ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240711

Adversarially Robust Neural Lyapunov Control

Li Wei[1], Yuankun Jiang[2], Chenglin Li[1,*], Wenrui Dai[2], Junni Zou[2] and Hongkai Xiong[1]

[1]Department of Electronic Engineering, Shanghai Jiao Tong University
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University

Abstract. State-of-the-art learning-based stability control methods for nonlinear robotic systems suffer from the issue of reality gap, which stems from discrepancy of the system dynamics between training and target (test) environments. To mitigate this gap, we propose an adversarially robust neural Lyapunov control (ARNLC) method to improve the robustness and generalization capabilities for Lyapunov theory-based stability control. Specifically, inspired by adversarial learning, we introduce an adversary to simulate the dynamics discrepancy, which is learned through deep reinforcement learning to generate the worst-case perturbations during the controller's training. By alternatively updating the controller to minimize the perturbed Lyapunov risk and the adversary to deviate the controller from its objective, the learned control policy enjoys a theoretical guarantee of stability. Empirical evaluations on five stability control tasks with the uniform and worst-case perturbations demonstrate that ARNLC not only accelerates the convergence to asymptotic stability, but can generalize better in the entire perturbation space.

1 Introduction

Designing a stable and robust controller to stabilize nonlinear dynamical systems has long been a challenge. Lyapunov stability theory performs a fairly significant role in the controller design for stability control of robotic systems [38, 32, 18, 25, 26]. However, many previous approaches are restricted to the polynomial approximation of system dynamics [15, 28], and suffer from sensitivity issues when searching for the Lyapunov functions [20]. Recently, by leveraging deep learning-based methods, some works have successfully incorporated the Lyapunov stability theory with the powerful expressiveness of neural networks and convenience of gradient descent for network learning [4, 1, 23, 6, 45, 41, 7]. One outstanding method among them is the neural Lyapunov control (NLC) [4], where both the Lyapunov function and controller policy are approximated by neural networks. In NLC, the networks are trained by minimizing a Lyapunov risk stemmed from the Lyapunov stability theorem. Nevertheless, most existing learning-based controllers are trained without any distinction between the training and test environments [5, 40]. Since the training simulator cannot perfectly model the target environment for testing, a reality gap will incur inevitably by such a modelling error (i.e., discrepancy of system dynamics), which degrades the performance of controller at the actual deployment. Hence, learning-based controller needs to consider the uncertainty of physical parameters (or external forces) that may cause the modelling error [17, 8, 11, 44]. Motivated by this, we focus in this paper on addressing the challenging problem of learning a controller to stabilize the nonlinear dynamical system in face of such a modelling error.

Over the years, several approaches have already been proposed to alleviate the controller's performance degradation incurred by modelling errors. The majority of them are built upon another splendid learning-based control method: deep reinforcement learning (RL) [34, 31]. These deep RL-based control methods treat the modeling error as an extra disturbance to the system [2], and have achieved a great success in controlling [29, 36, 42, 43, 22, 13, 14]. For example, in robust adversarial reinforcement learning (RARL), the policy learning is formulated as a zero-sum game between the controller and an adversary that generates disturbances to the system, where the learned controller is proved to have improved capability of robustness and generalization. Since RL methods train policies by maximizing the sum of expected rewards that the agent obtains during the interaction with environment, its performance depends greatly on the manually designed reward function while the learned policy is sensitive to the preset control interval [35, 27]. Hence, RL is prone to fail in the control tasks with a relatively small control interval, as will later be verified in our experiments. While our aim is to find the control policy that can enable a stable control, which is also robust to the choice of control intervals.

In this paper, we present a novel method that can automatically learn robust control policies with a provable guarantee of stability. Specifically, we formulate a perturbed Lyapunov risk for learning a controller in the dynamical system, which is imposed with the adversary's perturbations in a certain range. To train the controller policy to resist to the worst-case perturbations within that range, we formulate the learning of adversary as a Markov decision process (MDP), and train the adversary policy by proximal policy optimization (PPO). In the case of known system dynamics, the action space in the MDP can be the range of external forces or space of physical parameters, which causes the modelling error. More practically for the unknown dynamics, the original NLC no longer works since update of the networks is infeasible without prior knowledge of the system dynamics. We therefore train an environment model by sampling from the system, while the adversary's action is set as the offset to the output of this environment model. We further formulate an adversarially robust controller learning problem, which is approximately solved by alternatively updating the controller policy with Lyapunov methods and the adversary policy by PPO. Our contributions can be summarized as follows.

- We propose a perturbed Lyapunov risk for learning the control policy under perturbations.
- We formulate an optimization problem for adversarially robust controller learning, to learn a policy in face of the worst-case

* Corresponding Author. Email: lcl1985@sjtu.edu.cn.

perturbations that are imposed by the RL-trained adversary.
- We propose an adversarially robust neural Lyapunov control (ARNLC) approach to approximately solve this problem, and demonstrate its performance on several stability control tasks.

2 Related Work

Adversarial training. The idea of viewing the gap between training and test scenarios as an extra disturbance of the system in reinforcement learning was first proposed in [24], with the problem formulated as finding a min-max solution of the value function that takes the perturbations into account. Inspired by [24], [29] proposes the robust adversarial reinforcement learning (RARL), where an adversary is learned simultaneously to prevent the agent from accomplishing its goal, while the agent's policy and the adversary policy are trained alternately. [42] proposes robust reinforcement learning based on perturbations on state observations, introducing an adversary to apply disturbances on the state observations of the agent. [36] focuses on a scenario where the agent attempts to perform an action, which behaves differently from expected due to disturbances. All of the above literature mainly studies training the adversary for RL settings, while our focus in this work is on introducing adversarial training to the Lyapunov stability control.

Neural Lyapunov stability control. [4] proposes the neural Lyapunov control, which uses neural networks to learn both the control and Lyapunov functions for nonlinear dynamical systems based on the Lyapunov stability theory. [30] learns a control law that stabilizes an unknown nonlinear dynamic system. However, it needs to design a Lyapunov function manually. [6] also proposes an approach for learning the robust nonlinear controller based on the robust convex optimization and Lyapunov theory, achieving generalization beyond system parameters seen during the training process. However, this approach only considers the control-affine dynamical systems, not the more general nonlinear ones. In this work, we focus on improving the robustness and generalization for control policies of nonlinear dynamical systems.

