UNIVERSITY OF CALIFORNIA SAN DIEGO

Neural Lyapunov Methods for Learning-based Control

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Ya-Chien Chang

Committee in charge:

Professor Sicun Gao, Chair
Professor Nikolay Atanasov
Professor Henrik Christensen
Professor Sylvia Herbert

2025

The Dissertation of Ya-Chien Chang is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2025

DEDICATION

I dedicate this dissertation to my family, whose love and support have carried me through every step of this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my advisor, Professor Sicun Gao, for his unwavering support, insightful guidance, and constant encouragement throughout my doctoral journey. He consistently placed my development and well-being at the forefront, always considering what was best for me academically, professionally, and personally. His mentorship has profoundly shaped my growth both as a researcher and as an individual. Through his student-centered advising and the freedom he gave me to explore research independently, he made the process intellectually fulfilling and deeply enjoyable. Having him as my advisor has been a true gift, and I will carry his guidance and the wisdom he shared with me throughout my life.

I would like to acknowledge my committee members, Professor Nikolai Atanasov, Prof. Henrik Christensen and Professor Sylvia Herbert, for their valuable discussions and insightful feedback to this dissertation. Their perspectives have helped shape my work, and I am sincerely grateful for their time and dedication.

I would also like to thank my co-authors, Nima Roohi, Milan Ganai, and Chiaki Hirayama, for their collaboration, ideas, and support throughout our research. I am especially grateful to my colleagues in Professor Gao's research group, who made our office a fun and supportive place to work. I feel lucky to have been part of such a vibrant, kind, and uplifting group.

To the friends I've made in San Diego, thank you for all the companionship, laughter, and unforgettable memories. Your presence brought balance and joy to my life beyond academics, and I will always treasure our time together.

Finally, I am deeply thankful to my family for their unconditional love and support. Their belief in me gave me strength, their encouragement motivated me to keep going, and their presence filled my life with purpose and happiness. I could not have done this without them.

Chapter 2, in full, is a modified reprint of material previously published in Ya-Chien Chang, Nima Roohi, and Sicun Gao. "Neural Lyapunov Control." Advances in Neural Information Processing Systems (NeurIPS), 2019. The dissertation author was the primary investigator and author of this material.

Chapter 3, in full, is a modified reprint of material previously published in Ya-Chien Chang and Sicun Gao, "Stabilizing Neural Control Using Self-Learned Almost Lyapunov Critics," IEEE International Conference on Robotics and Automation (ICRA), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 4, in full, is a modified reprint of material previously published in 3. Milan Ganai, Chiaki Hirayama, Ya-Chien Chang, and Sicun Gao, "Learning Stabilization Control from Observations by Learning Lyapunov-like Proxy Models," IEEE International Conference on Robotics and Automation (ICRA), 2023. The material in this chapter represents collaborative work.

Chapter 5, in full, is a modified reprint of material previously published in Ya-Chien Chang, and Sicun Gao, "Extremum-Seeking Action Selection for Accelerating Policy Optimization," IEEE International Conference on Robotics and Automation (ICRA), 2024. The dissertation author was the primary investigator and author of this material.

VITA

2013 M.S. in Applied Mathematics, National Tsing Hua University, Taiwan

2025 Ph.D in Computer Science, University of California San Diego, USA

PUBLICATIONS

Ya-Chien Chang, Nima Roohi, and Sicun Gao. "Neural Lyapunov Control." Advances in Neural Information Processing Systems 32 (2019).

Ya-Chien Chang, and Sicun Gao. "Stabilizing neural control using self-learned almost lyapunov critics." 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

Milan Ganai, Chiaki Hirayama, **Ya-Chien Chang**, and Sicun Gao. "Learning Stabilization Control from Observations by Learning Lyapunov-like Proxy Models." 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023.

Ya-Chien Chang, and Sicun Gao. "Extremum-Seeking Action Selection for Accelerating Policy Optimization." 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Artificial Intelligence
Professor Sicun Gao

ABSTRACT OF THE DISSERTATION

Neural Lyapunov Methods for Learning-based Control

by

Ya-Chien Chang

Doctor of Philosophy in Computer Science

University of California San Diego, 2025

Professor Sicun Gao, Chair

Learning-based control methods have demonstrated strong capabilities in solving complex control tasks in robotics. However, their broader adoption is often limited by sample inefficiency and the lack of formal guarantees regarding the stability and safety of the learned controllers. This dissertation introduces a set of frameworks that enhance the reliability and efficiency of learning-based control by extending the use of Lyapunov methods and introducing adaptive action-selection strategies.

In this thesis, we first introduce neural Lyapunov methods that jointly optimize stabilizing controllers and Lyapunov functions, providing provable guarantees for the stability of dynamical systems. These methods significantly simplify nonlinear control design and achieve substantially

larger regions of attraction compared to existing control methods. We then develop algorithms that extend neural Lyapunov methods to a variety of control settings, including model-based stabilization of nonlinear systems, model-free reinforcement learning, and imitation learning from limited expert demonstrations. In all cases, the learned controllers demonstrate enhanced stability and robustness compared to existing learning-based techniques. To further address the inefficiencies of exploration in reinforcement learning, we propose novel methods based on Extremum-Seeking Control that improve action selection by leveraging local optimization signals. These approaches enable more efficient learning by adaptively directing exploration toward the most informative regions within the state space.

Taken together, the methods developed in this thesis form a principled framework for learning-based control that is both robust and scalable across simulated and real-world robotic systems. By applying neural Lyapunov techniques to ensure formal stability guarantees, and adaptive exploration strategies to improve learning efficiency, this work closes the gap between theoretical guarantees and practical deployment. As a result, it advances the safety, efficiency, and real-world applicability of learning-based control in robotics.

Chapter 1

Introduction

1.1 Background

Stability is a foundational concept in control theory, ensuring that dynamical systems behave predictably and remain safe in the presence of disturbances or uncertainty. This property is especially critical in modern applications such as robotic locomotion, autonomous vehicles, and aerospace systems, where safety and robustness are non-negotiable. As these systems become more complex and high-dimensional, the need for scalable, flexible, and reliable control synthesis methods has grown substantially. In response, recent advances in machine learning have introduced powerful tools for modeling unknown dynamics and generating control policies directly from data. However, integrating such data-driven methods with formal stability guarantees remains an open challenge.

In control theory, Lyapunov functions are powerful tools for analyzing the stability of nonlinear systems as well as designing controllers with provable guarantee of stability, by providing explicit representation or estimations of the *regions of attraction* (ROA). Lyapunov functions can be considered as a general form of energy functions, which are scalar functions whose values decrease along the system trajectories, so that the states of the dynamical system are forced to approach the equilibrium or the goal state of the system, and eventually stabilize. When Lyapunov functions can be constructed for a controlled dynamical system, then we can rigorously prove that the system is stable in this way. Moreover, the level sets of the Lyapunov

function can be used to characterize the region of attraction of the system, which is also called a stability region, guaranteeing that all states starting inside will never escape and converge to the goal point at the end.

Although Lyapunov functions are powerful tools once we have them, the most difficult challenge is to find them in the first place. For even mildly nonlinear systems, there is no general analytic form for Lyapunov functions. In fact, even *checking* that a function is a Lyapunov function is nontrivial (NP-hard in fact), because we need to ensure that the function is positive definite and its derivative over time along the system trajectories is negative everywhere. Numerical methods for searching Lyapunov functions have been attracting much attention for decades [1]. One mainstream approach is to use sum-of-squares (SOS) representations [2] for the Lyapunov functions, and semidefinite programming [3] for the search through convex optimization. The sum-of-squares approach can only work with dynamical system that can be expressed with polynomial dynamics, and Lyapunov functions that are SOS polynomials [4–6]. The applicability of SOS polynomial Lyapunov functions has been widely explored for different kinds of system in control. However, many theoretical results [7–9] show that no polynomial Lyapunov function of any degree can be found for a very simple stable polynomial systems. Practically, scalability and numerical sensitivity issues restrict the use of SOS-based approaches to only a few classes of low-dimensional dynamical systems.

Since the 1990s, neural networks have been explored as function approximators for constructing Lyapunov functions in dynamical systems [10–13]. Early work leveraged gradient-based optimization to train multi-layer perceptrons that approximate scalar functions satisfying Lyapunov conditions. The general strategy was to iteratively update the network parameters to produce non-decreasing functions over a sampled training set, offering a data-driven alternative framework to construct Lyapunov functions. These early efforts demonstrated the potential of neural networks as Lyapunov function approximators and laid the groundwork for more general frameworks in this direction. However, most early approaches [14–16] do not attempt to evaluate the validity of the learned neural Lyapunov functions because it is very hard to

produce global guarantees on these functions. In response, a growing body of research [17–21] began to develop neural Lyapunov design frameworks that integrate learning objectives with stability-aware training processes, including methods that approximate ROA boundaries more effectively and improve generalization across the state space. To further improve the soundness of neural Lyapunov functions, learner-verifier frameworks have emerged, aiming to ensure that Lyapunov conditions are satisfied not just over a sampled state set but throughout the entire state space [22–24]. These frameworks consist of a learner that proposes candidate Lyapunov functions and a verifier that searches for counterexamples that violate the Lyapunov conditions. The identified counterexamples are added to the training set to guide the learner to focus its training on the regions surrounding them. This counterexample-guided training significantly enhances the robustness of the neural Lyapunov functions.

Building on these developments, and as learning-based control techniques have emerged to address the limitations of analytic control in complex, high-dimensional, and uncertain environments, recent research has incorporated neural Lyapunov functions as certificates to ensure the stability of learned control policies during training and exploration. These neural Lyapunov certificates are embedded directly into learning objectives or integrated into guided training loops, enabling policies that achieve high performance while satisfying formal stability guarantees. By bridging the gap between learning-based control architectures and rigorous stability analysis, neural Lyapunov functions provide a scalable and generalizable framework for certifying the behavior of complex dynamical systems. This neural Lyapunov framework opens new opportunities for designing and implementing certified neural controllers, and has driven active research in safe reinforcement learning with neural Lyapunov methods.

1.2 Dissertation Overview

This dissertation explores neural Lyapunov methods for learning-based control, emphasizing stability guarantees and their applicability across a broad range of control settings. The

work begins with a framework that co-optimizes Lyapunov functions and stabilizing controllers, resulting in provably stable control policies and significantly enlarged regions of attraction compared to traditional analytic approaches. Building upon this core contribution, we extend the methodology to diverse settings, including model-based stabilization, model-free reinforcement learning, and imitation learning from limited demonstrations. In each case, the proposed controllers exhibit improved stability and robustness relative to standard learning-based methods. To further address the challenge of sample inefficiency in reinforcement learning, we introduce an Extremum-Seeking Control strategy that guides action selection through local optimization signals. These strategies enable more efficient learning by directing exploration toward the most informative regions of the state space.

In Chapter 2, we propose a learner-verifier framework for synthesizing both control policies and neural Lyapunov functions for nonlinear systems, with provable guarantees of stability. The approach significantly simplifies the process of Lyapunov-based control design, provides end-to-end correctness guarantees, and achieves much larger regions of attraction compared to existing methods such as linear-quadratic regulators, sum-of-squares and semidefinite programming.

Chapter 3, extends neural Lyapunov methods to the model-free reinforcement learning setting by developing algorithms for training neural control policies and neural Lyapunov critic functions. We show that the learned Lyapunov critics can be used to estimate the regions of attraction of the controllers based on Almost Lyapunov condition. The effectiveness of the proposed methods is demonstrated across a range of nonlinear control tasks.

In Chapter 4, the model-free Lyapunov-based method introduced in the previous chapter is further developed to accelerate Learning from Observation (LfO) for stabilization control problems. In LfO, the objective is to learn control policies from expert demonstrations that provide only state trajectories, without corresponding action information. We propose a new formulation that first learns a Lyapunov landscape model from expert demonstration and then leverages it to guide the policy learning process. The approach enables efficient learning with

significantly fewer expert observations than state-of-the-art LfO methods, demonstrating high performance in both real-world autonomous vehicle scenarios and simulated environments.

In Chapter 5, we address the challenge of improving action selection in model-free reinforcement learning setting by introducing additional adaptive control steps based on Extremum-Seeking Control (ESC). For each action sampled from stochastic policies, we apply sinusoidal perturbations and query for estimated Q-values as the response signal. Based on ESC, we then adaptively adjust the sampled actions toward nearby optima before applying them to the environment. This methods can be easily integrated into standard policy optimization to enhance learning efficiency. We demonstrate its effectiveness across a range of continuous control tasks.

Finally, Chapter 6 concludes the dissertation by summarizing the contributions and their broader impact. It also outlines promising directions for future exploration within the area of neural Lyapunov methods for learning-based control.

Chapter 2

Neural Lyapunov Control

We propose new methods for learning control policies and neural network Lyapunov functions for nonlinear control problems, with provable guarantee of stability. The framework consists of a learner that attempts to find the control and Lyapunov functions, and a falsifier that finds counterexamples to quickly guide the learner towards solutions. The procedure terminates when no counterexample is found by the falsifier, in which case the controlled nonlinear system is provably stable. The approach significantly simplifies the process of Lyapunov control design, provides end-to-end correctness guarantee, and can obtain much larger regions of attraction than existing methods such as LQR and SOS/SDP. We show experiments on how the new methods obtain high-quality solutions for challenging control problems.

2.1 Introduction

Learning-based methods hold the promise of solving hard nonlinear control problems in robotics. Most existing work focuses on learning control functions represented as neural networks through repeated interactions of an unknown environment in the framework of deep reinforcement learning, with notable success. However, there are still well-known issues that impede the immediate use of these methods in practical control applications, including sample complexity, interpretability, and safety [25]. Our work investigates a different direction: Can learning methods be valuable even in the most classical setting of nonlinear control design? We

focus on the challenging problem of designing feedback controllers for stabilizing nonlinear dynamical systems with provable guarantee. This problem captures the core difficulty of underactuated robotics [26]. We demonstrate that neural networks and deep learning can find provably stable controllers in a direct way and tackle the full nonlinearity of the systems, and significantly outperform existing methods based on linear or polynomial approximations such as linear-quadratic regulators (LQR) [27] and sum-of-squares (SOS) and semidefinite programming (SDP) [2]. The results show the promise of neural networks and deep learning in improving the solutions of many challenging problems in nonlinear control.

The prevalent way of stabilizing nonlinear dynamical systems is to linearize the system dynamics around an equilibrium, and formulate LQR problems to minimize deviation from the equilibrium. LQR methods compute a linear feedback control policy, with stability guarantee within a small neighborhood where linear approximation is accurate. However, the dependence on linearization produces extremely conservative systems, and it explains why agile robot locomotion is hard [26]. To control nonlinear systems outside their linearizable regions, we need to rely on Lyapunov methods [28]. Following the intuition that a dynamical system stabilizes when its energy decreases over time, Lyapunov methods construct a scalar field that can force stabilization. These fields are highly nonlinear and the need for function approximations has long been recognized [28]. Many existing approaches rely on polynomial approximations of the dynamics and the search of sum-of-squares polynomials as Lyapunov functions through semidefinite programming (SDP) [2]. A rich theory has been developed around the approach, but in practice the polynomial approximations pose much restriction on the systems and the Lyapunov landscape. Moreover, well-known numerical sensitivity issues in SDP [29] make it very hard to find solutions that fully satisfy the Lyapunov conditions. In contrast, we exploit the expressive power of neural networks, the convenience of gradient descent for learning, and the completeness of nonlinear constraint solving methods to provide full guarantee of Lyapunov conditions. We show that the combination of these techniques produces control designs that can stabilize various nonlinear systems with verified regions of attraction that are much larger than

what can be obtained by existing control methods.

We propose an algorithmic framework for learning control functions and neural network Lyapunov functions for nonlinear systems without any local approximation of their dynamics. The framework consists of a learner and a falsifier. The learner uses stochastic gradient descent to find parameters in both a control function and a neural Lyapunov function, by iteratively minimizing the *Lyapunov risk* which measures the violation of the Lyapunov conditions. The falsifier takes a control function and Lyapunov function from the learner, and searches for *counterexample* state vectors that violate the Lyapunov conditions. The counterexamples are added to the training set for the next iteration of learning, generating an effective curriculum. The falsifier uses delta-complete constraint solving [30], which guarantees that when no violation is found, the Lyapunov conditions are guaranteed to hold for all states in the verified domain. In this framework, the learner and falsifier are given tasks that are difficult in different ways and can not be achieved by the other side. Moreover, we show that the framework provides the flexibility for fine-tuning the control performance by directly enlarging the region of attraction on demand, by adding regulator terms in the learning cost.

We experimented with several challenging nonlinear control problems in robotics, such as drone landing, wheeled vehicle path following, and n-link planar robot balancing [31]. We are able to find new control policies that produce certified region of attractions that are significantly larger than what can be established previously. We provide a detailed analysis of the performance comparison between the proposed methods and the LQR/SOS/SDP methods.

Related Work. Compared to the control-theoretic approaches, neural Lyapunov control provides a much simpler design process, relying purely on gradient-based methods for the learning. The saving is similar to the reduction of feature engineering and specific optimization methods in other areas of AI. The recent work of Richards *et. al.* [17] has also proposed and shown the effectiveness of using neural networks to learn safety certificates in a Lyapunov framework, but our goals and approaches are different. Richards *et. al.* focus on discrete-time nonlinear systems and the use of neural networks to learn the region of attraction of a given controller.

The Lyapunov conditions are validated in relaxed forms through sampling. Special design of the neural architecture is required to compensate the lack of complete checking over all states. In comparison, we focus on learning the control and the Lyapunov function together with provable guarantee of stability in larger regions of attraction. Our approach directly handles continuous nonlinear dynamical systems, does not assume control functions are given other than an initialization, and uses generic feed-forward network representations without manual design. Our approach successfully works on many more nonlinear systems, and find new control functions that enlarge regions of attraction obtainable from standard control methods. Related learning-based approaches for finding Lyapunov functions include [32–35]. There is strong evidence that linear control functions are all we need for solving highly nonlinear control problems through reinforcement learning as well [36], suggesting convergence of different learning approaches. In the control and robotics community, similar learner-falsifier frameworks have been proposed by [37, 38] without using neural network representations. The common assumption is the Lyapunov functions are high-degree polynomials. In these methods, an explicit control function and Lyapunov function can not be learned together because of the bilinear optimization problems that they generate. Our approach significantly simplifies the algorithms in this direction and has worked reliably on much harder control problems compared to existing methods. Several theoretical results on asymptotic Lyapunov stability [7, 8, 39, 40] show that some very simple dynamical systems do not admit a polynomial Lyapunov function of any degree, despite being globally asymptotically stable. Such results further motivates the use of neural networks as a more suitable function approximator. A large body of work in control uses SOS representations and SDP optimization in the search for Lyapunov functions [2, 41–44]. However, scalability and numerical sensitivity issues have been the main challenge in practice. As is well known, the number of semidefinite programs from SOS decomposition grows quickly for low degree polynomials [2].

2.2 Preliminaries

We consider the problem of designing control functions to stablize a dynamical system at an equilibrium point. We make extensive use of the following results from Lyapunov stability theory.

Definition 1 (Controlled Dynamical Systems)**.** An n-dimensional controlled dynamical system is

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f_u(x), \quad x(0) = x_0 \tag{2.1}$$

where $f_u : \mathscr{D} \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, and $\mathscr{D} \subseteq \mathbb{R}^n$ is an open set with $0 \in \mathscr{D}$ that defines the state space of the system. Each $x(t) \in \mathscr{D}$ is a state vector. The feedback control is defined by a continuous function $u : \mathbb{R}^n \to \mathbb{R}^m$, used as a component in the full dynamics f_u.

Definition 2 (Asymptotic Stability)**.** We say that system of (1) is stable at the origin if for any $\varepsilon \in \mathbb{R}^+$, there exists $\delta(\varepsilon) \in \mathbb{R}^+$ such that if $\|x(0)\| < \delta$ then $\|x(t)\| < \varepsilon$ for all $t \geq 0$. The system is asymptotically stable at the origin if it is stable and also $\lim_{t\to\infty} \|x(t)\| = 0$ for all $\|x(0)\| < \delta$.

Definition 3 (Lie Derivatives)**.** The Lie derivative of a continuously differentiable scalar function $V : \mathscr{D} \to \mathbb{R}$ over a vector field f_u is defined as

$$\mathsf{L}_{f_u} V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} [f_u]_i(x)$$

It measures the rate of change of V along the direction of the system dynamics.

Proposition 1 (Lyapunov Functions for Asymptotic Stability)**.** *Consider a controlled system* (1) *with equilibrium at the origin, i.e.,* $f_u(0) = 0$*. Suppose there exists a continuously differentiable function* $V : \mathscr{D} \to \mathbb{R}$ *that satisfies the following conditions:*

$$V(0) = 0, \textit{ and, } \forall x \in \mathscr{D} \setminus \{0\}, V(x) > 0 \textit{ and } \mathsf{L}_{f_u} V(x) < 0. \tag{2.2}$$

Then, the system is asymptotically stable at the origin and V is called a Lyapunov function.

Linear-Quadratic Regulators (LQR) is a widely-adpoted optimal control strategy. LQR controllers are guaranteed to work within a small neighborhood around the stationary point where the dynamics can be approximated as linear systems. A detailed description can be found in [27].

2.3 Learning to Stabilize with Neural Lyapunov Functions

We now describe how to learn both a control function and a neural Lyapunov function together, so that the Lyapunov conditions can be rigorously verified to ensure stability of the system. We provide pseudocode of the algorithm in Algorithm 1.

2.3.1 Control and Lyapunov Function Learning

We design the hypothesis class of candidate Lyapunov functions to be multilayered feedforward networks with tanh activation functions. It is important to note that unlike most other deep learning applications, we can not use non-smooth networks, such as with ReLU activations. This is because we will need to analytically determine whether the Lyapunov conditions hold for these neural networks, which requires the existence of their Lie derivatives.

For a neural network Lyapunov function, its input is any state vector of the system in Definition (8) and the output is a scalar value. We write θ to denote the parameter vector for a Lyapunov function candidate V_θ. For notational convenience, we write u to denote both the control function and the parameters that define the function. The learning process updates both the θ and u parameters to improve the likelihood of satisfying the Lyapunov conditions, which we formulate as a cost function named the *Lyapunov risk*.

The Lyapunov risk measures the degree of violation of the following Lyapunov conditions, as shown in Proposition (11). First, the value of $V_\theta(x)$ is positive; Second, the value of the Lie derivative $\mathsf{L}_{f_u} V_\theta(x)$ is negative; Third, the value of $V_\theta(0)$ is zero. Conceptually, the overall

Lyapunov control design problem is about minimizing the minimax cost of the form

$$\inf_{\theta,u} \sup_{x\in\mathscr{D}} \Big(\max(0,-V_\theta(x)) + \max(0, \mathsf{L}_{f_u} V_\theta(x)) + V_\theta^2(0) \Big).$$

The difficulty in control design problems is that the violation of the Lyapunov conditions can not just be estimated, but needs to be fully guaranteed over all states in $\mathscr{D}$. Thus, we need to rely on global search with complete guarantee for the inner maximization part, which we delegate to the falsifier explained in Section 3.2. For the learning step, we define the following Lyapunov risk function.

Definition 4 (Lyapunov Risk)**.** Consider a candidate Lyapunov function V_θ for a controlled dynamical system f_u from Definition 8. The Lyapunov risk is a defined by the following function

$$L_\rho(\theta,u) = \mathbb{E}_{x\sim\rho(\mathscr{D})} \Big(\max(0,-V_\theta(x)) + \max(0, \mathsf{L}_{f_u} V_\theta(x)) + V_\theta^2(0) \Big), \tag{2.3}$$

where x is a random variable over the state space of the system with a distribution ρ. In practice, we work with the Monte Carlo estimate named the *empirical Lyapunov risk* by drawing samples:

$$L_{N,\rho}(\theta,u) = \frac{1}{N} \sum_{i=1}^{N} \Big(\max(-V_\theta(x_i),0) + \max(0, \mathsf{L}_{f_u} V_\theta(x_i)) \Big) + V_\theta^2(0), \tag{2.4}$$

where $x_i, 1 \le i \le N$ are samples of the state vectors sampled according to $\rho(\mathscr{D})$.

It is clear that the empirical Lyapunov risk is an unbiased estimator of the Lyapunov risk function. It is clear that $L_{N,\rho}$ is an unbiased estimator of L_ρ.

Note that L_ρ is positive semidefinite, and any (θ,u) that corresponds to a true Lyapunov function satisfies $L(\theta,u)$=0. Thus, Lyapunov functions define global minimizers of the Lyapunov risk function.

