_0 \in \mathcal{R}_\pi^{\mathbf{0}}$. The continuity of the map $\psi \to \nabla_\psi V$ together with the above result on vanishing $\nabla_\psi V$ at the equilibrium point completes the proof.* ■

The effect of the other terms of Equation (19) on the learning signal is discussed below.

F.2. The Component $\frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_p}$

This term determines how state information propagates forward through the trajectory. More precisely, it shows how perturbing a state at time p affects the downstream state $\mathbf{x}_k$ after $k - p$ time steps when the parameters of the system are kept fixed. In the following, we take a closer look at the constituent terms of (18). We first define the function

$$J_i(\mathbf{x}_i, \mathbf{u}_i) = [\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_i}, \frac{\partial f}{\partial \mathbf{u}}|_{\mathbf{u}=\mathbf{u}_i}]^\mathsf{T} \tag{22}$$

that is different form (18) in the sense that $\mathbf{u}_i$ is not necessarily a function of $\mathbf{x}_i$. Due to the fact that $\partial \mathbf{x}_k / \partial \mathbf{x}_p$ equals the product of $J_i = J_i(\mathbf{x}_i, \mathbf{u}_i)$ along the trajectory $\{(\mathbf{x}_i, \mathbf{u}_i)\}_{i=1}^{i=k-1}$, a measure of the size of J_i would be informative about the influence of $\mathbf{x}_p$ on $\mathbf{x}_k$. First, we consider the vanishing gradient issue when the influence becomes too small. Notice that the size of J_i as defined in (22) is determined by two terms as (we drop the index i for convenience)

$$\|J(\mathbf{x}, \mathbf{u})\| \leq \|\frac{\partial f}{\partial \mathbf{x}}\| + \|\frac{\partial f}{\partial \mathbf{u}}\|. \tag{23}$$

Let f be the dynamics of the closed-loop system, i.e., $f = f_\pi$. If f is Lipschitz continuous (as it is assumed in Section 2 to ensure the existence and uniqueness of the solution), both terms on the r.h.s. of the inequality (23) will be bounded. However, the condition of Equation (22) is a more general case when $\mathbf{u}$ is not necessarily a function of the states. In this case, f must be Lipschitz in both $\mathbf{x}$ and $\mathbf{u}$, that is, there exists positive constants $L_{f_\mathbf{x}}$ and $L_{f_\mathbf{u}}$ such that:

$$\|\partial f(\mathbf{x},\mathbf{u})/\partial \mathbf{x}\| \leq L_{f_\mathbf{x}},\ \forall \mathbf{u} \in \mathcal{U} \text{ and } \forall \mathbf{x} \in \mathcal{X} \tag{24}$$

$$\|\partial f(\mathbf{x},\mathbf{u})/\partial \mathbf{u}\| \leq L_{f_\mathbf{u}},\ \forall \mathbf{u} \in \mathcal{U} \text{ and } \forall \mathbf{x} \in \mathcal{X}. \tag{25}$$

Intuitively, this means that the open-loop dynamics $f(\mathbf{x},\mathbf{u})$ rolls out smoothly and does not respond too harshly to the changes in the control input. If a component of the system breaks down under some control input, $\mathbf{u} \in \mathcal{U}$, the above conditions do not hold. In addition, it might be the case that the dynamics of the system show high-frequency vibrations under some specific control inputs (e.g., when the controller excites the natural frequency of the system). These conditions occur rarely in physical systems when the controller remains within a reasonable working regime but may happen frequently under *adversarial* regimes when the system is intentionally attacked by an unauthorized user. Such regimes are beyond the scope of this paper. Hence, we can reasonably assume that J_i is upper bounded by $2 \times \max(L_{f_{\mathbf{x}_i}}, L_{f_{\mathbf{u}_i}})$.