Robust model predictive control (Robust MPC). Robust MPC is another research branch to deal with the uncertainty in physical parameters [33, 10, 12]. It looks for the optimal feedback law among all the feasible feedback laws within a given finite horizon, in terms of a given control performance criterion at every sampling instant [9]. However, it is usually restricted to the additive disturbances [19] and is computationally expensive [3].

3 Preliminaries and Background

We consider a continuous-time, time-invariant nonlinear dynamical system of the form:

$$\dot{x} = f(x, a), \tag{1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state and $a \in \mathcal{A} \subseteq \mathbb{R}^m$ is the control input, respectively, and $\dot{x}$ denotes the first-order time derivative of x. The system is feedback controlled by a policy function: $a = \pi(x)$. We aim to stabilize this system at an equilibrium point $x = 0 \in \mathcal{X}$, by finding a policy to build a closed-loop controlled dynamical system $\dot{x} = f_\pi(x)$ with $f_\pi(0) = 0$, such that the equilibrium point $x = 0$ achieves asymptotic stability as defined below.

Definition 1 (**Asymptotic stability in the sense of Lyapunov** [16])**.** *The equilibrium point $x = 0$ of f_π is stable in the sense of Lyapunov if $\forall \varepsilon > 0$, $\forall t_0 > 0$, there exists $\delta(\varepsilon) > 0$ such that if $\|x(t_0)\| \leq \delta(\varepsilon)$ then $\|x(t)\| \leq \varepsilon$ for all $t \geq t_0$. The equilibrium point $x = 0$ of f_π is asymptotically stable if it is stable and there exists a positive constant $c = c(t_0)$ such that $x(t) \to 0$ as $t \to \infty$, for all $\|x(t_0)\| \leq c$.*

We assume the system is locally Lipschitz continuous in x.

Assumption 1 (continuity)**.** *The dynamics $f(\cdot)$ in Equ. (1) is L_f Lipschitz continuous with respect to the 1-norm, whereby $f(\cdot)$ satisfies the inequality:*

$$\|f_{\pi_\mu(x)}(x) - f_{\pi_\mu(x)}(y)\|_1 \leq L_f \|x - y\|_1, \tag{2}$$

for all $x(t), y(t) \subseteq \mathbb{R}^n$, with a finite constant $L_f > 0$.

3.1 Stability Guarantee with Lyapunov Functions

Lyapunov stability theory provides an elegant way to guarantee the stability, as follows.

Theorem 1 (**Lyapunov stability theorem** [16])**.** *Suppose $f_\pi : \mathcal{X} \to \mathbb{R}^n$ is locally Lipschitz in $\mathcal{X} \subseteq \mathbb{R}^n$. For a continuous-time controlled dynamical system f_π, if there exists a continuous function $V : \mathcal{X} \to \mathbb{R}$ such that*

$$V(0) = 0; \text{ and } V(x) > 0, \forall x \in \mathcal{X} - \{0\}; \text{ and } \dot{V}(x) < 0; \tag{3}$$

then the system is asymptotically stable at $x = 0$, where V is called a Lyapunov function.

The time derivative of $V(x)$ can be derived as $\dot{V}(x) = \sum_{i=1}^n \frac{\partial V}{\partial x_i}\dot{x}_i = \sum_{i=1}^n \frac{\partial V}{\partial x_i}[f_\pi]_i(x)$, which depends on both $V(x)$ and the controlled dynamics f_π. Theorem 1 then states that the trajectories of the system's state will eventually reach the equilibrium $x = 0$, if we can design a control policy π such that the Lyapunov function $V(x)$ exists and satisfies the conditions in Eq. (3).

3.2 Neural Lyapunov Control

Neural Lyapunov control (NLC) [4] leverages neural networks to approximate both the control policy π and Lyapunov function $V_\theta(x)$, which are parameterized by θ^π and θ, respectively. The network parameters θ^π and θ are learned by minimizing the following Lyapunov risk:

$$\begin{aligned} L_\rho(\theta, \theta^\pi) = \mathbb{E}_{x\sim\rho(\mathcal{X})}\Big(& \max(0, -V_\theta(x)) \\ & + \max(0, \dot{V}_\theta(x)) + V_\theta^2(0)\Big), \end{aligned} \tag{4}$$

where x is a random variable following a uniformly random distribution ρ over the state space $\mathcal{X}$. In its physical meaning, this Lyapunov risk quantifies the degree of violation of the Lyapunov conditions in Eq. (3) over the state space $\mathcal{X}$, given a certain policy and Lyapunov function.

4 Adversarially Robust Neural Lyapunov Control

In this paper, we aim to narrow down the reality gap incurred by the modelling discrepancy in the system dynamics f between training and test environments, by learning a policy of controller μ that is better stabilizing the system (i.e., achieving the asymptotic stability faster) and more robust (i.e., resisting to variations of the system dynamics). Specifically, we consider modeling such a discrepancy by an adversary, with the system function given by:

$$\dot{x} = f(x, a^\mu, a^\nu), \tag{5}$$

where $a^\mu \in \mathcal{A}$ and $a^\nu \in \mathcal{A}_{adv}$ are the controller's action and adversary's action at time t following their policies π_μ and π_ν, respectively, while the rest notations follow the definition in Eq. (1). In the view of the controller, a^ν imposes a variation to the dynamics f, which can be rewritten as $\dot{x} = f_{\pi_\nu(x)}(x, a^\mu)$. Hence, introducing the adversary ν imposes a modelling error to Eq. (1) during training of the controller. A practical example is where the controller is applied to manipulate locomotion of a robot outdoor, while the adversary may be the weather that produces unpredictable wind or rain to disturb this controller. Note that the system dynamics in view of the controller reduces to Eq. (1) when $a^\nu = \pi_\nu(x) = 0$ for any time t. In this section, we propose the adversarially robust neural Lyapunov control (ARNLC) method to train a policy π_μ for the controller μ, such that the system governed by Eq. (5) is stabilized in face of the adversary ν.

4.1 Perturbed Controller Learning

In our adversarial control setting, both the controller μ and adversary ν observe the system state x, and then take actions $a^\mu \sim \pi_\mu(x)$ and $a^\nu \sim \pi_\nu(x)$. After that, the system evolves according to Eq. (5). Here, we utilize neural networks to learn the controller policy $\pi_\mu(x)$, adversary policy $\pi_\nu(x)$ and candidate Lyapunov function $V_\theta(x)$, as parameterized by θ^μ, θ^ν and θ, respectively. Our objective is to leverage the Lyapunov stability theory to find a controller policy π_μ that can achieve the stability of system in the presence of a certain adversary. Namely, the resulting closed-loop controlled dynamical system $\dot{x} = f_{\pi_\mu(x),\pi_\nu(x)}(x)$ is asymptotically stable at the equilibrium point $x = 0$. Motivated by this, our proposed ARNLC seeks to minimize the following perturbed Lyapunov risk w.r.t. θ and θ_μ, to update the controller policy together with the candidate Lyapunov function.