Proposition 2. *Let V_{θ_o} be a Lyapunov function for dynamical system f_{u_o} where u_o is the control parameters. Then (θ_o,u_o) is a global minimizer for L_ρ and $L_\rho(\theta_o,u_o) = 0$.*

Note that both V_θ and f_u are highly nonlinear (even though u is almost always linear in practice), and thus $L(\theta, u)$ generates a highly complex landscape. Surprisingly, multilayer feedforward tanh networks and stochastic gradient descent can quickly produce generalizable Lyapunov functions with nice geometric properties, as we report in detail in the experiments. In Figure 2.1 (b), we show an example of how the Lyapunov risk is minimized over iterations on the inverted pendulum example.

Initialization and improvement of control performance over LQR.

Because of the local nature of stochastic gradient descent, it is hard to learn good control functions through random initialization of control parameters. Instead, the parameters u in the control function are initialized to the LQR solution, obtained for the linearized dynamics in a small neighborhood around the stationary point. On the other hand, the initialization of the neural network Lyapunov functions can be completely random. We observe that the final learned controller often delivers significantly better control solutions than the initalization from LQR. Figure 2.1(a) shows how the learned control reduces oscillation of the system behavior in the 2-link planar robot balancing example and achieve more stable control.

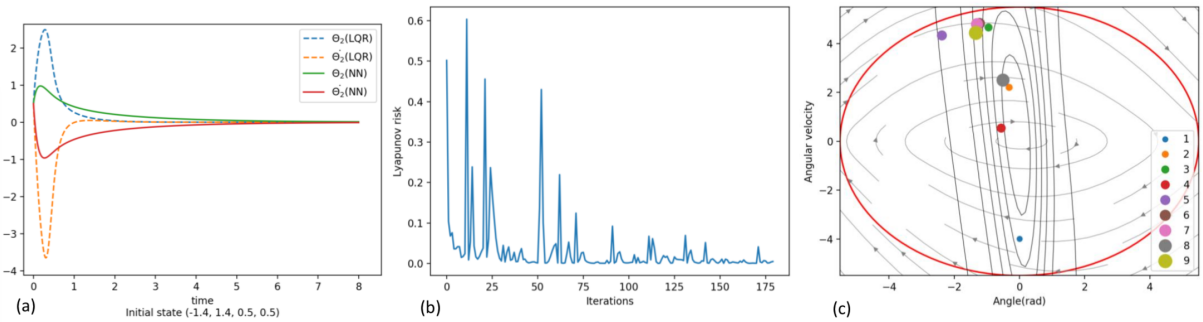

Figure 2.1. (a) Comparison between LQR and deep-learned controllers for 2-link planar robot balancing. (b) The Lyapunov risk decreases quickly over iterations. (c) Counterexamples returned by falsifiers from several epochs, which quickly guides the learner to focus on sepcial regions in the space.

2.3.2 Falsification and Counterexample Generation

For each control and Lyapunov function pair (V_θ, u) that the learner obtains, the falsifier's task is to find states that violate the Lyapunov conditions in Proposition 11. We formulate the *negations* of the Lyapunov conditions as a nonlinear constraint solving problem over real numbers. These *falsification constraints* are defined as follows.

Definition 5 (Lyapunov Falsification Constraints)**.** Let V be a candidate Lyapunov function for a dynamical system defined by f_u defined in state space $\mathscr{D}$. Let $\varepsilon \in \mathbb{Q}^+$ be a small constant parameter that bounds the tolerable numerical error. The Lyapunov falsification constraint is the following first-order logic formula over real numbers:

$$\Phi_\varepsilon(x) := \left(\sum_{i=1}^{n} x_i^2 \geq \varepsilon\right) \wedge \left(V(x) \leq 0 \vee \mathsf{L}_{f_u} V(x) \geq 0\right)$$

where x is bounded in the state space $\mathscr{D}$ of the system. The numerical error parameter ε is explicitly introduced for controlling numerical sensitivity near the origin. Here ε is orders of magnitude smaller than the range of the state variables, i.e., $\sqrt{\varepsilon} \ll \min(1, ||\mathscr{D}||_2)$.

Remark 1. The numerical error parameter ε allows us to avoid pathological problems in numerical algorithms such as arithmetic underflow. Values inside this tiny ball correspond to disturbances that are physically insignificant. This parameter is important for eliminating from our framework the numerical sensitivity issues commonly observed in SOS/SDP methods. Also note the ε-ball does not affect properties of the Lyapunov level sets and regions of attraction outside it (i.e., $\mathscr{D} \setminus B_\varepsilon$).

The falsifier computes solutions of the falsification constraint $\Phi_\varepsilon(x)$. Solving the constraints requires global minimization of a highly nonconvex functions (involving Lie derivatives of the neural network Lyapunov function), and it is a computationally expensive task (NP-hard). We rely on recent progress in nonlinear constraint solving in SMT solvers such as dReal [30], which has been used for similar control design problems [38] that do not involve neural networks.

Example 1. *Consider a candidate Lyapunov function* $V(x) = \tanh(a_1x_1 + a_2x_2 + b)$ *and dynamics* $\dot{x_1} = -x_2^2$ *and* $\dot{x_2} = \sin(x_1)$. *The falsification constraint is of the following form*

$$\Phi_\varepsilon(x) := (x_1^2 + x_2^2) \geq \varepsilon \wedge (\tanh(a_1x_1 + a_2x_2 + b) \leq 0 \vee a_1(1 - \tanh^2(a_1x_1 + a_2x_2 + b))(-x_2^2) + a_2(1 - \tanh^2(a_1x_1 + a_2x_2 + b))\sin(x_1) \geq 0))$$

which is a nonlinear non-polynomial disjunctive constraint system. The actual examples used in our experiments all use larger two-layer tanh *networks and much more complex dynamics.*

To completely certify the Lyapunov conditions, the constraint solving step for $\Phi_\varepsilon(x)$ can never fail to report solutions if there is any. This requirement is rigorously proved for algorithms in SMT solvers such as dReal [45], as a property called delta-completeness [30].

Definition 6 (Delta-Complete Algorithms)**.** Let C be a class of quantifier-free first-order constraints. Let $\delta \in \mathbb{Q}^+$ be a fixed constant. We say an algorithm $\mathscr{A}$ is δ-complete for C, if for any $\varphi(x) \in C$, $\mathscr{A}$ always returns one of the following answers correctly: φ does not have a solution (unsatisfiable), or there is a solution $x = a$ that satisfies $\varphi^\delta(a)$. Here, φ^δ is defined as a small syntactic variation of the original constraint (precise definitions are in [30]).

In other words, if a delta-complete algorithm concludes that a formula $\Phi_\varepsilon(x)$ is unsatisfiable, then it is guaranteed to not have any solution. In our context, this is exactly what we need for ensuring that the Lyapunov condition holds over all state vectors. If $\Phi_\varepsilon(x)$ is determined to be δ-satisfiable, we obtain counterexamples that are added to the training set for the learner. Note that the counterexamples are simply state vectors without labels, and their Lyapunov risk will be determined by the learner, not the falsifier. Thus, although it is possible to have spurious counterexamples due to the δ error, they are used as extra samples and do not harm correctness of the end result. In all, when delta-complete algorithms in dReal return that the falsification constraints are unsatisfiable, we conclude that the Lyapunov conditions are satisfied by the candidate Lyapunov and control functions. Figure 2.1(c) shows a sequence of counterexamples

found by the falsifier to improve the learned results.

Remark 2. When solving $\Phi_\varepsilon(x)$ with δ-complete constraint solving algorithms, we use $\delta \ll \varepsilon$ to reduce the number of spurious counterexamples. Following delta-completeness, the choice of δ does not affect the guarantee for the validation of the Lyapunov conditions.

2.3.3 Tuning Region of Attraction

An important feature of the proposed learning framework is that we can adjust the cost functions to learn control and Lyapunov functions favoring various additional properties. In fact, the most practically important performance metric for a stabilizing controller is its region of attraction (ROA). An ROA defines a forward invariant set that is guaranteed to contain all possible trajectories of the system, and thus can conclusively prove safety properties. Note that the Lyapunov conditions themselves do not directly ensure safety, because the system can deviate arbitrarily far before coming back to the stable equilibrium. Formally, the ROA of an asymptotically stable system is defined as:

Definition 7 (Region of Attraction)**.** Let f_u define a system asymptotically stable at the origin with Lyapunov function V for domain $\mathscr{D}$. A region of attraction R is a subset of $\mathscr{D}$ that contains the origin and guarantees that the system never leaves R. Any level set of V completely contained in $\mathscr{D}$ defines an ROA. That is, for $\beta > 0$, if $R_\beta = \{V(x) \leq \beta\} \subseteq \mathscr{D}$, then R_β is an ROA for the system.

To maximize the ROA produced by a pair of Lyapunov function and control function, we add a cost term to the Lyapunov risk that regulates how quickly the Lyapunov function value increases with respect to the radius of the level sets, by using $L_{N,p}(\theta,u) + \frac{1}{N}\sum_{i=1}^{N} \|x_i\|_2 - \alpha V_\theta(x_i)$ following Definition 12. Here α is tunable parameter. We observe that the regulator can have major effect on the performance of the learned control functions. Figure 2.2 illustrates such an example, showing how different control functions are obtained by regulating the Lyapunov risk to achieve larger ROA.

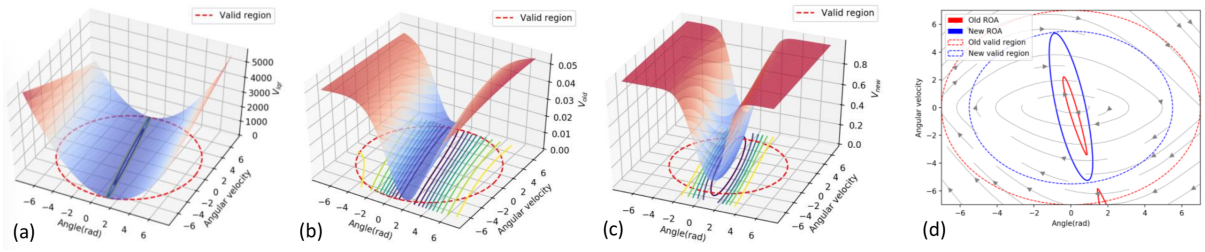

Figure 2.2. (a) Lyapunov function found by the initial LQR controller. (b) Lyapunov function found by learning without tuning the ROA. (c) Lyapunov function found by learning after adding the ROA tuning term. (d) Comparison of ROA for the different Lyapunov functions.

Algorithm 1. Neural Lyapunov Control

function LEARNING(X, f, q^{lqr})
  Set learning rate (0.01), input dimension (# of state variables), output dimension (1)
  Initialize feedback controller u to LQR solution q^{lqr}
  Repeat:
    $V_\theta(x), u(x) \leftarrow \text{NN}_{\theta,u}(x)$ ▷ Forward pass of neural network
    $\mathsf{L}_{f_u} V_\theta(x) \leftarrow \sum_{i=1}^{D_{in}} \frac{\partial V}{\partial x_i}[f_u]_i(x)$
    Compute Lyapunov risk $L(\theta, u)$
    $\theta \leftarrow \theta + \alpha \nabla_\theta L(\theta, u)$
    $u \leftarrow u + \alpha \nabla_u L(\theta, u)$ ▷ Update weights using SGD
  Until convergence
  return V_θ, u
end function
function FALSIFICATION($f, u, V_\theta, \varepsilon, \delta$)
  Encode conditions in Definition 5
  Using SMT solver with δ to verify the conditions
  return satisfiability
end function
function MAIN( )
  Input: dynamical system (f), parameters of LQR (q^{lqr}), radius (ε), precision (δ) and an initial set of randomly sampled states in D
  while Satisfiable **do**
    Add counterexamples to X
    $V_\theta, u \leftarrow$ LEARNING(X, f, q^{lqr})
    CE$\leftarrow$ FALSIFICATION($f, u, V_\theta, \varepsilon, \delta$)
  end while
end function

2.4 Experiments

We demonstrate that the proposed methods find provably stable control and Lyapunov functions on various nonlinear robot control problems. In all the examples, we use a learning rate of 0.01 for the learner, an ε value of 0.25 and δ value of 0.01 for the falsifier, and re-verify the result with smaller ε in Table 2.1. We emphasize that the choices of these parameters do not affect the stability guarantees on the final design of the control and Lyapunov functions. We show that the region of attraction is enlarged by 300% to 600% compared to LQR results in these examples. Full details of the results and system dynamics are provided in the Appendix. Note that for the Caltech ducted fan and n-link balancing examples, we numerically relaxed the conditions slightly when the learning has converged, so that the SMT solver dReal does not run into numerical issues. More details on the effect of such relaxation can be found in the paper website [46].

Table 2.1. Runtime statistics of the full procedures on four nonlinear control examples.

Benchmarks	Learning time	falsification time	# samples	# iterations	ε
Inverted Pendulum	25.5	0.6	500	430	0.04
Path Following	36.3	0.2	500	610	0.01
Caltech Ducted Fan	1455.16	50.84	1000	3000	0.01
2-Link Balancing	6000	458.27	1000	4000	0.01

Inverted pendulum. The inverted pendulum is a standard nonlinear control problem for testing different control methods. This system has two state variables, the angular position θ, angular velocity $\dot{\theta}$ and one control input u. θ and $\dot{\theta}$ represent the angular position from the inverted position and angular velocity. The system dynamics can be described as

$$\ddot{\theta} = \frac{mg\ell\sin(\theta) + u - 0.1\dot{\theta}}{m\ell^2} \tag{2.5}$$

Using constants $g = 9.81$, $m = 0.15$ and $\ell = 0.5$, our learning procedure finds the following

neural Lyapunov function: $V = \tanh(W_2 \tanh(W_1 x + B_1) + B_2)$, where $x = [\theta\ \dot{\theta}]^T$, with

$$W_1 = \begin{bmatrix} -1.1751 & 0.0265 & 0.0439 & -0.5518 & -0.0067 & 0.4446 \\ 0.0288 & -0.0007 & -0.0030 & -0.0348 & -0.0067 & 0.2599 \end{bmatrix}^T$$

$$W_2 = \begin{bmatrix} 0.6047 & -0.6942 & -1.1177 & -1.1330 & 0.7800 & -0.2621 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} -1.2251 & -0.8158 & -0.5308 & 0.8925 & 0.9339 & 1.0895 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0.1592 \end{bmatrix}.$$

The linear control function is given by $u = -23.1717\theta - 6.7996\dot{\theta}$. The neural Lyapunov function is proved to be valid within the domain $\|x\|_2 \leq 6$. In contrast, the ROA found by SOS/SDP techniques is an ellipse with large diameter of 1.75 and short diameter of 1.2. Using LQR control on the linearized dynamics, we obtain an ellipse with large diameter of 6 and short diameter of 0.1. We observe that among all the examples in our experiments, this is the only one where the SOS Lyapunov function has passed the complete check by the constraint solver, so that we can compare to it. The Lyapunov function obtained by LQR gives a larger ROA if we ignore the linearization error. The different regions of attractions are shown in Figure 2.3. These values are consistent with the approximate maximum region of attraction reported in [17]. In particular, Figure 2.3 (c) shows that the SOS function does not define a big enough ROA, as many trajectories escape its region.

Caltech ducted fan in hover mode. The system models the motion of a landing aircraft in hover mode, controlled by two input forces u_1 and u_2. The state variables x, y, θ denote the position and orientation of the centre of the fan. The full equations of motion, involving the six-dimensional state vector $[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$, can be written as:

$$\begin{aligned} \ddot{x} &= \frac{-d\dot{x} + u_1 \cos(\theta) - u_2 \sin(\theta)}{m}, \\ \ddot{y} &= \frac{-d\dot{y} + u_1 \sin(\theta) - u_2 \cos(\theta) - mg}{m}, \\ \ddot{\theta} &= \frac{r u_1}{I}, \end{aligned} \tag{2.6}$$

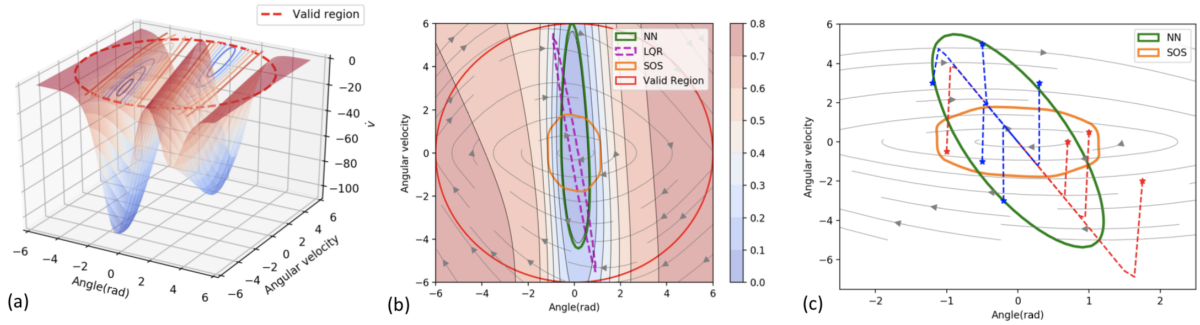

Figure 2.3. Results of Lyapunov functions for inverted pendulum. (a) Lie derivative of learned Lyapunov function over valid region. Its value is negative over the valid region, satisfying the Lyapunov conditions. (b) ROA estimated by different Lyapunov functions. Our method enlarges the ROA from LQR three times. (c) Validation of ROAs. Stars represent initial states. It shows trajectories start near border of the ROA defined by the learned neural Lyapunov function are safely bounded within the green region. On the contrary, many trajectories (red) starting inside the SOS region can escape, and thus the region fails to satisfy the ROA properties.

where $g = 0.28$, $m = 11.2$, $I = 0.0462$, $r = 0.156$ and $d = 0.1$. We find a neural Lyapunov function: $V = \tanh(W_2 \tanh(W_1 x + B_1) + B_2)$, where $x = [x\ y\ \theta\ \dot{x}\ \dot{y}\ \dot{\theta}]^T$, with

$$W_1 = \begin{bmatrix} 0.0314 & 0.0190 & -0.1893 & 0.2532 & 0.0177 & -0.0890 \\ -0.0397 & 0.0242 & 0.1094 & -0.1346 & 0.0186 & 0.1177 \\ -0.1221 & 0.0584 & 0.1417 & -0.0897 & -0.0658 & 0.0060 \\ -0.0853 & -0.0682 & -0.0680 & 0.1741 & 0.2397 & 0.0061 \\ 0.0847 & 0.0065 & 0.0952 & -0.1782 & 0.3689 & 0.0006 \\ -0.1239 & 0.2481 & -0.0991 & 0.2475 & -0.0408 & 0.0017 \end{bmatrix}$$

,

$$W_2 = \begin{bmatrix} 0.0563 & 0.0368 & 0.0218 & -0.0158 & -0.0093 & -0.0186 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} -0.6099 & -0.5518 & 0.1146 & 0.1873 & 0.2220 & 0.4308 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0.0666 \end{bmatrix}$$

and two neural controllers:

$$u_1 = 0.5000x + 0.000002y - 2.1339\theta + 2.7899\dot{x} - 0.00000003\dot{y} - 1.3992\dot{\theta}$$

$$u_2 = 0.000001x - 1.0000y - 0.000003\theta - 0.000003\dot{x} - 5.0407\dot{y} - 0.000001\dot{\theta}$$

In Figure 2.4(a), we show that the ROA is significantly larger than what can be obtained from LQR.

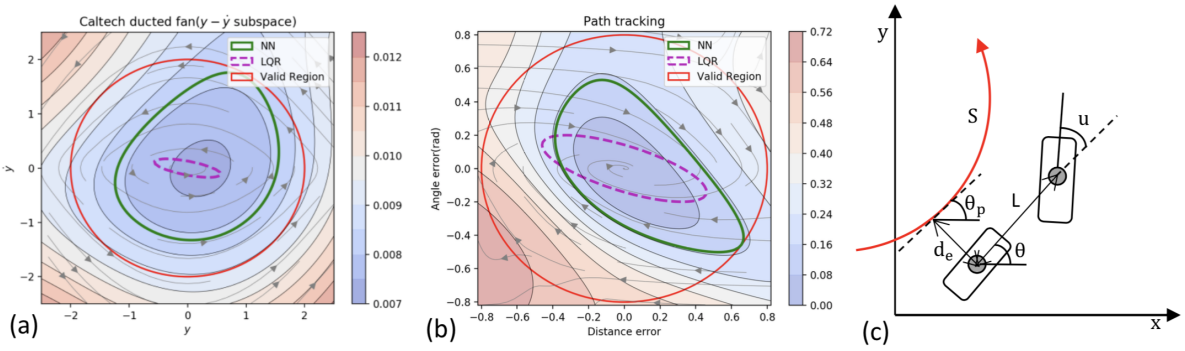

Figure 2.4. (a) Comparison of ROAs for Caltech ducted fan. (b) Comparison of ROAs for path following. (c) Schematic diagram of wheeled vehicle to show the nonlinear dynamics.

Wheeled vehicle path following. We consider the path tracking control using kinematic bicycle model (see Figure 2.4(c)). The system state is defined by the angle error θ_e and the distance error d_e, where $\theta_e = \theta - \theta_p$. With this formulation, the system dynamics can be written as:

$$\begin{aligned} \dot{s} &= \frac{v\cos(\theta_e)}{1-\dot{d}_e\kappa(s)}, \\ \dot{d}_e &= v\sin(\theta_e), \\ \dot{\theta}_e &= \frac{v\tan(u)}{L} - \frac{v\kappa(s)\cos(\theta_e)}{1-\dot{d}_e\kappa(s)}. \end{aligned} \tag{2.7}$$

Assume a target path is a unit circle, then we obtain the following Lyapunov function within $\|x\|_2 \leq 0.8$, $V = \tanh(W_2\tanh(W_1x+B_1)+B_2)$, where $x = [d_e\ \theta_e]^T$, with

$$W_1 = \begin{bmatrix} -2.5250 & -0.4774 & -0.5239 & -0.0232 & -0.0627 & 1.3562 \\ -0.1841 & -0.6964 & -0.5862 & -0.5032 & -0.5620 & 2.5184 \end{bmatrix}^T$$

$$W_2 = \begin{bmatrix} 0.6251 & -1.0490 & -1.0708 & 0.4644 & 0.7019 & -1.1287 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} -1.2776 & -0.4641 & -0.3699 & 0.9194 & 0.9758 & 1.3282 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0.0997 \end{bmatrix}$$

and the neural controller is $u = -0.8471 d_e - 1.6414 \theta_e$.

