Proceedings of Machine Learning Research vol 144:1–21, 2022

Adversarially Regularized Policy Learning Guided by Trajectory Optimization

Zhigen Zhao ZHIGEN.ZHAO@GATECH.EDU
Simiao Zuo SIMIAOZUO@GATECH.EDU
Tuo Zhao[*] TOURZHAO@GATECH.EDU
Ye Zhao[*] YE.ZHAO@ME.GATECH.EDU
Georgia Institute of Technology, North Ave NW, Atlanta, GA 30332

Abstract

Recent advancement in combining trajectory optimization with function approximation (especially neural networks) shows promise in learning complex control policies for diverse tasks in robot systems. Despite their great flexibility, the large neural networks for parameterizing control policies impose significant challenges. The learned neural control policies are often overcomplex and non-smooth, which can easily cause unexpected or diverging robot motions. Therefore, they often yield poor generalization performance in practice. To address this issue, we propose ad**ve**rsarially **r**egularized p**o**licy lear**ni**ng guided by traje**c**tory optimiz**a**tion (VERONICA) for learning smooth control policies. Specifically, our proposed approach controls the smoothness (local Lipschitz continuity) of the neural control policies by stabilizing the output control with respect to the worst-case perturbation to the input state. Our experiments on robot manipulation show that our proposed approach not only improves the sample efficiency of neural policy learning but also enhances the robustness of the policy against various types of disturbances, including sensor noise, environmental uncertainty, and model mismatch.

Keywords: Adversarial Regularization, Policy Learning, Trajectory Optimization

1. Introduction

Robust and generalizable motion planning enables robotic systems to handle various uncertainties and accomplishes diverse tasks. However, learning a dynamically consistent neural control policy (i.e., a neural-network control policy) and executing it reliably remain challenging. First, the function approximators used to model the policy can be highly complex and non-smooth, causing poor generalization performance. Second, the dynamics models involved often have some mismatch between the physical robot and the environment, leading for the need to learn a robust policy.

Trajectory optimization (TO) (Betts, 1998; Kuindersma et al., 2016; Tassa et al., 2014; Posa et al., 2014) is a powerful model-based approach to generate optimal control sequences for complex robotic systems. However, existing methods for solving TO problems with full robot dynamics require solving large nonlinear programs, resulting in high computational cost. This difficulty prevents the use of TO methods in real-time robot control settings. As such, to alleviate the computational burden at run-time, it is preferable to have a parametric representation of a robot control policy. In comparison, model-free policy search, as in (Deisenroth et al., 2013), aims to automatically learn the controller through random exploration. However, a majority of these methods fail to explore the model dynamics, which causes sample inefficiency.

To take advantage of both TO and policy search, Mordatch and Todorov (2014) and Levine and Koltun (2013a) train a robot control policy supervised by optimized trajectory samples, and meanwhile adapting TO to the learned policy. The work in Mordatch and Todorov (2014) observes that

[*] Co-corresponding Author

the derivatives of a neural control policy can behave irregularly even when the policy matches the optimal trajectory baseline. This is because neural networks have high complexity and flexibility, which makes them highly non-smooth — a small change in the networks' input can cause a large variation in the output. To mitigate this limitation, existing works attempt to impose some smoothness constraints on the policy. For instance, Mordatch and Todorov (2014) matches the gradient for policy and trajectory samples via tangent propagation. However, tangent propagation requires Jacobian computation on each trajectory point, which does not scale well to large datasets.

To alleviate these issues, we propose a new approach: ad**ve**rsarially **r**egularized p**o**licy lear**ni**ng guided by traje**c**tory optimiz**a**tion (VERONICA). Specifically, our approach improves the local Lipschitz continuity of the neural control policy via adversarial regularization, which improves generalization performance for inputs not seen during training. We focus on promoting smoothness in policy for non-hybrid robotics tasks that are often governed by differential equations with high-order continuity. For hybrid systems where non-smooth dynamics might occur during physical contact, several works in TO (Brubaker et al., 2009; Todorov, 2011; Mordatch et al., 2012) and physical simulation MuJoCo (Todorov et al., 2012) propose to model contact with a smoothed model, where contact forces diminish gradually with contact distance. The work of Drnach and Zhao (2021) proposes a risk-sensitive cost function to represent a stochastic, smoothed variant of the original complementarity contact problem (Chen and Mangasarian, 1996). In this work, we show that the VERONICA framework also provides robustness benefits for a hybrid locomotion system with physical contacts.

The VERONICA framework is related to existing works (Miyato et al., 2018; Zhang et al., 2019; Hendrycks et al., 2019; Xie et al., 2019; Jiang et al., 2019; Shen et al., 2020). These works consider similar regularization techniques, but target at other applications with different motivations, e.g., semi-supervised learning, unsupervised domain adaptation, harnessing adversarial examples, fine-tuning pre-trained models and model-free reinforcement learning. Morimoto and Doya (2000) solves a similar min-max problem to improve the robustness of reinforcement learning.

We further observe that besides promoting policy smoothness, adversarial regularization improves the robustness of the policy against modeling errors and perturbations in the environment. We verify that the VERONICA framework produces stable robot behaviors under sensor noise, environmental uncertainty, and model mismatch.

Conventionally, adversarial regularization involves a min-max game, which is solved by alternating gradient descent-ascent. During training, neither of the players can be advantageous, such that the generated perturbations can be over-strong and hinder model generalization. To resolve this issue, we employ Stackelberg adversarial regularization (SAR), as proposed in Zuo et al. (2021), which formulates adversarial regularization as a Stackelberg game. In SAR, the policy (i.e., the leader) has a higher priority than the perturbation (i.e., the follower). The leader procures its advantage by considering how the follower will respond after observing the leader's decision, such that the leader anticipates the predicted move of the follower when optimizing its strategy. We note that prioritizing the policy optimization is reasonable and beneficial because we target the performance of the learned policy, instead of the adversary.

Our contributions are: I) We propose VERONICA, an adversarial regularization method for learning smooth neural control policies guided by TO. This improves the generalization performance of the learned policy; II) We show that the learned policy achieves better robustness under disturbances such as sensor noise, environmental uncertainty, and model mismatch; III) We reformulate adversarial regularization as a Stackelberg game, which further improves generalization and robustness of the policy compared with the conventional formulation.

2. Related Works

Adversarial Training in Robot Learning: Adversarial training has previously been used to improve safety in robot visuomotor control scenarios (Chen et al., 2020). The work in Lechner et al. (2021) argues that adversarial training induces unexplored error profiles in *vision-based* robot learning, which studies classification tasks that are not Lipschitz continuous. In contrast, our work focuses on adversarial regularization for neural control policy in *dynamics-based* robot learning, which are intrinsically smooth. Therefore, *vision-based* adversarial training studies fundamentally different problems than ours.

Imitation Learning: Behavioral cloning (BC) uses supervised learning to directly imitate expert trajectories without interacting with the environment (Schaal et al., 1997). However, BC is particularly vulnerable to error compounding (Ross et al., 2011). In our work, we solve a BC problem for policy learning in each iteration of the Alternating Direction Method of Multipliers (ADMM) method, while the ADMM framework offers a coupling mechanism to allow the trajectory optimizer (i.e., the teacher) to not only guide the learned policy (i.e., the student) towards better solutions but also adapt to the student. More importantly, we incorporate an adversarial regularizer to improve policy smoothness, which significantly eases the effect of error compounding.

Along another line of research, generative adversarial imitation learning (Ho and Ermon, 2016; Zolna et al., 2019) uses generative adversarial networks (GAN) to directly generate policies that imitate expert demonstrations. In contrast, the adversaries in our work are the direct perturbations on the input (i.e., the robot state), rather than the discriminator network.

Trajectory-Optimization-Guided Policy Learning: Trajectory optimization has been used to aid and stabilize value function learning in the reinforcement learning (RL) context (Lowrey et al., 2018), while Landry et al. (2021) use a bilevel optimization to learn the value function with adversarial samples. In this work, we focus on supervised learning approaches that train neural control policies from TO.

Guided policy search (GPS) (Levine and Koltun, 2013a,b, 2014) iteratively updates guiding sample using differential dynamic programming (DDP) and trains policies on the distribution over the guiding samples. In contrast, the work of Mordatch and Todorov (2014) seeks consensus between neural network policy and trajectory optimization using ADMM (Boyd et al., 2011). Duburcq et al. (2020) similarly solve for ADMM consensus, but aim to learn a trajectory sequence rather than policy. The ADMM formulation in our work is closely related to Mordatch and Todorov (2014), but we focus on adversarial regularization for policy learning.

