Robust Adversarial Reinforcement Learning with Dissipation Inequation Constraint

Peng Zhai[1,4], **Jie Luo**[1,5], **Zhiyan Dong**[1,2,5*], **Lihua Zhang**[1,3,4*], **Shunli Wang**[1,3], **Dingkang Yang**[1,4]

[1] Academy for Engineering and Technology, Fudan University
[2] Ji Hua Laboratory, Foshan, China
[3] Engineering Research Center of AI and Robotics, Ministry of Education, Shanghai 200433, China
[4] Jilin Provincial Key Laboratory of Intelligence Science and Engineering, Changchun, China
[5] Engineering Research Center of AI and Robotics, Shanghai, China
{pzhai18, 19210860032, dongzhiyan, lihuazhang, slwang19, dkyang20}@fudan.edu.cn

Abstract

Robust adversarial reinforcement learning is an effective method to train agents to manage uncertain disturbance and modeling errors in real environments. However, for systems that are sensitive to disturbances or those that are difficult to stabilize, it is easier to learn a powerful adversary than establish a stable control policy. An improper strong adversary can destabilize the system, introduce biases in the sampling process, make the learning process unstable, and even reduce the robustness of the policy. In this study, we consider the problem of ensuring system stability during training in the adversarial reinforcement learning architecture. The dissipative principle of robust H_∞ control is extended to the Markov Decision Process, and robust stability constraints are obtained based on L_2 gain performance in the reinforcement learning system. Thus, we propose a dissipation-inequation-constraint-based adversarial reinforcement learning architecture. This architecture ensures the stability of the system during training by imposing constraints on the normal and adversarial agents. Theoretically, this architecture can be applied to a large family of deep reinforcement learning algorithms. Results of experiments in MuJoCo and GymFc environments show that our architecture effectively improves the robustness of the controller against environmental changes and adapts to more powerful adversaries. Results of the flight experiments on a real quadcopter indicate that our method can directly deploy the policy trained in the simulation environment to the real environment, and our controller outperforms the PID controller based on hardware-in-the-loop. Both our theoretical and empirical results provide new and critical outlooks on the adversarial reinforcement learning architecture from a rigorous robust control perspective.

1 Introduction

Deep reinforcement learning (DRL) has become a popular method for training continuous controllers. It is widely used in the fields of robotic control and navigation (Duan et al. 2016; Lee et al. 2020; Hodge, Hawkins, and Alexander 2021). Although the performance of DRL is better than that of the traditional methods in simulation, realistic application examples are rare (Zhao, Queralta, and Westerlund 2020). There are available methods for direct operation in actual physical tasks to collect data (Haarnoja et al. 2018; Hwangbo et al. 2019), the policy is unstable in the initial training period, and repeated training may result in aging and failure of the actuator (Li et al. 2020). Another cheaper methods are to deploy the policies trained in simulation directly in the real environment. However, the differences (such as modeling error and disturbance) between the simulation and the real environment reduce the performance of the policies (Christiano et al. 2016). To this end, learning policies that are robust to environmental change, mismatched configurations, and mismatched control actions are becoming increasingly more preferable for sim-to-real tasks (Kamalaruban et al. 2020).

One effective method to learn robustness is domain randomization (Peng et al. 2018; Tobin et al. 2017) whereby a professionally knowledgeable designer determines the uncertain model components in the task. Thereafter, a set of training environments is constructed, and the uncertain components are randomly assigned to ensure the average robustness of the agent assigned to the set. Nevertheless, it requires significantly designers' experience in the test domain (Vinitsky et al. 2020); the uncertainty in the training environment will also lead to the instability of policy learning. Another method to learn robustness that is easier to automate is to model environmental differences as adversarial disturbances (Ilahi et al. 2020). Through the alternate update of the normal and adversarial agents, a two-player-zero-sum game is constructed. This makes the normal agent robust to the disturbance of the adversary. The method does not require much knowledge of domain. Nonetheless, it increases the difficulty of the training domain. This may make the training unstable, causing the policy to fall into a local suboptimal solution, or resulting in a non-convergence (Tessler, Efroni, and Mannor 2019). The instability during training is because adversarial-policy acquisition is significantly faster than stable-control-policy acquisition for an underactuated or unstable system. For example, in an inverted double pendulum system, a disturbance acting on the pendulum arm can easily make the system unstable (Mackenroth 2008). If the adversary can always destroy the stability of the system during learning, it will lead to task failure. In addition, the

*Corresponding author

cost of optimization cannot be appropriately defined, halting the learning process (Zhang, Hu, and Basar 2020). Such issues will be aggravated in sample-based learning, because the stochasticity in the data will bring more instability.

In this study, we consider certain constraints on the normal and adversarial agents in the adversarial architecture to ensure the stability of the system during training. First, we extend the dissipative inequality of the H_∞ theory to the reinforcement learning(RL) system and derive the robust stability constraint of RL. Second, we propose a new robust stability RL architecture, which we call dissipation-inequation-constraint-based adversarial reinforcement learning (DICARL). Our architecture consists of three modules: a normal agent, an adversarial agent, and a data-driven Lyapunov network. The Lyapunov network is used to constrain the robust stability of the two agents during training. The robust stability is measured by L_2 gain. By introducing constraints into the objective function of the policies update, the direction of the adversarial agent's gradient that reduces the performance of the L_2 gain is penalized, and the direction of the normal agent's gradient is updated to simultaneously increase the L_2 gain and performance metrics (reward function).