Definition 2 (**Perturbed Lyapunov risk for controller**). *We consider a candidate Lyapunov function V_θ for a continuous-time dynamical system in Eq. (5). In the presence of an adversary policy π_ν parameterized by θ_ν, the perturbed Lyapunov risk for the controller μ is defined by:*

$$L_\rho(\theta, \theta^\mu, \theta^\nu) = \mathbb{E}_{x\sim\rho(\mathcal{X})}\Big(\max(-V_\theta(x), 0) + \max(0, \dot{V}_\theta(x)) + V_\theta^2(0)\Big), \tag{6}$$

where $\rho(\mathcal{X})$ is the state distribution, and $\dot{V}_\theta(x) = \sum_{i=1}^n \frac{\partial V_\theta}{\partial x_i}[f_{\pi_\mu,\pi_\nu}]_i(x)$.

The learning of π_μ and V_θ can then be formulated as the following optimization problem:

$$\begin{aligned} \min_{\theta,\theta^\mu} \quad & L_\rho(\theta, \theta^\mu, \theta^\nu), \\ \text{s.t.} \quad & \dot{x} = f(x, a^\mu, a^\nu), a^\mu \sim \pi_\mu, a^\nu \sim \pi_\nu. \end{aligned} \tag{7}$$

This perturbed Lyapunov risk L_ρ differs from the conventional Lyapunov risk in that the time derivative $\dot{V}_\theta(x)$ now depends on f_{π_μ,π_ν} instead of f_π, which makes the closed-loop dynamical system variable for the controller. In practice, we use the following empirically perturbed Lyapunov risk, which is an unbiased estimator of Eq. (6):

$$L_{N,\rho}(\theta, \theta^\mu, \theta^\nu) = \frac{1}{N}\sum_{i=1}^N \Big(\max(-V_\theta(x_i), 0) + \max(0, \dot{V}_\theta(x_i)) + V_\theta^2(0)\Big). \tag{8}$$

However, this practical estimator cannot guarantee satisfaction of the conditions in Theorem 1 on the entire state space $\mathcal{X}$. We thus apply additionally a falsifier to constantly find the counter-examples that violate these conditions during the training process, which is a common strategy also used in NLC. Specifically, the falsifier finds counter-example states according to the following criterion:

$$V_\theta(x) \le 0 \vee \dot{V}_\theta(x) \ge 0,\ \forall x \in \mathcal{X} - \{0\}, \tag{9}$$

which specifies the negation of conditions in Eq. (3). During the training of π_μ and V_θ, the falsifier constantly finds counter-examples and adds them into the training dataset.

4.2 Adversary Learning

Compared with the conventional Lyapunov risk in Eq. (4), the perturbed counterpart L_ρ in Eq.(6) presents some new challenges to the learning of controller policy. Due to the perturbation from adversary, the dynamical system in view of the controller becomes variable, which prevents the learning of its policy π_μ from reaching the stability as will be shortly shown in experiments. Inspired by the idea of adversarial training, the proposed ARNLC leverages reinforcement learning method to train the adversary policy. The intuition behind this is that if we can train a controller under the worst-case perturbation (which degrades the performance of its policy to the most) in a certain range, the controller then obtains a conservative policy that is robust to any perturbation within that same range.

We formulate training of the adversary as a discrete-time Markov decision process (MDP) with a fixed control interval, defined as the tuple $(\mathcal{X}, \mathcal{A}_{adv}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where γ is the discount factor. The adversary agent observes the system state $x \in \mathcal{X}$ at each time step and takes action $a^\nu \in \mathcal{A}_{adv}$, while the controller acts $a^\mu \in \mathcal{A}$. The system then evolves to the next state x' according to transition probability $\mathcal{P}(\cdot|x, a^\mu, a^\nu)$ between time steps, and the adversary agent receives reward $r(x, a^\mu, a^\nu)$.

Adversary's action design. Depending on whether or not the system dynamics f can be accessed, ARNLC applies different action design for the adversary agent. *i*) For **known dynamics** f, the action space $\mathcal{A}_{adv}$ of adversary can be range of the external disturbing force or space of the environment parameters, which changes the system dynamics in view of the controller. The adversary action can then be directly imposed on the system, which simulates the external force (e.g., strong wind) and change of environment parameters (e.g., friction coefficient) that the controller may encounter in the test environment. *ii*) For **unknown dynamics** f, NLC is unable to back propagate the gradients to update π_μ and V_θ, since f is required to compute L_ρ. Alternatively, we use supervised learning to train an environment model M_η that is approximated by the neural network for the unknown dynamical system. Then, π_μ and V_θ can be updated by the gradients of M_η. Since coefficients of M_η do not have a clear physical meaning of the environment (which are weights and biases of the network), we define actions of the adversary as the additive error to the output of M_η. However, perturbations imposed by the adversary's action may lead to an unstable training, or even the nonexistence of an asymptotically stable equilibrium. Hence, we limit the adversary's action to a certain range, which can be tuned practically to balance the stable training and adversary learning.

Adversary's reward design. Adversary should be assigned a higher reward if it leads the system to an unstable state at each time step, which is contrary to the controller's goal. For example, in the task where we aim to design a controller to swing the pendulum into an upright position, the reward of the adversary can be set as the

Table 1. Types of perturbations for each task

Task	Perturbation Type
Pendulum	Mass of Ball, Length of Pole, Friction Coefficient, Gravity
Cart Pole	Mass of Cart, Length of Pole, Mass of Pole, Gravity
Car Trajectory Tracking	Velocity of Car, Radius of Path
2-link Pendulum	Pole1 Length, Pole1 Position of the Center of Mass

square of the normalized angle between the pendulum and the vertical direction. The reward functions for training the adversary in all the tasks used in the paper are shown in Table A-3 in Appendix [39]. In general, the controller's action that tends to stabilize the system will decrease the reward for the adversary, while the adversary policy that achieves a higher reward will prevent the controller from minimizing L_ρ.