3. Method

We introduce VERONICA, our proposed adversarially regularized approach which combines the strength of policy learning and trajectory optimization. First, we define an adversarial regularizer and explain how it improves smoothness and robustness of neural control policies; Second, we describe an ADMM-based algorithm that solves the full joint optimization problem; Third, we develop an extension to our proposed adversarial regularization approach — Stackelberg adversarial regularization. We consider the neural control policy learning process guided by N optimal trajectories $\{\mathbf{X}, \mathbf{U}\} = \{\mathbf{X}_i, \mathbf{U}_i \mid i = 1, \cdots, N\}$, and each optimal trajectory $\{\mathbf{X}_i, \mathbf{U}_i\}$ consists of T state-control pairs $\{\mathbf{x}_i^t \in \mathbb{R}^{d_x}, \mathbf{u}_i^t \in \mathbb{R}^{d_u} \mid t = 1, \cdots, T\}$, where $\mathbf{x}_i^t$ and $\mathbf{u}_i^t$ denote the robot state and the control, respectively. In this study, the robot state corresponds to the joint positions, velocities and task parameters such as goal configurations, while the control corresponds to the joint torque. Moreover, let $\pi(\cdot|\mathbf{W})$ denotes the neural control policy, where $\mathbf{W}$ denotes the associated parameters.

3.1. Adversarial Regularization for Neural Control Policy

To promote smoothness of the neural control policy, we consider the following adversarial discrepancy measure:

$$r_{\text{adv}}(\mathbf{x}, \mathbf{W}) = \max_{\|\boldsymbol{\delta}\| \le \epsilon} r(\mathbf{x}, \mathbf{W}, \boldsymbol{\delta}) = \max_{\|\boldsymbol{\delta}\| \le \epsilon} \|\pi(\mathbf{x}|\mathbf{W}) - \pi(\mathbf{x} + \boldsymbol{\delta}|\mathbf{W})\|^2,$$

where $\|\cdot\|$ denotes the ℓ_2 norm, $\boldsymbol{\delta} \in \mathbb{R}^{d_x}$ is the adversarial perturbation injected to the state vector $\mathbf{x}$, and $\epsilon > 0$ is the perturbation strength. Such an adversarial discrepancy measure $r_{\text{adv}}(\mathbf{x}, \mathbf{W})$ essentially computes the maximal deviation of the neural control policy output at state $\mathbf{x}$ given an input perturbation $\boldsymbol{\delta}$ whose ℓ_2 norm is bounded by ϵ.

We then apply the adversarial discrepancy measure to control the smoothness of the neural control policy. Specifically, we solve the following joint optimization problem:

$$\begin{aligned} &\min_{\mathbf{X},\mathbf{U},\mathbf{W}} \sum_{i=1}^{N} \mathcal{L}(\mathbf{X}_i, \mathbf{U}_i) + \mathcal{Q}_{\text{BC}}(\mathbf{X}, \mathbf{U}, \mathbf{W}) + \alpha \mathcal{R}_{\text{adv}}(\mathbf{X}, \mathbf{W}), \\ &\text{s.t.} \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t), \mathbf{x}^0 = \mathbf{x}_{\text{init}}, \mathbf{X} \in \mathcal{X}, \mathbf{U} \in \mathcal{U}, \end{aligned} \tag{1}$$

where $\mathcal{L}(\mathbf{X}_i, \mathbf{U}_i)$ denotes the loss function of the trajectory optimization (TO) for the i^{th} trajectory, $\mathcal{Q}_{\text{BC}}(\mathbf{X}, \mathbf{U}, \mathbf{W})$ denotes the loss function for policy learning:

$$\mathcal{Q}_{\text{BC}}(\mathbf{X}, \mathbf{U}, \mathbf{W}) = \frac{1}{N} \sum_{i,t} ||\pi(\mathbf{x}_i^t|\mathbf{W}) - \mathbf{u}_i^t||^2,$$

$\mathcal{R}_{\text{adv}}(\mathbf{X}, \mathbf{W})$ is the adversarial regularizer for controlling the smoothness of the policy:

$$\mathcal{R}_{\text{adv}}(\mathbf{X}, \mathbf{W}) = \frac{1}{N} \sum_{i,t} r_{\text{adv}}(\mathbf{x}_i^t, \mathbf{W}) = \frac{1}{N} \sum_{i,t} \max_{\|\boldsymbol{\delta}_i^t\| \le \epsilon} \|\pi(\mathbf{x}_i^t|\mathbf{W}) - \pi(\mathbf{x}_i^t + \boldsymbol{\delta}_i^t|\mathbf{W})\|^2,$$

and α is the regularization coefficient weighting between the $\mathcal{Q}_{\text{BC}}(\mathbf{X}, \mathbf{U}, \mathbf{W})$ and $\mathcal{R}_{\text{adv}}$.

Solving the optimization problem in Eq. (1) learns a neural control policy that not only minimizes the TO loss and the behavior cloning loss, but also encourages the adversarial discrepancy measure of the policy to be small at every state of the optimal trajectories.

(I) Adversarial Regularization Improves Generalization: Existing methods usually train neural control policies by only minimizing the trajectory optimization loss and behavior cloning loss. Due to the high capacity of deep neural networks, the learned neural control policies are often over-complex and highly non-smooth. This is inconsistent with observations that many optimal control policies for robots are smooth. Here we exclude the problem involving physical contact dynamics, which exhibits discontinuous and non-smooth phenomenon. Smoothness requires a small perturbation to the state vector $\mathbf{x}$ to only yield a small change to the policy output (Figure 1). Such a property can improve generalization of the learned policy.

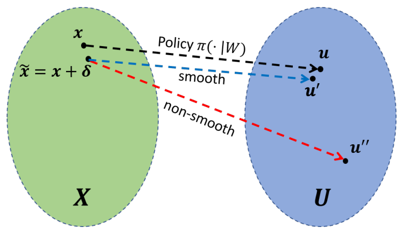

Figure 1: Illustration of policy smoothness at state $\mathbf{x}$ and control $\mathbf{u}$. If the policy $\pi(\cdot|W)$ is smooth around $\mathbf{x}$, the perturbed state $\tilde{\mathbf{x}}$ will produce a control $\mathbf{u}'$ similar to u. If the policy $\pi(\cdot|\mathbf{W})$ is non-smooth around $\mathbf{x}$, the output control $\mathbf{u}''$ would deviate significantly from $\mathbf{u}$.

VERONICA naturally promotes the desired smoothness by imposing a high penalty when the adversarial perturbation $\boldsymbol{\delta}$ yields a large deviation to the policy output. More precisely, $r_{\text{adv}}(\mathbf{x}, \mathbf{W})$ essentially upper bounds the deviation of the policy output due to the adversarial perturbation $\boldsymbol{\delta}$ with respect to the state $\mathbf{x}$, and therefore can be viewed as a measure of the local Lipschitz constant within a small neighborhood of $\mathbf{x}$, i.e., $C_{\mathbf{x}} = \sup_{\|\boldsymbol{\delta}\| \leq \epsilon} \frac{\|\pi(\mathbf{x}|\mathbf{W}) - \pi(\mathbf{x}+\boldsymbol{\delta}|\mathbf{W})\|}{\|\boldsymbol{\delta}\|}$. Accordingly, our proposed adversarial regularizer penalizes the average discrepancy measures of the neural control policy at all trajectory points, which enforces its local Lipschitz continuity.

(II) Adversarial Regularization Gains Robustness: Robot systems measure their states from sensors, which are prone to stochastic or systematic sensor errors. VERONICA naturally gains robustness against such disturbances. Specifically, the adversarial perturbation in VERONICA can be viewed as a proxy to the errors. Therefore, our approach does not require prior knowledge of them. In comparison, existing methods for handling such errors usually assume specific forms, e.g., independent Gaussian noise, which can be restrictive in practice.

Moreover, as suggested in Asadi et al. (2018), the Lipschitz continuity is essential to robustness, especially for control and reinforcement learning problems. This is because for policies without the Lipschitz continuity property, a small error in sensor measurement or state transition potentially leads to a drastic change to the policy output. Due to the dynamic nature of the control problem, it will further yield significant error compounding during policy roll-out. Moreover, when the models used to describe robot dynamics mismatch the real robot, such compounding system errors can be catastrophic. Quantitatively, the upper bound for policy robustness under state disturbance, measured by compounding value function discrepancy, is proportional to the Lipschitz constant of the neural control policy. As the VERONICA approach can effectively control the local Lipschitz continuity of the neural control policy, such an issue can be mitigated. A theoretical analysis can be found in Appendix G of the supplementary material (Zhao et al., 2021).