Extensive experiments are conducted in MuJoCo and GymFc environments. The GymFc environment is an attitude flight controller training simulation platform. We demonstrate the performance of the trained policies both in simulation and on a real quadrotor. The results on MuJoCo show that our algorithm is more robust to the changes in the environmental parameters than that of the baselines. We empirically demonstrate that a stronger adversary will reduce the performance of the adversarial architecture during training, and our method effectively alleviates this problem. The results of the GymFc simulation show that our algorithm is more robust to an actuator disturbance. Experiments on a real quadrotor show that our trained neural network controller is better than the hardware-in-the-loop (HIL) based PID controller, demonstrating the potential of our method in sim-to-real tasks.

Related Work. The adversarial architecture is originated from the H_∞ control theory, which makes the controller robust against model uncertainty by introducing adversarial disturbances into the controlled system (Modares, Lewis, and Sistani 2014; Wu and Luo 2012) or real robot system (Pinto, Davidson, and Gupta 2017). (Morimoto and Doya 2005) converted the H_∞ problem into a differential game and solved the minimax value function of the normal and adversary agents to update the policies. It was the first study wherein the adversarial architecture was introduced into RL theory. This idea of minimax was then extended in the robust adversarial reinforcement learning (RARL) architecture (Pinto et al. 2017), with considerable empirical success, subsequently adopted and improved in (Kamalaruban et al. 2020; Gleave et al. 2019). However, as discussed earlier, the adversarial architecture sometimes fails to improve the robustness of the normal agent because a strong adversary affects the stability of the training process (Zhang, Hu, and Basar 2020). To reduce the influence of the adversary on the training stability, the main methods include manual adjustment of frequency (Pan et al. 2019) or magnitude (Tessler, Efroni, and Mannor 2019) of the interaction between the adversary and the environment or reducing the update frequency of the adversarial policy (Gu, Jia, and Choset 2019). These methods ensure the stability of training to a certain extent, but it is difficult to achieve an effective trade-off between robustness and training stability.

More closely related work from a theoretical perspective is that of (Han et al. 2019) who introduces and extends the idea of Lyapunov stability and H_∞ control to design policies with robustness guarantee. This work introduces a Lyapunov function as a critic to provide the policy gradient, simultaneously find the Lyapunov function and policy that can guarantee the robust stability of the closed-loop system. Our work differs from this work in a number of respects: (1) Their critic is represented by a Lyapunov function which tends to bring the system state to a balance point, it is difficult to apply this method to robot locomotion tasks, or it is necessary to design a specific reward function. Our method adopts the auxiliary optimization strategy and retains the reward-driven mechanism to be applied in more general environments. (2) The Lagrangian term from (Han et al. 2019) is only used for the constraints of the normal agent, whereas the Lagrangian term in our formulation constrains both the normal and the adversarial agents simultaneously to ensure the stability of the system during the training process.

2 Preliminaries

2.1 Adversarial Reinforcement Learning

An adversarial environment may be expressed as a two-player γ discounted zero-sum Markov game (Perolat et al. 2015). The Markov Decision Process (MDP) of this game can be expressed as a tuple $(S, A_1, A_2, P, r, \gamma, s_0)$, where A_1 and A_2 are the continuous action sets for the normal and adversarial agents, respectively. Moreover, $P : S \times A_1 \times A_2 \times S \rightarrow \mathbb{R}$ is the transition probability function, and $r : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ is the reward signal of the two agents. If the policy of the normal agent is π_μ, and the policy of the adversarial agent is π_ω, the reward function can be expressed as $r_{\pi_\mu,\pi_\omega} = E_{a,\omega}[r(s,a,\omega)]$, where $s \in S, a \sim \pi_\mu(\cdot|s) \in A_1, \omega \sim \pi_\omega(\cdot|s) \in A_2$. The two-player zero-sum game can be considered as an environment, where the goal of the normal agent is to maximize the return of γ discount (accumulative reward) $G = E_{s_0,a,\omega}[\sum_{t=0}^{T-1} \gamma^t r(s,a,\omega)]$, where T is the horizon length of each episode; whereas, that of the adversarial agent is to minimize the return of γ discount. (Perolat et al. 2015) demonstrated that for a game with optimal equilibrium return G^*, minimax and Nash equilibriums are equivalent,

$$G^* = \min_{\pi_\omega} \max_{\pi_\mu} G(\pi_\mu, \pi_\omega) = \max_{\pi_\mu} \min_{\pi_\omega} G(\pi_\mu, \pi_\omega), \quad (1)$$

(Pinto et al. 2017) indicated that optimal return might be approached by directly learning the optimal policy π_μ^*, π_ω^*.

2.2 Characteristics of L_2 Gain in a Reinforcement Learning System

The basic idea of robust H_∞ control is to suppress the maximum gain of the system to impair the influence of the disturbances signal on the evaluation signal, which describes the system quality (i.e., the reward signal in RL system)(YingMin 2007). The L_2 norm is introduced into the signal space to measure the evaluation signal. The maximum gain of the system can be expressed as the induced norm of the operator from the system input signal space (disturbance space) to the output signal space (reward space). Therefore, if the maximum gain of the system is designed to be small enough, the influence of any input disturbance signal in L_2 space on the evaluation signal will be suppressed within the allowable range, which means that the system is Bounded Input Bounded Output (BIBO) stable. To solve the problem of achieving the robustness stability of an agent under the RL system, (Han et al. 2019) extended the L_2 gain in robustness control theory as:

Definition 1. If an RL system is mean square stable (MSS) when $\omega = 0$, and $\sum_{t=0}^{\infty} E[\|r_\pi(s_t)\|] / \sum_{t=0}^{\infty} E[\|\omega(s_t)\|] \leq \lambda^2$ holds for all $\omega \in L_2 = \{\omega \,|\sum_{t=0}^{\infty} \omega^2(s_t) < \infty\}$, the system is referred to as the MSS with an L_2 gain is less than or equal to λ.

