Truly Deterministic Policy Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we present a policy gradient method that avoids exploratory noise injection and performs policy search over the deterministic landscape. By avoiding noise injection all sources of estimation variance can be eliminated in systems with deterministic dynamics (up to the initial state distribution). Since deterministic policy regularization is impossible using traditional non-metric measures such as the KL divergence, we derive a Wasserstein-based quadratic model for our purposes. We state conditions on the system model under which it is possible to establish a monotonic policy improvement guarantee, propose a surrogate function for policy gradient estimation, and show that it is possible to compute exact advantage estimates if both the state transition model and the policy are deterministic. Finally, we describe two novel robotic control environments—one with non-local rewards in the frequency domain and the other with a long horizon (8000 time-steps)—for which our policy gradient method (TDPO) significantly outperforms existing methods (PPO, TRPO, DDPG, and TD3). Our implementation with all the experimental settings is available at https://anonymous.4open.science/r/code_tdpo-D23A.

Policy Gradient (PG) methods can be broadly characterized by three defining elements: the policy gradient estimator, the regularization measures, and the exploration profile. For gradient estimation, episodic [46], importance-sampling-based [38], and deterministic [42] gradients are some of the most common estimation oracles. As for regularization measures, either an Euclidean distance within the parameter space [46, 42, 28], or dimensionally consistent non-metric measures [38, 19, 40, 20, 47] have been frequently adapted. Common exploration profiles include Gaussian [38] and stochastic processes [28]. These elements form the basis of many model-free and stochastic policy optimization methods successfully capable of learning high-dimensional policy parameters.

Both stochastic and deterministic policy search can be useful in applications. A stochastic policy has the effect of smoothing or filtering the policy landscape, which is desirable for optimization. Searching through stochastic policies has enabled the effective control of challenging environments under a general framework [38, 40]. The same method could either learn robotic movements or play basic games (1) with minimal domain-specific knowledge, (2) regardless of function approximation classes, and (3) with less human intervention (ignoring reward engineering and hyper-parameter tuning) [8]. Using stochasticity for exploration, although it imposes approximations and variance, has provided a robust way to actively search for higher rewards. Despite many successes, there are practical environments which remain challenging for current policy gradient methods. For example, non-local rewards (e.g., those defined in the frequency domain), long time horizons, and naturally-resonant environments all occur in realistic robotic systems [27, 30, 36] but can present issues for policy gradient search.

To tackle challenging environments such as these, this paper considers policy gradient methods based on deterministic policies and deterministic gradient estimates, which could offer advantages by allowing the estimation of global reward gradients on long horizons without the need to inject noise into the system for exploration. To facilitate a dimensionally consistent and low-variance deterministic

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

policy search, a compatible policy gradient estimator and a metric measure for regularization should be employed. For gradient estimation we focus on Vine estimators [38], which can be easily applied to deterministic policies. As a metric measure we use the Wasserstein distance, which can measure meaningful distances between deterministic policy functions that have non-overlapping supports (in contrast to the Kullback-Liebler (KL) divergence and the Total Variation (TV) distance).

The Wasserstein metric has seen substantial recent application in a variety of machine-learning domains, such as the successful stable learning of generative adversarial models [3]. Theoretically, this metric has been studied in the context of Lipschitz-continuous Markov decision processes in reinforcement learning [16, 9]. Pirotta et al. [35] defined a policy gradient method using the Wasserestein distance by relying on Lipschitz continuity assumptions with respect to the policy gradient itself. Furthermore, for Lipschitz-continuous Markov decision processes, Asadi et al. [4] and Rachelson and Lagoudakis [37] used the Wasserstein distance to derive model-based value-iteration and policy-iteration methods, respectively. On a more practical note, Pacchiano et al. [33] utilized Wasserstein regularization for behavior-guided stochastic policy optimization. Moreover, Abdullah et al. [1] has proposed another robust stochastic policy gradient formulation. Estimating the Wasserstein distance for general distributions is more complicated than typical KL-divergences [44]. This fact constitutes and emphasizes the contributions of Abdullah et al. [1] and Pacchiano et al. [33]. However, for our deterministic observation-conditional policies, closed-form computation of Wasserstein distances is possible without any approximation.