3.2. Combined Trajectory Optimization and Adversarially Regularized Policy Learning

We apply ADMM (Zhao et al., 2020; Zhou and Zhao, 2020) to solve the optimization problem in Eq. (1). Specifically, we reparameterize Eq. (1) into a decomposable form by introducing two auxiliary sets of state and control variables: $(\mathbf{X}^{\mathrm{TO}}, \mathbf{U}^{\mathrm{TO}})$ represents the trajectory samples generated by trajectory optimization (TO), and $(\mathbf{X}^{\mathrm{PL}}, \mathbf{U}^{\mathrm{PL}})$ are copies of $(\mathbf{X}^{\mathrm{TO}}, \mathbf{U}^{\mathrm{TO}})$ for policy learning. Accordingly, the optimization problem in Eq. (1) is reformulated as:

$$\min_{\mathbf{X}^{\mathrm{TO,PL}}, \mathbf{U}^{\mathrm{TO,PL}}, \mathbf{W}} \sum_{i=1}^{N} \mathcal{L}(\mathbf{X}_i^{\mathrm{TO}}, \mathbf{U}_i^{\mathrm{TO}}) + \mathcal{Q}_{\mathrm{BC}}(\mathbf{X}^{\mathrm{PL}}, \mathbf{U}^{\mathrm{PL}}, \mathbf{W}) + \alpha \mathcal{R}_{\mathrm{adv}}(\mathbf{X}^{\mathrm{PL}}, \mathbf{W})$$
$$\text{s.t.} \;\; \mathbf{X}^{\mathrm{TO}} = \mathbf{X}^{\mathrm{PL}}, \mathbf{U}^{\mathrm{TO}} = \mathbf{U}^{\mathrm{PL}}. \tag{2}$$

ADMM splits the above optimization problem into N individual TO problems and a policy learning problem to be solved in an iterative manner. Let $\boldsymbol{\lambda}_{\mathbf{X}_i}^p, \boldsymbol{\lambda}_{\mathbf{U}_i}^p$ denote the dual variables at the p^{th} iteration and $\rho_x, \rho_u > 0$ denote the penalty parameters. The ADMM primal and policy updates are:

$$\mathbf{X}_i^{\mathrm{TO},p+1}, \mathbf{U}_i^{\mathrm{TO},p+1} = \operatorname*{arg\,min}_{\mathbf{X}_i, \mathbf{U}_i} \; \mathcal{L}(\mathbf{X}_i, \mathbf{U}_i) + \frac{\rho_x}{2}\|\mathbf{X}_i - \mathbf{X}_i^{\mathrm{PL},p} + \boldsymbol{\lambda}_{\mathbf{X}_i}^p\|^2$$
$$+ \frac{\rho_u}{2}\|\mathbf{U}_i - \mathbf{U}_i^{\mathrm{PL},p} + \boldsymbol{\lambda}_{\mathbf{U}_i}^p\|^2, \quad \textbf{(primal TO update)} \tag{3}$$

$$\mathbf{W}^{p+1} = \operatorname*{arg\,min}_{\mathbf{W}} \; \mathcal{Q}_{\mathrm{BC}}(\mathbf{X}^{\mathrm{PL},p}, \mathbf{U}^{\mathrm{PL},p}, \mathbf{W}) + \mathcal{R}_{\mathrm{adv}}(\mathbf{X}^{\mathrm{PL},p}, \mathbf{W}), \quad \textbf{(policy update)} \tag{4}$$

$$\mathbf{X}_i^{\mathrm{PL},p+1}, \mathbf{U}_i^{\mathrm{PL},p+1} = \operatorname*{arg\,min}_{\mathbf{X}_i, \mathbf{U}_i} \; \mathcal{Q}_{\mathrm{BC}}(\mathbf{X}_i^{\mathrm{PL},p}, \mathbf{U}_i^{\mathrm{PL},p}, \mathbf{W}^{p+1}) + \frac{\rho_x}{2}\|\mathbf{X}_i^{\mathrm{TO},p+1} - \mathbf{X}_i + \boldsymbol{\lambda}_{\mathbf{X}_i}^p\|^2$$
$$+ \frac{\rho_u}{2}\|\mathbf{U}_i^{\mathrm{TO},p+1} - \mathbf{U}_i + \boldsymbol{\lambda}_{\mathbf{U}_i}^p\|^2. \quad \textbf{(primal PL update)} \tag{5}$$

Primal TO update: The update in Eq. (3) involves TO and is solved by either direct optimization methods Drnach and Zhao (2021) or indirect methods such as differential dynamic programming (DDP), as described in Tassa et al. (2014) and Jacobson and Mayne (1970). We defer details of the DDP algorithm to Appendix A in the supplementary material (Zhao et al., 2021).

Policy update: Note that Eq. (4) is a min-max optimization problem. For notation simplicity, we omit the iteration index p, and we rewrite it as

$$\mathbf{W} = \arg\min_{\mathbf{W}} \mathcal{Q}_{\mathrm{BC}}(\mathbf{X}^{\mathrm{PL}}, \mathbf{U}^{\mathrm{PL}}, \mathbf{W}) + \frac{\alpha}{N}\sum_{i,t} \max_{\|\boldsymbol{\delta}_i^t\| \le \epsilon} r(\mathbf{x}_i^{\mathrm{PL},t}, \mathbf{W}, \boldsymbol{\delta}_i^t). \tag{6}$$

To solve Eq. (6), we apply an alternating gradient descent/ascent algorithm. Specifically, at the s^{th} iteration, we first apply the projected gradient ascent algorithm to update $\boldsymbol{\delta}_i^t$ for K steps,

$$\boldsymbol{\delta}_i^{t,s} = \boldsymbol{\delta}_i^{t,s,K}, \text{ where } \boldsymbol{\delta}_i^{t,s,k} = \Pi\left[\boldsymbol{\delta}_i^{t,s,k-1} + \eta_\delta \nabla_\delta r(\mathbf{x}_i^{\mathrm{PL},t}, \mathbf{W}^s, \boldsymbol{\delta}_i^{t,s,k-1})\right] \text{ for } k = 2, \cdots, K.$$

Here, $\boldsymbol{\delta}_i^{t,s,1}$ is randomly sampled from $\mathcal{N}(0, \sigma^2\mathbb{I})$, Π denotes projection to the ℓ_2 ball with a radius ϵ, and $\eta_\delta > 0$ denotes the step size. Then we apply a gradient descent (or stochastic gradient descent) step to $\mathbf{W}$,

$$\mathbf{W}^s = \mathbf{W}^{s-1} - \eta_W[\nabla_{\mathbf{W}} \mathcal{Q}_{\mathrm{BC}}(\mathbf{X}^{\mathrm{PL}}, \mathbf{U}^{\mathrm{PL}}, \mathbf{W}^s) + \frac{\alpha}{N}\sum_{i,t} \nabla_{\mathbf{W}} r(\mathbf{x}_i^{\mathrm{PL},t}, \mathbf{W}^s, \boldsymbol{\delta}_i^{t,s})]. \tag{7}$$

Primal PL update: The update in Eq. (5) solves an unconstrained differentiable optimization subproblem, which can be efficiently solved for each trajectory using stochastic gradient descent.

Dual update: After the above three updates, we perform the dual update as follows:

$$\boldsymbol{\lambda}_{\mathbf{X}_i}^{p+1} = \boldsymbol{\lambda}_{\mathbf{X}_i}^{p} + \mathbf{X}_i^{\mathrm{TO},p+1} - \mathbf{X}_i^{\mathrm{PL},p+1}, \quad \boldsymbol{\lambda}_{\mathbf{U}_i}^{p+1} = \boldsymbol{\lambda}_{\mathbf{U}_i}^{p} + \mathbf{U}_i^{\mathrm{TO},p+1} - \mathbf{U}_i^{\mathrm{PL},p+1}. \tag{8}$$

After a certain number of iterations of the above primal-dual policy updates, the joint optimization in Eq. (2) achieves a consensus and the primal and dual residuals meet the ADMM stopping criteria. The overall algorithm is summerized in Algorithm 1 in Appendix B (Zhao et al., 2021).

3.3. Stackelberg Adversarial Regularization

One major limitation of the adversarial regularizer in Eq. (6) is that it solves a min-max-game-based optimization, where neither of the players can be advantageous. This is problematic because the adversarial player may generate over-strong perturbations that hinder generalization. To mitigate this issue, we employ Stackelberg adversarial regularization (Zuo et al., 2021) to solve the policy update in Eq. (6) through a Stackelberg game formulation. In a Stackelberg game, there are two players, a leader (the policy) and a follower (the perturbations). The leader acknowledges the strategy of the follower, such that it is always in an advantageous position. This effectively eliminates the over-strong perturbations.

To simplify the notation, we omit the indices on the trajectory sample points $\mathbf{x}$. We solve

$$\begin{aligned} \min_{\mathbf{W}} \mathcal{Q}_{\mathrm{SAR}}(\mathbf{W}) = \mathcal{Q}_{\mathrm{BC}}&(\mathbf{X}, \mathbf{U}, \mathbf{W}) \\ &+ \frac{\alpha}{N}\sum r(\mathbf{x}, \mathbf{W}, \boldsymbol{\delta}^K), \\ \text{s.t. } \boldsymbol{\delta}^K(\mathbf{W}) = U^K &\circ U^{K-1} \circ \cdots \circ U^1(\boldsymbol{\delta}^0). \end{aligned} \tag{9}$$

The policy parameter $\mathbf{W}$ in Eq. (9) is the leader, and the perturbation $\boldsymbol{\delta}(\mathbf{W})$ is the follower. Here, $\circ$ denotes operator composition, i.e., $f(\cdot) \circ g(\cdot) = f(g(\cdot))$. Each U^k for $k = 1, \cdots, K$ represents the k^{th} step update operator for the follower's strategy. The operators are defined by pre-selected optimization algorithms such as stochastic gradient descent (SGD) or Adam (Kingma and Ba, 2014).