Where $\|\cdot\|$is the 2-norm of the signal and $\omega(s_t)$ is the uncertainty of system, which may contain modeling errors and external disturbances of the system. λ is a positive constant and r_π is the reward signal of policy π. Definition 1 guarantees that the influence of uncertainty on reward function is always bounded.

2.3 Dissipation Inequation in H_∞ Controlled Theory

H_∞ control is an important method for solving robust control problems. It reduces the influence of disturbance on the reward/quality signals by restraining the H_∞ norm of the system (Hongye 2010; Luo, Wu, and Huang 2014; Modares, Lewis, and Sistani 2014). Consider the following nonlinear uncertain systems:

$$\begin{cases} \dot{s} = f(s) + g(s)\omega, \\ r = h(s). \end{cases} \tag{2}$$

Where $s^T = [s^1, s^2, \ldots, s^n] \in M$ is a state vector, and the superscript representing each component of the vector is distinguished from the subscript representing the time step. In a small abuse of notation, ω is an uncertain disturbance signal and denote the adversarial action in Section 3, M is the invariant set of the system (2), r is a reward signal, $f(s)$, $h(s)$ and $g(s)$ are differentiable functions. Note that because the control variable can be expressed as a function of the state variable, we incorporate the control variable into the functions $f(s)$ and $h(s)$ for clarity. The following theorem exist (TieLong 1996):

Theorem 1. For a given $\lambda > 0$, the sufficient condition for L_2 gain being less than or equal to λ in the nonlinear uncertain system (2) is that a continuous differentiable Lyapunov function $F(s) \geq 0$ exists, and $\forall s \in M$ satisfies the following dissipation inequation:

$$\frac{\partial F}{\partial s}(s) f(s) + \frac{\partial F}{\partial s}(s) g(s) \omega \leq \frac{1}{2}\{\lambda^2 \|\omega\| - \|r\|\} \tag{3}$$

Where the left side of the inequation can be viewed as the total derivative of the Lyapunov function. Therefore, (3) can be further simplified as:

$$\frac{\mathrm{d}F(s)}{\mathrm{d}t} \leq \frac{1}{2}\{\lambda^2 \|\omega\| - \|r\|\} \tag{4}$$

Thus, it is clear that there is a close correlation between the characteristics of the L_2 gain and the dissipativeness of an uncertain system. Please refer to (Willems 1972; Byrness, Isidori, and Willems 1991) for a specific proof.

3 Methodology

In this section, we introduce the L_2 gain constraint into the RL architecture to ensure the stability of the system during training and enhance the robustness of normal agent. The constraint conditions of a robust RL based on dissipative inequality are presented in Section 3.1. In Section 3.2, we show the method of introducing this constraint into the adversarial architecture. Finally, in Section 3.3, we introduce the overall architecture and optimization process of the proposed algorithm.

3.1 Dissipation Inequation-Based Robust Reinforcement Learning Constraint

In this section, we will extend the relationship between L_2 gain and dissipativeness of H_∞ control theory (Theorem 1) to the RL system described in Section 2.1. First, the basic assumption is presented.

Assumption 1. The limit of k-step transition probability from state s_0 to state s under policy π exists: $\lim_{k\to\infty} Pr(s\,|s_0, k, \pi)$.

Assumption 2. For any policy π which belongs to policy space Π, the rewards as received in each step during an interaction with the environment are always bounded, i.e., $\forall s \in \{s\,|r_\pi(s) < \infty\}, \pi \in \Pi$.

Assumption 1 is the basic assumption in reinforcement learning theory. Because real systems are always bounded, Assumption 2 is easy to satisfy. On this assumption, we propose sufficient conditions for the MSS of the RL system.

Theorem 2. Given that $\lambda > 0$, the sufficient condition for L_2 gain being less than or equal to λ under Definition 1 with regard to the RL system as defined in Section 2.1 is that there exist positive constants a, b, and continuous differentiable Lyapunov function $F(s)$ such that $0 \leq a r_\pi(s) \leq F(s) \leq b r_\pi(s)$ and satisfies:

$$E_{\mu(s)}[E_\pi[F(s')] - F(s)] \leq E_{\mu(s)} \frac{1}{2}[\lambda^2 \|\omega(s)\| - \|r_\pi(s)\|] \tag{5}$$

Where s' is the successor state, and $\mu(s)$ is the state distribution of policy π. Intuitively, the left-hand side of Eq.(5) can be regarded as the difference of the Lyapunov function,

that is, the concept of derivative in the discrete-time system. Therefore, the Eq.(5) has the same structure as the dissipation inequation (4) and is a generalization in the RL discrete-time systems, and Theorem 2 can be regarded as an alternative statement of Theorem 1 in the MSS. It should be noted that Theorem 2 in our paper has a similar structure and assumptions to Theorem 1 of (Han et al. 2019). However, Theorem 1 of (Han et al. 2019) gives constraints from the perspective of Lyapunov stability theory, while Theorem 2 in our paper is a generalization of the dissipative inequality in the robust H_∞ control theory. The dissipative theory is an idea of disturbance suppression, which suppresses the influence of disturbance on the reward of the RL system to the desired minimum. The complete proof is given in Appendix A.1. In the next subsection, we will show how to introduce Theorem 2 into the policies update functions of the adversarial architecture and approximate the Lyapunov function $F(s)$.