N-Link Planar Robot Balancing. The n-link pendulum system has n control inputs and $2n$ state variables, as follows:

$$[\theta_1, \theta_2, \ldots, \theta_n, \dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_n]$$

, representing the n link angles and n angle velocities. Each link has mass m_i and length ℓ_i, and the moments of inertia I_i are computed from the link pivots, where $i = 1, 2, \ldots, n$, then the dynamics has the form:

$$M(\theta)\,\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + \tau(\theta) = Bu, \tag{2.8}$$

where

$$\theta = [\theta_1, \theta_2, \ldots, \theta_n]^{\mathrm{T}} \in \mathbb{R}^n, u \in \mathbb{R}^n$$

$$M(\theta) = \left[a_{ij} \cos\left(\theta_j - \theta_i\right)\right], M(\theta) \in \mathbb{R}^{n \times n}$$

$$C(\theta, \dot{\theta}) = \left[-a_{ij} \dot{\theta}_j \sin\left(\theta_j - \theta_i\right)\right], C(\theta, \dot{\theta}) \in \mathbb{R}^{n \times n},$$

$$\tau(\theta) = [-b_i \sin \theta_i], G(\theta) \in \mathbb{R}^n,$$

$$B = [1, 1, \ldots, 1]^{\mathrm{T}}$$

$$\begin{cases} a_{ii} = I_i + m_i \ell_{ci}^2 + \ell_i^2 \sum_{k=i+1}^{n} m_k, 1 \leq i \leq n \\ a_{ij} = a_{ji} = m_j \ell_i \ell_{cj} + \ell_i \ell_j \sum_{k=j+1}^{n} m_k, 1 \leq i < j \leq n \end{cases}$$

$$b_i = \left(m_i \ell_{ci} + \ell_i \sum_{k=i+1}^{n} m_k \right) g, 1 \leq i \leq n,$$

For the 2-link pendulum system, our approach can find a neural Lyapunov function that is valid within domain $\mathscr{D} : \|x\|_2 \leq 0.5$: $V = \tanh(W_2 \tanh(W_1 x + B_1) + B_2)$, where $x = [\theta_1\ \theta_2\ \dot{\theta}_1\ \dot{\theta}_2]^T$

$$W_1 = \begin{bmatrix} -0.3578 & -0.2339 & -0.5153 & -0.2648 \\ 0.4244 & 0.3886 & 0.1041 & 0.0195 \\ -0.4218 & -0.4314 & -0.4371 & -0.2353 \\ -0.0042 & -0.0020 & -0.0013 & -0.0077 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 0.1670 & -0.1353 & -0.2582 & 0.5208 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} -0.4547 & 0.0263 & 0.6899 & 0.7721 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0.7633 \end{bmatrix},$$

and two neural controllers are

$$u_1 = -49.5249\theta_1 - 20.0854\theta_2 - 21.4826\dot{\theta}_1 - 10.0516\dot{\theta}_2$$

$$u_2 = -18.2287\theta_1 - 19.143\theta_2 - 9.2905\dot{\theta}_1 - 6.6695\dot{\theta}_2$$

Additionally, the learning procedure finds a neural Lyapunov function for the 3-link pendulum system on the valid domain $\mathscr{D} : \|x\|_2 \leq 0.5$ with precision $\delta = 0.01$. The neural Lyapunov function: $V = \tanh(W_2 \tanh(W_1 x + B_1) + B_2)$, where $x = [\theta_1\ \theta_2\ \theta_3\ \dot{\theta}_1\ \dot{\theta}_2\ \dot{\theta}_3]^T$

$$W_1 = \begin{bmatrix} -0.1919 & 0.1715 & -0.0481 & 0.0707 & 0.1923 & 0.0548 \\ 0.0943 & 0.0112 & 0.0027 & 0.0102 & -0.0005 & 0.0002 \\ 0.0942 & -0.2393 & 0.0932 & -0.0692 & -0.1582 & -0.0221 \\ -0.1136 & -0.1927 & -0.0753 & -0.0407 & -0.1289 & 0.0246 \\ -0.1645 & 0.2017 & 0.0412 & -0.1091 & -0.1892 & -0.1396 \\ 0.0868 & 0.0103 & 0.0030 & 0.0094 & -0.0002 & -0.0007 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 0.0017 & 0.4299 & 0.0023 & -0.0021 & 0.0002 & -0.5047 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} -0.5246 & -0.3993 & -0.3698 & 0.1214 & 0.2343 & 0.4633 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 0.3918 \end{bmatrix},$$

and three neural controllers are

$$u_1 = -101.7856\theta_1 - 8.9265\theta_2 - 3.467\theta_3 - 28.5081\dot{\theta}_1 - 14.0951\dot{\theta}_2 - 7.3643\dot{\theta}_3$$

$$u_2 = 15.8736\theta_1 - 62.5769\theta_2 - 4.0104\theta_3 - 7.8591\dot{\theta}_1 - 12.6341\dot{\theta}_2 - 7.3690\dot{\theta}_3$$

$$u_3 = 5.1672\theta_1 + 7.2750\theta_2 - 42.4820\theta_3 - 2.6997\dot{\theta}_1 - 4.9186\dot{\theta}_2 - 11.8446\dot{\theta}_3$$

In Figure 5, we show the shape of the neural Lyapunov functions on two of the dimensions, and

the ROA that the control design achieves. We also provide a video of the control on the 3-link model.

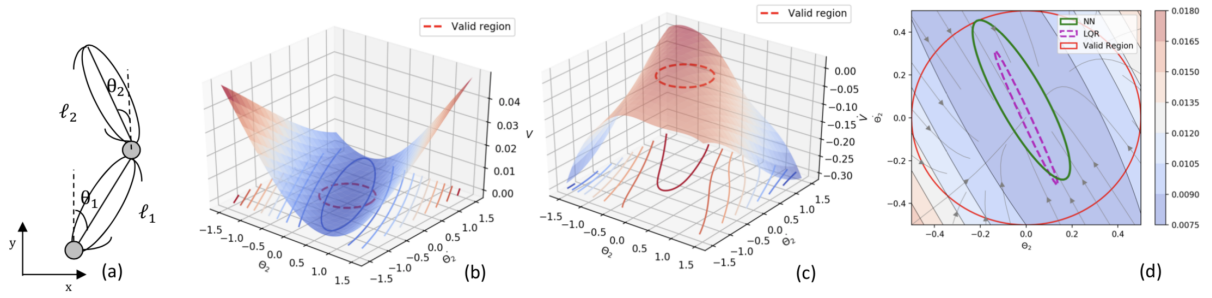

Figure 2.5. Results of n-link planar robot balancing. (a) Schematic diagram. (b) Learned Lyapunov function. (c) Lie derivative of Lyapunov function. (d) Comparison of the region of attraction.

2.5 Conclusion

We proposed new methods to learn control policies and neural network Lyapunov functions for highly nonlinear systems with provable guarantee of stability. The approach significantly simplifies the process of nonlinear control design, provides end-to-end provable correctness guarantee, and can obtain much larger regions of attraction compared to existing control methods. We show experiments on challenging nonlinear problems central to various nonlinear control problems. The proposed methods demonstrate clear advantage over existing methods. We envision that neural networks and deep learning will lead to better solutions to core problems in robot control design.

2.6 Acknowledgement

Chapter 2, in full, is a modified reprint of material previously published in Ya-Chien Chang, Nima Roohi, and Sicun Gao. “Neural Lyapunov Control.” Advances in Neural Information Processing Systems (NeurIPS), 2019. The dissertation author was the primary investigator and author of this material.

Chapter 3

Stabilizing Neural Control Using Self-Learned Almost Lyapunov Critics

The lack of stability guarantee restricts the practical use of learning-based methods in core control problems in robotics. We develop new methods for learning neural control policies and neural Lyapunov critic functions in the model-free reinforcement learning (RL) setting. We use sample-based approaches and the Almost Lyapunov function conditions to estimate the region of attraction and invariance properties through the learned Lyapunov critic functions. The methods enhance stability of neural controllers for various nonlinear systems including automobile and quadrotor control.

3.1 Introduction

Ensuring stability of neural control policies is critical for the practical use of learning-based methods for control design in robotics. There has been exciting progress towards introducing control-theoretic approaches for enhancing stability in reinforcement learning and imitation learning, such as using Lyapunov methods [24, 33, 34, 47–51], control barrier functions [52–54], or control-theoretic regularization [55–57]. However, three major questions are still open. First, while giving direct control-theoretic prior or guidance can improve performance, to what extent can a learning agent *self-supervise* to achieve certifiable stability? Second, stability requirements should be inherent to a system and not affected by the choice of reward functions, which often

do not focus on stability. Can we achieve stability without restricting the flexibility of reward engineering in reinforcement learning? Third, the typical formulation of stability in learning is in the form of reducing expected violation of certain constraints asymptotically, which does not easily translate to certifiable behaviors when the learned policies are practically deployed. Is it possible to obtain stronger stability claims in the standard control-theoretic sense, such as for estimating region of attraction and forward invariance properties? Our goal is to positively answer these questions to improve the reliability and usability of learning-based methods for practical control problems in robotics.

We propose new methods for incorporating Lyapunov methods in deep reinforcement learning. We design a model-free policy optimization process in which the agent attempts to formulate a Lyapunov-like function through *self-supervision*, in the form of a special critic function that is then used for improving stability of the learned policy, without using or being affected by the environment reward. Our work follows the framework of actor-critic Lyapunov methods recently proposed in [49] and extends it in the following ways. We allow the agent to self-learn the Lyapunov critic function by minimizing the Lyapunov risk [24] over its experience buffer without accessing the rewards. This Lyapunov critic function is represented as a generic feed-forward neural network, randomly initialized. This design differs from the typical choice of using a positive definite neural network to construct the Lyapunov candidate, to capture values learned from appropriately designed cost functions that only reflect stability objectives, as in [17, 49]. The lack of restriction and guidance turns out to be beneficial. In our approach, the learning agent can formulate Lyapunov landscapes that better enforce stability, compared to existing methods. In fact, this form of self-learned Lyapunov candidate can often be shown to satisfy the Almost Lyapunov conditions [58], which allows us to use sample-based analysis to estimate its region of attraction and forward invariance properties. We show that the new design is important for learning certifiably stable control policies for practical control problems such as automobile path-tracking and quadrotor control.

Our work builds on the recent progress of model-free and sample-based Lyapunov

methods [17, 49, 58, 59]. Such methods allow us to use sampled Lie derivatives of candidate Lyapunov functions to estimate stability properties of neural control policies without using analytic forms of the system dynamics. We exploit the expressiveness of neural networks for capturing Lyapunov landscapes that are too complex for conventional choices such as sum-of-squares polynomials. We believe the proposed methods further advance the promising direction of developing rigorous neural control and certification methods in reinforcement learning.

In all, we make the following contributions. We propose new methods for training neural Lyapunov critic functions (Section IV.A) and use it to improve stability properties of neural control policies (Section IV.B) in a model-free policy optimization setting. We show that the learned Lyapunov critic function can be analyzed through sample-based methods based on the Almost Lyapunov conditions, for estimating its region of attraction and invariance properties (Section IV.C). We demonstrate the benefits of the proposed methods in comparison with standard policy optimization and existing Lyapunov-based actor-critic methods (Section V).

3.2 Related Work

Besides the most closely related work [49] discussed above, our work is connected to various recent progress in safe reinforcement learning, Lyapunov methods in reinforcement learning, and data-driven methods for the analysis of control and dynamical systems. Safe reinforcement learning is now a large and active field [60, 61] focusing on learning with various forms of soft or hard safety constraints, often formulated as Constrained Markov Decision Processes (CMDP) [62]. Lyapunov methods have seen various applications in this context. It was first introduced in RL by the work of [47], which uses predefined controllers and Lyapunov-based methods for learning a switching policy with safety and performance guarantees. [33] proposed a model-based RL framework that uses Lyapunov functions to guide safe exploration and uses Gaussian Process models of the dynamics to obtain high-performance control policies with provable stability certificates. [34] developed methods for constructing Lyapunov function

in the tabular setting that can guarantee global safety of a behavior policy during training. The approach is extended to policy optimization for the continuous control setting in [48], showing benefits in various high-dimensional control tasks. The work in [50] formulates the state-action value function for safety costs as candidate Lyapunov functions and model its derivative with Gaussian Processes with statistical guarantees on the control performance. Similarly in [49], candidate Lyapunov functions are constructed from value functions with benefit in stabilizing the control performance. In the work of [17, 24], neural networks are used to learn Lyapunov functions for establishing certificates for stability and safety properties.

Many other control-theoretic methods have also been proposed to improve reinforcement learning [51–57]. These methods typically involve introducing strong control-theoretic priors, or use the neural policies as an oracle to extract low-variance policies in simpler hypothesis classes. For instance, the work in [52] uses control barrier functions to ensure safety of learned control policies. The work in [55] proves stability properties throughout learning by taking advantage of the robustness of control-theoretic priors. Our goal is to turn control-theoretic methods into general self-supervision methods for enhancing stability.

3.3 Preliminaries

Definition 8 (Dynamical Systems)**.** An n-dimensional controlled dynamical system is defined by

$$\dot{x}(t) = f(x(t), u(t)),\ u(t) = g(x(t)),\ x(0) = x_0, \tag{3.1}$$

where $f : D \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, $g : D \to \mathbb{R}^m$ is a control function, and $D \subseteq \mathbb{R}^n$ with $0, x_0 \in D$ defines the state space of the system. Each $x(t) \in D$ is called a state vector and $u(t) \in \mathbb{R}^m$ is a control vector.

Definition 9 (Stability)**.** We say that the system of (1) is stable at the origin if for any $\varepsilon \in \mathbb{R}^+$, there exists $\delta(\varepsilon) \in \mathbb{R}^+$ such that if $\|x(0)\| < \delta$ then $\|x(t)\| < \varepsilon$ for all $t \geq 0$. The system is asymptotically stable at the origin if it is stable and also $\lim_{t\to\infty} x(t) = 0$ for all $\|x(0)\| < \delta$.

Definition 10 (Lie Derivatives)**.** Consider the system in (4.1) and let $V : D \to \mathbb{R}$ be a continuously differentiable function. The Lie derivative of V over f is defined as

$$\mathsf{L}_{f_u} V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \dot{x}_i(t). \tag{3.2}$$

It measures the rate of change of V over time along the direction of the system dynamics of $x(t)$.

Definition 11 (Lyapunov Conditions for Asymptotic Stability)**.** Consider a controlled system (1) with an equilibrium at the origin, i.e., $\exists\ u \in \mathbb{R}^m$ s.t. $f(0,u) = 0$. Suppose there exists a continuously differentiable function $V : D \to \mathbb{R}$ satisfying $V(0) = 0$, and $\forall x \in D \setminus \{0\}, V(x) > 0$, and $\mathsf{L}_{f_u} V(x) < 0$. Then V is a Lyapunov function. The system f is asymptotically stable at the origin if such Lyapunov function V can be found.

The recent work [58] proposed an approximate notion of Lyapunov methods named *Almost Lyapunov functions*, which allows the Lyapunov conditions to be violated in restricted subsets of the space while still ensuring stability properties. It is the basis of our sample-based approach for validating learned Lyapunov candidates. We will discuss this notion and our approach in detail in Section IV.C.

We will use the standard notations for reinforcement learning in Markov Decision Processes (MDP) with state space S, action space A and transition model P: $S \times S \times A \to [0,1]$. A reward function defines the reward for taking action a in state s and transitioning into s'. Let π_ϕ denote a stochastic policy parameterized by ϕ. The goal of the learning agent is to maximize the expected γ-discounted cumulative return $J(\phi) = \mathbb{E}_{s_0,a_0,\ldots}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})]$. Policy optimization methods [63–66] estimate policy gradient and use stochastic gradient ascent to directly improve policy performance. A standard gradient estimator is

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\phi \log \pi_\phi(a_t|s_t) \hat{A}_t \right], \tag{3.3}$$

where π_ϕ is a stochastic policy and $\hat{A}_t$ estimates the advantage that represents the difference

between the Q value of an action compared with the expected value of a state, to indicate whether an action should be taken more frequently in the future. The gradient steps will move the distribution over actions in the right direction accordingly. The expectation $\hat{\mathbb{E}}_t[\ldots]$ is estimated by the empirical average over finite batch of samples. The proximal policy optimization algorithm (PPO) [67] applies clipping to the objective function to remove incentives for the policy to change dramatically, using:

$$J^{\text{CLIP}}(\phi) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\phi)\hat{A}_t, \text{clip}(r_t(\phi), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right], \tag{3.4}$$

where $r_t(\phi) = \pi_\phi(a_t|s_t)/\pi_{old}(a_t|s_t)$ and ε is a hyperparameter. The clipping ensures the gradient steps do not overshoot in the policy parameter space in each policy update.

3.4 Policy Optimization with Lyapunov Critics

We now describe how to use self-learned candidate Lyapunov functions to improve stability of neural control policies. We will name these candidate Lyapunov functions as *Lyapunov critics*, following [49]. We will first describe how to learn Lyapunov critics through sampled trajectories, and then how to integrate this critic values in advantage estimation for policy optimization. We describe how sampling-based certification of stability using Lyapunov critics, following the framework of Almost Lyapunov functions. The full procedure is as shown in Algorithm 1. The overall loop is close to standard PPO [67, 68] with only additional steps at Line 11-12, for learning the Lyapunov critic, and Line 17, for policy optimization guided by the new critic. We will explain these steps in the following sections.

3.4.1 Self-Learning Lyapunov Critics

We represent candidate Lyapunov functions using neural networks, draw samples of the system states, and use gradient descent to learn network parameters that maximize the satisfaction of the Lyapunov conditions in Definition 11. Following [24], such learning can be achieved by

Algorithm 2. Policy Optimization with Self-Learned Almost Lyapunov Critics (POLYC)

Initialize policy (actor) network π_ϕ and the reward value function network V_η^r
Initialize Lyapunov function network V_θ randomly
Initialize replay buffer B as empty set
for episodes $= 1, \ldots, K$ **do**
    for $t = 1, \ldots, T$ **do**
        Sample $a_t \sim \pi_\phi(a_t|s_t)$
        Sample $s_{t+1} \sim P(s_t + 1|s_t, a_t)$
        $B \leftarrow B \cup (s_t, a_t, r_t, s_{t+1})$
    end for
    Sample mini-batches of size N from B
    Compute Lyapunov risk $R_{f,N,\rho,\Delta t}(\theta)$ under $\pi_{\phi_{new}}$
    $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta R_{f,N,\rho,\Delta t}(\theta)$
    for each policy optimization step **do**
        Sample mini-batches of transitions from B
        $\delta_t \leftarrow r_t + \gamma V_\eta^r(s_{t+1}) - V_\eta^r(s_t)$ ▷ cf. [68]
        $\hat{A}(s_t, a_t) \leftarrow \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t+1}\delta_{T-1}$
        $\hat{A}_\beta^L \leftarrow \beta \min(0, -\mathsf{L}_{f_{\pi_\phi},\Delta t} V_\theta(s_t)) + (1-\beta)\hat{A}(s_t, a_t)$
        $r(\phi) \leftarrow \pi_{\phi_{new}}(a|s)/\pi_\phi(a|s)$
        $J^{\text{CLIP}}(\phi, \beta) \leftarrow \hat{\mathbb{E}}\left[\min\left(r(\phi)\hat{A}_\beta^L, \text{clip}(r(\phi), 1-\varepsilon, 1+\varepsilon)\hat{A}_\beta^L\right)\right]$
        $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi J^{\text{CLIP}}(\phi, \beta)$
        $\eta \leftarrow \eta - \alpha_\eta \nabla_\eta \hat{\mathbb{E}}[(V_\eta^r(s_t) - G(s_t))^2]$ ▷ $G(s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$
    end for
end for

minimizing the following loss function named as the Lyapunov risk:

Definition 12 (Empirical Lyapunov Risk [24])**.** Consider dynamical system f with domain D. Let $V_\theta : D \to \mathbb{R}$ be a continuously differentiable function parameterized by θ. The empirical Lyapunov risk of V_θ over f with a sampling distribution $\rho(D)$, written as $R_{f,N,\rho}(\theta)$, is defined as:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\max(-V_\theta(s_i), 0) + \max(0, \mathsf{L}_{f_u} V_\theta(s_i))\right) + V_\theta^2(0) \tag{3.5}$$

where $s_1, ..., s_N$ are states sampled according to the sampling distribution ρ. The empirical Lyapunov risk is nonnegative, and when V_θ is a true Lyapunov function for f, this empirical risk attains its global minimum $R_{f,N,\rho}(\theta) = 0$, regardless of the choice of the sample size N and distribution ρ.

In this original formulation, evaluating the Lyapunov risk requires the full knowledge of the system dynamics f for computing $\mathsf{L}_{f_u} V$. Our current setting is different. The learning agent does not know the system dynamics f and can only approximate the Lie derivative $\mathsf{L}_{f_u} V$ along sampled trajectories of the system, through finite differences:

$$\dot{V}(s) = \frac{1}{\Delta t}\Big(V(s') - V(s)\Big), \tag{3.6}$$

where s and s' are two consecutive states and Δt is the time difference between them and $\lim_{\Delta t \to 0} \dot{V}(s) = \mathsf{L}_{f_u} V(s)$. We define the corresponding discretized Lyapunov risk objective $R_{f,N,\rho,\Delta t}$ where the only change from (3.5) above is that the Lie derivative $\mathsf{L}_{f_u} V_\theta(s)$ is replaced by the sampled estimate $\dot{V}(s)$ for all states.

At each iteration of the policy update (the main loop from Line 4 to 23 in Algorithm 1), we collect trajectories of samples from the controlled system f_π, and perform stochastic gradient descent to minimize the discretized Lyapunov risk $R_{f,N,\rho,\Delta t}(\theta)$ to learn the Lyapunov critic function V_θ (Line 11-12). Note that the dependency of the dynamics f on the behavior policy π_ϕ is important. When the policy is updated, the controlled system changes its dynamics, and the candidate Lyapunov function should be learned correspondingly. Thus, we need to be able to sample from the buffer of previous trajectories, and estimate their Lie derivative values using the new policies (Line 10-11). This dependency makes our approach inherently on-policy in the current formulation, in the sense that the critic is always learned from the behavior policy π_ϕ and thus not very sample-efficient. Off-policy learning of the Lyapunov critic is possible through importance sampling and keeping track of the policy and Lyapunov critic updates, which is a promising direction that we leave open.

3.4.2 Stabilization via Policy Optimization

Once the Lyapunov critic is formulated, we use the learned candidate Lyapunov functions as an additional critic value for policy optimization. For a temporarily fixed candidate Lyapunov

function V_θ, the only term in the Lyapunov risk that is affected by policy change is the Lie derivative term V_θ. Thus, in policy optimization we now impose the additional goal of ensuring $V(\theta) < 0$ to encourage the policy update to improve the Lyapunov landscape for stabilization. We combine the standard advantage estimate $\hat{A}(s_t, a_t)$ [67] and a clipped Lie derivative term in each policy optimization step:

$$\hat{A}^L_\beta(s_t, a_t) = (1-\beta)\hat{A}(s_t, a_t) + \beta \min(0, -V(s_t)) \tag{3.7}$$

where $\beta \in [0,1]$ balances the weights. With policy gradient methods on this advantage estimate, the second term penalizes actions that produce a positive Lie derivative which makes $\min(0, -V(s_t)) < 0$. When the Lie derivative is negative, it does not bias the advantage estimate. We emphasize that although the Lie derivative term is dependent on π_ϕ, it does not have a functional form but only accessed through sampled values based on Equation (3.6). The true dynamics of the system remains unknown to the learner.

With the definition of the advantage with Lyapunov critic in (3.7), we can easily replace the standard advantage estimators in various on-policy algorithms. For instance, the standard policy gradient estimator becomes $\mathbb{E}_{\pi_\phi}[\nabla_\phi \log \pi_\phi(a_t|s_t)\hat{A}^t_\beta(s_t, a_t)]$. In Algorithm 1 we use the PPO version [67] of policy update (Line 19-20), which is what we use in the experiments. Optionally, we can update the β parameter as a Lagrange multiplier, by also taking gradient steps on β at some learning rate α using $\beta \leftarrow \beta - \alpha\mathbb{E}[V(s)]$, clipped between $[1,0]$. We have not observed much performance difference in doing so, and have excluded this step in Algorithm 1 for simplicity.

3.4.3 Validating Almost Lyapunov Conditions

A major benefit of the proposed approach is that the self-learned Lyapunov critic allows us to estimate the region of attraction of the controlled system when learning is successful. Since we do not have access to the dynamics of the system and can only estimate the Lyapunov risk

at sampled states, we can not expect to certify that the learned Lyapunov critic functions are true Lyapunov functions in the standard sense. However, recent progress in relaxed conditions for Lyapunov methods enabled the use of sample-based analysis to find region of stability. In particular, the Lyapunov critic functions can be analyzed through sampling using the Almost Lyapunov conditions [58] defined as follows:

Proposition 3 (Almost Lyapunov Conditions [58])**.** *Consider a dynamical system in (1) defined by f with domain $D \subseteq \mathbb{R}^n$ and a continuously differentiable positive definite function $V : D \to [0,\infty)$. Let $c_1, c_2 > 0$ be two constants and define B as the region between two sublevel sets $B = \{x \in D : c_1 \leq V(x) \leq c_2\}$ for V. Let $\Omega \subseteq B$ be a measurable set. Suppose for some $a > 0$, $\max_{x\in B} \mathsf{L}_{f_u} V(x) < a \min_{x\in B} V$, and $\forall x \in \Omega, \mathsf{L}_{f_u} V(x) \geq -aV(x)$ and $\forall x \in \Omega \setminus B, \mathsf{L}_{f_u} V(x) < -aV(x)$. Then there exists $\hat{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \hat{\varepsilon})$, if the volume of each connected component Ω^* of Ω satisfies $\mathsf{vol}(\Omega^*) \leq \varepsilon$, then there exists $T > 0$ such that for any $x_0 \in D$ with $V(x_0) < c_2 - r(\varepsilon)$, $x(t)$ stays within B for all $t > 0$ and moreover, it converges to the sublevel set $V(x) \leq c_1 + r(\varepsilon)$ for any $t > T$. Here $r(\varepsilon) = h\varepsilon^{1/n} + g\varepsilon$ for some constants h and g.*

The full proof is in [58] where all constants are explicitly constructed. Conceptually, the theorem relaxes the standard Lyapunov conditions to allow a set Ω that contains the violation states where the Lie derivative can be positive ($\forall x \in \Omega, \mathsf{L}_{f_u} V(x) \geq -aV(x)$). As long as each component of Ω is small enough (with volume less than ε), the violations do not affect stability. Under mild conditions, the appropriate region between sublevel sets of the system (written as B in the definition) defines a forward invariant set for all trajectories, and an approximate form of contraction where the trajectory converges to near the lower level set of the region B.