State transition. The state transition is deterministic by the system dynamics f. Given a fixed control interval Δt, it can be derived as $x_{t+1} = x_t + f(x_t, a_t^\mu, a_t^\nu)\Delta t$.

Given the controller policy π_μ, the goal of RL is to find the optimal adversary policy π_ν^* that maximizes the following objective function:

$$\pi_\nu^* = arg\max_{\pi_\nu} \mathbb{E}_{a_h^\mu \sim \pi_\mu, a_h^\nu \sim \pi_\nu, x_h \sim \mathcal{P}} \left[\sum_{h=0}^{\infty} \gamma^h r(x_h, a_h^\mu, a_h^\nu) \right], \tag{10}$$

which are the expected cumulative discounted reward starting from initial state x_0, following a certain distribution over the whole state space. Provided with an appropriate adversary's reward design, π_ν^* can produce the worst-case perturbation sequence, which adversarially destabilizes the system to the most extent.

4.3 Adversarially Robust Controller Learning

Given an adversary policy π_ν^* trained by RL that performs the worst-case perturbation to the controller, we formulate the adversarially robust controller learning problem as:

$$\begin{aligned} \min_{\theta,\theta^\mu} \quad & L_\rho(\theta, \theta^\mu, \theta^\nu), \\ \text{s.t.} \quad & \dot{x} = f(t, x, a^\mu, a^\nu), a^\mu \sim \pi_\mu, \\ & a^\nu \sim \pi_\nu^* = arg\max_{\pi_\nu} V^{\pi_\mu}(\pi_\nu). \end{aligned} \tag{11}$$

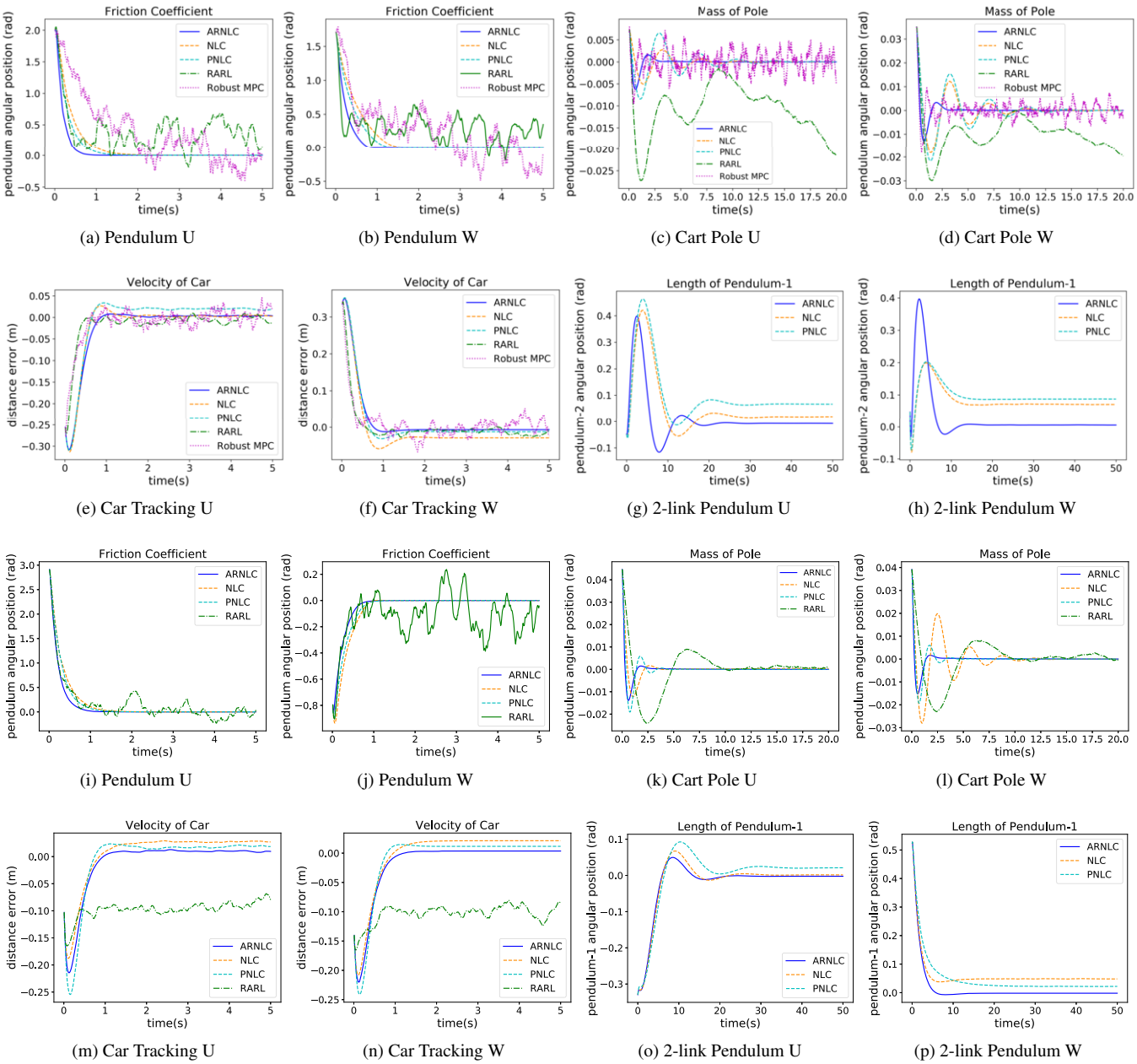
Note that formulations of the objective functions for neural Lyapunov-controller and RL-adversary are totally different, hence the adversarial learning problem here cannot be formulated as a two-player zero-sum game. The proposed ARNLC in Algorithm 1 uses an alternating procedure to solve this problem approximately, where we summarize it for the case of unknown system dynamics f. **Training environment model:** at each iteration e, we sample M_1 transitions in the environment with a random policy and update the environment model by minimizing the error between its prediction $\mathcal{M}_\eta(x, a)$ and the next state x' on the transitions. **Training controller's and adversary's policies:** at each outer iteration i, we perform a two-stage optimization. *i*) We train the controller policy and Lyapunov function while the adversary policy is fixed based on the Lyapunov theory. We initialize a state dataset $\mathcal{S}$ by randomly sampling from $\mathcal{X}$. At each inner iteration j_μ, we construct $\{x_k, a_k^\mu, a_k^\nu, x_k'\}_k$ on the state subset of size M_3 from $\mathcal{S}$, and update π_μ and V_θ by performing stochastic gradient descent (SGD) w.r.t. Eq. (7). *ii*) We train the adversary policy while the controller policy is fixed. At each inner iteration j_ν, we generate transitions on learned environment model $\mathcal{M}_\eta$ with the controller's and adversary's policies. We then perform the policy optimization method from RL to update π_ν on these generated transitions. For known system dynamics, learning of the environment model at Lines 2-5 in Algorithm 1 is not required, and the transition generation in the two stages follows the known system dynamics f. Due to space limit, the proposed ARNLC for known system dynamics is provided in Appendix [39].