3.2 Policy Updating Functions with Lagrange Multiplier Method

In our adversarial game, at each moment t, the normal and adversarial agents observe the state s_t at the same time and select actions $a_t \sim \pi_\mu(s_t)$ and $\omega_t \sim \pi_\omega(s_t)$, where π_μ and π_ω are the policies of the normal and adversarial agents, respectively. The normal agent maximizes the long-term return whereas the adversarial agent minimizes the long-term return. The two agents are constrained by inequation (5). Let θ^μ and θ^ω denote the parameters of the policies π_μ and π_ω respectively. The theoretical update will be:

$$\begin{cases} \theta^\mu_{k+1} = arg\max\limits_{\theta^\mu} \underset{s,a\sim\pi_{\theta^\mu_k}}{E} \left[\mathcal{G}_\mu(s,a,\theta^\mu_k,\theta^\mu)\right], \\ \theta^\omega_{k+1} = arg\max\limits_{\theta^\omega} \underset{s,\omega\sim\pi_{\theta^\omega_k}}{E} \left[-\mathcal{G}_\omega(s,\omega,\theta^\omega_k,\theta^\omega)\right], \end{cases}$$
$$s.t.\ E_{s'\sim\pi_{\theta^\mu_k}}\left[F(s')\right] - F(s) \le \frac{1}{2}\left[\lambda^2\left\|\omega(s)\right\| - \left\|r_\pi(s)\right\|\right] \tag{6}$$

Where $\mathcal{G}_\mu(s,a,\theta^\mu_k,\theta^\mu)$, $\mathcal{G}_\omega(s,\omega,\theta^\omega_k,\theta^\omega)$ are the "surrogate" objectives for updating the policies of the normal and adversarial agents, respectively. We will let $\mathcal{G}_\mu$, $\mathcal{G}_\omega$ denote above "surrogate" objectives for concise. They represent different optimization objectives in different RL algorithms such as the clipped advantage function in the proximal policy optimization (PPO) algorithm (Schulman et al. 2017). The Lagrange multiplier method is then used to bring the constraint term into the update equation as,

$$\begin{cases} \theta^\mu_{k+1} = arg\max\limits_{\theta^\mu} \underset{s,a\sim\pi_{\theta^\mu_k}}{E} \left[\mathcal{G}_\mu - \alpha\Delta L(s,a,\omega,r,s')\right], \\ \theta^\omega_{k+1} = arg\max\limits_{\theta^\omega} \underset{s,\omega\sim\pi_{\theta^\omega_k}}{E} \left[-\mathcal{G}_\omega - \alpha\Delta L(s,a,\omega,r,s')\right], \end{cases} \tag{7}$$

Where α is the Lagrangian multiplier of the corresponding items, $\Delta L(s,a,\omega,r,s')$ is the Lagrangian of dissipation inequation (5) and we will let ΔL denote this for concise,

$$\Delta L = F(s', \tau_{\theta^\mu}(s')) - F(s,a) + \frac{1}{2}\left[\|r\| - \lambda^2\|\omega\|\right] \tag{8}$$

Where $\tau_{\theta^\mu}(s')$ is the parameterized policy of the normal agent. When the adversarial agent is updated, the parameterized policy $\tau_{\theta^\omega}(s)$ of the adversarial agent is adopted to substitute ω in (8).

Note that (8) is derived from the right-half shift term of the constraint inequality in (6). This Lagrangian has different meanings in the update equations of the normal and adversarial agents. For the normal agent, the Lagrangian multiplier can give the agent an additional gradient with respect to the stability of the system, whereas for the adversarial agent, the Lagrangian multiplier restrains the update of the policy that leads to system instability. We will discuss this mechanism in more detail in Appendix A.2. The Lyapunov function is calculated using a neural network for fitting through the data-driven method. The objective function of the Lyapunov network F_θ is defined as:

$$J_{Lya}(F_\theta) = E_{(s,a)\sim D_\mu}\left[\frac{1}{2}(F_\theta(s,a) - F_{target}(s,a))^2\right] \tag{9}$$

The target function of the Lyapunov function can be defined differently (Mayne et al. 2000). In this study, we use a generalized advantage function (Schulman et al. 2015): $F_{target}(s,a) = \hat{A}(s_t,a_t)$. (Luo, Wu, and Huang 2014) proved that the approximation of the Lyapunov function obtained from a model-free data-driven calculation would not influence the performance of the H_∞ controlled policies.

3.3 Dissipation Inequation Constraint-Based Adversarial Reinforcement Learning Method

Our algorithm optimizes both the normal and adversarial agents using the following alternating procedure. In each rollout, the trajectories of the normal and adversarial agents interacting with the environment are collected. Following that, the advantage function is estimated by the trajectories, and the normal agent and Lyapunov network are updated. The Lyapunov network and the trajectories are then used to update the adversarial agent. This sequence is repeated until convergence.

Algorithm 1 describes our method in detail, where θ^F is the parameter of the Lyapunov network; the subscripts of all the network parameters represent the iteration step; N_{iter} is the total number of the rollout; N_{Sample} is the length of each rollout trajectory; $Optimizer(\cdot)$ is a policy optimizer that can use different algorithms according to different tasks and "surrogate" objectives such as PPO, trust region policy optimization (TRPO), and soft actor-critic (SAC). The Lyapunov network is only optimized with the normal agent, and the network parameters are fixed in the optimization process of the adversary. Furthermore, before update the adversary, the successor action a' is resampled using the updated normal agent network and stored in D_ω buffer in order to calculate the first term in Eq. (8).

4 Experiment

In this section, we describe the three sets of experiments performed. The first and second sets of experiments were carried out in MuJoCo. The purpose of the first group of ex-

Algorithm	Pendulum	Double	HalfCheetah	Hopper
Vanilla PPO	59.2±0.4	36.3±1.19	66.7±2.24	14.6±0.49
RARL	72.8±0.75	40.6±1.62	82.5±1.96	20.4±0.66
NR-MDP	71.2±0.4	26.9±1.37	46.8±2.18	12.9±1.14
Oracle	**78.0±0.2**	33.3±1.49	84.3±1.27	22.0±0.45
DICARL(ours)	77.1±0.7	**44.1±1.87**	**87.3±0.9**	**38.4±1.28**

Table 1: Success rates and standard deviations of different algorithms with 100 mass combinations are compared. Where the Pendulum and Double denote the InvertedPendulum and InvertedDoublePendulum tasks, respectively. The trained policies are initialized by 7 random seeds, and 700 episodes are tested for each mass group. Significantly better results from a t-test with $p < 1\%$ are highlighted in bold.