With the Almost Lyapunov conditions, we can use an ε-net over the space, chosen based on the Lipschitz constant of the Lie derivative, such that using sampled V value at the center of each cell we can identify the set Ω of states where the Lie derivatives violate the standard conditions. In Figure 3.1, we show the results of such computation to compare four different types of candidate Lyapunov functions for the inverted pendulum controlled by neural

network policies. Each plot visualizes the sign of the Lie derivative of the proposed Lyapunov candidate at uniformly sampled points over the $\theta/\dot{\theta}$ space. The grey dots represent cells where the candidate Lyapunov function has negative Lie derivative values ($\mathrm{L}_{f_u} V < -aV(x)$ as required in Proposition 3), and the red dots indicate cells that violate such condition. The value of the Lyapunov candidate itself can be shown to be always nonnegative in the domain in all four cases, and the black contours represent the level sets of increasingly positive values. The red dots indicate states where the standard Lyapunov conditions are violated, and the patterns are different.

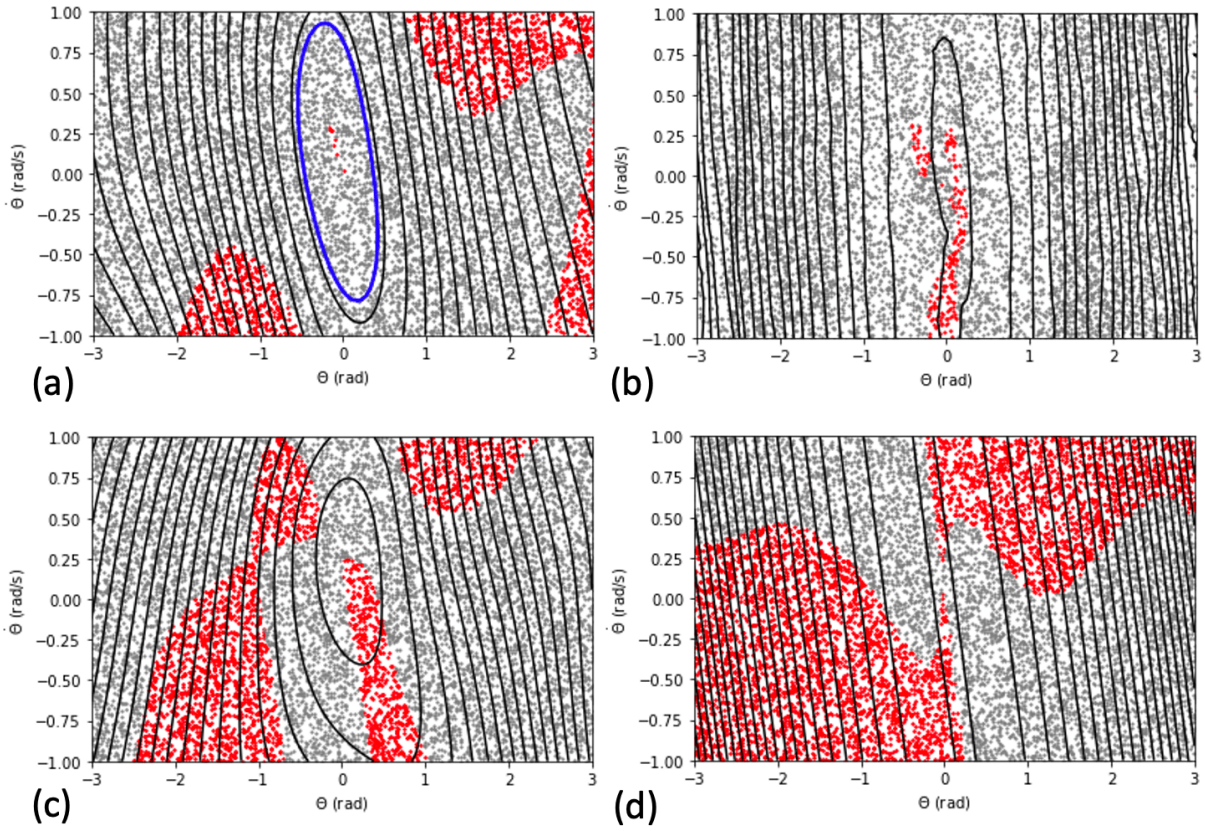

Figure 3.1. The landscapes of four different Lyapunov candidates for the inverted pendulum controlled by neural network policies.

• Figure 3.1(a) shows the self-learned Lyapunov critic obtained after learning the neural control policy using Algorithm 1. We see that we can find a sublevel set of the Lyapunov critic (inside the blue circle) where the Lie derivative is positive only at a very small number of sparse

regions (a few red dots near the center). This landscape satisfies the Almost Lyapunov conditions and the sublevel set defines forward invariant set with attraction (Proposition 3).

• Figure 3.1(b) shows the landscape generated by the Lyapunov actor-critic method (LAC) [49]. We see that the Lyapunov conditions are satisfied more globally, although with more violations closer to the origin. The function can also be established as an Almost Lyapunov function where the violation is sparse, which does not include the innermost sublevel set. The level sets is also much larger than those in (a), making it harder to find sublevel sets that are forward invariant. On the other hand, as shown later in the next section, the learned controller does always stabilize the system in all sampled trajectories. This indicates that there may exist better Lyapunov candidates for certifying the stability.

• Figure 3.1(c) shows the Lyapunov candidate fit for the control policy learned by standard PPO. We see that much more violation states are observed and it does not allow us to find a region where Almost Lyapunov Conditions can be validated. This indicates that the lack of Lyapunov critic makes it hard to enforce stability properties.

• Figure 3.1(d) uses the quadratic Lyapunov function for the linearized inverted pendulum obtained through LQR directly as a Lyapunov candidate for the neural controller trained by Algorithm 1, same as the one used in (a). We see that the simple Lyapunov function does not satisfy the Almost Lyapunov conditions and fails to capture the stability properties of the learned controller.

3.5 Experiments

We now show experimental results on the proposed methods for various nonlinear control problems. Our implementation follows Algorithm 1 (referred to as LY) and optionally uses an additional entropy term in the advantage estimates. We compare the performance with the closely related work of Lyapunov-based Actor-Critic (LAC) [49], and also standard implementations of Soft Actor-Critic (SAC) [69] and PPO [67]. We use the following control environments:

the inverted pendulum, quadrotor control, automobile path-tracking, and Mujoco Hopper and Walker.

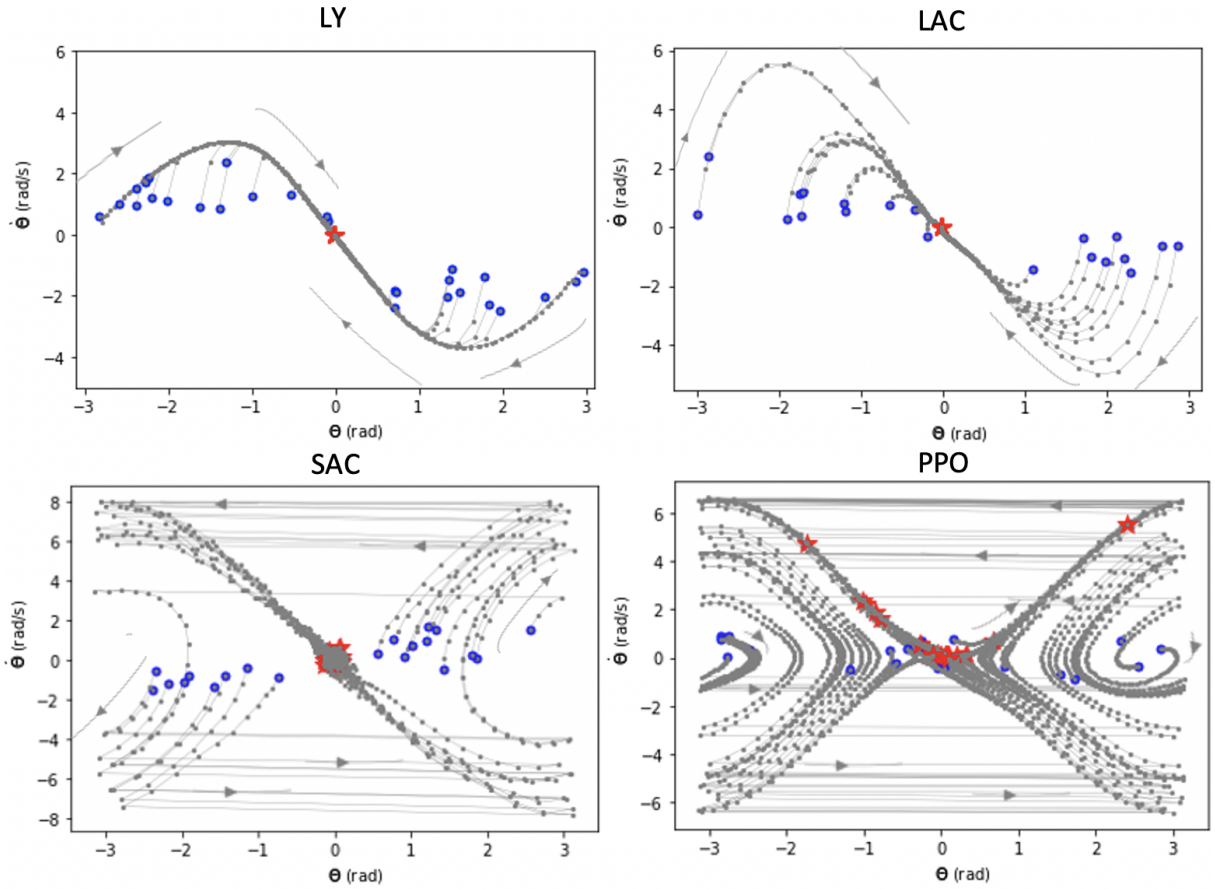

Figure 3.2. Sample trajectories generated by the policies learned with each algorithm in the inverted pendulum environment.

Inverted Pendulum. For the standard inverted pendulum (Pendulum-v0 in OpenAI Gym), Figure 3.2 shows the system trajectories under learned policies plotted in the θ-$\dot{\theta}$ space. The blue dots indicate initial positions and the red dots indicate positions after stabilization. Both LY and LAC learn stable controllers that reach the upright position without falling down, i.e., showing monotonically decreasing θ values. In contrast, the policies trained with SAC and PPO show horizontal lines that indicate falling down and crossing $-\pi$, by first swinging downwards and then back to the upright position. As discussed in the previous section in Fig 1, both LY and LAC learn Almost Lyapunov functions that are consistent with the stable control behaviors.

Quadrotor control. We learn neural controller for the 6-DOF quadrotor model, which has four control inputs and twelve state variables. The control inputs $\Omega_1, \Omega_2, \Omega_3$ and Ω_4 are the angular velocity of each rotor. The state variables are the inertia frame positions (x, y, z), velocities $(\dot{x}, \dot{y}, \dot{z})$, rotation angles (ϕ, θ, ψ) and angular velocities $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. The equations of motion of the quadrotor are given in [70] and Figure 3.3(a) shows the schematic. The control problem is known to be hard for policy gradient methods, typically requiring imitation learning steps. In Figure 3.4, we see that the LY method can learn stable tracking controller for the system. Moreover, the learned Lyapunov critic is shown in Fig 4(a) and its Lie derivatives is in Fig 4(b). Almost Lyapunov conditions can be validated with in the blue level set which certifies stable behavior shown in the trajectory. In comparison, LAC fails to learn a working controller.

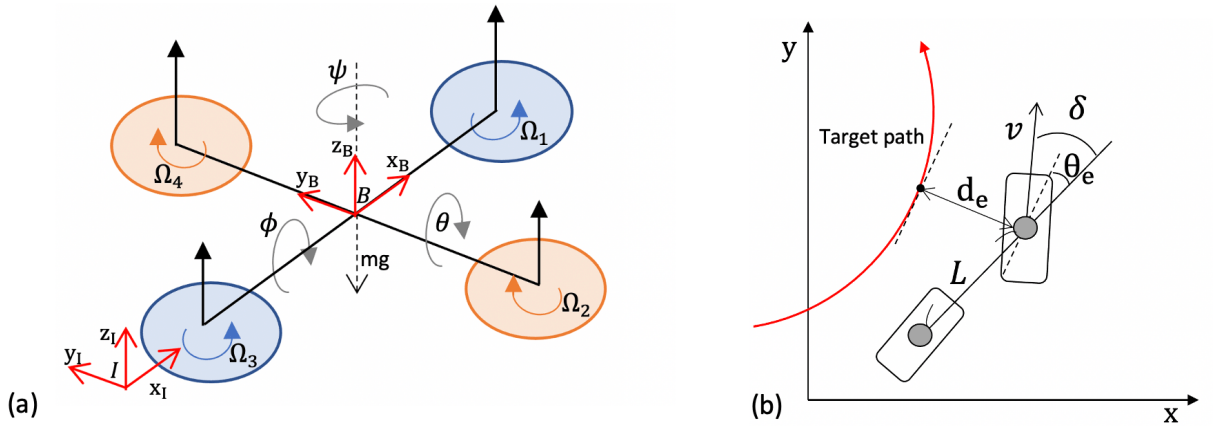

Figure 3.3. (a) Schematic of 6-DOF quadrotor system with body frame B and inertia frame I. (b) Schematic of wheeled vehicle.

Automobile path-tracking control. The control problem of following a target path and speed command is the core of autonomous driving [16]. In the training environment, the state has four dimensions including target velocity V_t, angular error $\theta_e(t)$, distance to the path $d_e(t)$, and the vehicle speed $v(t)$. The action space contains acceleration $a(t)$ and a steering control $\delta(t)$. The car dynamics follows standard bicycle model [16] and Figure 3.3(b) shows a schematic. In this experiment, we observe that all methods can obtain working control policies in the training environment in Figure 3.5(a). However, the Lyapunov landscape matters when applying the

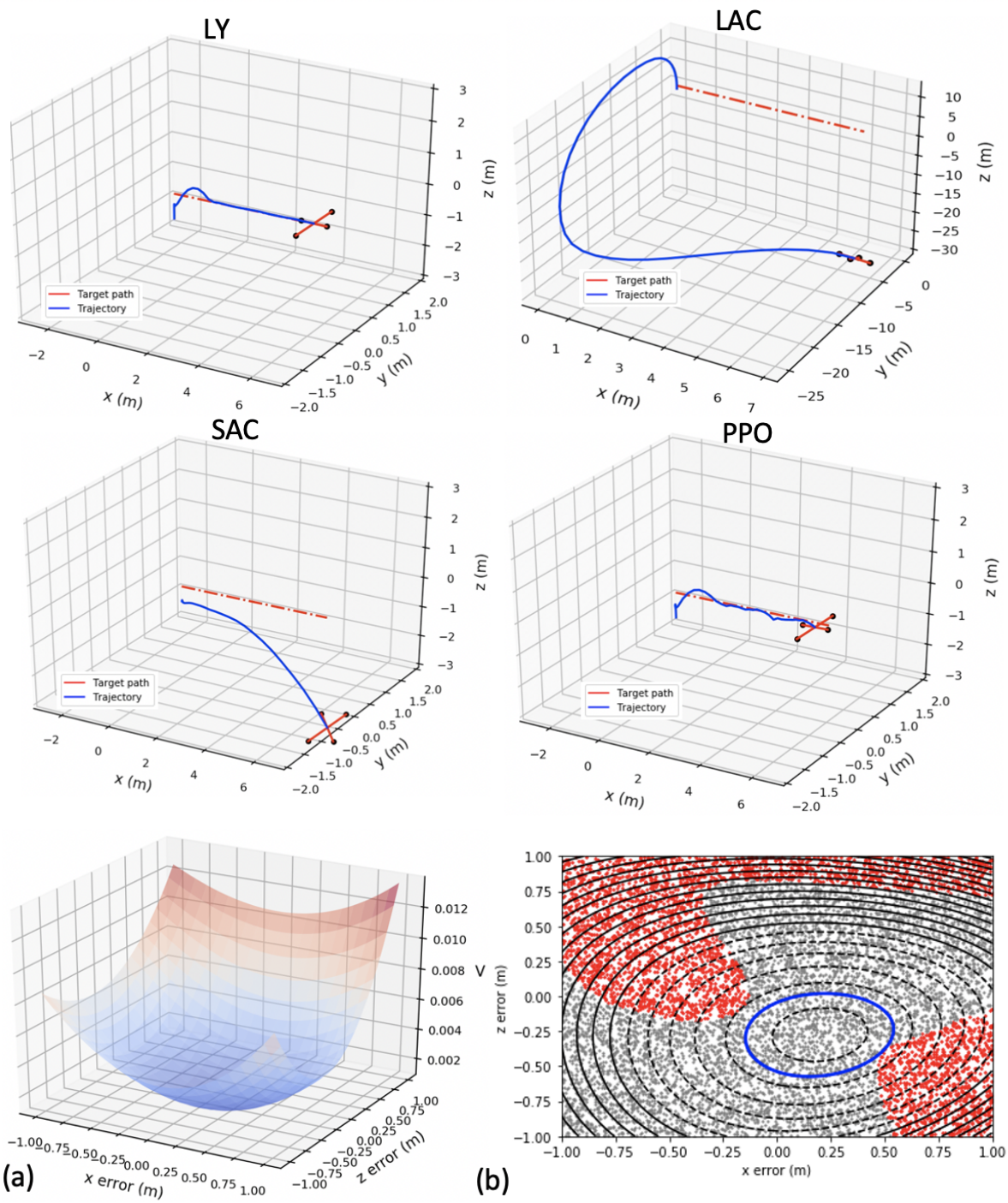

Figure 3.4. Quadrotor control for tracking a horizontal path. (a) shows the Lyapunov critic learned in LY and (b) shows where Almost Lyapunov conditions are validated (within the blue level set).

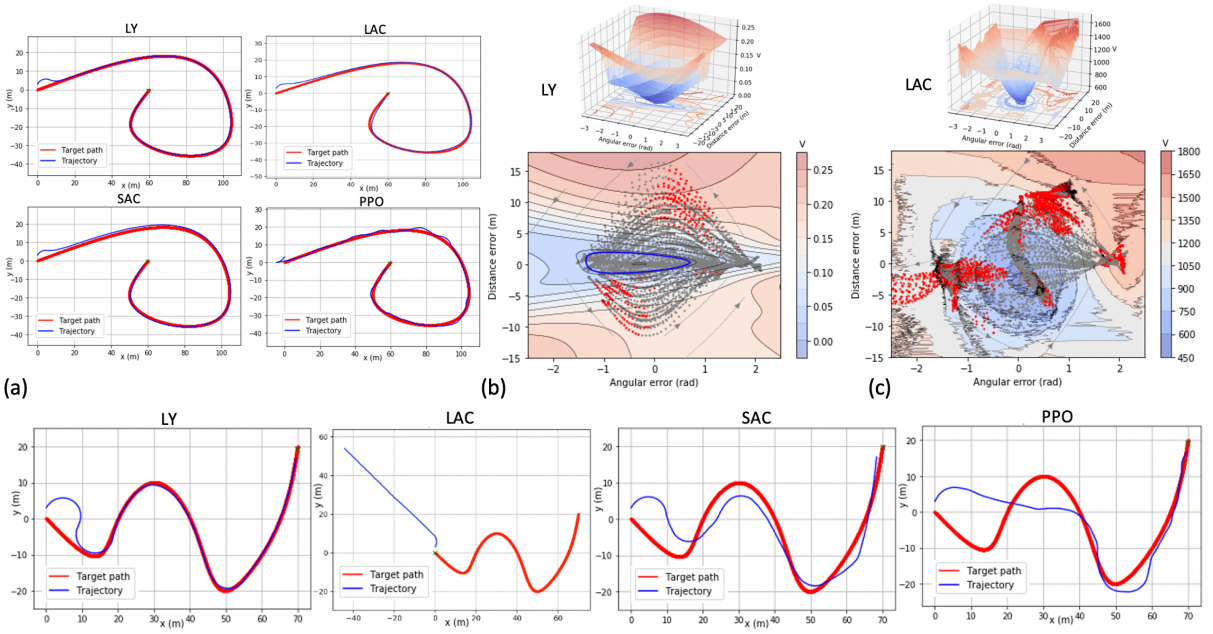

Figure 3.5. Experiments for automobile path tracking. (a) All methods learn to control well in the training environment. (b) 3D and 2D landscape generated by the learned Lyapunov function from LY. Almost Lyapunov conditions are validated within the blue level set. (c) Landscape generated by the learned Lyapunov critic from LAC. Second row: the control performance when tracking an unseen path (the red curve). The blue curves indicate the trajectory of the vehicle, starting from the left and going towards the right.

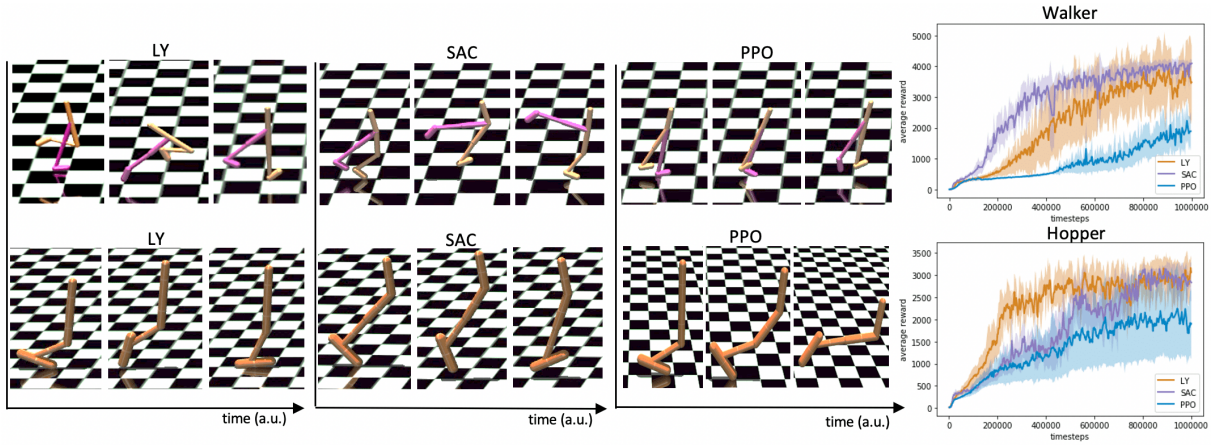

Figure 3.6. Control performance in the Walker and Hopper environments and learning curves over 5 random seeds. Our method typically learn faster than PPO, and comparable to SAC. All methods can achieve high rewards, but the learned control behaviors are different.

policy on a different path, as illustrated in the bottom row in Fig 3.5. LY methods generalizes well to unseen paths because of its region of attraction, validated via the Almost Lyapunov

conditions within the blue level set of the learned Lyapunov critic. In Figure 3.5(c), we see that the Lyapunov critic learned in LAC does not create a landscape that enforces a region of attraction in the sense of Almost Lyapunov conditions.

Mujoco Walker and Hopper. Walker and Hopper are standard high-dimensional locomotion environment. For both the goal is move forward as fast as possible and not stabilization. The LAC method requires using cost functions for stabilization objectives only, and thus does not work in these environments. We can learn Lyapunov critics that are independent from the reward, just focusing on stabilizing the joint angles to hold the upright positions. As shown in Fig 3.6, the LY controller maintains better pose and gaits compared to SAC and PPO. The learning curves show that LY does not slow down learning, and can be used in generic high-dimensional control tasks to improve performance.

3.6 Conclusion

We proposed new methods for training stable neural control policies using Lyapunov critic functions. We showed that the learned Lyapunov critics can be used to estimate regions of attraction for the controllers based on Almost Lyapunov conditions. We demonstrated the benefits of the proposed methods in various nonlinear control problems. Future work includes further improving sample complexity of the Lyapunov critic learning as well as the validation process for ensuring the Almost Lyapunov conditions.

3.7 Acknowledgement

Chapter 3, in full, is a modified reprint of material previously published in Ya-Chien Chang and Sicun Gao, "Stabilizing Neural Control Using Self-Learned Almost Lyapunov Critics," IEEE International Conference on Robotics and Automation (ICRA), 2021. The dissertation author was the primary investigator and author of this material.

Chapter 4

Learning Stabilization Control from Observations by Learning Lyapunov-like Proxy Models

The deployment of Reinforcement Learning to robotics applications faces the difficulty of reward engineering. Therefore, approaches have focused on creating reward functions by Learning from Observations (LfO) which is the task of learning policies from expert trajectories that only contain state sequences. We propose new methods for LfO for the important class of continuous control problems of learning to stabilize, by introducing intermediate proxy models acting as reward functions between the expert and the agent policy based on Lyapunov stability theory. Our LfO training process consists of two steps. The first step attempts to learn a Lyapunov-like landscape proxy model from expert state sequences without access to any kinematics model, and the second step uses the learned landscape model to guide in training the learner's policy. We formulate novel learning objectives for the two steps that are important for overall training success. We evaluate our methods in real automobile robot environments and other simulated stabilization control problems in model-free settings, like Quadrotor control and maintaining upright positions of Hopper in MuJoCo. We compare with state-of-the-art approaches and show the proposed methods can learn efficiently with less expert observations.