In Stackelberg adversarial training, the leader acknowledges the strategy of the follower by treating the perturbations (the follower) as a function of the policy parameters (the leader). Correspondingly, we solve for the policy parameters using gradient descent, where the Stackelberg gradient is

$$\frac{\mathrm{d}\mathcal{Q}_{\mathrm{SAR}}(\mathbf{W})}{\mathrm{d}\mathbf{W}} = \underbrace{\frac{\mathrm{d}\mathcal{Q}_{\mathrm{BC}}(\mathbf{X},\mathbf{U},\mathbf{W})}{\mathrm{d}\mathbf{W}} + \alpha\frac{\partial r(\mathbf{x},\mathbf{W},\boldsymbol{\delta}^K)}{\partial \mathbf{W}}}_{\text{leader}} + \underbrace{\alpha\frac{\partial r(\mathbf{x},\mathbf{W},\boldsymbol{\delta}^K)}{\partial \boldsymbol{\delta}^K}\frac{\mathrm{d}\boldsymbol{\delta}^{\mathrm{K}}}{\mathrm{d}\mathbf{W}}}_{\text{leader-follower interaction}}. \tag{10}$$

In comparison, the conventional adversarial regularization in Eq. (6) uses only the leader term and does not consider the leader-follower interaction.

The most expensive term to compute in Eq. (10) is $\mathrm{d}\boldsymbol{\delta}^{\mathrm{K}}/\mathrm{d}\mathbf{W}$. Recall that we have $\boldsymbol{\delta}^k = U^k(\boldsymbol{\delta}^{k-1})$, where U^k is an update operator, e.g., a one-step gradient ascent. As a short-hand, we write

$$\boldsymbol{\delta}^k(\mathbf{W}) = \boldsymbol{\delta}^{k-1}(\mathbf{W}) + \Delta(\mathbf{x}, \boldsymbol{\delta}^{k-1}(\mathbf{W}), \mathbf{W}),$$

where $\Delta(\mathbf{x}, \boldsymbol{\delta}^{k-1}(\mathbf{W}), \mathbf{W})$ signifies the update from $\boldsymbol{\delta}^{k-1}$ to $\boldsymbol{\delta}^k$. Then we have

$$\frac{\mathrm{d}\boldsymbol{\delta}^{\mathrm{k}}}{\mathrm{d}\mathbf{W}} = \frac{\mathrm{d}\boldsymbol{\delta}^{\mathrm{k}-1}}{\mathrm{d}\mathbf{W}} + \frac{\partial\Delta(\mathbf{x},\boldsymbol{\delta}^{k-1},\mathbf{W})}{\partial\mathbf{W}} + \frac{\partial\Delta(\mathbf{x},\boldsymbol{\delta}^{k-1},\mathbf{W})}{\partial\boldsymbol{\delta}^{k-1}}\frac{\mathrm{d}\boldsymbol{\delta}^{\mathrm{k}-1}}{\mathrm{d}\mathbf{W}}.$$

This recursive differentiation can be efficiently computed using deep learning libraries, such as *PyTorch* (Paszke et al., 2019). Please refer to Zuo et al. (2021) for more details. The overall adversarial regularization algorithm is shown in Algorithm 2 in Appendix C (Zhao et al., 2021).

4. Experiments

We evaluate VERONICA on cart-pole swing-up, Kuka arm manipulation, and hopper locomotion tasks. The experiments are shown in the video[1]. We compare smoothness, generalization, and robustness of policies trained with Gaussian perturbations, conventional adversarial regularization (VERONICA-AR), and SAR (VERONICA-SAR). We do not include tangent propagation due to the excessive computational requirements to compute the Jacobian. We also demonstrate that the neural control policy is able to handle simple multi-modal dynamics for the pick and place task.

For Kuka manipulation tasks, the simulation environment is implemented in *PyBullet* (Coumans and Bai, 2016–2021). We solve for TO described in Eq. (3) using DDP implemented in *Crocoddyl* (Mastalli et al., 2020). For hopper locomotion tasks, we implement both the simulation environment and a direct TO algorithm in *Drake* (Tedrake and the Drake Development Team, 2019). The adversarially regularized policy learning algorithm is implemented in *PyTorch* (Paszke et al., 2019) and *Higher* (Grefenstette et al., 2019). The implementation details can be found in Appendix D (Zhao et al., 2021).

Policy Smoothness: We qualitatively examine the smoothness of our neural control policy by inspecting a typical policy roll-out for cart-pole swing-up and Kuka arm reaching tasks, as shown in Figure 2. Figure 2(a) shows the smoothness comparison during a cart-pole swing-up. VERONICA produced visually smoother force sequences comparing to Gaussian perturbation. Figure 2(b) displays the torque sequence of Kuka joint 2 during a reaching task. The policy trained by Gaussian perturbation generates a non-smooth torque profile around the initial position of the task, indicating that the Gaussian perturbation is not sufficient to prevent overfitting at the initial phase of the trajectory, where the torque changes relatively quickly with respect to state. In comparison, the

1. The link to the video is `https://youtu.be/2zlAC9Xs8Bg`.

VERONICA-AR and VERONICA-SAR policies produce smoother control sequences that track the baseline closely. To inspect the smoothness of the neural control policies, we plot the torque output on Kuka joint 2 against the joint angle in Figure 2(c). VERONICA successfully penalize against the non-smooth peak that appeared in the torque profile of the Gaussian perturbed policy.

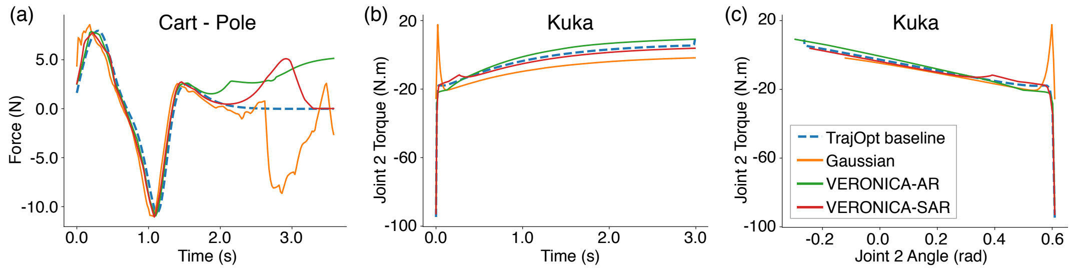

Figure 2: Comparison of control output smoothness for cart-pole and Kuka arm reaching tasks. Trajectory optimization baseline is marked as a dashed line. (a) Time sequence of forces applied onto the cart during swing-up. (b-c) Torque output for Kuka joint 2 with respect to time and joint 2 angle.

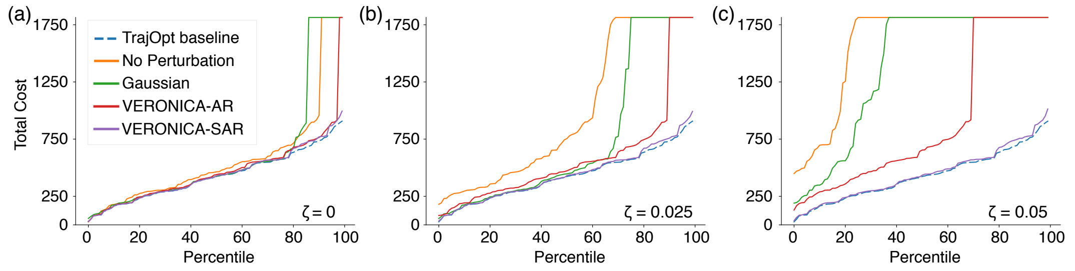

Figure 3: Cost percentile plot for 3-DOF arm reaching task with 100 different initializations and under different disturbances on sensor measurement. Disturbances are drawn from a uniform distribution bounded by ζ. Policies trained with no perturbation, Gaussian perturbation, VERONICA-AR, and VERONICA-SAR are compared against an undisturbed TO baseline. The plot is capped at 2 times the maximum baseline cost. A cost curve that exceeds the plotting cap indicates that a percentage of policy roll-outs lead to unstable robot motion.

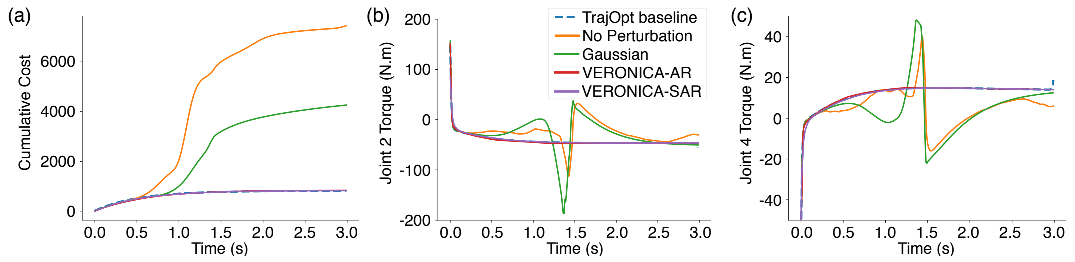

Figure 4: Example of an undisturbed policy roll-out for a 3-DOF manipulator reaching task where Gaussian perturbation fails. (a) Cumulative cost for policy roll-out (b-c) Torque outputs on joints 2 and 4.