Algorithm 1 Adversarially robust neural Lyapunov control (ARNLC) for unknown dynamics

Require: unknown environment $\mathcal{M}$ and state space $\mathcal{X}$
Ensure: learned policies π_μ and π_ν
1: **Initialize:** η for environment model $\mathcal{M}_\eta$, θ^μ for controller μ, θ^ν for adversary ν, θ for Lyapunov function V_θ, control interval Δt, uniformly random policy π_r
2: **for** $e = 1, 2, \dots N_e$ **do** ▷ train an environment model for the system
3: Sample transitions $\{x_i, a_i, x_i'\}_{i=1}^{M_1}$ from $\mathcal{M}$ using π_r with Δt
4: Update $\mathcal{M}_\eta$ by minimizing $\frac{1}{M_1}\sum_{i=1}^{M_1} |\mathcal{M}_\eta(x_i, a_i) - x_i'|^2$ w.r.t. η
5: **end for**
6: **for** $i = 1, 2, \dots N_{iter}$ **do** ▷ train controller
7: Randomly sample M_2 states $\mathcal{S} = \{x_k\}_{k=1}^{M_2}$ from state space $\mathcal{X}$
8: **for** $j_\mu = 1, 2, \dots N_\mu$ **do**
9: $a_k^\mu = \pi_\mu(x_k)$, $a_k^\nu = \pi_\nu(x_k)$ and $x_k' = M_\eta(x_k, a_k^\mu) + a_k^\nu$ on $\{x_k\}_{k=1}^{M_3}$ sampled from $\mathcal{S}$
10: $\pi_\mu, V_\theta \leftarrow \min_{\theta,\theta^\mu} L_\rho(\theta, \theta^\mu, \theta^\nu)$ on $\{x_k, a_k^\mu, a_k^\nu, x_k'\}_{k=1}^{M_3}$ ▷ use SGD to update
11: Find counter-example set Ω of size M_4 following criterion in Eq. (9) and $\mathcal{S} \leftarrow \mathcal{S} \cup \Omega$
12: **end for**
13: **for** $j_\nu = 1, 2, \dots N_\nu$ **do** ▷ train adversary
14: $\{x_h, a_h^\mu, a_h^\nu, r_h\}_{h=1}^{N_{traj}} \leftarrow$ generate$(\mathcal{M}_\eta, \pi_\mu, \pi_\nu)$
15: $\pi_\nu \leftarrow$ policyOptimize $\left(\{x_h, a_h^\mu, a_h^\nu, r_h\}_{h=1}^{N_{traj}}, \pi_\nu\right)$
16: **end for**
17: **end for**

5 Experiment

We evaluate our proposed ARNLC algorithm on several control tasks, where the system dynamics are variable by external forces or perturbations on environment parameters. We compare our ARNLC with NLC [4], PNLC (perturbed NLC) in Section 4.1, RARL [29] and Robust MPC [19, 21]. We use proximal policy optimization (PPO) [31] as our baseline RL algorithm for the adversary training at Line 15 in Algorithm 1. In NLC and PNLC, we build neural networks for the controller policy and Lyapunov function, which are updated by minimizing the Lyapunov risk in Eq. (4) and Perturbed Lyapunov risk in Eq. (6), respectively. A uniformly random adversary policy is applied to impose perturbations in PNLC. In ARNLC and RARL, we build a neural network for the adversary policy and update it with PPO. In ARNLC, the controller policy is optimized together with the empirically perturbed Lyapunov function in Eq. (6), where the



(a) Pendulum U (b) Pendulum W (c) Cart Pole U (d) Cart Pole W

(e) Car Tracking U (f) Car Tracking W (g) 2-link Pendulum U (h) 2-link Pendulum W

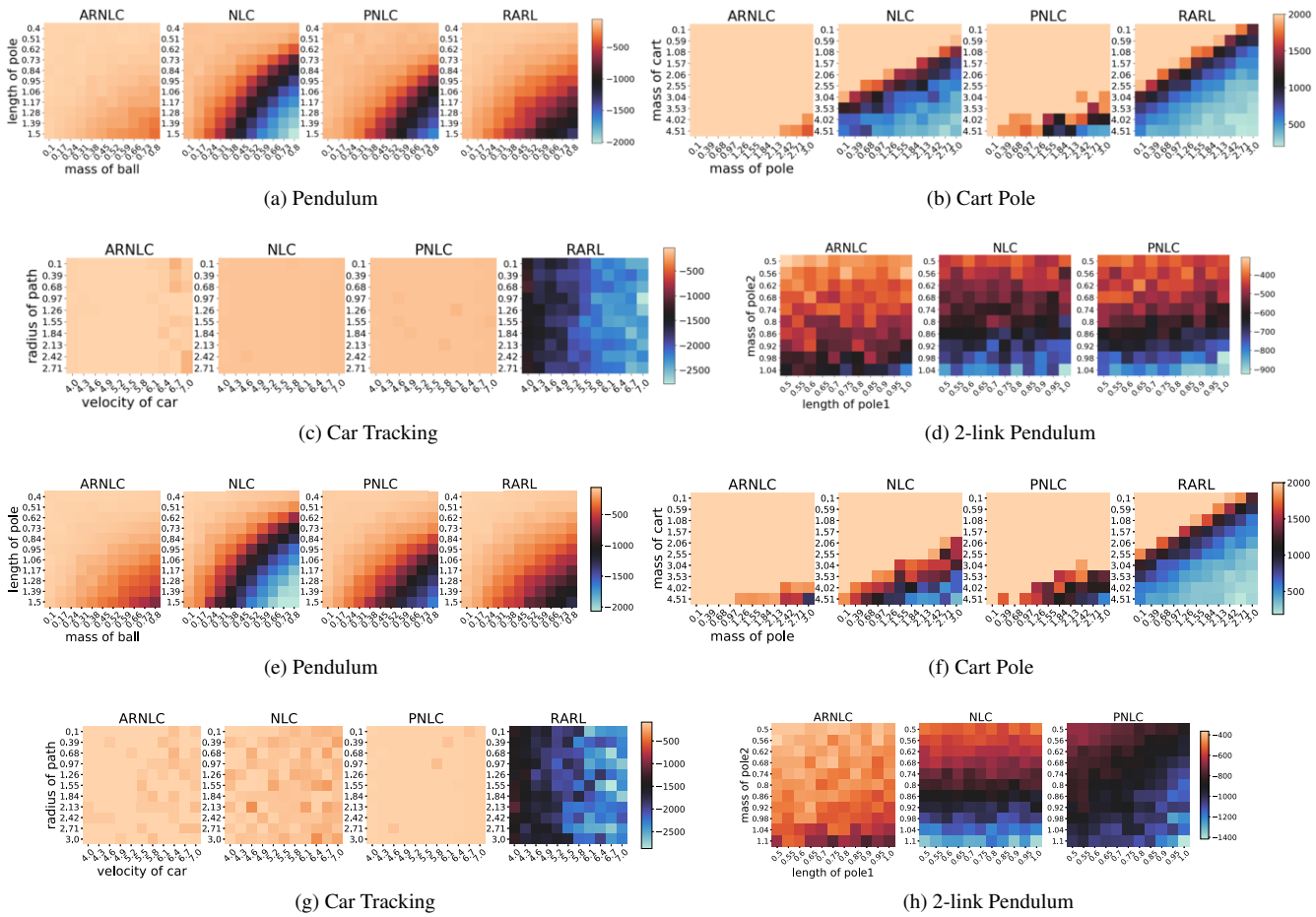
(i) Pendulum U (j) Pendulum W (k) Cart Pole U (l) Cart Pole W