4.1 Introduction

Reward engineering remains a challenge in applying Reinforcement Learning to robotic control problems [71]. Manual design of suitable reward functions can be complicated, requiring extra sensors [72, 73] and may guide learner to acquire unintended behavior [25]. Consequently, imitation learning has become an indispensable approach that allows usage of expert demonstrations to replace reward functions, typically in the setting of learning from demonstrations (LfD) [74, 75], where the learner has access to several state-action pair sequences of desired behavior generated by the expert. However, actions may be difficult to retrieve (like with human experts) and there can be mismatch between expert and learner action spaces. Therefore, methods have been proposed for the problem of learning from observations (LfO) where access to expert behavior is limited to state sequences [76]. LfO can model many challenging forms of imitation such as learning from videos to acquire control skills. State-of-the-art methods for LfO typically adapt methods from LfD to the state observation setting. For instance, the Generative Adversarial Imitation from Observation algorithm (GAIfO) [76, 77] adapts Generative Adversarial Imitation Learning (GAIL) [78] by using Generative Adversarial Networks (GANs) [79] to learn policies that mimic expert state transition distributions. However, because of the absence of actions, models, and reward signals, LfO problems are often too challenging in the general continuous control settings [77]. Our goal is to develop efficient LfO methods for a specific but important class of continuous control problems: stabilization control, such as controlling a robot to maintain an upright standing pose, or controlling an autonomous car to track a given path. Stabilization is the basis of all advanced control problems [28]. Complex tasks can almost always be decomposed into a motion planning part and a stabilization control part, which is used for following the plans.

We propose a novel two-step LfO procedure for stabilization control to achieve efficient imitation learning, supported by control-theoretic principles. We exploit the framework of Lyapunov stability theory, which provides a general structure for stabilization in nonlinear control

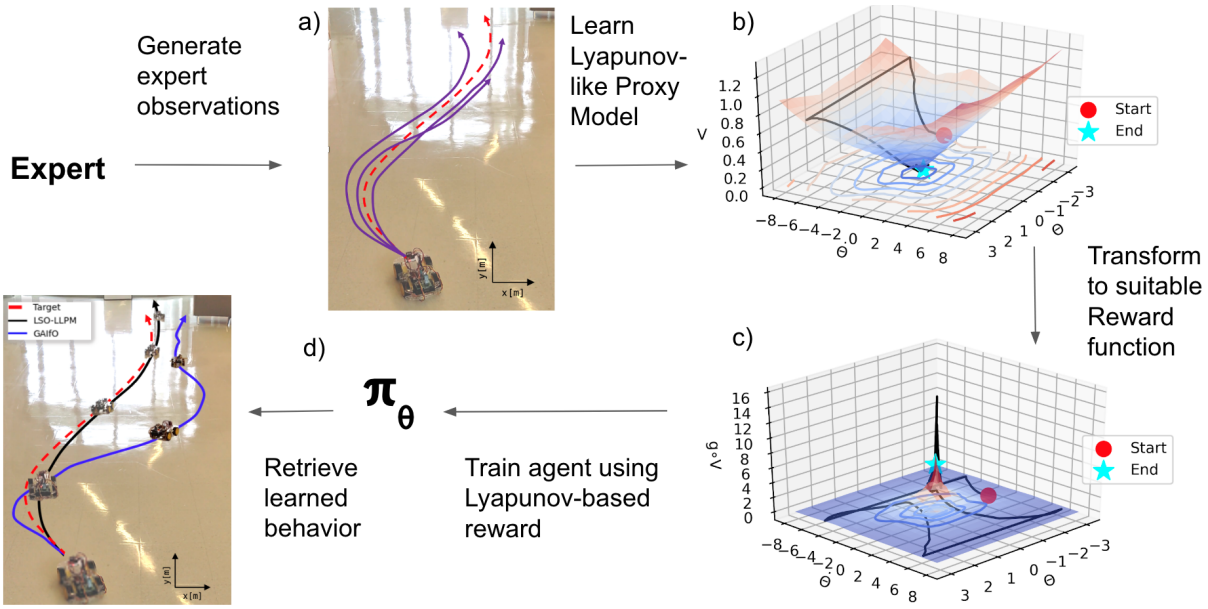

Figure 4.1. (a) We collect some expert state trajectories. (b) We learn a Lyapunov-like proxy model from the state trajectories. (c) We transform the model to a reward function. (d) We subsequently train the agent to learn a policy using the reward function. In the bottom left image, we show a trajectory from proposed approach compared with that of state-of-the-art GAIfO [77].

systems [28]. The core idea of Lyapunov methods is to construct an energy-like landscape that provides sufficient conditions for control systems to converge to and stabilize at their equilibrium points. In the LfO setting, we propose methods that learn Lyapunov-like functions as an intermediate goal, which we name as *Lyapunov-like proxy models*. From observations of expert state sequences (Figure 4.1 (a)), we first learn a neural network model that attempts to construct an approximate Lyapunov landscape to explain convergence of expert states (Figure 4.1 (b)). We transform this learned landscape model (Figure 4.1 (c)) to guide the training of the learner policy (Figure 4.1 (d)). For this procedure to succeed, it is critical for the proxy model to capture the geometry of the region of attraction and convergence rates (i.e. Lie derivatives of Lyapunov function) so they are consistent with expert behavior dynamics. We test our proposed procedure in real and simulated environments.

In the context of GAN-based approaches for LfO and LfD [78, 80–83], we can consider the Lyapunov-like proxy models as introducing a special hypothesis class for the discriminators,

as opposed to using generic distributions, to achieve more robust modeling of the expert behaviors for the specific context of stabilization problems.

We describe our technical approach in Section 4.4. In Section 4.5, we evaluate our approach in challenging environments for learning to stabilize from observations, including Acrobot, Quadrotor control, Automobile path-tracking, and stabilization version of MuJoCo Hopper robot. We evaluate our approach on real automobile robot environments in Section 4.6. Compared to state-of-the-art methods, the proposed methods can learn more efficiently from less observations of expert trajectories and produce more stable control policies.

4.2 Related Work

Imitation Learning. Learning from Demonstrations (LfD) problems have been the most well-studied form of imitation learning in MDPs. The learner has access to state-action trajectories of the expert without knowledge of the transition dynamics or the reward function in the MDP. Existing approaches in LfD generally fall into three categories: Behavioral Cloning [84, 85], Inverse Reinforcement Learning [86, 87], and Adversarial Imitation Learning [78, 80–83]. Our approaches are most related to the last category, where imitation learning is formulated in an adversarial framework of learning the policy as a generative model from the simultaneous training of a generator model and a discriminator model. The state-of-the-art algorithms are typically based on the method of Generative Adversarial Imitation Learning (GAIL) [78], which uses Generative Adversarial Networks (GANs) [79] to train a generative model that can create trajectories with a state-action occupancy measure similar to that of the expert, while the discriminator learns to provide feedback signals by differentiating the behavior distributions between the expert and the learner. Discriminator Actor-Critic (DAC) [88] is an off-policy version of GAIL that uses state-transition samples to train off-policy to achieve mode-covering in distribution matching. In general, LfD does not handle the most challenging forms of imitation, like learning from visual data of only expert state observations.

Learning from Observations (LfO). GAN-based approaches from LfD can be extended to the LfO setting since such methods can be used to only imitate state distributions without access to actions from the expert. The Generative Adversarial Imitation from Observation (GAIfO) algorithm [76, 77] uses a similar GAN framework as GAIL. Instead of training the discriminator using state-action pairs, GAIfO uses state transitions so that the imitator policy, which is the generator, produces a similar distribution of state transitions as expert. Another approach to LfO is Behavior Cloning from Observation (BCO) [89] where the agent acquires inverse dynamics experience in a self-supervised manner, which is then used to create a model to perform a task based on expert state observations. This approach does not require post-demonstration environment interactions, so it reduces the delay before the imitating agent is successful and avoids training and learning in risky environments (like autonomous vehicles). BCO nevertheless faces the issue of inaccuracies in that the learned inverse dynamics model accumulates error over time [90–92]. GAIfO has been shown to perform better than BCO by alleviating this compounding error issue [76, 77]. Off-policy learning methods have also been introduced to the GAN-based approaches for LfO. The works of Off-Policy Imitation Learning from Observations (OPOLO) [93] and Inverse Dynamics Disagreement Minimization (IDDM) [94] are able to accelerate the training of the learner/generator in the GAN framework by leveraging off-policy training of the inverse dynamics or action models of the environment before imitating from the expert. The works of Forward Adversarial Imitation Learning (FAIL) [95] and Imitating Latent Policies from Observation (ILPO) [96] also learn forward dynamics models and assume environment dynamics is deterministic with discrete action spaces. In contrast to this line of work, we focus on efficient learning in the setting of on-policy LfO without learning models or leveraging off-policy samples. The performance gain from off-policy methods can be used orthogonal to ours.

Lyapunov-based Methods in Reinforcement Learning and Imitation Learning. Lyapunov-based approaches have been recently introduced in model-free learning tasks [34, 48, 49, 97–101]. [34, 48] solves constrained MDPs with Lyapunov methods to constrain a policy search space

during each training iteration. They formulate a constraint value function as a Lyapunov function and update the policy with Lyapunov constraints. The work of [49] constructs candidate Lyapunov functions from value functions in an actor-critic framework, using Lyapunov decreasing condition as critic value to enhance stability properties of neural control policies. The work of [98] performs self-learning of almost-Lyapunov functions, used as a critic function to accelerate policy training.

A Lyapunov-based approach for LfD problems by [99] relies on Lyapunov theory and local quadratic constraints to establish safety and stability guarantees for deep neural network control systems. The methods assume that the environment is a linear dynamical system and do not consider more complex environments. Other Lyapunov-based methods in [101–103] learn globally asymptotic system dynamics (transition model) and then plan for trajectories toward the goal state by leveraging the prediction models, with error compounding issues similar to BCO [89–92].

4.3 Preliminaries

Markov Decision Processes and Learning from Observations. We consider imitation learning in Markov Decision Processes (MDPs) [104, 105]. MDPs are defined as $\langle \mathscr{S}, \mathscr{A}, P, r, \gamma \rangle$, where $\mathscr{S}$ is the state space, and $\mathscr{A}$ is the action space. $P(s_{t+1}|s_t, a_t)$ is the transition probability of reaching state s_{t+1} after action a_t is taken at state s_t, where t denotes a time-step but does not directly affect the transition probability. In the standard RL setting, the agent receives $r(s, a, s')$ where $r : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ is the reward function, and γ is the discount factor. In the imitation learning setting, the reward function is assumed unknown, and the goal is to train the agent to perform the task given expert observation trajectories, so we can write the MDP for imitation problems as $\langle \mathscr{S}, \mathscr{A}, P, C \rangle$ where C is the cost function that measures the deviation. In the LfO setting, there is no action information in the expert trajectories, so the cost function for the learner is defined as $C : \mathscr{S} \times \mathscr{S} \to \mathbb{R}$ which only assigns cost after comparing states between the

learner and the expert. The LfO agent attempts to learn a policy $\pi_\phi : \mathscr{S} \times \mathscr{A} \to [0,1]$ such that sampling from policy π_ϕ produces a distribution of state-action-cost tuples $\{(s_i, a_i, c_i)\}$ under environment dynamics. The goal of LfO is to minimize cumulative of the cost function C along trajectories generated by the agent's policy π_ϕ.

Stability and Lyapunov Methods. In stabilization problems, the agents control dynamical systems:

$$\dot{x}(t) = f(x(t), u(t)), u(t) = h(x(t)), x(0) = x_0, \tag{4.1}$$

where $x(t)$ takes values in an n-dimensional state space $X \subseteq \mathbb{R}^n$, $f : X \to \mathbb{R}^n$ is a Lipschitz-continuous vector field, $h : X \to \mathbb{R}^m$ is a control function. Each $x(t) \in X$ is called a state vector and $u(t) \in \mathbb{R}^m$ is a control vector. The notion of stability is then formally defined as:

Definition 13 (Lyapunov and Asymptotic Stability)**.** A system of Eq. 4.1 is *Lyapunov stable* at the origin $x = 0$, if for all $\varepsilon > 0$, there exists $\delta = \delta(\varepsilon) > 0$ such that for all $\|x(0)\| < \delta$, then $\|x(t)\| < \varepsilon$ for all $t \geq 0$. The system is *locally asymptotically stable* at the origin if it is Lyapunov stable and there exists $\delta > 0$ such that if $\|x(0)\| < \delta$, then $\lim_{t\to\infty} x(t) = 0$. $\|\cdot\|$ is typically the Euclidean norm.

Definition 14 (Lie Derivatives)**.** Consider the system in Eq. 4.1 and let $V : X \to \mathbb{R}$ be a continuously differentiable function. The *Lie derivative* of V over f is defined as

$$L_f V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \dot{x}_i(t). \tag{4.2}$$

The Lie derivative measures the change of V over time along the direction of the system dynamics.

Definition 15 (Lyapunov Conditions for Asymptotic Stability)**.** Consider a controlled system Eq. 4.1 with an equilibrium at the origin, i.e. $\exists u \in \mathbb{R}^m$ so $f(0,u) = 0$. Suppose there is a continuously differentiable function $V : X \to \mathbb{R}$ satisfying $V(0) = 0$; $\forall x \in X \setminus \{0\}, V(x) > 0$; and $L_f V(x) < 0$. Then V is a *Lyapunov function*. The system f is asymptotically stable at the origin if Lyapunov function V can be found.

We train neural networks to learn approximate Lyapunov landscapes based on the expert state observations by enforcing Lyapunov conditions to be satisfied along expert trajectories. Because the procedure is learning-based and cannot guarantee the Lyapunov conditions throughout the entire state space, we use the name *Lyapunov-like proxy model* for our LfO setting.

GAN-based Approaches in Imitation Learning. Adversarial Imitation Learning approaches [78, 80–83] rely largely on the usage of Generative Adversarial Networks (GANs) [79]. The GAN architecture pits two neural networks against each other in order to make one neural network produce data distributions similar to that of the training data. The two neural networks are called the Generator (G) and Discriminator (D) respectively. G attempts to fool D by making its output distribution p_g similar to training data distribution p_{data} given data from a prior on input noise variables p_z. D is trained to maximize the probability of assigning the correct label to both training data examples and samples from G, while G is concurrently trained to minimize $\log(1-D(G(z)))$. The minimax two-player game can thus be formulated with value function $W(D,G)$ [79]:

$$\min_G \max_D W(D,G) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z)))] \tag{4.3}$$

In GAIL [78], p_{data} is the expert trajectory of state-action pairs, and G is the imitation policy π_ϕ that needs to be trained and from which state-action pairs are sampled. The goal is to make the learner's state-action distribution be close to that of the expert. For LfO problems, GAIfO [77] uses similar paradigm, replacing state-action pairs with state transitions.

4.4 Learning Stabilization Control

At a high-level, our algorithm comprises two steps. We first train a neural Lyapunov-like proxy model using the expert trajectories. This step of training uses violation of the Lyapunov conditions as the loss function, adapting the conditions for asymptotic stability from Definition 15 to the LfO setting. The learned Lyapunov function subsequently provides the reward signal

for training the learner's agent. For this second step of training to succeed, we need to define a transformation of the Lyapunov-like proxy model from the first step using convex functions. The overall algorithm called LSO-LLPM, short for Learning Stabilization Control from Observations through Learning Lyapunov-like proxy Models, is shown in Algorithm 3. Line $1-3$ is the first step of training the Lyapunov-like proxy model (Section 4.4.1), and Line $4-7$ is the second step of using Proximal Policy Optimization (PPO) [67] to optimize the learner's policy given a reward function derived from the Lyapunov-like proxy model (Section 4.4.2).

Algorithm 3. LSO-LLPM

Require: Expert state-only trajectories: $\tau_E = \{(s, s')\}$, randomly initialized policy network π_ϕ, randomly initialized Lyapunov-like proxy model V_θ, and hyperparameters $c, \beta_1, \beta_2 \in \mathbb{R}^+$.

1: **for each** $(s, s') \in \tau_E$ **do**

2: Update V_θ by taking gradient descent steps on the following loss function:

$$V_\theta^2(0) + \max(0, -V_\theta(s)) + \beta_1 (c + (V_\theta(s') - V_\theta(s))/\Delta t)^2$$

3: **end for**

4: **for** $i = 0, 1, 2, \ldots$ **do**

5: Sample trajectories $\tau_i \sim \pi_\phi$

6: Update π_ϕ by performing PPO update steps with the following reward function:

$$g(V_\theta(s)) + \beta_2 \min(0, (V_\theta(s) - V_\theta(s'))/\Delta t)$$

7: **end for**

4.4.1 Learning Neural Lyapunov-like proxy Models

In the first step, we learn Lyapunov-like proxy models to capture the stabilization behavior of the expert. The shape of the region of attraction [28] and the convergence rate of this approximate Lyapunov landscape are important because we need to avoid misleading the learner in the second training step to pursue behaviors that cannot be achieved because of constraints in the underlying dynamics. For instance, when controlling a vehicle to track a given path, if we define an aggressive Lyapunov landscape to aim for a fast convergence rate, then the vehicle may be forced to attain a high speed and fail to stabilize. On the other hand, if the Lyapunov

landscape is too smooth, then the learner may attempt to take conservative actions that are not sufficient to drive the system to the target equilibrium.

We represent the Lyapunov-like proxy model over state space, $V_\theta : S \to \mathbb{R}$, as a neural network (typically two-layer fully connected networks are sufficient in our examples), which is randomly initialized. We sample observations from the expert trajectories and perform stochastic gradient descent on the parameters to minimize the violations of Lyapunov conditions using the following loss function:

$$V_\theta(0)^2 + \max(0, -V_\theta(s)) + \beta_1(c + L_f V_\theta(s))^2 \tag{4.4}$$

Recall that $L_f V_\theta$ is the Lie derivative of V_θ along the system dynamics f that we do not have knowledge of. Instead, we approximate the Lie derivative by the finite difference of the Lyapunov function over each discrete time step, $L_f V_\theta = (V_\theta(s') - V_\theta(s))/\Delta t$, where s and s' are consecutive state observations in the trajectory. We know that this is a good approximation when Δt is small.

The first term $V_\theta(0)^2$ ensures that the equilibrium point corresponds to a Lyapunov value of zero, which is the lowest across the state space because the second term $\max(0, -V_\theta(s))$ requires that the Lyapunov function value be non-negative at all sampled points, which ideally generalizes to other regions in the state space. The third term $\beta_1(c + L_f V_\theta(s))^2$ is a critical design. It controls the convergence rate as measured by the Lie derivative $L_f V_\theta$. Here we deviate from the standard Lyapunov conditions of only requiring the Lie derivative to be negative by forcing it to take the value of some positive constant rate c. With this requirement, the overall landscape V_θ will be shaped through learning so each step taken by the expert will be considered as a unit step toward stabilization. In this way, we capture the convergence rate by the Lyapunov-like proxy model which can then properly guide the learner in the second training step. We stop gradients for $V_\theta(s')$ in calculating $L_f V_\theta$.

4.4.2 Policy Learning from the Lyapunov-like proxy Model

After learning the Lyapunov-like proxy model, we transform it into a reward function that the learner can maximize using standard policy optimization procedures such as PPO [67]. The reward is defined as:

$$g(V_\theta(s)) + \beta_2 \min(0, (V_\theta(s) - V_\theta(s'))/\Delta t) \tag{4.5}$$

where $g(x)$ is a convex function (fixed through all environments) for scaling the values of the Lyapunov-like proxy models so that the learner can receive sufficiently strong reward feedback in each step. We also want to maintain stability when the agent is already close to the target equilibrium point. Suitable choices of $g(x)$ include $-\log(x)$ and $-\log(1-e^{-kx^2})$ for some $k > 0$. The second term of Eq. 4.5 reduces the reward when the Lie derivative becomes positive, which prevents the learner from taking large steps near the equilibrium state. In this manner, we observe fast converge of the agent to the equilibrium state. Overall, the Lyapunov candidate function acts as a proxy for the reward function so that PPO will take steps to increase the reward and attempt to reproduce stabilization control policies that converge at a similar rate as the trajectories from the expert.

4.5 Experiments in Simulated Environments

We evaluate our algorithm LSO-LLPM for nonlinear control problems and compare with the state-of-the-art algorithm GAIfO. For fair comparison, we implement GAIfO using PPO without entropy loss. Evaluation environments include Automobile path-tracking control [106] and Acrobot [107], as well as high dimensional tasks like Quadrotor [70], and Hopper Standing, and Walker Standing. Acrobot is a classical control task simulated within OpenAI Gym [108], Automobile path-tracking control and Quadrotor were simulated using PythonRobotics [109], and Hopper and Walker Standing environments were simulated with MuJoCo [110]. We used

NNs with $2-3$ hidden layers with $64-2048$ neurons.

Baselines and Evaluation Metrics. We focus on comparing with the state-of-the-art on-policy LfO approach GAIfO [77] because of the same assumptions of having access to observations only and on-policy training of the learner without modeling the environment transitions. In particular, GAIfO has been shown to perform consistently better than BCO [89]. There are a range of other imitation learning baselines with different additional assumptions, like LfD methods with access to actions (GAIL [78], DAC [88]), and off-policy LfO methods (OPOLO [93]) that assume ability to pre-train inverse transition models to accelerate learning progress. This performance gain is orthogonal to on-policy learning objective in our setting. Consequently, we focus on comparing with GAIfO in the evaluation and analysis.

We evaluate the performance along the following metrics: First, we measure the overall performance of the learner with respect to a varying number of sampled expert trajectories that are provided to both algorithms. Second, we analyze the learning efficiency by the learning curves over time.

Environments. The nonlinear control tasks in each environment are specified as follows:

- *Acrobot:* The Acrobot environment consists of two links and two joints. In the initial state, both links are hanging down. The goal is to swing the links up so the tip of the link farthest from the pivot reaches the threshold in the shortest time. The state consists of information on the angles from the joints as well as their angular velocity, and the agent can actuate the joint between the links.

- *Automobile path-tracking control:* Autonomous driving is a control problem in which using speed commands the agent needs to follow a target path. In the environment, the state space is four dimensions, namely the difference between target speed and vehicle speed $V_t - v(t)$, the angular error $\theta_e(t)$, the distance to the path $d_e(t)$, and vehicle speed $v(t)$. The action space has two dimensions, namely the acceleration $a(t)$ and a steering control $\delta(t)$.

- *Quadrotor control:* We also test our algorithm in the 6-degree-of-freedom Quadrotor model. This has four control inputs and twelve state variables. The control inputs Ω_1, Ω_2, Ω_3, and

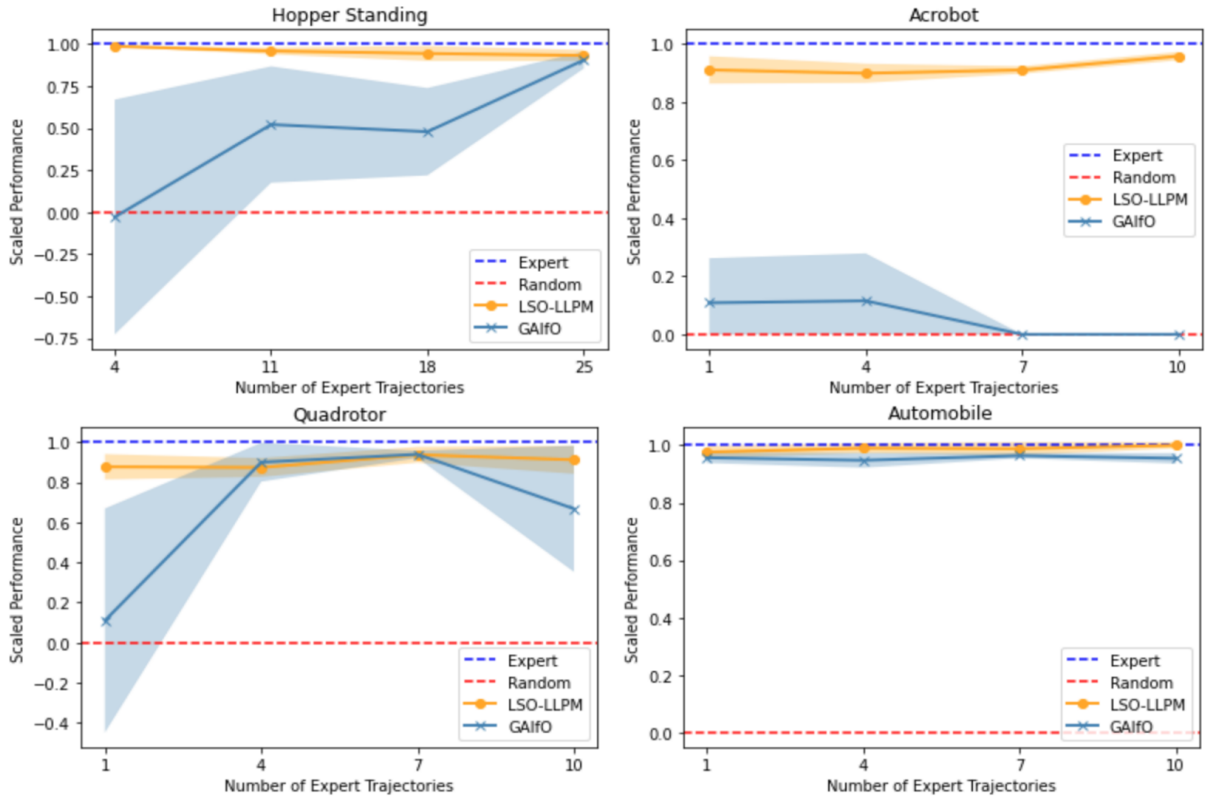

Figure 4.2. Performance of learned policies with varying numbers of expert trajectories. The performance is normalized to be between 0 (average reward of random policy) and 1 (average reward of expert policy). The shaded areas show variance over 5 random seeds. We observe the proposed methods perform much better especially in environments that are harder to control, like Hopper Standing, Acrobot, and Quadrotor.