Generalization Performance: To evaluate the generalization performances of VERONICA, we perform policy roll-outs with 100 different initializations in an undisturbed environment (see Figure 3(a)). The adversarially regularized policies produce lower costs because the policies trained with no perturbation or Gaussian perturbation are unable to generate stable robot motions under some initializations. Figure 4 displays an example of an arm reaching task that Gaussian perturbation cannot handle. Although a vast majority of roll-outs with the VERONICA-AR policy are stable, a small percentage (2%) produces unstable robot motions that fail to achieve the task. In contrast, the VERONICA-SAR policy leads to stable and near-optimal robot motions across all attempts, demonstrating that VERONICA-SAR enhances numerical stability comparing to VERONICA-AR.

We further compare VERONICA with proximal policy optimization (PPO) - a model free reinforcement learning method. PPO is trained for 3e6 timesteps. VERONICA is trained with 1000 trajectories and 300 timesteps each, matching the sample size of PPO. Figure 5 shows that VERONICA consistently achieves lower cost than PPO, demonstrating its superior sample-efficiency.

Policy Robustness: We evaluate our policies' robustness against three different kinds of disturbances. For sensor noise and environmental uncertainty, we add a uniform noise bounded by an ℓ_∞-norm ball with radius ζ onto the sensor measurement and state transition, respectively. As for model mismatch, we modify the URDF file used in policy roll-out by decreasing the mass of each robot link by 0.25 kg.

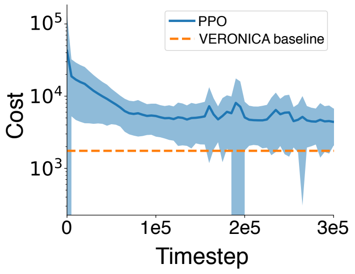

Figure 5: Learning curve of PPO in the 3-DOF Kuka reaching task. The baseline is the 95 percentile cost for VERONICA-AR across 100 different tests.

We first compare the policies' robustness against different magnitudes of sensor noise, as shown in Figure 3(b-c). While Gaussian perturbation does provide some robustness comparing to the unregularized policy, VERONICA-AR and VERONICA-SAR consistently outperforms the Gaussian perturbation. Furthermore, VERONICA-AR deviates significantly from the undisturbed TO baseline under a strong sensor noise ($\zeta = 0.05$), while VERONICA-SAR remains able to produce stable robot motion and closely track the TO baseline.

Table 1: Task Error for 3-DOF Manipulator Reaching Task ($\zeta = 0.01$, *Unit: m*)

	Gaussian	VERONICA-AR	VERONICA-SAR
Undisturbed	1.62e-1 ± 5.05e-2	6.26e-2 ± 3.96e-2	6.39e-2 ± 2.70e-2
Sensor Error	1.75e-1 ± 7.85e-2	7.11e-2 ± 7.57e-2	6.61e-2 ± 2.42e-2
Environment Uncertainty	1.73e-1 ± 7.59e-2	8.66e-2 ± 1.03e-1	7.75e-2 ± 3.99e-2
Model Mismatch	2.14e-1 ± 8.26e-2	5.24e-2 ± 3.27e-2	1.23e-1 ± 2.16e-2

Table 1 shows the average task errors - the distance between the goal and the actual final positions for the robot arm's end-effector - and their standard deviation for 100 manipulator reaching tasks under different types of disturbances. VERONICA provides significantly lower task errors across all clean and disturbed experiments. Furthermore, VERONICA-SAR leads to a lower standard deviation than VERONICA-AR, indicating that the policy learned by VERONICA-SAR is less prone to outliers comparing to VERONICA-AR.

Application to Higher-DOF Manipulators: We investigate how the performance of VERONICA-SAR scales to higher state and control dimensions by evaluating the task errors of manipulator reaching tasks for 3, 5, and 7-DOF Kuka arms (Table 2). The task error increases with the dimensionality of the problem, but not significantly. Note that the 5 and 7-DOF experiments involve manipulation in the 3-D space, which lead to much higher problem complexity than the planar 3-DOF configuration, and require larger neural control policies. Figure 6 indicates that similar to the 3-DOF cases, the proposed Stackelberg adversarial regularization benefits both generalization and robustness performance compared to Gaussian regularization in the 7-DOF Kuka arm reaching tasks.

Table 2: Median Task Errors for M-DOF Manipulator (*Unit: m*)

$M = 3$	$M = 5$	$M = 7$
6.39e-2	1.23e-1	1.32e-1

Preliminary Study of Learning Multimodal Dynamics: In the pick and place task, we train a policy to handle the control of the Kuka arm for both free-moving or object-holding scenarios. In order to train the policy applicable for both cases simultaneously, we include a discrete variable

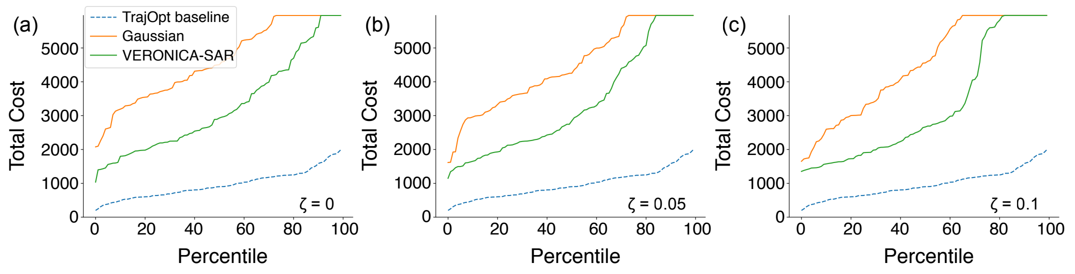

Figure 6: Cost percentile plot for 7-DOF arm reaching task with 100 different initializations and under different disturbances on sensor measurements. The plot is capped at 3 times the maximum baseline cost.

in the network input to signify the grasping state of the object. Figure 7 shows the arm's torque output for the same initialization, with or without an object. For simplicity, this experiment assumes that only one object with a known mass, and the object is fixed to a pre-specified position in the gripper when grasped by the arm. In the future, the adaptability of the policy can be improved by augmenting the input with more information such as the weight of the object and the relative position between the object and the gripper.

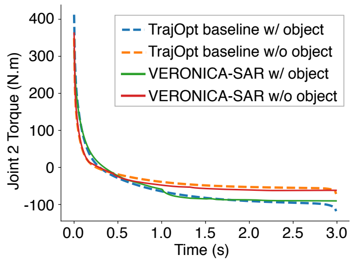

Figure 7: Comparison for the control policy outputs with or without grasping a 5kg object.

Application to Hybrid Locomotion Systems: We apply VERONICA in hopper locomotion tasks to evaluate the performance of VERONICA in a single leg 5-DOF hopper system, where the hybrid locomotion trajectories involve intermittent contacts with the terrain. We compare the cost percentile plot between the TO baseline and VERONICA-SAR, as displayed in Figure 8. Note that the open-loop rollout of trajectories generated by TO baseline performs poorly in simulation due to the model mismatch between TO and simulation environments. In contrast, the policy trained with VERONICA-SAR generates a lower cost hopper motions due to the robustness against model mismatch provided by adversarial perturbation. A visual comparison can be found in the video.

5. Conclusion

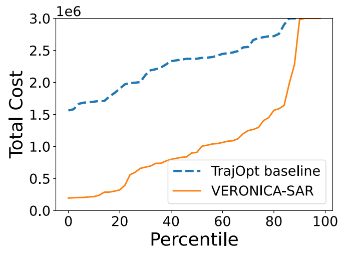

Figure 8: Cost percentile plot for hopper locomotion task with 100 different initializations.

We present VERONICA, an adversarial regularization framework for combined trajectory optimization and policy learning. We show that the proposed regularizer improves generalization and robustness by enforcing Lipschitz continuity of the policy. Additionally, we propose to further stabilize training by formulating the adversarial regularization as a Stackelberg game. The experiment results in robot manipulation scenarios show that our approach helps to improve the smoothness of the learned policy, which results in a more stable robot motions and lower policy execution costs. Additionally, we demonstrate that policies trained with VERONICA are able to robustly handle various types of disturbances.

Our future work will (i) evaluate the performance of VERONICA in the presence of more types of perturbations and uncertainties, such as varying link moment of inertia and kinematic parameters; (ii) extend VERONICA to solve more complex manipulation and locomotion problems involving physical contact and enhance robustness to contact uncertainties. Adaptive adversarial training, where perturbations are generated by an additional network, can be incorporated to generate variable perturbation radius around contact points.