Algorithm 1: DICARL (proposed algorithm)

Initialize Environment $\mathcal{E}$, rollout buffer D_μ, D_ω, Lagrangian multiplier α **Initialize** Lyapunov network $F(s,a)$, constant λ, normal agent policy $\pi_\mu(a\,|s)$, adversarial agent policy $\pi_\omega(\omega\,|s)$ and corresponding parameters $\theta_0^F, \theta_0^\mu, \theta_0^\omega$

for k in N_{iter} **do**
  $\theta_k^\mu \leftarrow \theta_{k-1}^\mu, \theta_k^\omega \leftarrow \theta_{k-1}^\omega, \theta_k^F \leftarrow \theta_{k-1}^F$
  for t in N_{Sample} **do**
    Run policies $a_t \sim \pi_\mu(a\,|s, \theta_k^\mu)$, $\omega_t \sim \pi_\omega(\omega\,|s, \theta_k^\omega)$ in environment $\mathcal{E}$
    Get successor states s_{t+1}, r_t, store $(s_t, a_t, w_t, r_t, s_{t+1})$ in D_μ, D_ω
  end for
  Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_{Sample}$ and stored in D_μ, D_ω
  $\theta_k^\mu, \theta_k^F \leftarrow Optimizer(D_\mu, \theta_k^\mu, \theta_k^F, \mathcal{G}_\mu, J_{Lya}(F_{\theta^F}))$
  $\theta_k^\omega \leftarrow Optimizer(D_\omega, \theta_k^\omega, \mathcal{G}_\omega, J_{Lya}(F_{\theta^F}))$
end for

periments was to verify the robustness of our method with respect to four baseline algorithms. The purpose of the second group of experiments was to prove our assertion (i.e., introducing a dissipative inequation into the adversarial architecture could stabilize the training). The last set of experiments was carried out in GymFc, an RL environment for developing attitude flight controllers for the unmanned aerial vehicle (UAV) (Koch et al. 2019). In this environment, we compared the robustness of our method and that of the baseline method under actuator attacks. Finally, we deployed the trained neural network controller in a real quadrotor attitude controller to demonstrate the potential of our method in sim-to-real tasks.

We used the PPO algorithm implemented by openAI baselines (Dhariwal et al. 2017) as the policy optimizer for all the algorithms to fairly compare the baseline algorithms to ours. Specifically, we compared Vanilla PPO, RARL, Noisy Action Robust MDP (NR-MDP) (Tessler, Efroni, and Mannor 2019), and domain randomization. We called the domain randomization method the Oracle method because it trained directly on the test domain. The pseudo codes of each algorithm are shown in Appendix B. All our experiments were run on a desktop computer equipped with an Intel Core i7-8700k CPU.

4.1 Robustness Under the Modeling Error

We chose four MuJoCo tasks: InvertedPendulumAdv-v1, InvertedDoublePendulumAdv-v1, HopperAdv-v1, and HalfCheetahAdv-v1 from an improved adversarial Gym (Pinto et al. 2017) to test our algorithm. In each task, the agent was a robot composed of several joints. Considering our method and the RARL, the adversary acted on different body parts of the robot. For the NR-MDP, the adversary acted on the actuator of the robot. For the Oracle method, the environmental variables (body parts of robots with different masses) in the test domain were initialized randomly before each rollout. The Vanilla PPO trained directly in the original environment without any adversaries. The detailed description of the environment can be found in Appendix C.1. In addition, all the shared hyper-parameters were the same, and we run all algorithms with the same amount of simulation steps. The detailed settings of the hyper-parameter of each algorithm are reported in Appendix C.2.

The modeling error between the simulation and the real world can cause controllers to fail, requiring a robust control policy. In the training domain, the body mass of the robot remained unchanged. In the test domain, we changed the masses of the two body parts of the robot. The range of change in the mass was the original mass multiplied by [0.1-2.1) with an interval of 0.2, and a total of 100 groups of different mass combinations. The threshold of time steps was set for each task. If the time steps of the agent is larger than the threshold, it will be regarded as a successful task. If otherwise, it will be a failure. Each mass group was trained on seven random seeds and is evaluated across 700 episodes in the training domain.

In safety-critical tasks or real robot tasks, any failure may be fatal and can cause damage to the robot. Therefore, we counted the number of combinations with a failure rate of less than 2% in different mass groups in the test domain as a measure of the adaptability of the controller. The results in Table 1 demonstrate that the proposed algorithm outperforms all the baselines in most environments. In the InvertedPendulum environment, our algorithm also has competitive performance. The heat-maps of accumulative reward and failure rates can be found in Appendix C.3. The training curves can be found in Appendix C.4. Moreover, we observed that the Oracle method is the most competitive base-

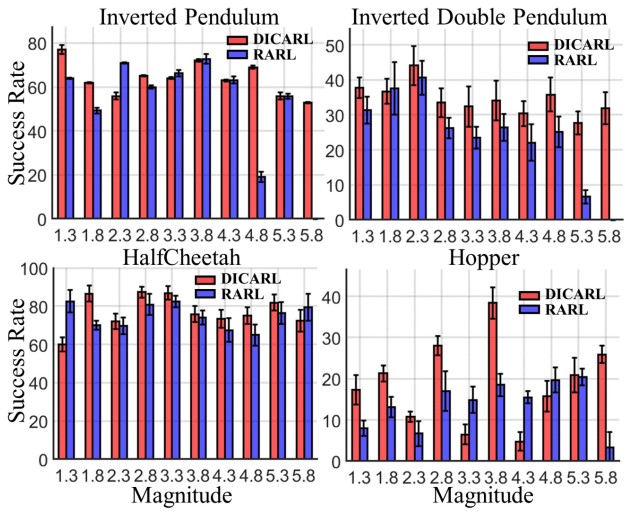

Figure 1: Success rates and thrice the standard deviations of DICARL and RARL in different adversary magnitudes are compared. The trained policies are initialized by 7 random seeds, and 700 episodes are tested for each mass group.