(m) Car Tracking U (n) Car Tracking W (o) 2-link Pendulum U (p) 2-link Pendulum W

Figure 1. Control curves of Pendulum, Cart Pole, Car Tracking and 2-link Pendulum with different perturbation types under uniform perturbations (U) and learned adversary's worst-case (W) perturbations in testing. (a)-(h): known system dynamics; (i)-(p): unknown system dynamics.

controller policy is learned by PPO with the negative adversary reward. Since the RL-based adversary and discrete-time predictor of environment model are learned in ARNLC, we compute the difference of Lyapunov function $V(x_{t+\Delta t}) - V(x_t)$ instead of the time derivative $\dot{V}(x_t)$ (see Appendix [39] for a detailed explanation). In robust MPC, a bounded perturbation is added to the system dynamics where the controller takes actions by searching the optimal feedback among all feasible horizons generated by the perturbed function. In our experiments, we find that the computation time for Robust MPC at each control step may exceed the control interval, which would lead to the controller failure in practice. But we simply neglect this and consider its simulation performance. For detail of the experimental settings, please refer to Appendix [39]. Our experiments are designed to answer the following questions.

- Can our proposed ARNLC still achieve asymptotic stability in face of the worst-case perturbations? Will ARNLC reach the stability faster than the other baseline algorithms?
- How will the controller's performance of ARNLC degrade in the entire perturbation space?
- Will ARNLC suffer from the issues of RL methods [35, 27], i.e., being sensitive to control intervals?



(a) Pendulum (b) Cart Pole

(c) Car Tracking (d) 2-link Pendulum

(e) Pendulum (f) Cart Pole

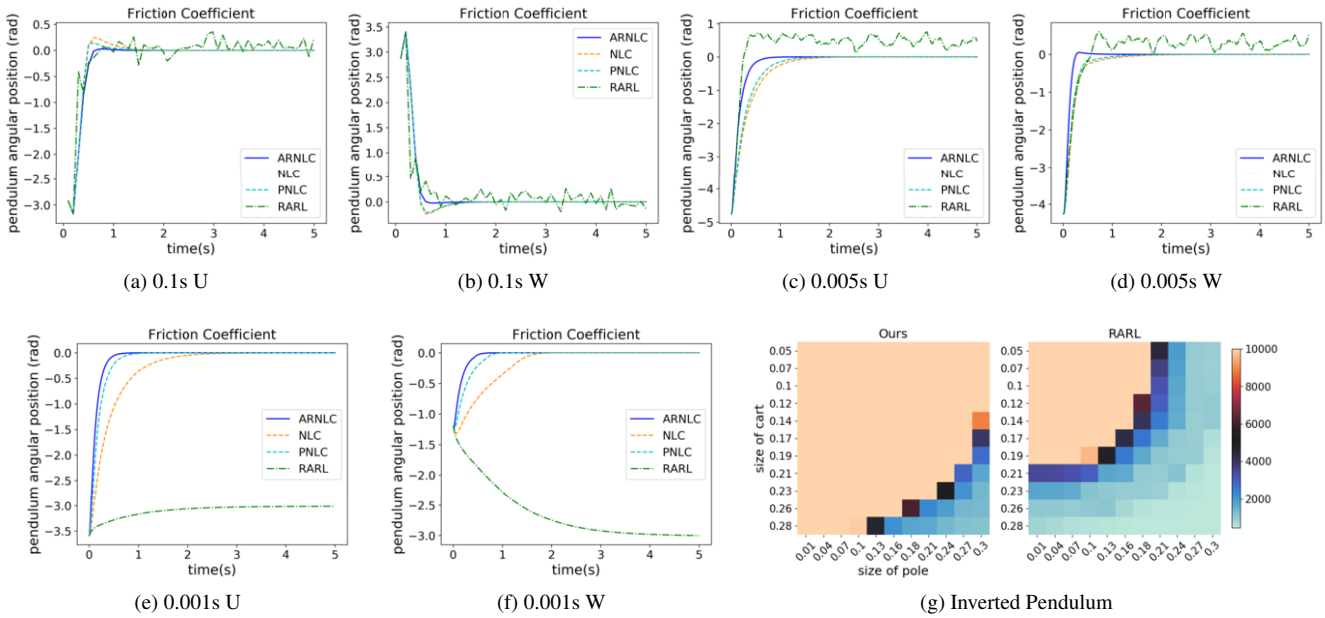
(g) Car Tracking (h) 2-link Pendulum

Figure 2. Heatmap of averaged cumulative negative adversary's reward with known and unkown system dynamics. In (d) and (h), RARL fails to converge. (a)-(d): known system dynamics; (e)-(h): unknown system dynamics.