A more general control signal consists of two parts. The first part is a function of states and the second part is open-loop. Therefore, we have:

$$\mathbf{u}_i = \pi(\mathbf{x}_i;\psi) + \tilde{\mathbf{u}}_i \ \text{ with } \ \frac{\partial \tilde{\mathbf{u}}_i}{\partial \mathbf{x}_i} = 0.$$

We can write the overall dynamics as $\mathbf{x}_{i+1} = f(\mathbf{x}_i,\mathbf{u}_i) = f(\mathbf{x}_i,(\pi(\mathbf{x}_i),\tilde{\mathbf{u}}_i)) = f(\mathbf{x}_i,\tilde{\mathbf{u}}_i)$ which transforms to the case of (22) with the difference that the policy function $\pi(\mathbf{x})$ is now absorbed in the first component of (22). Hence, $J_i(\mathbf{x}_i,\tilde{\mathbf{u}}_i)$ decomposes as

$$J_i(\mathbf{x}_i,\tilde{\mathbf{u}}_i) = [\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_i} + \frac{\partial f}{\partial \pi(\mathbf{x})}|_{\pi(\mathbf{x})=\pi(\mathbf{x}_i)}, \frac{\partial f}{\partial \tilde{\mathbf{u}}}|_{\tilde{\mathbf{u}}=\tilde{\mathbf{u}}_i}]^\mathsf{T}$$

Observe that Equation (18) depends only on the first block of $J_i(\mathbf{x}_i,\tilde{\mathbf{u}}_i)$ and $\partial f/\partial \tilde{\mathbf{u}}$ does not affect the gradient even though it affects the trajectory of the system. As a result, the influence of the open-loop control signal $\tilde{\mathbf{u}}_t$ on $\partial \mathcal{L}/\partial \psi$ is via the final states of the trajectory $\mathbf{x}_T$ as well as $\partial \mathcal{L}/\partial \mathbf{x}_T$ in (19).

F.3. The Component $\frac{\partial^+ \mathbf{x}_k}{\partial \psi}$

This component captures the effect of the policy parameters on the next state of the system when the current state is kept fixed. Using chain rule we have

$$\frac{\partial^+ \mathbf{x}_k}{\partial \psi} = \frac{\partial f(\mathbf{x}_k,\pi)}{\partial \pi}|_{\pi=\pi(\mathbf{x}_k;\psi)} \times \frac{\partial \pi(\mathbf{x}_k;\psi)}{\partial \psi}. \tag{26}$$

The first term in the r.h.s. depends on how sensitive is the dynamics function f with respect to its control argument. Lower bounding this sensitivity can ensure that the effect of the controller remains visible throughout the trajectory. More rigorously, we assume there exists sufficiently large $\kappa > 0$ such that

$$\|\frac{\partial f(\mathbf{x},\pi)}{\partial \pi}|_{\pi=\pi(\mathbf{x};\psi)}\| > \kappa, \qquad \text{for every } x \in \mathcal{D}$$

where $\mathcal{D}$ is the domain of interest in which the system operates.

The second term on the r.h.s of Equation (26) is independent of the rest of the system and shows the sensitivity of the control signal with respect to the parameters of the controller. For example, if the controller is implemented by a neural network with tanh activation functions in the hidden layers, this term can vanish if the weights diverge and push the activation values towards the saturation regimes of tanh.

Appendix G. Training Details

G.1. Lyapunov Function

A parametric candidate Lyapunov function $V(\cdot;\theta)$ must satisfies the conditions of (2). It must be positive definite on a domain $\mathcal{D}$ and its value must decrease along the trajectories of the system. As the second condition depends on the system, it is embedded in the optimization loss function (5). The first condition (positive definiteness) though does not depend on the system. It must be enforced for every $\mathbf{x} \in \mathcal{D}$. Rather than including it in the loss function, we restrict the hypothesis set $V(\cdot;\theta) \in \mathcal{H}$ to the class of positive definite functions. In kernel methods, this property can be achieved by a proper choice of kernels. With neural networks, $V(\cdot;\theta)$ can be represented by $V(\mathbf{x};\theta) = v(\mathbf{x};\theta)^\mathsf{T} v(\mathbf{x};\theta)$ as a Lyapunov candidate function where $v(\mathbf{x};\theta)$ is a multilayer perceptron. This guarantees the non-negativeness of $V(\cdot;\theta)$. To ensure $V(\mathbf{x};\theta)$ does not vanish at any point other than the origin, first suppose $\mathbf{z}_\ell$ and $\mathbf{z}_{\ell+1}$ are the input and output of the ℓ^{th} layer respectively, i.e., $\mathbf{z}_{\ell+1} = \zeta_\ell(W\mathbf{z}_\ell)$ where W is the weight matrix and $\zeta_\ell(\cdot)$ is the activation function. To guarantee the strict positiveness of $v(\mathbf{x};\theta)$ for $\mathbf{x} \neq \mathbf{0}$, both activation function ζ_ℓ and weight matrix $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ must have trivial nullspaces. Activation functions such as tanh and ReLU meet this condition. The weight matrix can be constructed as the following:

$$\mathbf{W}_\ell = \begin{bmatrix} \mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1} + \varepsilon \mathbf{I}_{d_{\ell-1}} \\ \mathbf{G}_{\ell 2} \end{bmatrix} \tag{27}$$

where $\mathbf{G}_{\ell 1} \in \mathbb{R}^{q_\ell \times d_{\ell-1}}$ for some $q_\ell \in \mathbb{N}_{\geq 1}, \mathbf{G}_{\ell 2} \in \mathbb{R}^{(d_\ell - d_{\ell-1}) \times d_{\ell-1}}, \mathbf{I}_{d_{\ell-1}} \in \mathbb{R}^{d_{\ell-1} \times d_{\ell-1}}$ is the identity matrix, and $\epsilon \in \mathbb{R}_+$ is a positive constant to ensure the upper block has full rank. See Remark 1 in the appendix of (Richards et al., 2018) for further details.

In our experiments, $v(\mathbf{x};\theta)$ is represented by a 3-layer multilayer perceptron each layer of dimension 64 followed by tanh activation functions and a linear last layer.

G.2. Discretized Time and State Space

We discretize the dynamics of the inverted pendulum with a time resolution of $\Delta T = 0.01$. The state space is also discretized in the rectangle $[\theta_{\text{min}}, \theta_{\text{max}}] = [-\pi/2, \pi/2], [\omega_{\text{min}}, \omega_{\text{max}}] = [-2\pi, 2\pi]$. Each dimension is divided equally into 100 sections to form a $2-$dimensional grid. Therefore, all the computations of Section 5 take place with a finite set of states. For example, the true RoA denoted by the green plot in Figure 1(Right) is computed by integrating forward all states of the state space grid. If the resolution of the grid is too coarse, the bound on the negative definiteness of $\Delta V(\mathbf{x})$ must change from 0 to a more conservative negative value in order to make sure the critical Lyapunov decrease condition will not be violated. The new bound depends on the Lipschitz constants of V and f (see Richards et al. (2018) for the detailed derivation of the bound).

G.3. Hyper-parameters

The pre-training phase is performed with the learning rate 0.001 and $10k$ training steps. Each run of the RoA estimation algorithm is performed with the learning rate 0.01 and $10k$ training steps. Each run of the policy update phase is performed with the learning rate 0.01 with 100 steps. The number of learning steps in the policy update phase is a proxy to the distance between the current policy and the updated policy. Hence, by keeping this number fairly small, we make sure the condition of Assumption 1 is likely to be fulfilled.

In Equation (5), λ_{RoA} is set to 1000 to enforce the Lyapunov decrease condition. In the same equation, λ_{monot} is set to 0.01 to encourage to monotonicity of the RoA estimation algorithm. In Equation (6), λ_u is set to 10 to put more emphasis on stabilizing the unstable states compared with keeping stable the already stable states.

The sampling mixture parameters β_r and β_p are both set to 0.6 in Algorithms 1 and 2 respectively.