line on most tasks, since this method is directly trained in the test domain. Nevertheless, in the InvertedDoublePendulum environment, the Oracle method performs poorly. This is because the inverted double pendulum system is more sensitive to changes in the model's parameter (Gluck, Eder, and Kugi 2013), and the randomization of the mass of the pendulum may lead to the appearance of a top-heavy unstable model, increasing the difficulty of training. This highlights a potential issue with domain randomization. Although training across a wide variety of dynamic parameters can increase robustness, naive parameterization may cause the system to become difficult to stabilize during training and lead to a vulnerable policy. We also observe that the performance of the NR-MDP algorithm is poor. The possible reason is that for multi-joint robots, the disturbance on the actuator may not be mapped to the mass change of the robot. On the other hand, when the actuator disturbance is too strong, normal agents may be prevented from successfully solving tasks in some environments (Tessler, Efroni, and Mannor 2019). We put the hyper-parameter setting and performance analysis of the SAC version of all algorithms in Appendix D.

4.2 Influence of Robustness Constraint on System Stability

This set of experiments is based on the following intuition: in adversarial training, the stronger the adversary is, the easier its instability. Therefore, with our method, the RARL algorithm was used as the baseline for comparison. We used the same environmental settings and shared hyper-parameters as in Section 4.1 to train the agents. The difference was that we set different upper limits of magnitude forces for the adversary (i.e., the action range of the adversarial agent). The detailed hyper-parameter settings of the algorithm are shown in Appendix E. We used the same test domain as mentioned in Section 4.1 to compare the robustness of the two algorithms. The upper limit of the magnitude of the adversary is set to $[1.3 - 5.8]$.

We calculated the adaptability of the two algorithms to the test domain under training with different magnitudes of the adversary. The evaluation method is the same as mentioned in Section 4.1; the result is shown in Figure 1. In the environment of InvertedPendulum and InvertedDoublePendulum, it can be seen that in the RARL algorithm, when the magnitude of the adversary increases to a certain extent, the performance of the controller degrades rapidly, and a stable control policy cannot be acquired. This result is consistent with the observation of (Tessler, Efroni, and Mannor 2019; Pan et al. 2019; Pinto et al. 2017) because these two environments are more sensitive to disturbance, and the powerful adversary in RARL can always prevent the normal agent from successfully performing tasks. However, DICARL can adapt to a more powerful adversary. This supports our viewpoint that DICARL can effectively use adversary attacks to improve robustness and maintain environmental stability during training. For the HalfCheetah environment, the increase in the magnitude of the adversary has no significant effect on the normal agent because the half-cheetah robot has a greater stability margin and is insensitive to disturbances. Moreover, we also observed that the performance of the two algorithms in the Hopper environment fluctuated greatly with the change in the magnitude of the adversary. For a forward hopping robot, the forward disturbance force is more likely to cause task failure than that of the backward disturbance. This indicates that the increase in the force of the magnitude does not necessarily obtain a sufficiently strong adversary, resulting in the difference in the robustness of the normal agent. In general, our algorithm has more advantages when used with a powerful adversary during training and has a controller with optimal adaptability to the environment. This provides a potential method for improving robustness through more complex and diversified adversarial training in the future.

4.3 Sim-to-Real Task

In the last set of experiments, we used the GymFc to train the quadrotor angular velocity controller based on a neural network. We compared the two baselines, including Vanilla PPO and NR-MDP. We set the same shared hyper-parameters for the three algorithms. Thereafter, the neural network controller was used to replace the PID angular velocity controller based on the HIL method and deployed to the actual quadrotor flight control system. Finally, we tested our method in a real environment. The environmental setting and real quadrotor parameters can be found in Appendix F.1 and Appendix F.2 detail the setting for the hyper-parameter of the algorithm.

Simulation Environment. In the simulation environment, the signal supplied to the motor is added to 10% random noise. (Fei et al. 2020) reported that this kind of attack on the motor signal could cause the aircraft to deviate from the normal operating point, resulting in serious performance degradation, requiring a robust control policy. We visually

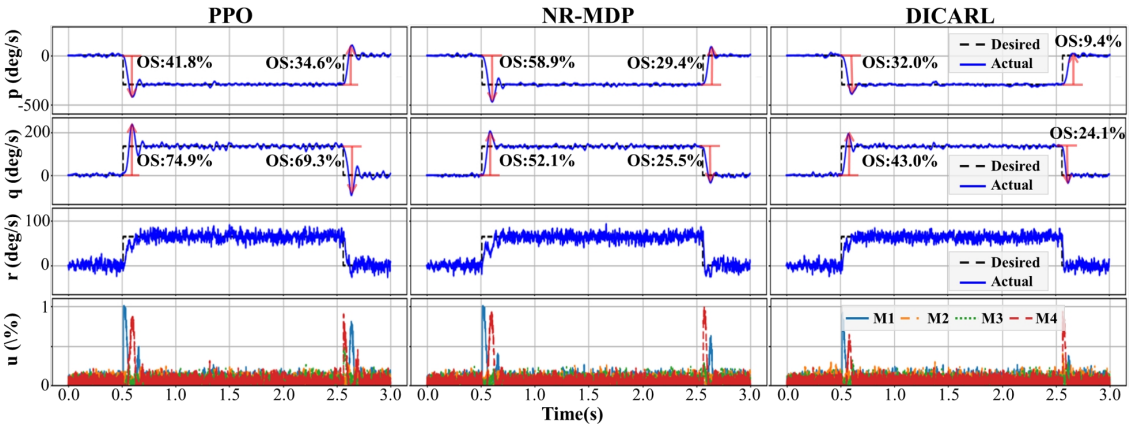

Figure 2: Step responses and control signals of the three algorithms in the GymFc training environment. OS is short for overshoot, whereas blue line represents the actual angular velocity, and dashed black line represents the desired angular velocity.