5.1 Control of Perturbed Nonlinear Systems

We consider four balancing tasks. 1) **Pendulum**: balance a pendulum (one end attached to the ground by a joint) by applying a force to a ball at the other end. The system has two state variables, angular position φ and angular velocity $\dot{\varphi}$ of pendulum, and one control input a^{μ} on the ball; 2) **Cart Pole**: control a cart (attached to a pole by a joint) by applying a force to prevent the pole from falling. State variables of the system are cart position x, cart velocity $\dot{x}$, pendulum angular position φ and pendulum angular velocity $\dot{\varphi}$. Control input is force a^{μ} applied to the cart; 3) **Car Trajectory Tracking**: control a car to follow a target circular path. The system has two state variables: the distance error x_e and angle error φ_e between current car position and the target path. The control input is force a^{μ} on the car; 4) **2-link Pendulum**: control a two-joint pendulum system (two pendulums are linked by a joint and one end is linked to ground by another joint) to keep both pendulums upright. The system has four state variables: the angular position φ_1 and angular velocity $\dot{\varphi}_1$ of the first pendulum, the angular position φ_2 and angular velocity $\dot{\varphi}_2$ of the second pendulum. Two control inputs are forces a_1^{μ} and a_2^{μ} on the two joints. We evaluate different comparison algorithms on these tasks with perturbed environment parameters as shown in Table 1, under both known and unknown system dynamics settings. For example, in the pendulum task, the friction coefficient can be changed at each time step. We set two test scenarios: *i*) perturbations are randomly generated w.r.t. a uniform distribution at each control step, which are larger than training ones; *ii*) perturbations are taken w.r.t. the trained adversary policy of ARNLC.

We run the training process of learning-based algorithms, i.e., ARNLC, NLC, PNLC and RARL on these tasks until the convergence. Here, we slightly modify and improve the original NLC and PNLC to make them compatible with the setting of unknown system dynamics. We then deploy the trained policies and robust MPC in the two test scenarios. **For known system dynamics**, control curves under uniform (U) perturbations are shown in Figs. 1(a), 1(c), 1(e) and 1(g), while control curves under worst-case (W) perturbations learned by the adversary are illustrated in Figs. 1(b), 1(d), 1(f) and 1(h). **For unknown system dynamics**, control curves under uniform (U) perturbations are shown in Figs. 1(i), 1(k), 1(m) and 1(o), while control curves under worst-case (W) perturbations learned by the adversary are illustrated in Figs. 1(j), 1(l), 1(n) and 1(p). The initial state is separately and randomly chosen under uniform (U) and worst-case (W) perturbations, hence they are different. Since we aim to study the comparison of all the algorithms separately under U and W, we keep the initial state the same for the algorithms inside U and W to remove the impact of different initial state. The horizontal axis is the control time, while the vertical axis is the system state. Here we set the fixed control interval to 0.01s, and state zero as the equilibrium point. We observe that by incorporating an RL-based ad-



(a) 0.1s U (b) 0.1s W (c) 0.005s U (d) 0.005s W

(e) 0.001s U (f) 0.001s W (g) Inverted Pendulum

Figure 3. (a)-(f) Control curves of Pendulum with control interval set to 0.1s, 0.005s and 0.001s, respectively; (g) heatmap of averaged cumulative negative adversary's reward with unknown system dynamics.

versary during training, our ARNLC can achieve asymptotic stability under both test scenarios in all the tasks, while it reaches the stability the fastest compared to the other baselines under both conditions of known and unknown system dynamics. Though NLC reaches the stability in some tasks, it fails to reach the equilibrium point in Car Tracking W, 2-link Pendulum U and W with known system dynamics and Car Tracking U and W, 2-link Pendulum W with unknown system dynamics. PNLC trained under uniform sampled perturbations outperforms NLC in some tasks (e.g., Pendulum and Cart Pole), but is worse in Cart Tracking U and 2-link Pendulum W and U with known system dynamics and 2-link Pendulum U with unknown system dynamics. RARL and robust MPC fail to reach the stability in the 2-link Pendulum task.

5.2 *Generalization in Perturbation Space*

We further evaluate the generalization capability of trained policies of learning-based algorithms in the entire perturbation space. We exclude the evaluation of robust MPC here, since it requires to know system dynamics under each perturbation setting, which is an unfair comparison. Besides, we additionally evaluate ARNLC and RARL for unknown system dynamics in the **Inverted Pendulum** task provided by MuJoCo [37], which controls a cart (attached to a pendulum) to balance the whole system and keep the pendulum upright. While NLC and PNLC are not compared, since they require known system dynamics during training and cannot be trained in the Inverted Pendulum task. We use the cumulative negative reward of adversary to evaluate the performance of controller policy in the environment with a certain perturbation, where a higher negative reward indicates the better performance of controller.

The performance heatmaps of these five tasks achieved by different algorithms are shown and compared in Fig. 2 with both known and unknown system dynamics, and 3(g) with unknown system dynamics, where the performance is averaged on ten equal-length runs. We observe that ARNLC achieves the best generalization performance under both conditions of known and unknown system dynamics except for Car Tracking with unknown system dynamics. PNLC generalizes better than NLC in Pendulum and CartPole with known system dynamics, while showing similar or even worse performance in other tasks. RARL presents the worst performance in all the tasks except for Pendulum.

5.3 *Impact of Control Intervals*

Eventually, we evaluate the impact of different control intervals to our ARNLC, which are set to 0.01s, 0.1s, 0.005s and 0.001s, respectively. The resulting control curves obtained for Pendulum are shown in Figs. 1(a)-1(b) and 3(a)-3(f). We observe that ARNLC and other Lyapunov-based baselines can achieve asymptotic stability with all different control intervals, while RARL is sensitive to the change of control intervals as also verified in [35] and fails to reach the equilibrium state. Note that here we report the most important results in the main text, please also see Appendix [39] for additional results.

6 Conclusions and Future Work

We have proposed ARNLC to improve the robustness and generalization in stability control tasks for nonlinear dynamical systems. Specifically, we formulated a perturbed Lyapunov risk stemmed from Lyapunov theorem to jointly update the controller and candidate Lyapunov function under perturbations generated by an adversary during training, where the adversary was trained by RL method to destabilize the system. We adopted an alternative training procedure to update the controller and adversary. We have empirically evaluated ARNLC in several stability control tasks, demonstrating its robustness under different perturbations and better generalization than state of the art approaches in entire perturbation space. An exciting future research direction could be to extend our ARNLC to non-stability control tasks for the dynamical systems that are not required to achieve an equilibrium point, where the reward function for the adversary can alternatively be designed based on the imitation error, like in the imitation learning. We hope to revisit this in our future works.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62125109, Grant 61931023, Grant 61932022, Grant 62371288, Grant 62320106003, Grant 62301299, Grant T2122024, Grant 62120106007.