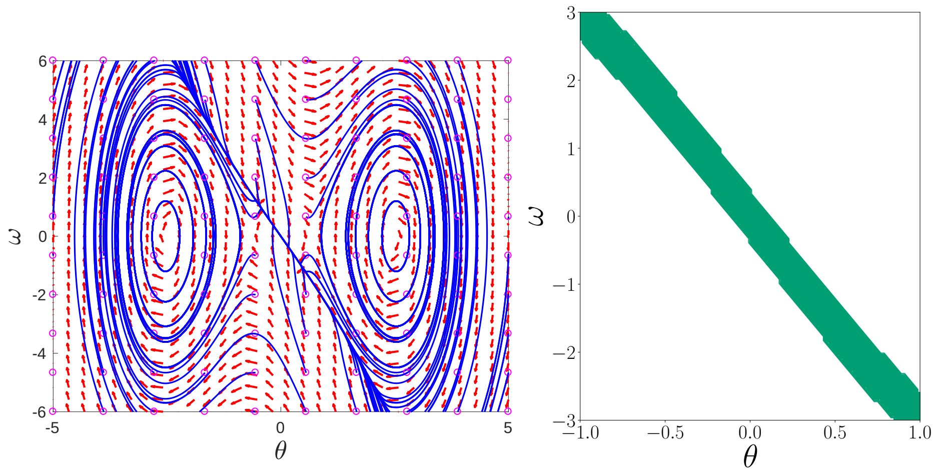
The length of the integrated trajectories (L_r, L_p) is set to 10 for both RoA estimation (see Algorithm 1) and policy update (see Algorithm 2) phases.

The multiplication constants γ_r and γ_p for both RoA estimation and policy update phases are set to 4.

The number of RoA estimation stages m in Algorithm 1 is set to 20. The total number of policy update phases is also set to 20. One can alternatively use a context-aware stopping criterion. For instance, updating the policy can be stopped when no significant change in the RoA or the policy parameters is observed.

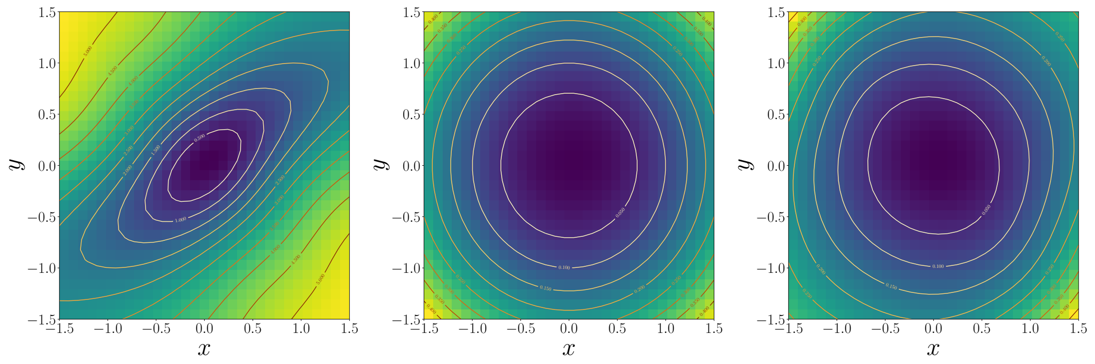
The number of sampled initial states N is initialized to 10 in Algorithms 1 and 2 and increases by 10 after each update of the policy. The heuristic reason is that after each policy update, the RoA enlarges and it takes more samples to obtain a good representative of the gap surrounding the RoA.

Appendix H. More Experimental Results



(a) Phase portrait (b) RoA

Figure 5: Dynamics of the inverted pendulum and its initial RoA for an LQR controller



(a) Untrained (b) Quadratic (c) Trained

Figure 6: Pre-training the neural network with a quadratic Lyapunov function. The background heatmap represents the value of the underlying function. Lighter regions correspond to larger values.(a) The level sets of the untrained initialized neural network. (b) The target quadratic function (2) The level sets of the neural network pre-trained with the quadratic function of Figure 6b.