Acknowledgments

This project was partially supported by the *Ford Motor Company*.

References

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR, 2018.

John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

Marcus A Brubaker, Leonid Sigal, and David J Fleet. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396. IEEE, 2009.

Chunhui Chen and Olvi L Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Computational Optimization and Applications*, 5(2):97–138, 1996.

Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Adversarial feature training for generalizable robotic visuomotor control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1142–1148. IEEE, 2020.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

Luke Drnach and Ye Zhao. Robust trajectory optimization over uncertain terrain with stochastic complementarity. *IEEE Robotics and Automation Letters*, 6(2):1168–1175, 2021.

Alexis Duburcq, Yann Chevaleyre, Nicolas Bredeche, and Guilhem Boéris. Online trajectory planning through combined trajectory optimization and function approximation: Application to the exoskeleton atalante. In *IEEE International Conference on Robotics and Automation*, 2020.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*, 2019.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4572–4580, 2016.

David H Jacobson and David Q Mayne. Differential dynamic programming. 1970.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

Benoit Landry, Hongkai Dai, and Marco Pavone. Seagul: Sample efficient adversarially guided learning of value functions. In *Learning for Dynamics and Control*, pages 1–13. PMLR, 2021.

Mathias Lechner, Ramin Hasani, Radu Grosu, Daniela Rus, and Thomas A Henzinger. Adversarial training is not ready for robot learning. *arXiv preprint arXiv:2103.08187*, 2021.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International conference on machine learning*, pages 1–9. PMLR, 2013a.

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. *Advances in neural information processing systems*, 26:207–215, 2013b.

Sergey Levine and Vladlen Koltun. Learning complex neural network policies with trajectory optimization. In *International Conference on Machine Learning*, pages 829–837. PMLR, 2014.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Igor Mordatch and Emo Todorov. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems*, volume 4, 2014.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. In *NIPS*, pages 1061–1067. Citeseer, 2000.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

Emanuel Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 1071–1076. IEEE, 2011.

Emanuel Todorov, Tom Erez, and Yuval 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.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019.

Tian Xu, Ziniu Li, and Yang Yu. On value discrepancy of imitation learning. *arXiv preprint arXiv:1911.07027*, 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

Zhigen Zhao, Ziyi Zhou, Michael Park, and Ye Zhao. Sydebo: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments. *arXiv preprint arXiv:2010.11078*, 2020.

Zhigen Zhao, Simiao Zuo, Tuo Zhao, and Ye Zhao. Supplementary material for adversarially regularized policy learning guided by trajectory optimization, 2021. URL https://tinyurl.com/3cnyubv9.

Ziyi Zhou and Ye Zhao. Accelerated admm based trajectory optimization for legged locomotion with coupled rigid body dynamics. In *American Control Conference*, pages 5082–5089, 2020.

Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarej, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*, 2019.

Simiao Zuo, Chen Liang, Haoming Jiang, Xiaodong Liu, Pengcheng He, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. Adversarial training as stackelberg game: An unrolled optimization approach. *arXiv preprint arXiv:2104.04886*, 2021.

Supplemental Materials

Appendix A. Differential Dynamic Programming

In order to generate each individual trajectory sample satisfying robot rigid body dynamics, we solve the following trajectory optimization (TO) problem formulated as:

$$\min_{\mathbf{X},\mathbf{U}} \quad \mathcal{L}(\mathbf{X},\mathbf{U}) = \sum_{t=1}^{T-1} \ell(\mathbf{x}^t,\mathbf{u}^t) + \ell_f(\mathbf{x}^T,\mathbf{u}^T) \tag{11a}$$

$$\text{s.t.} \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t,\mathbf{u}^t), \mathbf{x}^0 = \mathbf{x}_{\text{init}}, \tag{11b}$$

$$\mathbf{X} \in \mathcal{X},\ \mathbf{U} \in \mathcal{U}, \tag{11c}$$

where $\ell(\mathbf{x}^t,\mathbf{u}^t)$ is the cost function at time-step t, $\ell_f(\mathbf{x}^T,\mathbf{u}^T)$ represents the terminal trajectory cost at time-step T, $\mathbf{x}^{t+1} = f(\mathbf{x}^t,\mathbf{u}^t)$ is the discretized system dynamics, and $\mathcal{X},\mathcal{U}$ represents additional path constraints on state and control. The running trajectory cost $\ell(\mathbf{x},\mathbf{u})$ is composed of the a goal tracking term, a control regularization term, and the ADMM residual terms:

$$\ell(\mathbf{x},\mathbf{u}) = \hat{\mathbf{x}}^\top \mathbf{Q}\hat{\mathbf{x}} + \mathbf{u}^\top \mathbf{R}\mathbf{u} + \frac{\rho_x}{2}\|\mathbf{x} - \mathbf{x}^{\text{PL}} + \boldsymbol{\lambda}_{\mathbf{x}}\|^2 + \frac{\rho_u}{2}\|\mathbf{u} - \mathbf{u}^{\text{PL}} + \boldsymbol{\lambda}_{\mathbf{u}}\|^2,$$

where $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{\text{goal}}$ represents the deviation between the trajectory state $\mathbf{x}$ and goal state $\mathbf{x}_{\text{goal}}$ and $\mathbf{Q},\mathbf{R} \succeq 0$ are the weighting matrices for the strength of the regularization. The ADMM residual terms $\frac{\rho_x}{2}\|\mathbf{x} - \mathbf{x}^{\text{PL}} + \boldsymbol{\lambda}_{\mathbf{x}}\|^2$ and $\frac{\rho_u}{2}\|\mathbf{u} - \mathbf{u}^{\text{PL}} + \boldsymbol{\lambda}_{\mathbf{u}}\|^2$ are initialized to be 0 at the first iteration, but eventually have the effect of regularizing the trajectory optimization to be closer to the policy output.

In the following we briefly describe the formulation of DDP, which is used in this work to compute trajectory samples. Jacobson and Mayne (1970) provides a detailed representation of DDP in the historical context, and Tassa et al. (2014) presents a control-constrained version of DDP that is widely used in robotics.

DDP solves the optimization described in Eq. (11) using a backward pass of Bellman's equation,

$$V(\mathbf{x}^t) = \min_{\mathbf{u}}[\ell(\mathbf{x}^t,\mathbf{u}^t) + V(\mathbf{x}^{t+1})]. \tag{12}$$

Let $Q(\delta\mathbf{x}^t,\delta\mathbf{u}^t)$ be the change in local cost function given a perturbation around the tth time-step:

$$Q(\delta\mathbf{x},\delta\mathbf{u}) = \ell(\mathbf{x}+\delta\mathbf{x},\mathbf{u}+\delta\mathbf{u}) - \ell(\mathbf{x},\mathbf{u}) + V(\mathbf{x}+\delta\mathbf{x}) - V(\mathbf{x}) \tag{13}$$

The DDP backward pass computes the second order Taylor expansion of Q and the optimal local perturbation $\delta\mathbf{u}^*$ is given by the local feedback control policy:

$$\delta\mathbf{u}^* = \mathbf{k} + \mathbf{K}\delta\mathbf{x}, \tag{14}$$

where $\mathbf{k} = -Q_{uu}^{-1}Q_u$ and $\mathbf{K} = -Q_{uu}^{-1}Q_{ux}$. After the backward pass is completed, the DDP forward pass simulates the system by rolling out the system dynamics $\mathbf{x}^{t+1} = f(\mathbf{x}^t,\mathbf{u}^t)$. The backward-forward passes are iterated until convergence.

Appendix B. Algorithm Overview of the Proposed Trajectory Optimization Guided by Adversarially Regularized Policy Learning

Algorithm 1 shows the complete procedure of jointly solving TO and policy learning using ADMM.