References

[1] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[2] T. Başar and P. Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[3] A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in Identification and Control*, volume 245 of *Lecture Notes in Control and Information Sciences*, pages 207–226. Springer, 1998.

[4] Y. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. In *Advances in Neural Information Processing Systems 32*, pages 3240–3249, 2019.

[5] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[6] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe nonlinear control using robust neural Lyapunov-barrier functions. In *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1724–1735. PMLR, 2021.

[7] V. Debauche, A. Edwards, R. M. Jungers, and A. Abate. Stability analysis of switched linear systems with neural Lyapunov functions. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, pages 21010–21018. AAAI Press, 2024.

[8] K. Garg and D. Panagou. Robust control barrier and control Lyapunov functions with fixed-time convergence guarantees. In *2021 American Control Conference, ACC*, pages 2292–2297. IEEE, 2021.

[9] B. Houska and M. E. Villanueva. *Robust Optimization for MPC*, pages 413–443. Springer International Publishing, Cham, 2019.

[10] J. Hu and B. Ding. Output feedback robust MPC for linear systems with norm-bounded model uncertainty and disturbance. *Automatica*, 108, 2019.

[11] S. Islam, P. X. Liu, and A. El-Saddik. Robust control of four-rotor unmanned aerial vehicle with disturbance uncertainty. *IEEE Transactions on Industrial Electronics*, 62(3):1563–1571, 2015.

[12] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 66(2): 794–801, 2021.

[13] E. Korkmaz. Adversarial robust deep reinforcement learning requires redefining robustness. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 8369–8377. AAAI Press, 2023.

[14] E. Korkmaz and J. Brown-Cohen. Detecting adversarial directions in deep reinforcement learning to make robust decisions. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17534–17543. PMLR, 2023.

[15] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. John Wiley & Sons, Inc., USA, 1972. ISBN 0471511102.

[16] E. Lavretsky and K. A. Wise. Robust adaptive control. In *Robust and adaptive control*, pages 317–353. Springer, 2013.

[17] C. Liu, J. Yang, K. An, M. Liu, and Q. Chen. Robust control of semi-passive biped dynamic locomotion based on a discrete control Lyapunov function. *Robotica*, 38(8):1345–1358, 2020.

[18] L. Liu, Y.-J. Liu, A. Chen, S. Tong, and C. Chen. Integral barrier Lyapunov function-based adaptive control for switched nonlinear systems. *Science China Information Sciences*, 63(3):1–14, 2020.

[19] J. Löfberg. Approximations of closed-loop minimax MPC. In *42nd IEEE Conference on Decision and Control*, pages 1438–1442. IEEE, 2003.

[20] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Trans. Autom. Control.*, 54(5):1007–1011, 2009.

[21] J. Löfberg. Automatic robust convex programming. *Optimization Methods and Software*, 27(1):115–129, 2012.

[22] D. J. Mankowitz, N. Levine, R. Jeong, A. Abdolmaleki, J. T. Springenberg, Y. Shi, J. Kay, T. Hester, T. A. Mann, and M. A. Riedmiller. Robust reinforcement learning for continuous control with model misspecification. In *International Conference on Learning Representations*, 2020.

[23] A. Mehrjou, M. Ghavamzadeh, and B. Schölkopf. Neural Lyapunov redesign. In *Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 459–470. PMLR, 2021.

[24] J. Morimoto and K. Doya. Robust reinforcement learning. *Neural Computation*, 17(2):335–359, 2005.

[25] A. Norouzi, A. Barari, and H. Adibi-Asl. Stability control of an autonomous vehicle in overtaking manoeuvre using wheel slip control. *International Journal of Intelligent Transportation Systems Research*, 18(2):320–330, 2020.

[26] A. K. Pal, S. Kamal, S. K. Nagar, B. Bandyopadhyay, and L. Fridman. Design of controllers with arbitrary convergence time. *Automatica*, 112: 108710, 2020.

[27] S. Park, J. Kim, and G. Kim. Time discretization-invariant safe action repetition for policy gradient methods. In *Advances in Neural Information Processing Systems 34*, pages 267–279, 2021.

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

[29] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[30] P. Saha, M. Egerstedt, and S. Mukhopadhyay. Neural identification for control. *IEEE Robotics and Automation Letters*, 6(3):4648–4655, 2021.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[32] A. Sharma and N. Kumar. Lyapunov stability theory based non linear controller design for a standalone pv system. In *2020 IEEE International Conference for Innovation in Technology (INOCON)*, pages 1–7. IEEE, 2020.

[33] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. Robust MPC for tracking constrained unicycle robots with additive disturbances. *Automatica*, 90:172–184, 2018.

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

[35] C. Tallec, L. Blier, and Y. Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.

[36] C. Tessler, Y. Efroni, and S. Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[37] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[38] N. Uddin, H. G. Harno, and R. A. Sasongko. Altitude control system design of bicopter using Lyapunov stability approach. In *2021 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–6. IEEE, 2021.

[39] L. Wei, Y. Jiang, C. Li, W. Dai, J. Zou, and H. Xiong. Appendix of Adversarially Robust Neural Lyapunov Control. Zenodo, Aug. 2024. doi: 10.5281/zenodo.13335611. URL https://doi.org/10.5281/zenodo.13335611.

[40] S. Witty, J. K. Lee, E. Tosch, A. Atrey, K. Clary, M. L. Littman, and D. Jensen. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2(4):e45, 2021.

[41] J. Wu, A. Clark, Y. Kantaros, and Y. Vorobeychik. Neural Lyapunov control for discrete-time systems. In *Advances in Neural Information Processing Systems 36*, 2023.

[42] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. S. Boning, and C. Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems 33*, 2020.

[43] H. Zhang, H. Chen, D. S. Boning, and C. Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.

[44] P. Zhao, Y. Mao, C. Tao, N. Hovakimyan, and X. Wang. Adaptive robust quadratic programs using control Lyapunov and barrier functions. In *59th IEEE Conference on Decision and Control, CDC*, pages 3353–3358. IEEE, 2020.

[45] R. Zhou, T. Quartz, H. D. Sterck, and J. Liu. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems 35*, 2022.