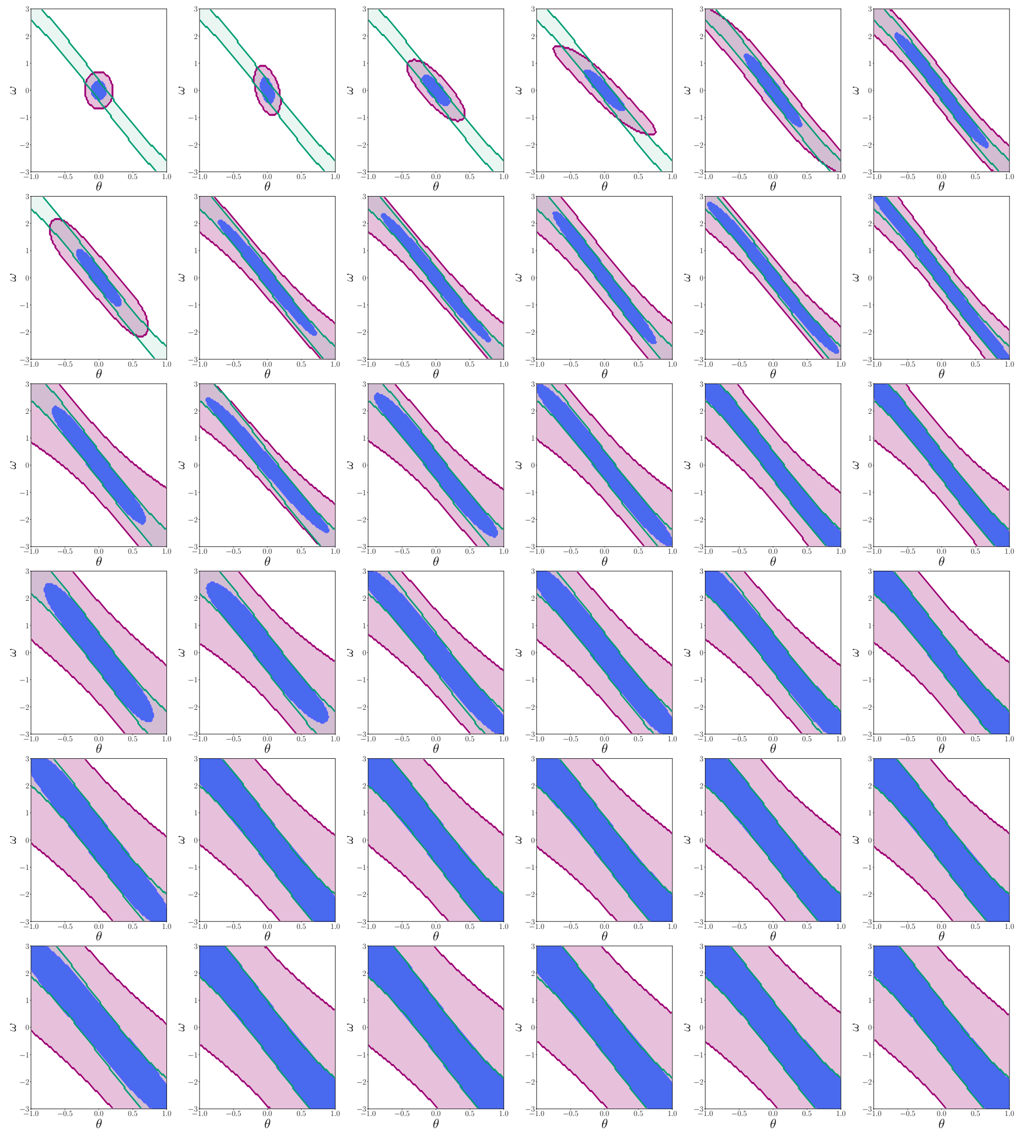


Figure 7: Visualizing the true ROA which is enlarged by the improved policy and is chased by a learned Lyapunov function. Each row corresponds to a policy update stage and each column corresponds to a RoA estimation stage. At each row, from left to right, the policy is fixed that results in a fixed true RoA (green plot). The columns from left to right are the stages of the RoA estimation phase (see Algorithm 1). The blue color is the estimated RoA $S_{c_n}(V_{\pi_n})$ and the pink color shows the gap $\mathcal{G} = S_{\gamma c_n}(V_{\pi_n}) \backslash S_{c_n}(V_{\pi_n})$ that is used in Algorithms 1 and 2 for sampling the initial states. After the RoA estimation phase is done (the rightmost figure of each row), the policy update phase is performed. The leftmost figure in the next row shows that the true RoA enlarges as a result of the policy update. The RoA estimation phase continues from its latest stage which is a decent initial approximate for the enlarged RoA. As a result of the alternate application of RoA estimation and policy update phases, the bottom rightmost figure shows a significantly larger RoA compared with the top leftmost figure (see the green boundary of the true RoA around the blue region that is the estimated RoA by the learned Lyapunov function). Note that the Lyapunov function is also learned such that its level set (blue region) almost perfectly matches the true RoA (green boundary) in the bottom rightmost figure.