Algorithm 1 TO-Guided Policy Learning Using ADMM

```
Input: $P$: total number of ADMM iterations; $N$: number of sample trajectories.
  $\mathbf{X}_{\text{init}} \leftarrow N$ trajectory initial conditions
  $\boldsymbol{\lambda}^0_{\mathbf{X}}, \boldsymbol{\lambda}^0_{\mathbf{U}} \leftarrow 0$
  for $p = 1, \cdots, P$ do
      $\mathbf{X}^{\text{TO},p}, \mathbf{U}^{\text{TO},p} \leftarrow$ compute N trajectories using Eq. (3)    (primal TO update)
      $\mathbf{W}^p \leftarrow$ solve min-max optimization in Eq. (4) using Algorithm 2    (policy update)
      $\mathbf{X}^{\text{PL},p}, \mathbf{U}^{\text{PL},p} \leftarrow$ optimize using Eq. (5)    (primal PL update)
      $\boldsymbol{\lambda}^p_{\mathbf{X}}, \boldsymbol{\lambda}^p_{\mathbf{U}} \leftarrow$ update using Eq. (8)    (dual update)
  end for
  return $\mathbf{W}^P$
```

Appendix C. Pseudo-Code for Adversarial Regularization

Algorithm 2 Adversarially Regularized Policy Learning.

```
Input: $\{\mathbf{X}, \mathbf{U}\}$: trajectory samples; $E$: number of epochs; $K$: number of perturbation updates.
  for epoch $= 1, \cdots, E$ do
      for $\{\mathbf{x}, \mathbf{u}\} \in \{\mathbf{X}, \mathbf{U}\}$ do
          Initialize $\boldsymbol{\delta}^0 \sim \mathcal{N}(0, \sigma^2\mathbb{I})$
          for $k = 1, \cdots, K$ do
              Compute $\mathrm{d}\mathcal{R}_{\text{adv}}/\mathrm{d}\boldsymbol{\delta}^{\text{k}-1}$
              $\boldsymbol{\delta}^k \leftarrow \text{Optimizer}(\mathrm{d}\mathcal{R}_{\text{adv}}/\mathrm{d}\boldsymbol{\delta}^{k-1})$
          end for
          Adv Reg:
              Compute $\mathrm{d}(\mathcal{Q}_{\text{BC}} + \mathcal{R}_{\text{adv}})/\mathrm{d}\mathbf{W}$
              Update $\mathbf{W}$ using (7)
          Stackelberg Adv Reg:
              Compute $\mathrm{d}\mathcal{Q}_{\text{SAR}}/\mathrm{d}\mathbf{W}$ using (10)
              $\mathbf{W} \leftarrow \text{Optimizer}(\mathrm{d}\mathcal{Q}_{\text{SAR}}/\mathrm{d}\mathbf{W})$
      end for
  end for
```

Appendix D. Implementation Details

We use a fully connected neural network with 2 hidden layers and 8 units per layer for the cart-pole example. The 3-DOF Kuka arm uses 2 hidden layers and 64 units each, the 5-DOF manipulator uses 3 hidden layers and 64 units each, while the 7-DOF manipulator uses a residual network with 3 hidden layers and 256 units. The Kuka arm simulation is illustrated in Figure 9. The hopper example, as shown in Figure 10, uses a fully connected network with 3 hidden layers and 24 units each.

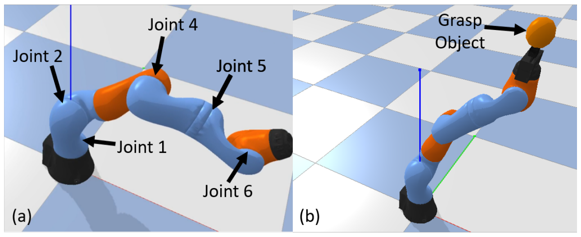

Figure 9: The Kuka arm manipulation scenarios in simulation. (a) Kuka IIWA arm reaching: the learned policy controls the arm to reach a predefined joint configuration. In 3-DOF reaching experiments, only joints 2, 4, and 6 are active degrees-of-freedom (DOFs), making the arm equivalent to a planar manipulator. In 5-DOF experiments, joints 1, 2, 4, 5, and 6 are active DOFs; (b) The Kuka arm pick and place task: an additional object is grasped by the Kuka arm during this task.

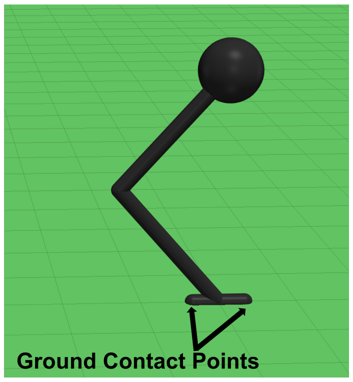

Figure 10: The hopper locomotion tasks in simulation. The single leg hopper has 5 degree-of-freedom, with two contact points with the ground located at the heel and the toe of the hopper.

In all experiments, we train the networks using AdamW (Loshchilov and Hutter, 2017) for policy optimization and stochastic gradient descent (SGD) for adversarial perturbation. The regularization coefficient α is set to 1. The learning rate for the policy learning lr_p is chosen between $\{1\text{e-}3, 5\text{e-}4\}$, and the learning rate for adversarial perturbation $\mathrm{lr}_{\mathrm{adv}}$ is chosen between $\{5\text{e-}4, 1\text{e-}4\}$. The number of adversarial update steps K is selected from $\{1, 3\}$, and the adversarial bound ϵ is chosen from $\{1\text{e-}2, 5\text{e-}3\}$. The policy is trained for at most 300 epochs, with model averaging in the last 1/4 of total epochs. Also, we apply gradient norm clipping of $\{\infty, 1\}$.

In ADMM, we apply a trajectory state penalty coefficient ρ_x of $\{1, 10, 50\}$ and a trajectory control penalty coefficient ρ_u of 1. We find that the behavioral cloning loss $\mathcal{Q}_{\mathrm{BC}}$ decreases over ADMM iterations, but the loss deduction is not significant after 5-10 iterations. Therefore, the ADMM is run until $\mathcal{Q}_{\mathrm{BC}}$ stops decreasing, which results in between 5-15 iterations in our experiments. The result for $\mathcal{Q}_{\mathrm{BC}}$ plotted with respect to ADMM iterations can be found in Appendix E.

D.1. 3-DOF Kuka Experiments

We use a fully connected network with 2 hidden layers and 64 units in each layer. The policy input for the reaching task is 9-dimensional, which consists of a 6-dimensional robot state and a 3-dimensional goal configuration. The policy input for the pick and place task is 10-dimensional, with 1 additional input dimension encoding the grasp state. The learning rate for policy parameters

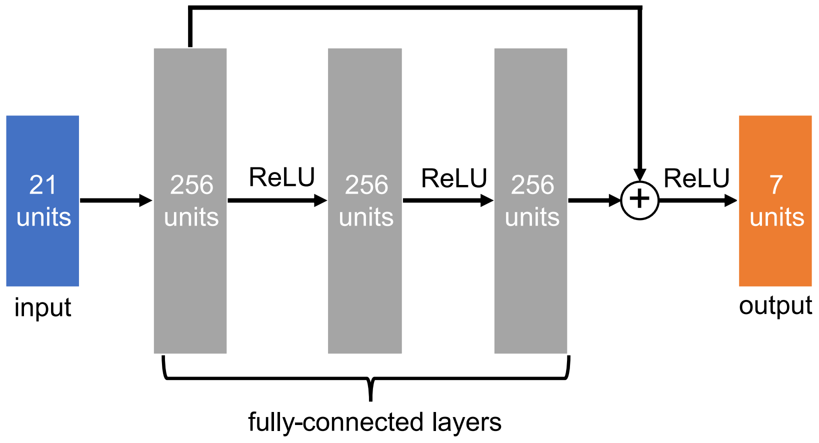

Figure 11: Illustration of the residual network used for 7-DOF Kuka manipulator experiments. The network consists of 3 hidden layers with 256 units each. A skip connection is included from the output of the 1$^{\text{st}}$ hidden layer to the output of the 3$^{\text{rd}}$ hidden layer.

lr_p is set to 1e-3, and the learning rate for adversarial perturbation lr_{adv} is set to 5e-3. The number of adversarial update step K is selected to be 1, and the adversarial bound ϵ is 5e-3. We use $N = 5000$ trajectory samples with 300 timesteps each. The policy is trained for 300 epochs, with model averaging in the last 75 epochs.

The PPO algorithm as compared in Figure 5 is implemented using *Stable Baseline 3* (Raffin et al., 2019).

D.2. 7-DOF Kuka Experiment

We use a residual network with 3 hidden layers (Figure 11) to learn the neural control policy for the 7-DOF Kuka experiment. The policy input is 21-dimensional, which consists of a 14-dimensional robot state and a 7-dimensional target joint angles. The learning rate for policy parameters lr_p is set to 1e-3, and the learning rate for adversarial perturbation lr_{adv} is set to 1e-4. The number of adversarial update step K is selected to be 1, and the adversarial bound ϵ is 5e-3. We apply a gradient norm clipping of 1. We use $N = 25000$ trajectory samples with 200 timesteps each. The policy is trained for 100 epochs, with model averaging in the last 25 epochs.

Appendix E. Policy Behavioral Cloning Loss Over ADMM Iterations

Figure 12 shows the behavioral cloning loss $\mathcal{Q}_{\text{BC}}$ plotted against ADMM iterations. In the cart-pole experiment shown in Figure 12(a), $\mathcal{Q}_{\text{BC}}$ decreases in the first 15 iterations, and gradually increases afterwards. In Kuka experiment (Figure 12(b)), $\mathcal{Q}_{\text{BC}}$ is improved significantly in the first 2 iterations, then only slowly decreases from the 3$^{\text{rd}}$ iteration onward.