Ω_4 are the angular velocities of each rotor. The state variables are the inertia frame positions (x, y, z), velocities $(\dot{x}, \dot{y}, \dot{z})$, rotation angles (ϕ, θ, ψ), and angular velocities $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. More details regarding the implementation and dynamics of the Quadrotor can be found in [70, 109]. This control problem is known to be hard for policy gradient methods but is solved with learning from demonstrations.

- *Hopper and Walker Standing:* We use MuJoCo Hopper and Walker and change task reward to formulate the stabilization control problem of standing in upright position. In Hopper Standing, there is one leg with 3 joints. In Walker Standing, there are two legs with 6 joints. In both environments, the agent's goal is to maintain standing state without falling down (losing balance)

for the longest time. We show graph results for Hopper Standing as the results are similar.

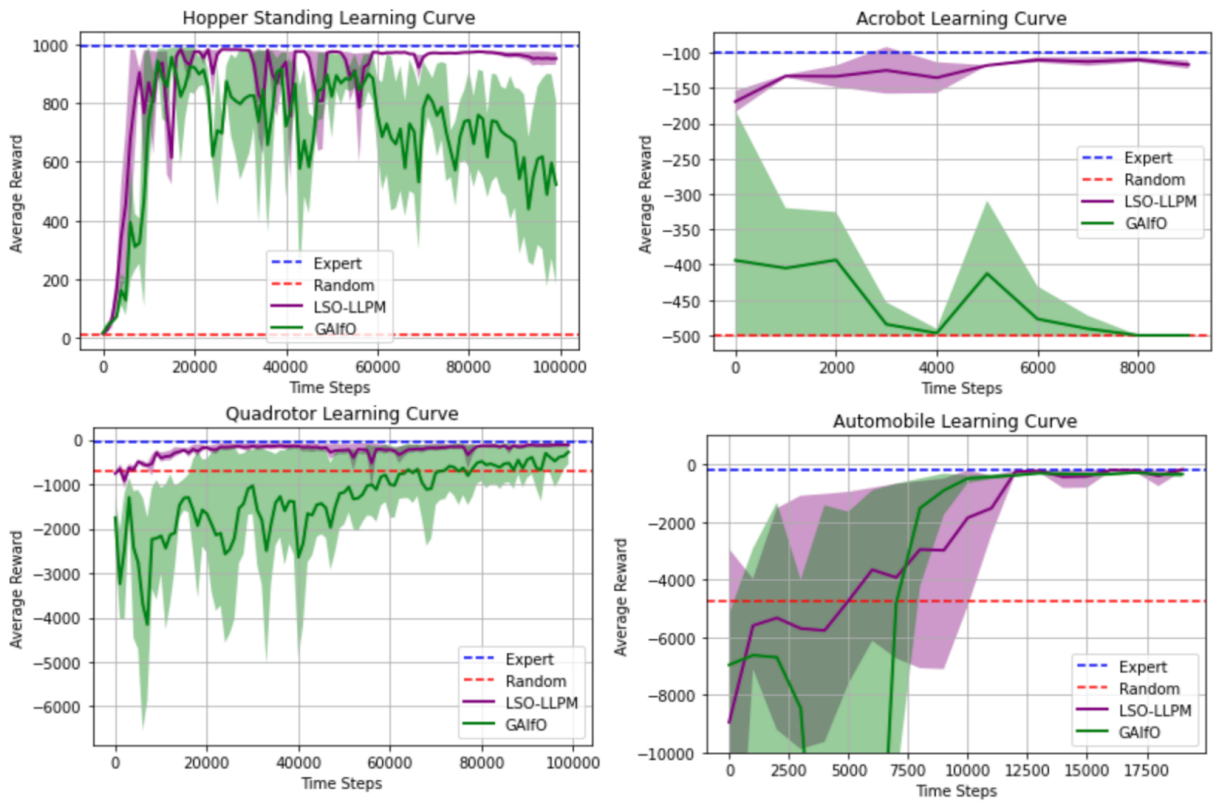

Figure 4.3. Comparing learning curves using $10-18$ expert trajectories (fixed number across different methods in each environment) over 5 random seeds. Environments in first row show distribution matching approaches in GAN-based methods often experience difficulty in making consistent learning progress while Lyapunov-like proxy models generate landscapes that are suitable for stabilization tasks and thus achieve stable learning performance.

4.5.1 Overall Performance

Results: LfO Performance. We examine the performance of the policies trained by LSO-LLPM and GAIfO with different number of expert trajectories in Figure 4.2. LSO-LLPM reaches at least 85% of the expert performance in all environments for all number of expert trajectories tested. As shown in the figure, LSO-LLPM consistently performs better than GAIfO for all environments and number of expert trajectories. In particular, in Hopper Standing and Acrobot, LSO-LLPM is able to perform well, even when GAIfO performs not much better than the random policy baseline. This difference illustrates the important use of Lyapunov-like proxy models

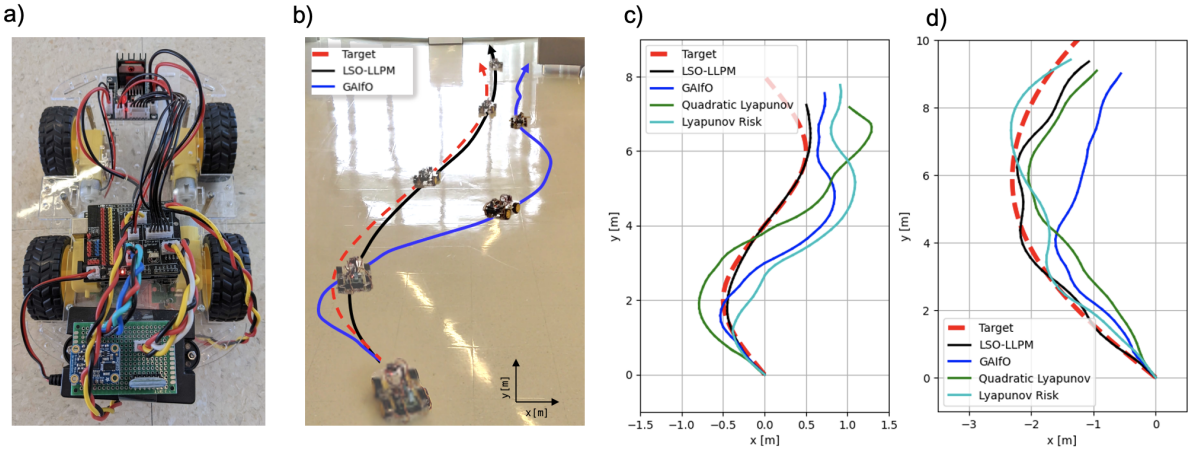

Figure 4.4. (a) Car robot used for the hardware experiments. (b) Tracking target path I. The car controlled by LSO-LLPM (black line) tracked the target path (red dash line) better than GAIfO (blue line). (c), (d) All trajectories driven by LSO-LLPM, GAIfO, Quadratic Lyapunov, and Lyapunov Risk. Our proposed method tracked both target paths fairly well compared with the other methods. (c) shows the result of tracking target path I and (d) shows the result of tracking target path II.

that capture challenging control problems in these environments due to underlying nonlinear dynamics. In these cases, GAN-based approaches performing distribution matching are often not sufficient for finding good control policies for stabilization.

Results: Learning Efficiency. The learning curves of LSO-LLPM and GAIfO in each environment are shown in Figure 4.3. In each environment, we use the number of expert trajectories (ranging between 10 to 18) that corresponds to the high performance cases in Figure 4.2. We see that across all environments, the LSO-LLPM methods can generally achieve fast learning progress with small variance across different random seeds. In comparison, GAIfO can learn well in simple control tasks such as simulated Automobile, but struggles to make progress in harder environments such as Hopper Standing and Acrobot. Note that this difficulty is not only determined by the dimensionality of the action space but also by how difficult it is to find control policies that can allow the learner to imitate the expert trajectories, and the LfO setting further increases that difficulty. For instance, although Acrobot is low-dimensional, the stabilization control problem is still very challenging. Overall, we observe Lyapunov-like proxy models can generate landscapes that are particularly suitable for stabilization tasks for efficient training of

the learner in hard control tasks.

4.6 Experiments in Hardware Environments

We present the evaluation of performance of automobile path-tracking control driven by LSO-LLPM, GAIfO, Quadratic Lyapunov, and Lyapunov Risk.

Quadratic Lyapunov In Lyapunov stability theory, a typical choice is the quadratic form (sum of squared differences between state features and their ideal locations/orientations) as default Lyapunov landscape. This is provably optimal for linear systems and often works well for mildly nonlinear systems. We consider whether a quadratic model can guide the learner without learning a Lyapunov-like proxy model.

Lyapunov Risk We also consider learning the Lyapunov Function with the generic Lyapunov Risk Loss as proposed in [24] instead of LSO-LLPM's loss expression in Eq. 4.4. This would demonstrate the importance of enforcing constant Lie derivative on expert trajectories in guiding the learner. **Hardware settings** We tested the control task using the car robot shown in Fig. 4.4 (a). One IMU sensor and two photoelectric encoders are on it to calculate its velocity $v(t)$ and orientation $\theta(t)$ each time step. With these values, we obtain the difference between target speed and vehicle speed $V_t - v(t)$, the angular error $\theta_e(t)$, and the distance to the path $d_e(t)$, which are used as each policy's input. Given these input values, the policies determine acceleration $a(t)$ and steering control $\delta(t)$ at the next time step. In this hardware experiment, we set $\Delta t = 0.5$ [s], target speed $V_t = 0.3$ [m/s]. The policies are trained under a simulator with these parameter settings, and they are directly deployed to the robot car for testing.

Results We tested the performance of automobile path-tracking with two different target paths, Target path I and II. Fig. 4.4 (b) and (c) are the results of tracking target path I, and Fig. 4.4 (d) shows the result of tracking target path II. The red dot lines are the target paths given in advance, and the closer the trajectories are to them, the better their control policies. As seen from these figures, the car driven by LSO-LLPM (black lines) tracked the target paths fairly well compared

with the other methods. In particular, GAIfO (blue lines) showed low path tracking performance as the trajectories controlled by it gradually deviated from the target paths. The main strength of our approach is obtaining stable control policies through LfO training process by utilizing Lyapunov-like proxy models. This enables more stable tracking performances even for the hardware experiments, in which are various disturbances such as sensor noises and discrepancies between simulator and real hardware modeling.

4.7 Discussion and Conclusion

We have introduced a novel model-free Lyapunov-based method to accelerate Learning from Observations for stabilization control problems by introducing intermediate proxy models between the expert and the agent policy based on Lyapunov stability theory. Our LfO training process first learns a Lyapunov landscape model from the expert state sequences and then transforms the learned model to guide the training of the learner's policy. We showed the proposed methods can capture stabilization control behaviors that take into account underlying dynamics so the learner's agent can successfully recover stable control policies through policy optimization. We evaluated the proposed methods in various real and simulated nonlinear stabilization control environments and observed better learning efficiency compared to the state-of-the-art GAN-based approaches.

We primarily focused on stabilization to a fixed equilibrium. To handle more general control tasks, it is possible to extend our approach by incorporating time in the Lyapunov-like proxy models. Such extension may work on more control environments with general locomotion tasks.

4.8 Acknowledgement

Chapter 4, in full, is a modified reprint of material previously published in 3. Milan Ganai, Chiaki Hirayama, Ya-Chien Chang, and Sicun Gao, "Learning Stabilization Control from

Observations by Learning Lyapunov-like Proxy Models," IEEE International Conference on Robotics and Automation (ICRA), 2023. The material in this chapter represents collaborative work.

Chapter 5

Extremum-Seeking Action Selection for Accelerating Policy Optimization

Reinforcement learning for control over continuous spaces typically uses high-entropy stochastic policies, such as Gaussian distributions, for local exploration and estimating policy gradient to optimize performance. Many robotic control problems deal with complex unstable dynamics, where applying actions that are off the feasible control manifolds can quickly lead to undesirable divergence. In such cases, most samples taken from the ambient action space generate low-value trajectories that hardly contribute to policy improvement, resulting in slow or failed learning. We propose to improve action selection in this model-free RL setting by introducing additional adaptive control steps based on Extremum-Seeking Control (ESC). On each action sampled from stochastic policies, we apply sinusoidal perturbations and query for estimated Q-values as the response signal. Based on ESC, we then dynamically improve the sampled actions to be closer to nearby optima before applying them to the environment. Our methods can be easily added in standard policy optimization to improve learning efficiency, which we demonstrate in various control learning environments.

5.1 Introduction

Deep reinforcement learning offers a promising solution for challenging control problems in robotics [98, 111–117]. It turns controller synthesis into stochastic optimization problems in

the parameter space of expressive control policies. In such policy optimization process, each vector of parameters defines a stochastic policy, from which we sample trajectories to estimate the policy gradient direction over the parameters, for local improvement of the policy towards higher overall performance. A wide range of techniques have been developed to ensure reliable gradient estimation and policy improvement [65, 67, 69, 118, 119]. However, it is still often observed that in many control problems, policy optimization can fail to make progress towards desirable performance [120–122].

A common design in policy optimization over continuous spaces that has not been challenged much is that the policies map states to Gaussian distributions over the action spaces. Sampling using such high-entropy stochastic policies enables exploration of actions, which is generally important in RL. However, many robotic control problems deal with complex unstable dynamics, where applying actions that are off the feasible control manifolds can quickly lead to undesirable divergence. Consequently, in such cases, most samples taken from the ambient action space generate low-value trajectories that hardly contribute to policy improvement, resulting in slow or failed learning. For instance, consider the problem of controlling the thrusts of a quadrotor for path tracking. Successful control requires symmetry in the thrusts of the four propellers. Exploration of actions under a Gaussian distribution over the four actions will most likely violate such constraints, and the quadrotor can quickly lose balance and fall off, making it hard to progress in learning. Such problems have led to much pre-processing and manual tuning in the practical use of RL [120–124], such as normalization of states and observations, reward engineering, and customized design of state and action spaces such that the feasible control becomes easier to sample with Gaussian distributions.

In this paper, we propose a general model-free approach for improving the quality of exploratory action samples for accelerating policy optimization. We utilize the methods of Extremum-Seeking Control (ESC), an adaptive feedback control strategy that performs real-time optimization of system performance [125–127]. ESC injects periodic perturbation signals in the control input to a system, and formulates feedback control laws to maintain dynamic estimations

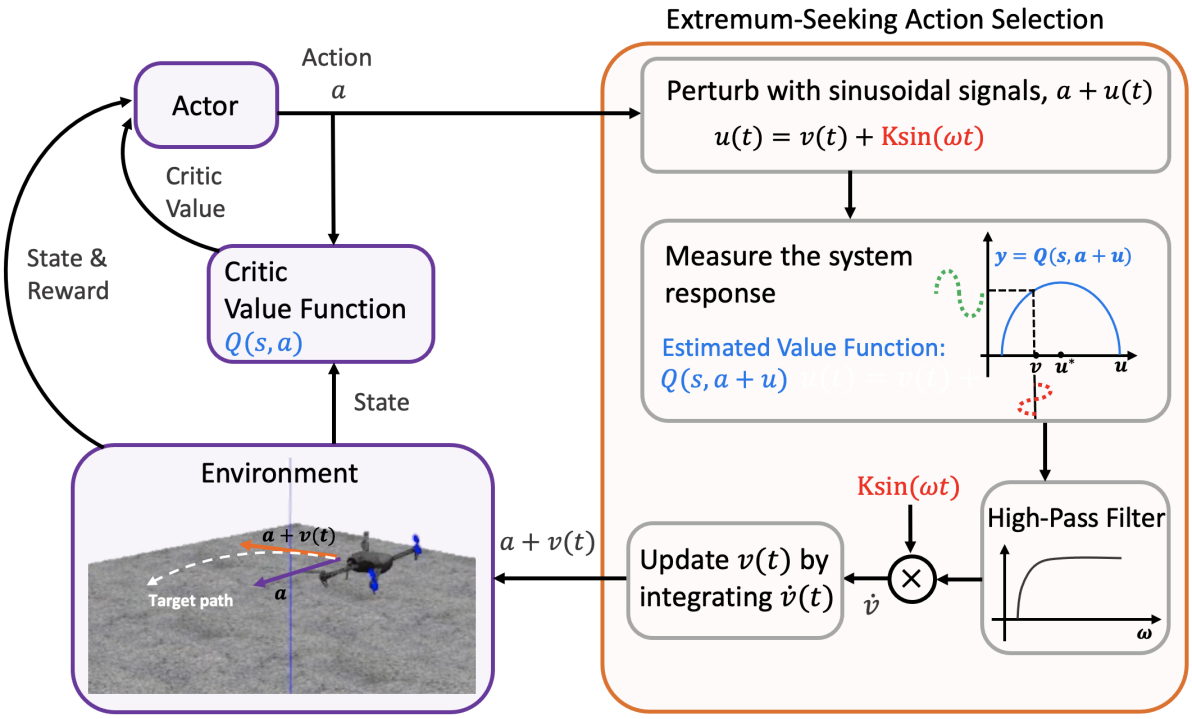

Figure 5.1. Diagram for Extremum-Seeking Action Selection (ESA) in the RL setting. We use Extremum-Seeking Control (ESC) strategies to improve the quality of exploratory actions, which reduces the sampling of low-value trajectories and accelerates policy optimization.

that adapt to the system response. It can improve the performance of the system by tracking its local optimum without analytic knowledge of the underlying dynamics. We do not directly change the practice of using Gaussian distributions as stochastic policies, but adapt ESC to the RL context to improve the quality of sampled trajectories. After each action is sampled from stochastic policies in the standard way, we apply sinusoidal perturbations and query for estimated Q-values as the response signal (Figure 5.1). Based on ESC, we then dynamically improve the sampled actions to be closer to nearby optima before applying them to the environment.

We will demonstrate how the ESC components can be easily added in standard policy optimization algorithms such as PPO [67] and SAC [69] to improve learning efficiency, without requesting any additional models or oracles that are often needed in DPI approaches [128].

In Section 5.2, we first use simple model-free optimization settings to demonstrate why ESC can achieve better sample quality and efficiency than policy gradient methods. Then

in Section 5.3, we describe our Extreme-Seeking Action Selection (ESA) method for policy optimization that uses ESC to improve action selection and reduces the sampling of low-value trajectories, to accelerate the learning process. In Section 5.4, we evaluate the proposed methods in various continuous control environments in MuJoCo [110], as well as a high-fidelity simulation environment for quadrotor control. We show the benefits of adding ESA in PPO and SAC, as well as other approaches such as adding parameter noise. We perform various ablation studies to show basic strategies for hyperparameter tuning in ESA, which is important in balancing between fast adaptive improvement on each sample and the conservative updates by policy gradient.

Related Work. The integration of adaptive control and reinforcement learning has been investigated in recent work [129–133]. Typically the focus is on using both methods for better control design in systems with specific structures such as control-affineness. Under such assumptions, strong theoretical guarantees can be obtained on stability and asymptotic convergence, which significantly improves standard RL approaches. The work in [130–132] uses RL approaches to first train a nearly optimal controller that is subsequently integrated into adaptive control methods to ensure global boundedness with asymptotic stability. We aim to propose a new direction for using methods from adaptive control to RL that focuses on improving learning efficiency in the general model-free setting. The dynamics perspectives and frequency-domain techniques from ESC can avoid the well-known challenge of maintaining complex distributions for sampling high-value trajectories, and allow us to improve each sampled action in a manner similar to real-time model-free control.

It is important to note that what we propose here is not just an exploration strategy, but a way for enhancing the quality of exploratory samples to accelerate the learning process in challenging control problems. Exploration is a major topic in RL [134]. Over continuous spaces, a widely used technique is to introduce noise into the policy action space [118, 119, 135–137] or parameter space [138–141]. We introduce sinusoidal perturbations based on ESC design to enable the use of frequency-domain techniques for improving each action sample, which is

different from further injecting random noise. We will further discuss the difference and compare the performance with the exploration through noise injection approaches in the experiments.

5.2 Preliminaries

5.2.1 Extreme-Seeking Control

Extremum-seeking control (ESC) is a model-free adaptive control method for adjusting inputs to a system to dynamically track its locally optimal performance. Here we explain it in the simplest setting of the system being an unknown but static objective function, in which case ESC can be viewed as a zeroth-order optimization method.

Consider a continuous objective function $J(u) : \mathbb{R}^n \to \mathbb{R}$. Starting from an initial input $u(0) \in \mathbb{R}^n$, ESC designs a control law for tracking a nearby local optimum u^* of J without accessing the analytic form of J. The core idea is that by injecting periodic perturbations on the input $u(t)$ and observing the change in $J(u(t))$, we can perform frequency-domain analysis to exploit the derivative information of J. To achieve this, the ESC introduces an additional estimation variables $v(t) \in \mathbb{R}^n$ that aim to converge to u^*. The feedback control law on $u(t)$ and $v(t)$ are designed as:

$$u(t) = K\sin(\omega t) + v(t) \tag{5.1}$$

$$\dot{v}(t) = -\alpha \mathbb{L}[\sin(\omega t)\mathbb{H}[J(u(t))]] \tag{5.2}$$

where $\mathbb{H}$ represents a high-pass filter, and $\mathbb{L}$ represents a low-pass filter. Note that $v(t)$ is updated through its time derivative $\dot{v}(t)$ in Equation (5.2), from initial condition $v(0) = u(0)$. Importantly, the actual input to the system $u(t)$ in Equation (5.1) is always oscillatory, and it is the estimation vector $v(t)$ that will converge to u^*. $K \in \mathbb{R}^+$ is the magnitude of the sinusoidal perturbation in $u(t)$, and $\alpha \in \mathbb{R}$ is a signed learning rate parameter.

At a high-level, the design uses the control input $u(t)$ to probe the system response

$J(u(t))$. Then for the estimation $v(t)$, we apply high-pass filtering to the response $\mathbb{H}[J(u(t))]$, and then demodulate with $\sin(\omega t)\mathbb{H}[J(u(t))]$, and finally use a low-pass filter $\mathbb{L}[\sin(\omega t)\mathbb{H}[J(u(t))]]$. After these steps, we will be able to make use of second-order properties of the objective J for dynamically updating $v(t)$ to approach u^*. For example, in the one-dimensional case, we can show (more details of the derivation are provided in the Appendix section):

$$\frac{\mathrm{d}}{\mathrm{d}t}(v(t)-u^*) = -\dot{v}(t) = \alpha K \mathbb{L}[\sin(\omega t)\mathbb{H}[J(u(t))]] \\ = -\frac{1}{2}\alpha K J''(u^*)\cdot(v(t)-u^*) \quad (5.3)$$

Thus, if J is convex around u^* (the concave case can be handled by changing the sign of α), then $-\frac{1}{2}\alpha K J''(u^*)$ is real and negative, and the error between $v(t)$ and u^* follows exponentially-stable linear dynamics that quickly converges to zero. Namely, the estimation $v(t)$ will converge to the optimum u^* of the objective J. Importantly, although the analysis uses $J''(u^*)$, it never needs to be known or estimated, since the algorithm only needs to iteratively update $u(t)$ and $v(t)$ according to Equation (5.1) and (5.2). In general, ESC methods ensure the following:

Proposition 4 (Convergence of ESC [125])**.** *With appropriate sinusoidal perturbations and the corresponding filters, the estimation $v(t)$ exponentially converges to a local optimum u^* of the objective function J in a neighborhood of $v(0)$.*

Although the method sketched above considers a static objective, the power of ESC methods lies in its ability of dealing with stochastic objectives. The idea is that as long as the plant dynamics has lower-frequency than the perturbation, the high-pass filter will remove the intrinsic frequency of the system dynamics, and the same derivation applies to the time-varying $J(u(t),t)$, possibly with stochasticity. In the dynamic case, the perturbation frequency should be chosen to be much higher than the frequency of $J(u^*,t)$, so that the high-pass filtering steps can be effective. Also, the methods can naturally be applied to multi-dimensional control inputs, using different frequencies for each input dimension. The general settings are discussed in detail

in [125, 126].