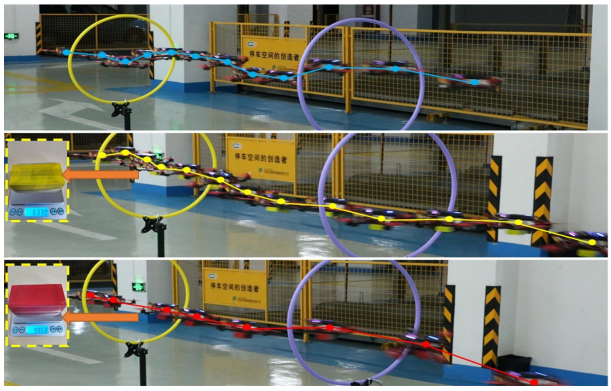

Figure 3: Trajectories of the quadcopter flying over the racing gate carrying payloads of different masses. Where the blue trajectory does not carry a payload, the yellow trajectory carries a payload of 63.3 g, and the red trajectory carries a payload of 98.68 g.

compare the step responses of PPO, NR-MDP, and DICARL angular velocity controllers under a disturbance of 10% random noise in Figure 2. DICARL has the smallest overshoot on the pitch and roll axes. Although NR-MDP has a better performance on the pitch and roll axes than that of the PPO algorithm, both of them have larger control signals(i.e., **u** in Figure 2), which will increase the motor load, reducing the service life. By contrast, our algorithm ensures lower overshoot and control signal, rendering it more suitable for an actual system. We notice that the three algorithms have large oscillations on the yaw axis. This may be because of the disturbance acting on the motors, causing each motor to have a random moment of inertia noise. This increases the difficulty of controlling the yaw axis.

Real Environment. Without further optimization, we evaluated the policy learned in the simulation on the real quadrotor. We used a quadcopter assembled with a QAV250 frame and pixhawk 2.4.6 flight controller. Details of the deployment method can be found in Appendix F.3. To test the robustness of the DICARL algorithm, we made the quadcopter perform the task of flying over the racing gates. We conducted three tests, where the quadcopter carried a different mass of payload in each test. For a small UAV, approximately 10-100 mW is required for 1 g additional take-off weight for hovering (Leutenegger et al. 2016). Even a small extra load will affect the performance. In this test, we tested the performance of the quadcopter under three payloads: no-load, load mass of 63.3 g and 98.68 g. We visualized the flight trajectories of these three tasks as shown in Figure 3. It can be seen that the quadcopter can smoothly and quickly cross the two racing gates under different payloads.

Controller	Pitch error	Roll error	Yaw error
PID	0.3595	0.3168	**0.0514**
DICARL	**0.2206**	**0.1158**	0.0721

Table 2: Normalized average error of pitch, roll, and yaw axes of DICARL and PID attitude controllers in the real world. Significantly better results from a t-test with $p < 1\%$ are highlighted in bold.

Finally, we compared the angular velocity error of the DICARL controller and the HIL-based PID controller (Dai et al. 2019) on the real quadcopter. The normalized mean error is shown in Table 2. Our method has a smaller tracking error on the pitch and the roll axes and delivers competitive performance with the PID controller on the yaw axis. See Appendix F.4 for the visualized tracking curve. In the actual test, the expected angular velocity tracked by our method is most greater than that of the PID controller. This proves that our controller has a faster response speed; however, to some extent, it also reduces the performance of the yaw axis.

5 Conclusion

In this study, we show that the effective method for improving the robustness of the policy and stabilizing the training

is to introduce robust stability constraints in the adversarial RL architecture. From the perspective of robust control, we propose a sufficient condition for the RL strategy to satisfy L_2 gain and develop a new robust adversarial reinforcement learning architecture based on this constraint. Theoretically, our architecture is suitable for the current mainstream RL algorithms, such as PPO, TRPO, and SAC. The PPO and SAC algorithms was used in the policy optimizer to instantiate our architecture and conduct extensive experiments to validate the effectiveness of our algorithm. The experiment on a real quadcopter proves the potential of our algorithm in a sim-to-real task. Both our theoretical and empirical results provide new and critical outlooks about adversarial RL architecture from a rigorous robust control perspective. We do not believe that our research will cause any social problem or put anyone at any disadvantage.

Experiments on Hopper show that the use of a single adversary to approximate the solution to a minimax problem does not consistently lead to improved robustness. Interesting future research directions include increasing the diversity of adversaries, such as using multiple adversaries to train together with a normal agent or attacking different functional components of the RL process. In this process, the stability of the system is also important to the performance of the normal agent. The increase in the number of adversaries may further deteriorate the stability of the system. It is necessary to introduce our method into the aforementioned multi-agent adversarial architecture.

Acknowledgments

This work was supported in part by Shanghai Municipality Science and Technology Major Project 2021SHZDZX0103, in part by Science and Technology Commission of Shanghai Municipality, Grant No. 19511132000, in part by National Natural Science Foundation of China under Grant No. 82090052, in part by Department of Science and Technology of Guangdong Province, Grant No. 2019A1515110352 and in part by Ningbo Science and Technology Bureau, Grant No. 2020Z073.