Appendix F. Effects of Adversarial Perturbation Bound Value

We evaluate the effect of adversarial perturbation bound ϵ by comparing the 3-DOF Kuka arm policies trained by VERONICA-SAR with a set of perturbation values $\epsilon \in \{0, 0.005, 0.01, 0.025, 0.05\}$. As seen in Figure 13, $\epsilon \in \{0.005, 0.01\}$ provides the best performances and closely track the TO

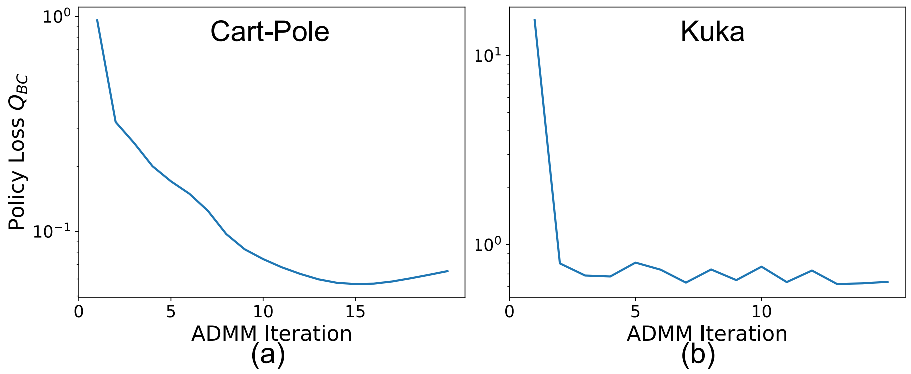

Figure 12: The behavioral cloning losses $\mathcal{Q}_{\mathrm{BC}}$ with respect to ADMM iterations. The policies are trained with VERONICA-SAR for (a) cart-pole and (b) 3-DOF Kuka manipulator.

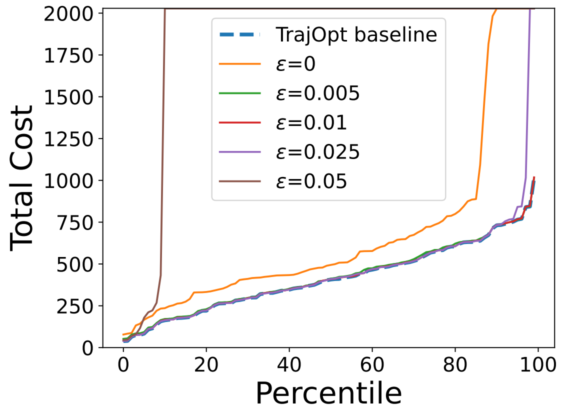

Figure 13: Cost percentile plot for 3-DOF arm policy rollout with 100 different initializations. The policies are trained with different adversarial perturbation bounds ϵ

.

baseline. $\epsilon = 0$ is equivalent to the policy trained without perturbation, which does not enjoy the generalization and robustness gains provided by VERONICA. In contrast, the policy performance decreases significantly when $\epsilon > 0.025$, indicating that the adversarial perturbation is too strong and causes underfitting.

Appendix G. Theoretical Analysis on Policy Smoothness and Robustness

In this section, we provide a theoretical analysis on how the Lipschitz continuity improves a neural control policy's robustness. We evaluate the policy's robustness against state disturbances via value discrepancy propagation analysis, as described in Xu et al. (2019), where the policy robustness is analyzed by studying how the error caused by state disturbance propagates in the value functions of the policy. As shown in Appendix G.3, the upper bound of the policy robustness (measured by value function discrepancy) is proportional to the Lipschitz constant of the policy. Therefore, controlling the Lipschitz continuity of the policy helps to improve its robustness.

We make the assumption that the poilcy π, the cost function $\ell(\mathbf{x}, \mathbf{u})$, and the system dynamics $f(\mathbf{x}, \mathbf{u})$ are globally Lipschitz continuous. Although these assumptions might not hold in all practical cases, the following discussion provides some insight and intuition about why controlling the smoothness of the policy enhances its robustness against various disturbances.

G.1. Definitions

$\pi(\cdot|\mathbf{W})$ denotes a neural control policy with network parameters $\mathbf{W}$. For notation simplicity, $\mathbf{W}$ are omitted in the following discussion. Let $\ell_\pi(\mathbf{x}^{(t)}) = \ell(\mathbf{x}^{(t)}, \pi(\mathbf{x}^{(t)}))$ denote the cost for policy π at state $\mathbf{x}^{(t)}$ on time-step t. Similarly, $f_\pi(\mathbf{x}^{(t)}) = f(\mathbf{x}^{(t)}, \pi(\mathbf{x}^{(t)}))$ represents the system dynamics under policy π at state $\mathbf{x}^{(t)}$. We define the value function J_π of policy $\pi(\mathbf{x})$ to be the infinite horizon cost with a discount factor $\gamma \in (0, 1)$,

$$J_\pi(\mathbf{x}^{(0)}) = \sum_{t=0}^{\infty} \gamma^t \ell_\pi(\mathbf{x}^{(t)}).$$

We consider the discount factor for convenience of analysis. The results can be extended to the average cost setting, but will be more involved.

The Lipschitz constant of π, ℓ_π, f_π, and J_π are denoted as C_π, C_{ℓ_π}, C_{f_π}, and C_{J_π} respectively. $C_\ell^{\mathbf{u}}$ and $C_f^{\mathbf{u}}$ represents the Lipschitz constant of $\ell(\mathbf{x}, \mathbf{u})$ and $f(\mathbf{x}, \mathbf{u})$ with respect to $\mathbf{u}$.

G.2. Lipschitz Continuity of Value Function

Lemma 1: *Given a neural control policy π with Lipschitz continuous cost function ℓ_π and dynamics f_π, and let $\gamma C_{f_\pi} < 1$. The value function J_π is Lipschitz continuous and the Lipschitz constant is $C_{J_\pi} = \frac{C_{\ell_\pi}}{1-(\gamma C_{f_\pi})^t}$.*

Proof:

$$\begin{aligned}
&\|J_\pi(\mathbf{x}^{(0)}) - J_\pi(\mathbf{y}^{(0)})\| \\
&= \sum_{t=0}^{\infty} \gamma^t \|\ell_\pi(f_\pi(\mathbf{x}^{(t)})) - \ell_\pi(f_\pi(\mathbf{y}^{(t)}))\| \\
&\le \sum_{t=0}^{\infty} C_{\ell_\pi} \gamma^t \|f_\pi(\mathbf{x}^{(0)}) - f_\pi(\mathbf{y}^{(0)})\| \\
&\le (\sum_{t=0}^{\infty} (\gamma C_{f_\pi})^t) C_{\ell_\pi} \|\mathbf{x}^{(0)} - \mathbf{y}^{(0)}\| \\
&= \frac{C_{\ell_\pi}}{1-(\gamma C_{f_\pi})^t} \|\mathbf{x}^{(0)} - \mathbf{y}^{(0)}\|
\end{aligned}$$

G.3. Value Discrepancy Under State Disturbances

Lemma 2 below shows that the value discrepancy for a policy π caused by a norm bounded perturbation is proportional to the Lipschitz constant of the policy.

Lemma 2: *Given a neural control policy π and let $\boldsymbol{\delta}^{(t)}$ be the state disturbance at time-step t norm bounded by $\|\boldsymbol{\delta}^{(t)}\| \leq \zeta$. Let $\pi'(\mathbf{x}^{(t)}) = \pi(\mathbf{x}^{(t)} + \boldsymbol{\delta}^{(t)})$ denote the disturbed neural control policy. The discrepancy between value functions $J_{\pi'}$ and J_π has an upper bound of $C_\pi(\frac{C_\ell^u + \gamma C_{J_\pi} C_f^u}{1-\gamma})\zeta$.*

Proof:

The value function J_π satisfies:

$$J_\pi(\mathbf{x}) = \ell_\pi(\mathbf{x}) + \gamma J_\pi(f_\pi(\mathbf{x})).$$

Therefore, the value discrepancy due to disturbances $\boldsymbol{\delta}$ can be written as the following:

$$\begin{aligned}
&J_{\pi'}(\mathbf{x}) - J_\pi(\mathbf{x}) \\
&= \ell_{\pi'}(\mathbf{x}) - \ell_\pi(\mathbf{x}) + \gamma(J_{\pi'}(f_{\pi'}(\mathbf{x})) - J_\pi(f_\pi(\mathbf{x}))) \\
&\leq C_\ell^{\mathbf{u}}\|\pi'(\mathbf{x}) - \pi(\mathbf{x})\| + \gamma(J_\pi(f_{\pi'}(\mathbf{x})) - J_\pi(f_\pi(\mathbf{x}))) + \gamma(J_{\pi'}(f_{\pi'}(\mathbf{x})) - J_\pi(f_{\pi'}(\mathbf{x}))) \\
&\leq C_\ell^{\mathbf{u}} C_\pi \zeta + \gamma C_{J_\pi} C_f^{\mathbf{u}} C_\pi \zeta + \gamma(J_{\pi'}(f_{\pi'}(\mathbf{x})) - J_\pi(f_{\pi'}(\mathbf{x}))) \qquad \textbf{(by \textit{Lemma 1})} \\
&\leq C_\pi(\frac{C_\ell^{\mathbf{u}} + \gamma C_{J_\pi} C_f^{\mathbf{u}}}{1-\gamma})\zeta.
\end{aligned}$$