5.2.2 Comparison with Policy Gradient

Policy gradient in RL can be considered a special case of the general strategy of search gradient [142], which we can directly compare with ESC. Again consider the setting of optimizing an unknown objective $J(u)$. The search gradient approach operates with a parameterized distribution $P_\theta(u)$ over the input space with density $p_\theta(u)$, and optimizes the following objective in the parameter space Θ:

$$\max_{\theta\in\Theta} \mathbb{E}_{u\sim P_\theta(u)}[J(u)]$$

by iteratively improves the parameters θ following the gradient of the stochastic objective, which is of the form:

$$\nabla_\theta \mathbb{E}_{u\sim P_\theta(u)}[J(u)] = \mathbb{E}_{u\sim P_\theta(u)}[J(u)\nabla_\theta \log(p(u))]. \tag{5.4}$$

With appropriate learning rates, following the search gradient ensures convergence to a distribution that centers at a local optimum of the objective J. The method benefits from reliable Monte Carlo estimation of the gradient in Eq (5.4) with enough samples, which is suitable for offline learning and conservative policy optimization. However, the dynamic updates in ESC can achieve much faster per-sample improvement, as we show in the following example.

Example 2. *In Figure 5.2, we compare ESC and policy gradient on simple objectives in both static and dynamic settings. We first use the static objective* $J(u) = (u_1 - 0.1)^2 + (u_2 - 0.5)^2$ *in Figure 5.2(a), where the initial input is at* $(2,2)$*. The blue curve at the bottom that shows the fastest convergence to* $J(u^*) = 0$ *is achieved by the estimate* $v(t)$ *in ESC, and the oscillating dotted curve around it is the response on the actual control input* $u(t)$*. In contrast, the other curves from policy gradient methods show much slower convergence that is only competitive when 100 samples are used for each update, whereas ESC only queries the objective with one input sample per iteration. In fact, ESC can almost achieve the same progress as gradient descent, which uses the analytic gradient of the function. In Figure 5.2(b), we consider a time-varying*

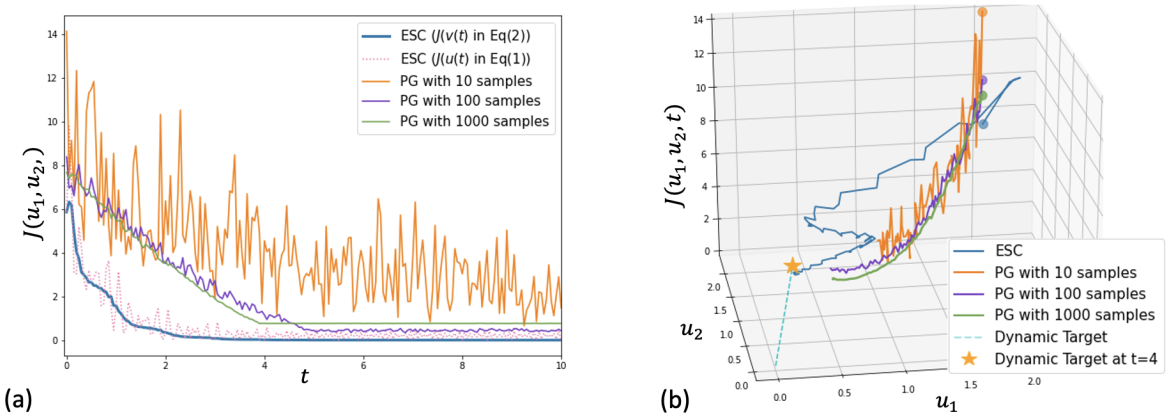

Figure 5.2. Illustration of the optimum tracking performance between ESC and PG in Example 2. It demonstrates that ESC (blue) exhibits the fastest convergence rate in both static and dynamic optimum examples. (a) The convergence speed in tracking a static objective function. (b) Comparison of the convergence in tracking a time-varying objective function. Trajectories show the convergence towards the optimum over time with varying objective values. The initial point is represented by circle dots, and the goal point at time $t = 4$ is denoted by a star.

objective $J(u,t) = (u_1 - 0.1t)^2 + (u_2 - 0.5t)^2$. *All methods start from* $(2,2)$ *which is far from the initial optimum of the objective, which is at* $(0,0)$. *We see that the blue curve representing the estimation with ESC quickly converges to the objective after* $t = 4$, *while policy gradient methods have a hard time quickly tracking the changing objective and the sample size needs to be very large.*

Consequently, it can be beneficial to use ESC to improve the quality of each sample, while maintaining an overall policy gradient framework for reliable improvement. This is the key approach that will be explained in the next section.

For the RL setting, we use the following standard notations for policy optimization. Markov Decision Processes are tuples $\mathscr{M} = \langle \mathscr{S}, \mathscr{A}, P, r, \gamma \rangle$ where $\mathscr{S}$ is the state space and $\mathscr{A}$ the action space, and both are continuous in our setting. The transition function $P : \mathscr{S} \times \mathscr{S} \times \mathscr{A} \to [0,1]$ determines the probability $P(s'|s,a)$ of transitioning into state s' from state s after taking action a. We consider general forms of reward functions $r : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ defined over transitions, and $\gamma \in [0,1)$ is the discount factor. We write π_θ to denote a stochastic policy $\pi_\theta : \mathscr{S} \times \mathscr{A} \to [0,1]$ parameterized by θ. The goal of policy optimization is to maximize

the expected γ-discounted cumulative return $J_{\rho(s_0)}(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\right]$ under some distribution of the initial states $\rho(s_0)$. The Q-value of a state-action pair (s,a) under a policy π_θ is the expectation of cumulative return of future trajectories after taking (s,a), defined as $Q^{\pi_\theta}(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a]$. It satisfies the Bellman equation $Q^{\pi_\theta}(s,a) = \mathbb{E}[r(s,a,s') + \gamma V^{\pi_\theta}(s')]$.

In policy gradient, policy parameters θ are updated at some learning rate α in the direction of

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[A^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)]$$

where $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$ is the advantage of the action a at state s, and $\hat{\mathbb{E}}[\cdot]$ denotes the empirical mean estimated through sampled trajectories. In continuous spaces, the behavior policy $\pi_\theta(a|s)$ at each state s typically is chosen to be a Gaussian distribution $\mathcal{N}(\mu_s, \Sigma_s)$ over the action space.

5.3 Extremum-Seeking Action Selection

We now describe the Extreme-Seeking Action Selection (ESA) method, which uses ESC strategies for improving each action sample to attain higher advantages locally, with exploration driven by both the sampling distribution from the behavior policy and the perturbations within ESC. The overall algorithm is shown in Algorithm 4.

ESA can be used as an add-on component for improving the quality of each action sample in typical policy optimization algorithms. We write $a_t \sim \pi_\theta(\cdot|s_t)$ as the action sample drawn at state s_t at time step t, according to the distribution determined by the current policy π_θ. Following ESC design in Equation (5.1) and (5.2), we need to maintain two vectors: $u(t)$ as the oscillatory control input, and $v(t)$ for updating an estimation that approaches the optimum. In the RL setting, persistent perturbation on the action hinders convergence and policy improvement. So instead of using oscillatory inputs as actions, we use $u(t)$ to probe estimations of Q-values, from which we find a reliable improvement $a_t + v(t)$ of the original action sample, which is then

applied to the environment. $u(0)$, $v(0)$, and t are all set to zero in the beginning of each episode.

Concretely, as shown in Algorithm 4, for each sampled action a_t, we first apply sinusoidal perturbation as $a_t + u(t)$, where $u(t) = v(t) + K\sin(\omega t)$. We use the Q-values determined by the current policy π_θ as the objective, and query for the value of $Q(s, a_t + u(t))$. Next, we use a modification of Equation (5.2) to update $v(t)$ as follows:

$$v(t+1) \leftarrow v(t) + \alpha \sin(\omega t)\mathbb{H}[Q(s, a_t + u(t))] \tag{5.5}$$

where $\mathbb{H}$ is a high-pass filter, and $\alpha > 0$ is a learning rate. We do not need the minus sign in Equation (5.2), which was for minimizing the objective and here we maximize. Importantly, we have also dropped the low-pass filter $\mathbb{L}[\cdot]$ from the standard design of ESC, because we want to still allow some high-frequency perturbations for enabling exploration, which is different from the ESC goal of tracking the extremum as precisely as possible. In this way, exploration of actions is achieved first by the original sample $a_t \sim \pi_\theta(\cdot|s_t)$, and then with sinusoidal components in $v(t)$ from the definition above. Finally, $a_t + v(t)$ will be the actual action applied to the environment, and we move onto the next state s_{t+1} by querying the environment with $(s_t, a_t + v(t))$.

We take advantage of the ability of ESC methods for directly tracking time-varying objectives, which is the Q-values that change over time steps within each episode. It bypasses a major challenge for exploration in continuous spaces, where we can not easily keep track of properties of individual states (such as visitation counts that are often used for exploration in discrete spaces). Instead, using the control-theoretic and frequency-domain analysis, we shift the focus to time-varying perturbation throughout the entire trajectories to achieve improvement in the overall performance.

Hyperparameters. The new hyperparameters introduced by the ESA component include the perturbation magnitude vector K, sinusoidal perturbation frequency vector ω, and the learning rate vector α. In particular, as long as the perturbation frequency is reasonably higher than the frequency of the Q-value function, the high-pass filter will be able to isolate the local

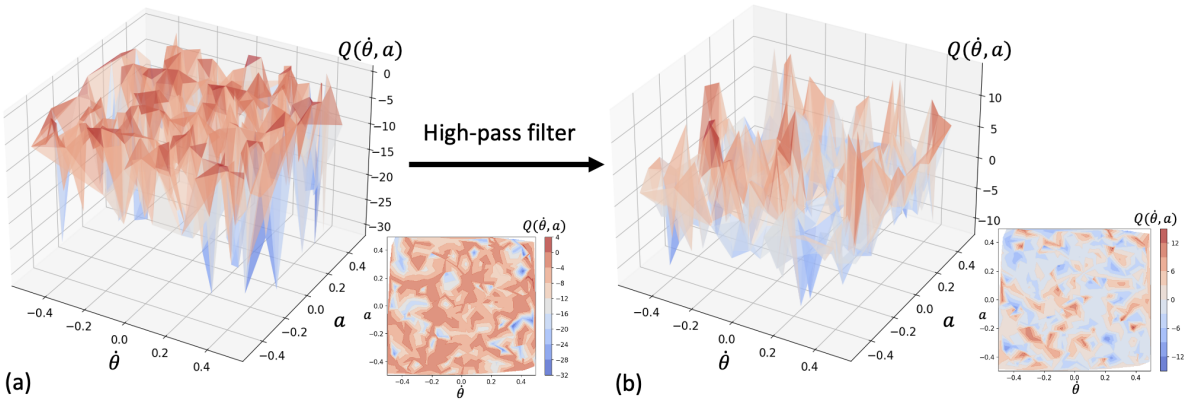

Figure 5.3. An illustration of the effect of using high-pass filters on the Q-value landscapes. (a) A Q-value landscape at a state in the inverted pendulum environment, plotted for a fixed policy π_θ at an intermediate stage of training. (b) Filtered Q-value landscape from (a).

Algorithm 4. Policy Optimization with ESA

Randomly initialize the policy network parameters θ and Q-network parameters ϕ, and empty replay buffer $\mathscr{D}$.
Choose hyperparameters: perturbation amplitudes vector K, frequency vector ω, and learning rate vector α,
for episodes $= 1, \ldots, N$ **do**
    $u(0) \leftarrow 0$ and $v(0) \leftarrow 0$
    for $t = 0, \ldots, T$ **do**
        Sample $a_t \sim \pi_\theta(\cdot|s_t)$
        $u(t) \leftarrow v(t) + K \sin \omega t$ ▷ Following Eq (5.1)
        $v(t+1) \leftarrow v(t) + \alpha K \sin(\omega t) \mathbb{H}[Q(s, a_t + u(t))]$
        ▷ Filter based on Q-values, as Eq. (5.5)
        $s_{t+1} \leftarrow \text{Env}(s_t, a_t + v(t))$
        $\mathscr{D} \leftarrow \mathscr{D} \cup (s_t, a_t + v(t), r_t, s_{t+1})$
    end for
    for each policy optimization step **do**
        Update θ and ϕ with $\mathscr{D}$ using standard policy optimization algorithms
    end for
end for

second-order information of the objective. Effects of the hyperparameter choices will be further discussed through ablation study in Section 5.4.

Benefits of High-Pass Filtering. High-pass filtering is an important step in ESC that ensures the convergence of the design of the algorithm. Intuitively, in the context of action selection, high-pass filters remove "flat" regions in the Q-value landscape, making it easier to locate actions

that lead to local peak Q-values. In Figure 5.3, we observe that high-pass filters enhance the visibility of peaks and enable faster local improvement towards the optimum.

Comparison with Using Analytic Gradient of Q-Networks. A natural approach of improving action samples to higher quality is to follow the gradient of the Q-value networks, i.e., using a first-order approach rather than the zeroth-order one proposed here. Namely, for each state s_t and sampled action a_t, we can query the Q-network for its gradient at (s_t, a_t) over actions, which in principle should indicate the direction of moving the sampled action towards higher Q-values. However, the Q-value models provided by deep neural networks have highly non-smooth landscapes over action inputs, as illustrated in Figure 5.3. Thus the analytic gradients are frequently misleading. Instead, ESC provides robust estimation through filtering and frequency analysis.

Pseudo-Lyapunov Clipping on Stabilization Problems. A main design factor in the effective use of ESA is in making sure that the sinusoidal perturbation does not hinder reliable policy updates. In stabilization control problems, there are special structures that allow systematic clipping of the ESA perturbations based on the performance of the behavior policy. We have developed such a method using pseudo-Lyapunov landscapes that are learned during the policy optimization process. Instead of relying on decreasing learning rates to balance the interaction between ESA action selection and policy gradient updates, we employ learned Lyapunov values to determine when to dampen sinusoidal perturbations in ESA. Based on the definition of Lyapunov functions, Lyapunov values can be employed as an evaluation metric to quantify the deviation between the current state and the desired goal state. When a value is above a certain threshold, we apply ESA to gather the next information of the environment. Conversely, if the deviation is below the threshold, we select a sample action from the current policy. Figure 5.4 shows how the overall algorithm works on the inverted pendulum example. We apply ESA when values above the purple level curve of the Lyapunov function, and observe a significant improvement in the stabilization performance when using ESA. In Figure 5.4 (c), we observe the different trajectories with ESA (blue line, stabilizing) and without (gray line, diverging).

It demonstrates that by clipping ESA based on the Lyapunov values, the training process can effectively respond to deviations from the desired states, leading to improved control and overall system stability.

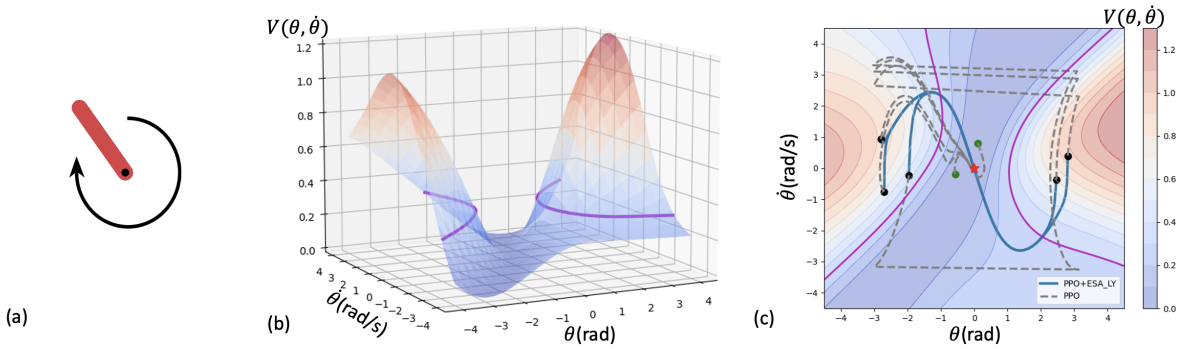

Figure 5.4. Illustration of the learned pseudo-Lyapunov landscape for clipping the ESA perturbations. (a) Inverted pendulum environment in OpenAI Gym. (b) The learned pseudo-Lyapunov function with a threshold (purple) Lyapunov value for damping perturbations. (b) Sampled trajectories from PPO and PPO+ESA. It shows that ESA is most effectively improving action selection when the pseudo-Lyapunov value exceeds a certain threshold. The black dots represent the starting position of each trajectory and the red star represents the goal point.

5.4 Experiments

We show experimental results to evaluate how ESA can improve the performance of policy optimization. We add ESA to the leading policy optimization methods including Proximal Policy Optimization (PPO) [67] and Soft Actor-Critic (SAC) [69] and benchmark the performance difference in various challenging control learning environments.

Environments. We consider continuous control environments in OpenAI Gym [108] and MuJoCo [110], including the inverted pendulum, hopper, and walker, as well as a Gazebo-based quadrotor control simulator enabled by the commercially-used autopilot framework PX4 [143]. The quadrotor control environment involves 12 state dimensions (inertia frame positions, velocities, rotation angles, and angular velocities) and 4 control inputs (thrust, roll, pitch, and yaw). Details of the equations of motion of the quadrotor can be found in [70]. The goal of the agent is to track an oriented point along a path, and the rewards are calculated based on the discrepancies

between their positions and orientations.

Baselines. We compare the performance of PPO+ESA and SAC+ESA with the standard PPO and SAC, as well as with the versions using additional parameter space noise, a widely-used approach for enhancing exploration [140]. We also show compare with DDPG incorporating time-correlated Ornstein–Uhlenbeck noise [118]. All algorithms are tested on 5 different random seeds in all environments.

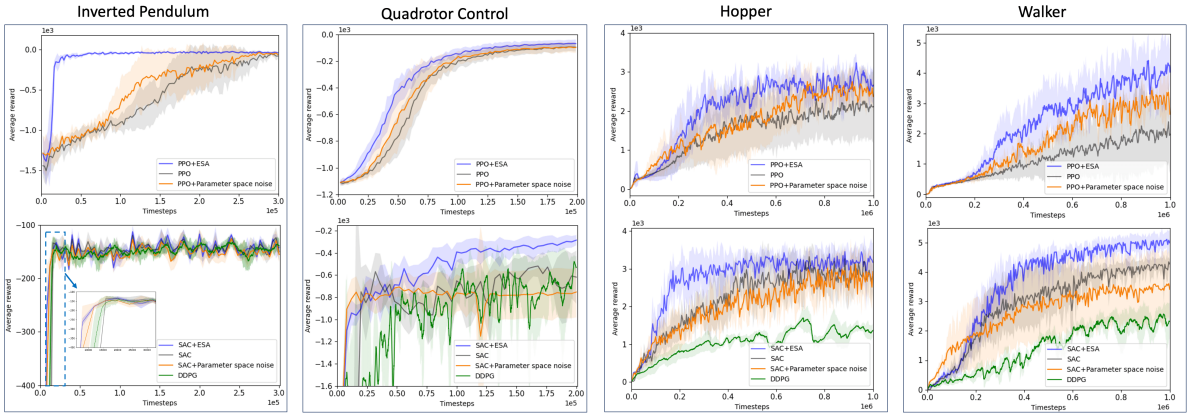

Figure 5.5. Performance comparison for all methods. PPO+ESA (blue, first row) and SAC+ESA (blue, second row) demonstrate higher learning efficiency and performance compared to other methods across all tasks. In comparison, adding random parameter noise (orange) leads to better exploration in the early stages of some tasks, but fails to sustain effective exploration throughout the entire training process.

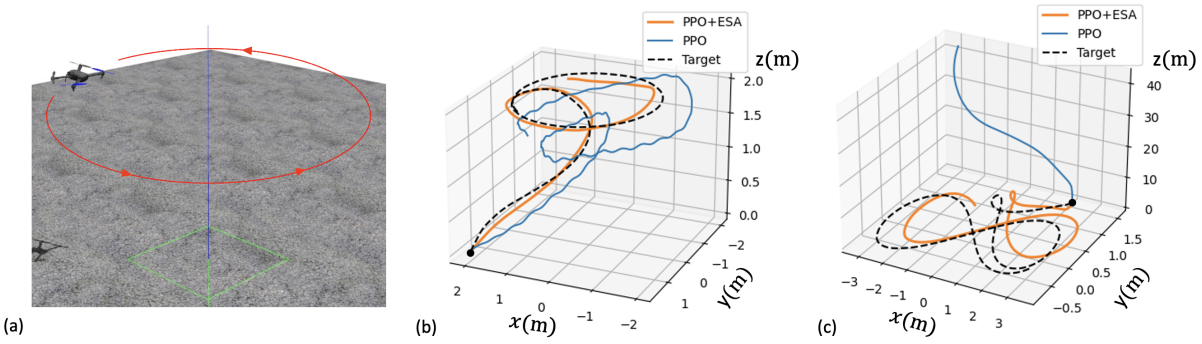

Figure 5.6. Illustration of how ESA improves the performance of PPO for quadrotor environment. We evaluated both policies trained after the same number of iterations. We observe that ESA improves the quality of sampled actions and accelerates learning. (a) Quadrotor control environment. (b) Performance comparison in a circle target path task. (c) Performance comparison in tracking an eight-shaped target path, where the PPO-trained policy diverges.

Overall Performance. Figure 5.5 shows comparisons of learning curves for all methods in benchmark environments. We observe that ESA accelerates learning and enhances the performance of both PPO and SAC, and outperforms other baselines. The computational cost of adding ESA is at most 50 percent longer runtime for each episode (2048 steps). In particular, Figure 5.6 demonstrates the specific improvement in performance in the quadrotor control environment. We visualize the behaviors of the trained control policies after the same number of training steps, and observe that PPO+ESA shows clear improvement in the control performance compared to the original PPO-trained policy.

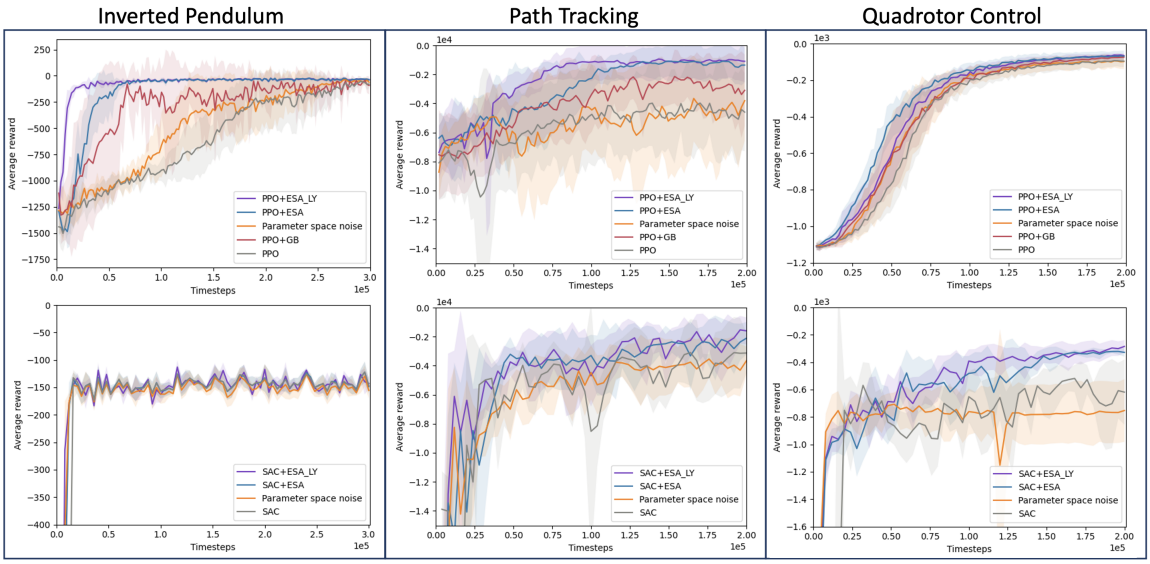

Figure 5.7. Performance comparison in stabilization control environments. We assess the effectiveness of the proposed method ESA_LY (purple) introduced in section 4.2 for adaptive clipping of ESA selection in stabilization tasks. Leveraging Lyapunov functions, ESA_LY provides direction suggestions based on the values of Lyapunov functions. In addition to reducing one tuning parameter of ESA, ESA_LY surpasses ESA and other existing approaches in terms of performance.