References

Byrness, C.; Isidori, A.; and Willems, J. 1991. Passivity, feedback equivalence and the global stabilization of minimum phase nonlinear systems. *IEEE T. Automatic Contr*, 36(11): 1228–1240.

Christiano, P.; Shah, Z.; Mordatch, I.; Schneider, J.; Blackwell, T.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2016. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv1610.03518*.

Dai, X.; Ke, C.; Quan, Q.; and Cai, K.-Y. 2019. Unified Simulation and Test Platform for Control Systems of Unmanned Vehicles. *arXiv preprint arXiv1908.02704*.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. *OpenAI Baselines*.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, 1329–1338. PMLR.

Fei, F.; Tu, Z.; Xu, D.; and Deng, X. 2020. Learn-to-Recover Retrofitting UAVs with Reinforcement Learning-Assisted Flight Control Under Cyber-Physical Attacks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 7358–7364. IEEE.

Gleave, A.; Dennis, M.; Wild, C.; Kant, N.; Levine, S.; and Russell, S. 2019. Adversarial policies Attacking deep reinforcement learning. *arXiv preprint arXiv1905.10615*.

Gluck, T.; Eder, A.; and Kugi, A. 2013. Swing-up control of a triple pendulum on a cart with experimental validation. *Automatica*, 49(3): 801–808.

Gu, Z.; Jia, Z.; and Choset, H. 2019. Adversary a3c for robust reinforcement learning. *arXiv preprint arXiv1912.00330*.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv1812.05905*.

Han, M.; Tian, Y.; Zhang, L.; Wang, J.; and Pan, W. 2019. H_∞ model-free reinforcement learning with robust stability guarantee. *arXiv preprint arXiv1911.02875*.

Hodge, V. J.; Hawkins, R.; and Alexander, R. 2021. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, 33(6): 2015–2033.

Hongye, S. 2010. *Basic Theory of Robust Control*. Science Press.

Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; and Hutter, M. 2019. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26).

Ilahi, I.; Usama, M.; Qadir, J.; Janjua, M. U.; Al-Fuqaha, A.; Hoang, D. T.; and Niyato, D. 2020. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *arXiv preprint arXiv2001.09684*.

Kamalaruban, P.; Huang, Y.-T.; Hsieh, Y.-P.; Rolland, P.; Shi, C.; and Cevher, V. 2020. Robust reinforcement learning via adversarial training with langevin dynamics. *arXiv preprint arXiv2002.06063*.

Koch, W.; Mancuso, R.; West, R.; and Bestavros, A. 2019. Reinforcement learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems*, 3(2): 1–21.

Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; and Hutter, M. 2020. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47).

Leutenegger, S.; Hurzeler, C.; Stowers, A. K.; Alexis, K.; Achtelik, M. W.; Lentink, D.; Oh, P. Y.; and Siegwart, R. 2016. Flying robots. In *Springer Handbook of Robotics*, 623–670. Springer.

Li, T.; Lambert, N.; Calandra, R.; Meier, F.; and Rai, A. 2020. Learning generalizable locomotion skills with hierarchical reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 413–419. IEEE.

Luo, B.; Wu, H.-N.; and Huang, T. 2014. Off-policy reinforcement learning for H_∞ control design. *IEEE transactions on cybernetics*, 45(1): 65–76.

Mackenroth, U. 2008. Robust high-performance disturbance rejection for an uncertain inverted double pendulum. In *2008 American Control Conference*, 2377–2383. IEEE.

Mayne, D. Q.; Rawlings, J. B.; Rao, C. V.; and Scokaert, P. O. 2000. Constrained model predictive control Stability and optimality. *Automatica*, 36(6): 789–814.

Modares, H.; Lewis, F. L.; and Sistani, M.-B. N. 2014. Online solution of nonquadratic two-player zero-sum games arising in the H_∞ control of constrained input systems. *International Journal of Adaptive Control and Signal Processing*, 28(3-5): 232–254.

Morimoto, J.; and Doya, K. 2005. Robust reinforcement learning. *Neural computation*, 17(2): 335–359.

Pan, X.; Seita, D.; Gao, Y.; and Canny, J. 2019. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 8522–8528. IEEE.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, 3803–3810. IEEE.

Perolat, J.; Scherrer, B.; Piot, B.; and Pietquin, O. 2015. Approximate dynamic programming for two-player zero-sum markov games. In *International Conference on Machine Learning*, 1321–1329. PMLR.

Pinto, L.; Davidson, J.; and Gupta, A. 2017. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1601–1608.

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

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv1707.06347*.

Tessler, C.; Efroni, Y.; and Mannor, S. 2019. Action Robust Reinforcement Learning and Applications in Continuous Control. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6215–6224. PMLR.

TieLong, S. 1996. *H_∞ Control theory and application*. Tsinghua University Press.

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

Vinitsky, E.; Du, Y.; Parvate, K.; Jang, K.; Abbeel, P.; and Bayen, A. 2020. Robust Reinforcement Learning using Adversarial Populations. *arXiv preprint arXiv2008.01825*.

Willems, J. C. 1972. Dissipative dynamical systems part I General theory. *Archive for rational mechanics and analysis*, 45(5): 321–351.

Wu, H.-N.; and Luo, B. 2012. Neural Network Based Online Simultaneous Policy Update Algorithm for Solving the HJI Equation in Nonlinear H_∞ Control. *IEEE Transactions on Neural Networks and Learning Systems*, 23(12): 1884–1895.

YingMin, J. 2007. *Robust H_∞ Control*. Science Press.

Zhang, K.; Hu, B.; and Basar, T. 2020. On the Stability and Convergence of Robust Adversarial Reinforcement Learning A Case Study on Linear Quadratic Systems. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 22056–22068. Curran Associates, Inc.

Zhao, W.; Queralta, J. P.; and Westerlund, T. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 737–744.