In the stabilization environments in Figure 5.7, the learned pseudo-Lyapunov landscape further improves the learning performance. We assess the effectiveness of the proposed method ESA_LY (purple) introduced in section 4.2 for adaptive clipping of ESA selection in stabilization tasks. Leveraging Lyapunov functions, ESA_LY provides direction suggestions based on the values of Lyapunov functions. In addition to reducing one tuning parameter of ESA, ESA_LY

surpasses ESA and other existing approaches in terms of performance.

Ablation Study: Perturbation Magnitude. The parameter amplitude K of the perturbation signal presents a trade-off between increasing convergence speed and reducing oscillation. Figure 5.8(a) shows how the learning performance changes as at various values of K for the perturbation signal in the inverted pendulum environment, when the frequency of the perturbation is fixed. We see that there the magnitude of $K = 0.2$ (red) achieves the best outcome. Reducing K to 0.1 leads to a slower convergence speed. Increasing K to be above 0.2 accelerates the initial progress of the learning curves but results in much higher variance in performance across different random seeds.

Ablation Study: Perturbation Frequency. The effective perturbation frequencies are affected by the environment dynamics and the responsiveness of the Q-value functions. In general, higher ω allows us to obtain a more accurate gradient estimate by applying a high-pass filter to the value. However, very high frequency may lead to non-smooth action choices that negatively impact policy learning. On the other hand, as shown in Figure 5.8(b), we observe that when the frequecy gets higher than 10π the effectiveness of ESA is reduced.

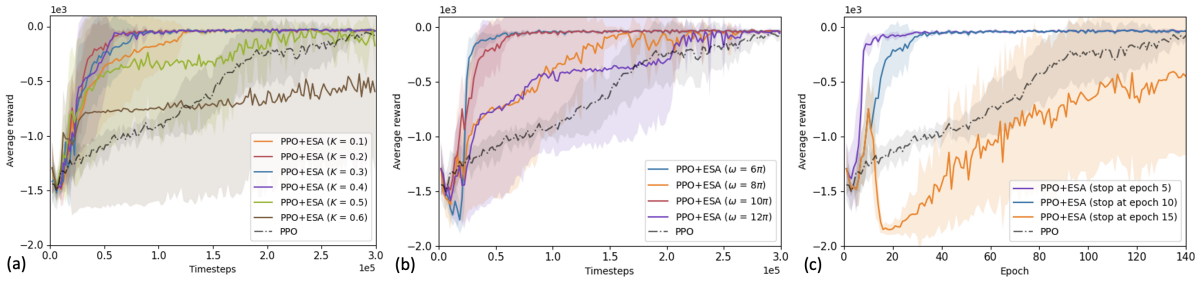

Figure 5.8. Ablation studies conducted on the inverted pendulum environment. (a) Training curves in relation to the magnitude of the perturbation signal. (b) Training curves in relation to the frequency of the perturbation signal. (c) Training curves comparing different numbers of ESA intervening episodes.

Ablation Study: Decay of ESA Learning Rate. In the policy optimization process, when the policy has reached near-peak performance, it also becomes very sensitive to perturbations. Thus the ESA-driven exploration should decay over time to avoid destabilizing policy learning. The results in Figure 5.8(c) illustrate the impact of different decay rates on performance.

5.5 Conclusion

We proposed the extremum-seeking action selection (ESA) method for improving both exploration and exploitation in sampling actions for policy optimization in continuous spaces. We follow the strategies in extremum-seeking control (ESC) by applying sinusoidal perturbations on the sampled actions in each step to obtain actions of higher action values and also improve exploration. We have shown that ESC methods can be particularly sample efficient for locally optimizing unknown objectives, compared to policy gradient methods. At the same time, the scale of ESA perturbations on the sampled actions needs to be carefully chosen to balance the trade-off between fast local improvement with ESC and reliable policy improvement over all states. The ability of tracking dynamic objectives makes ESC methods particularly suitable for handling problems in the continuous domain by shifting the focus from states to improving entire trajectories over time. Moreover, for stabilization problems, we show that further learning approximate pseudo-Lyapunov landscapes can provide a simple way of determining the clipping of the sinusoidal perturbations on the actions. We observed clear benefits of adding ESA methods in PPO and SAC in improving the learning performance in various continuous control problems.

Appendix: More Details in the Derivation of ESC

Because ESC techniques have rarely been introduced in the context of reinforcement learning, we provide more derivation details on how ESC ensures that its estimate $v(t)$ converges locally to an optimum, following Equation (5.1-5.2) We use the one-dimensional input for simplicity, and more general derivations can be found in standard references such as [125].

Consider minimizing $J(u)$ near a strict local minimizer u^*, which means that the first-order derivative $J'(u^*) = 0$, and the second-order derivative $J''(u^*) > 0$ in a local neighborhood of u^*. Since the analysis is local, we can approximate $J(u(t))$ with the second-order Taylor

expansion:

$$\begin{aligned} J(u(t)) &\cong J(u^*) + \frac{1}{2}J''(u^*)(u(t)-u^*)^2 \\ &= J(u^*) + \frac{1}{2}J''(u^*)(K\sin(\omega t) + v(t) - u^*)^2 \end{aligned}$$

where the first-order term $J'(u^*) = 0$, and the second equality is by plugging in $u(t)$ from Equation (5.1). Now, a key step is to focus on the error dynamics, which is how the difference between $v(t)$ and u^* changes, by defining $\xi(t) = u^* - v(t)$. The previous expansion can then be rewritten as:

$$\begin{aligned} J(u(t)) = J(u^*) + \frac{1}{2}J''(u^*)\xi(t)^2 - KJ''(u^*)\sin(\omega t)\xi(t) \\ + \frac{1}{4}K^2J''(u^*)(1-\cos 2\omega t) \end{aligned}$$

where we use $\sin^2(\omega t) = \frac{1}{2}(1-\cos 2\omega t)$. Now, by applying a high-frequency filter $\mathbb{H}$ on $J(u(t))$, we can remove the terms at lower frequencies such as $J(u^*)$ and $\frac{1}{4}K^2J''(u^*)$. We also remove the second-order term $\xi(t)^2$ dominated by $\xi(t)$ in the local analysis:

$$\mathbb{H}[J(u(t))] \cong -KJ''(u^*)\sin(\omega t)\xi(t) - \frac{1}{2}K^2J''(u^*)\cos(2\omega t)$$

Here we see why the filtering mechanism is useful: it allows us to inspect the signal only at certain frequencies that carry the information we need, in this case $J''(u^*)$. To fully do that, we will further demodulate the signal with $\sin(\omega t)$ and then apply low-pass filtering and arrive at the approximation:

$$\mathbb{L}[\sin(\omega t)\mathbb{H}[J(u(t))]] \cong -\frac{1}{2}KJ''(u^*)\xi(t)$$

where the sinusoidal terms are all filtered out by the low-pass filter $\mathbb{L}[\cdot]$ because of their high frequency. Now, plugging it in the definition of $\xi(t)$ and $\dot{v}(t)$ from Equation, we arrive at Equation 5.3 in Section 5.2.1. Consequently, the error dynamics $\xi(t)$ follows exponentially stabilizes to zero. Through such frequency-domain analysis, we see that $J''(u^*)$ does not need to be estimated, and convergence is implicitly guaranteed by following the control law for $u(t)$ and $v(t)$.

5.6 Acknowledgement

Chapter 5, in full, is a modified reprint of material previously published in Ya-Chien Chang, and Sicun Gao, “Extremum-Seeking Action Selection for Accelerating Policy Optimization,” IEEE International Conference on Robotics and Automation (ICRA), 2024. The dissertation author was the primary investigator and author of this material.

Chapter 6

Conclusion and Future Work

6.1 Summary of Accomplishments in This Dissertation

This dissertation aimed to advance the design of learning-based control systems that provide formal guarantees of stability, with a central focus on neural Lyapunov methods. Motivated by the limitations that the lack of stability guarantees places on the practical deployment of data-driven controllers, we developed a principled framework that combines insights from control theory with the flexibility of deep learning. Through the construction of neural Lyapunov certificates, we demonstrated how stability and robustness can be formally verified across a variety of learning-based control settings. These methods were successfully applied to model-based stabilization, model-free reinforcement learning, and imitation learning, highlighting the potential of neural Lyapunov approaches to bridge the gap between theoretical guarantees and empirical performance. In sum, these contributions mark a significant step toward enabling the reliable and scalable deployment of learning-based control in real-world, safety-critical systems.

The work presented in this dissertation introduced a sequence of methods that progressively broaden the scope and applicability of neural Lyapunov approaches. Chapter 2 introduced a learner-verifier framework for learning provably stable control policies and neural Lyapunov functions by directly addressing system nonlinearities, without relying on local approximations of the dynamics. This approach simplifies Lyapunov-based control design and offers flexibility to enhance control performance by enlarging the region of attraction through regularization during learning. In Chapter 3, we extended this neural Lyapunov framework to the model-free reinforcement learning setting, where the agent learns a Lyapunov-like function through self-supervision in the form of a specialized critic. This critic

improves stability and provides estimates of the region of attraction under Almost Lyapunov conditions for the learned policy, enabling stability-aware policy learning without requiring an explicit model of the environment. Chapter 4 built on this model-free foundation to address the Learning from Observation (LfO) problem, where only state trajectories from expert demonstrations are available. We introduced a novel Lyapunov landscape learning formulation that enables efficient policy learning with significantly fewer expert samples than prior LfO methods. Finally, Chapter 5 proposed an Extremum-Seeking Action Selection (ESA) strategy to improve action selection during training. By leveraging perturbation and filtering techniques, ESA enhances local peak Q-values and guides sampled actions toward nearby optima, thereby improving both exploration and sample efficiency and outperforming policy gradient methods in continuous control tasks.

Across all chapters, this dissertation contributes a systematic methodology for incorporating Lyapunov stability theory into learning-based control. Each component, from learner-verifier training and Lyapunov critics to Lyapunov landscape proxy model and adaptive action refinement, broadens the scope of neural Lyapunov approaches across diverse learning paradigms while maintaining theoretical soundness. In combination, these techniques demonstrate that it is possible to design controllers that are not only effective and data-efficient but also certifiably stable. Moreover, this work advances the practical applicability of safe learning-based control and lays the groundwork for future developments in reliable autonomous systems.

6.2 Future Research Directions

6.2.1 Scalability to High-Dimensional Systems

Future work could extend this framework by developing scalable learning and verification algorithms for constructing neural Lyapunov functions in high-dimensional control problems. In Chapter 2, we relied on the dReal SMT solver to verify nonlinear constraints, leveraging its sound and complete guarantees. However, such solvers are known to suffer from increased computation time as the dimensionality of the state space grows, limiting their practical scalability. In Chapter 3, we addressed this issue by shifting to a sampling-based approach using Almost Lyapunov conditions, which enabled approximate stability verification in higher-dimensional settings. While this relaxation improves tractability, it introduces

statistical uncertainty and lacks formal guarantees across the entire state space.

To push toward scalability without sacrificing reliability, future work could draw on advances in neural network verification, where researchers have developed scalable over-approximation techniques, such as zonotope-based reachability, abstract interpretation, and convex relaxations, to efficiently bound neural network outputs over input sets. Adapting these techniques to the Lyapunov verification setting would allow stability conditions to be checked without exhaustively sampling the entire state space, improving scalability while maintaining formal correctness. These advances could lead to more accurate and reliable safety certificates in high-dimensional control systems, enhancing the ability of learned policies to correct or avoid unsafe behaviors during training and deployment.

6.2.2 Multi-Agent Systems and Distributed Control

Another promising direction for future work is the extension of neural Lyapunov methods to multi-agent systems. Ensuring safety and stability in learning-based multi-agent control remains an open challenge, particularly in settings that demand decentralization, partial observability, or asynchronous updates. Recent research has explored Lyapunov-inspired frameworks for multi-agent learning, including decentralized reinforcement learning with shared Lyapunov constraints and graph-structured policies that stabilize formation behaviors. However, the integration of neural Lyapunov functions into learning-based distributed control architectures is still in its early stages. Future work could develop decentralized Lyapunov critics, where agents adapt local certificates in response to both environmental feedback and neighboring agent behavior. By enabling certifiably stable coordination in distributed systems, this research direction has the potential to support large-scale, safety-critical applications in swarm robotics, autonomous fleets, and distributed cyber-physical systems.

6.2.3 Orbital Stability in Periodic Control Tasks

Throughout this dissertation, the focus has primarily been on stabilization problems, where the objective is to drive the system toward an equilibrium. However, many real-world control tasks, such as dynamic locomotion, involve non-equilibrium behaviors, like periodic gaits in quadrupeds, that require a different notion of stability, orbital stability around limit cycles. While orbital Lyapunov functions have been explored in theory and low-dimensional examples, the integration of neural Lyapunov methods

into learning-based locomotion control, particularly for certifying gait safety in high-dimensional, real-world systems, remains an open and promising direction. Extending these methods to learn orbital stability certificates could involve enforcing Lyapunov risk decrease along periodic trajectories or adapting verification tools to certify local contraction within trajectory tubes. Moreover, integrating orbital Lyapunov critics with reinforcement or imitation learning could help guide policy training toward robust and safe locomotion behaviors. Extending stability analysis to dynamic behaviors would strengthen the theoretical grounding of learning-based locomotion controllers, particularly in the face of environmental variability and modeling uncertainty.

Bibliography

[1] Peter Giesl and Sigurdur Hafstein. Review on computational methods for lyapunov functions. *Discrete & Continuous Dynamical Systems-B*, 20(8):2291, 2015.

[2] Pablo A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[3] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[4] James Anderson and Antonis Papachristodoulou. Advances in computational lyapunov analysis using sum-of-squares programming. *Discrete & Continuous Dynamical Systems-B*, 20(8):2361, 2015.

[5] Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 3482–3487. IEEE, 2002.

[6] Antonis Papachristodoulou and Stephen Prajna. A tutorial on sum of squares techniques for systems analysis. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2686–2700. IEEE, 2005.

[7] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A Parrilo. A globally asymptotically stable polynomial vector field with no polynomial lyapunov function. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7579–7580. IEEE, 2011.

[8] Amir A. Ahmadi and Pablo A. Parrilo. Stability of polynomial differential equations: Complexity and converse lyapunov questions. *CoRR*, abs/1308.6833, 2013.

[9] Amir A. Ahmadi and Raphaël M. Jungers. Lower bounds on complexity of lyapunov functions for switched linear systems. *CoRR*, abs/1504.03761, 2015.

[10] Y Long and MM Bayoumi. Feedback stabilization: Control lyapunov functions modelled by neural networks. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2812–2814. IEEE, 1993.

[11] Danil V Prokhorov. A lyapunov machine for stability analysis of nonlinear systems. In *Proceedings*

of 1994 IEEE International Conference on Neural Networks (ICNN'94), volume 2, pages 1028–1031. IEEE, 1994.

[12] Danil V Prokhorov and Lee A Feldkamp. Analyzing for lyapunov stability with adaptive critics. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 2, pages 1658–1661. IEEE, 1998.

[13] Danil V Prokhorov and Lee A Feldkamp. Application of svm to lyapunov function approximation. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 1, pages 383–387. IEEE, 1999.

[14] G. Serpen. Empirical approximation for lyapunov functions with artificial neural nets. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 735–740 vol. 2, 2005.

[15] V. Petridis and S. Petridis. Construction of neural network based lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065, 2006.

[16] Navid Noroozi, Karimaghaee Paknoosh, Safaei Fatemeh, and Javadi Hamed. Generation of lyapunov functions by neural networks. *Lecture Notes in Engineering and Computer Science*, 2170, 07 2008.

[17] Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 466–476, 29–31 Oct 2018.

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

[19] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

[20] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. *arXiv preprint arXiv:2109.06697*, 2021.

[21] Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for lyapunov function approximation. *arXiv preprint arXiv:2109.13359*, 2021.

[22] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[23] Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. Counter-example guided synthesis of neural network lyapunov functions for piecewise linear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1274–1281. IEEE, 2020.

[24] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 3240–3249, 2019.

[25] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[26] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832).* 2019.

[27] Huibert Kwakernaak. *Linear Optimal Control Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1972.

[28] Wassim Haddad and Vijaysekhar Chellaboina. Nonlinear dynamical systems and control: A lyapunov-based approach. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*, 01 2008.

[29] Johan Löfberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, 2009.

[30] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-Complete decision procedures for satisfiability over the reals. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, pages 286–300, 2012.

[31] Yannian Liu and Xin Xin. Controllability and observability of n-link planar robot with a single actuator having different actuator-sensor configurations. *IEEE Transactions on Automatic Control*, 61:1–1, 12 2015.

[32] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4661–4666, Dec 2016.

[33] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 908–918. Curran Associates, Inc., 2017.

[34] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8092–8101. Curran Associates, Inc., 2018.

[35] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006.

[36] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies

is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1805–1814. Curran Associates, Inc., 2018.

[37] Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

[38] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 133–142. ACM, 2014.

[39] Amir Ali Ahmadi. On the difficulty of deciding asymptotic stability of cubic homogeneous vector fields. In *American Control Conference, ACC 2012, Montreal, QC, Canada, June 27-29, 2012*, pages 3334–3339, 2012.

[40] Amir Ali Ahmadi and Raphaël M Jungers. Lower bounds on complexity of lyapunov functions for switched linear systems. *Nonlinear Analysis: Hybrid Systems*, 21:118–129, 2016.

[41] D. Henrion and A. Garulli. *Positive Polynomials in Control*, volume 312 of *Lecture Notes in Control and Information Sciences*. Springer Berlin Heidelberg, 2005.

[42] G. Chesi and D. Henrion. Guest editorial: Special issue on positive polynomials in control. *IEEE Transactions on Automatic Control*, 54(5):935–936, May 2009.

[43] Z. Jarvis-Wloszek, R. Feeley, Weehong Tan, Kunpeng Sun, and A. 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 Vol.5, Dec 2003.

[44] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[45] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, pages 208–214, 2013.

[46] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov control (project website), https://yachienchang.github.io/NeurIPS2019.

[47] Theodore J. Perkins and Andrew G. Barto. Lyapunov design for safe reinforcement learning. *J. Mach. Learn. Res.*, 3:803–832, 2002.

[48] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh, and Edgar A. Duéñez-Guzmán. Lyapunov-based safe policy optimization for continuous control. *CoRR*, abs/1901.10031, 2019.

[49] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters and IROS*, 2020.

[50] Ming Jin and Javad Lavaei. Control-theoretic analysis of smoothness for stability-certified reinforcement learning. In *57th IEEE Conference on Decision and Control, CDC 2018, Miami, FL, USA, December 17-19, 2018*, pages 6840–6847. IEEE, 2018.

[51] Marco Gallieri, Seyed Sina Mirrazavi Salehian, Nihat Engin Toklu, Alessio Quaglino, Jonathan Masci, Jan Koutník, and Faustino Gomez. Safe interactive model-based learning, 2019.

[52] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 3387–3395, 2019.

[53] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions, 2019.

[54] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2669–2678, 2018.

[55] Richard Cheng, Abhinav Verma, Gábor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 1141–1150. PMLR, 2019.

[56] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

[57] Anqi Liu, Guanya Shi, Soon-Jo Chung, Anima Anandkumar, and Yisong Yue. Robust regression for safe exploration in control. volume 120 of *Proceedings of Machine Learning Research*, pages 608–619, The Cloud, 10–11 Jun 2020. PMLR.

[58] Almost lyapunov functions for nonlinear systems. *Automatica*, 113:108758, 2020.

[59] R. Bobiti and M. Lazar. Automated-sampling-based stability verification and doa estimation for nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3659–3674, 2018.

[60] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

[61] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW,*

Australia, 6-11 August 2017, pages 22–31, 2017.

[62] Eitan Altman. Constrained markov decision processes, 1999.

[63] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[64] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML 2016*, pages 1928–1937, 2016.

[65] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML 2015*, pages 1889–1897, 2015.

[66] Yuhuai Wu, Elman Mansimov, Roger B. Grosse, Shun Liao, and Jimmy Ba. Second-order optimization for deep reinforcement learning using kronecker-factored approximation. In *NIPS 2017*, pages 5285–5294, 2017.

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

[68] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[69] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[70] Bartomeu Rubí, Ramon Pérez, and Bernardo Morcego. A survey of path following control strategies for uavs focused on quadrotors. *Journal of Intelligent Robotic Systems*, 05 2020.

[71] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

[72] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 79–86, 2017.

[73] Petar Kormushev, Sylvain Calinon, and Darwin Gordon Caldwell. Robot motor skill coordination with em-based reinforcement learning. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010.

[74] Brenna Argall, S. Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57:469–483, 2009.

[75] Ahmed Hussein, Mohamed Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey

of learning methods. *ACM Computing Surveys*, 50, 04 2017.

[76] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *CoRR*, abs/1905.13566, 2019.

[77] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *CoRR*, abs/1807.06158, 2018.

[78] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[79] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[80] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *ICLR*, 2018.

[81] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *AAAI*, 2018.

[82] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[83] Jiankai Sun, Lantao Yu, Pinqian Dong, Bo Lu, and Bolei Zhou. Adversarial inverse reinforcement learning with self-attention dynamics model. *IEEE Robotics and Automation Letters*, 6(2):1880–1886, 2021.

[84] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

[85] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

[86] Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 09 2004.

[87] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, 05 2000.

[88] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *ICLR 2018*, 2017.

[89] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *CoRR*, abs/1805.01954, 2018.

[90] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T. Pokorny, Anca D. Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 462–469, 2016.

[91] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[92] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010.

[93] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12402–12413. Curran Associates, Inc., 2020.

[94] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[95] Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6036–6045. PMLR, 09–15 Jun 2019.

[96] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1755–1763. PMLR, 09–15 Jun 2019.

[97] Ming Jin and Javad Lavaei. Stability-certified reinforcement learning: A control-theoretic perspective. *IEEE Access*, 8:229086–229100, 2020.

[98] Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[99] He Yin, Peter Seiler, Ming Jin, and Murat Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 6:409–414, 01 2022.

[100] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems

with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[101] S. Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014.

[102] Yonadav Shavit, Nadia Figueroa, Seyed Sina Mirrazavi Salehian, and Aude Billard. Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach. *IEEE Robotics and Automation Letters*, 3(3):2718–2725, 2018.

[103] Nadia Figueroa and Aude Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In *CoRL*, 2018.

[104] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

[105] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[106] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. 2009.

[107] Alborz Geramifard, Christoph Dann, Robert H. Klein, William Dabney, and Jonathan P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(46):1573–1578, 2015.

[108] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[109] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.

[110] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS 2012*, pages 5026–5033. IEEE, 2012.

[111] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In Marc Toussaint, Antonio Bicchi, and Tucker Hermans, editors, *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020.

[112] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, 2018.

[113] Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions*

on Vehicular Technology, 68(3):2124–2136, 2019.

[114] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

[115] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In Marc Toussaint, Antonio Bicchi, and Tucker Hermans, editors, *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020.

[116] Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2023.

[117] Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Lee Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[118] 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.

[119] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[120] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

[121] Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*, 2020.

[122] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[123] Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 2020.

[124] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

[125] Kartik B. Ariyur and Miroslav Krstic. *Real Time Optimization by Extremum Seeking Control*. John Wiley Sons, Inc., USA, 2003.

[126] Chunlei Zhang and Raúl Ordóñez. *Extremum-seeking control and applications: a numerical optimization-based approach*. Springer Science & Business Media, 2011.

[127] Dragan Nešić. Extremum seeking control: Convergence analysis. *European Journal of Control*, 15(3-4):331–347, 2009.

[128] Wen Sun, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Dual policy iteration. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7059–7069, 2018.

[129] Said G Khan, Guido Herrmann, Frank L Lewis, Tony Pipe, and Chris Melhuish. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual reviews in control*, 36(1):42–59, 2012.

[130] Spencer M Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. Adaptive-control-oriented meta-learning for nonlinear systems.

[131] Tyler Westenbroek, Eric Mazumdar, David Fridovich-Keil, Valmik Prabhu, Claire J Tomlin, and S Shankar Sastry. Adaptive control for linearizable systems using on-policy reinforcement learning. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 118–125. IEEE, 2020.

[132] Anuradha M Annaswamy, Anubhav Guha, Yingnan Cui, Sunbochen Tang, Peter A Fisher, and Joseph E Gaudio. Integration of adaptive control and reinforcement learning for real-time control and learning. *IEEE Transactions on Automatic Control*, 2023.

[133] Yongshuai Wang, Chen Zheng, Mingwei Sun, Zengqiang Chen, and Qinglin Sun. Reinforcement-learning-aided adaptive control for autonomous driving with combined lateral and longitudinal dynamics. In *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 840–845. IEEE, 2023.

[134] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *CoRR*, abs/2109.00157, 2021.

[135] Pawel Wawrzynski. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5(2):91, 2015.

[136] Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[137] Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber.

Exploring parameter space in reinforcement learning. *Paladyn*, 1(1):14–24, 2010.

[138] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, 2018.

[139] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

[140] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[141] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7611–7622, 2019.

[142] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014.

[143] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.