Existing deterministic policy gradient methods (e.g., DDPG and TD3) use *deterministic policies* [42, 28, 10], meaning that they learn a deterministic policy function from states to actions. However, such methods still use *stochastic search* (i.e., they add stochastic noise to their deterministic actions to force exploration during policy search). In contrast, we will be interested in a method which not only uses *deterministic policies*, but also uses *deterministic search* (i.e., without constant stochastic noise injection). We call this method *truly deterministic policy optimization* (TDPO) and it may have lower estimation variances and better scalability to long horizons, as we will show in numerical examples.

Scalability to long horizons is one of the most challenging aspects for policy gradient methods that use stochastic search. This issue is sometimes referred to as the *curse of horizon* in reinforcement learning [29]. General worst-case analyses suggests that the sample complexity of reinforcement learning is exponential with respect to the horizon length [21, 25, 24]. Deriving polynomial lower-bounds for the sample complexity of reinforcement learning methods is still an open problem [18]. Lower-bounding the sample complexity of reinforcement learning for long horizons under different settings and simplifying assumptions has been a topic of theoretical research [6, 45]. Some recent work has examined the scalability of importance sampling gradient estimators to long horizons in terms of both theoretical and practical estimator variances [29, 22, 23]. All in all, long horizons are challenging for all reinforcement learning methods, especially the ones suffering from excessive estimation variance due to the use of stochastic policies for exploration, and our truly deterministic method may have advantages in this respect.

In this paper we focus on continuous-domain robotic environments with reset capability to previously visited states. The main contributions of this work are: (1) we introduce a Deterministic Vine (DeVine) policy gradient estimator which avoids constant exploratory noise injection; (2) we derive a novel deterministically-compatible surrogate function and provide monotonic payoff improvement guarantees; (3) we show how to use the DeVine policy gradient estimator with the Wasserstein-based surrogate in a practical algorithm (TDPO: Truly Deterministic Policy Optimization); (4) we illustrate the robustness of the TDPO policy search process in robotic control environments with non-local rewards, long horizons, and/or resonant frequencies.

1 Background

MDP preliminaries. An infinite-horizon discounted Markov decision process (MDP) is specified by $(\mathcal{S}, \mathcal{A}, P, R, \mu, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics, $R : \mathcal{S} \times \mathcal{A} \to [0, R_{\max}]$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\mu(s)$ is the initial state distribution of interest (where $\Delta(\mathcal{F})$ denotes the set of all probability distributions over $\mathcal{F}$, otherwise known as the Credal set of $\mathcal{F}$). The transition dynamics P is defined as an operator which produces a distribution over the state space for the next state $s' \sim P(s, a)$. The transition dynamics can be easily generalized to take distributions of states or actions as input

(i.e., by having P defined as $P : \Delta(\mathcal{S}) \times \mathcal{A} \to \Delta(\mathcal{S})$ or $P : \mathcal{S} \times \Delta(\mathcal{A}) \to \Delta(\mathcal{S})$). We may abuse the notation and replace δ_s and δ_a by s and a, where δ_s and δ_a are the deterministic distributions concentrated at the state s and action a, respectively. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ specifies a distribution over actions for each state, and induces trajectories from a given starting state s as follows: $s_1 = s$, $a_1 \sim \pi(s_1)$, $r_1 = R(s_1, a_1)$, $s_2 \sim P(s_2, a_2)$, $a_2 \sim \pi(s_2)$, etc. We will denote trajectories as state-action tuples $\tau = (s_1, a_1, s_2, a_2, \ldots)$. One can generalize the dynamics (1) by using a policy instead of an action distribution $\mathbb{P}(\mu_s, \pi) := \mathbb{E}_{s\sim\mu_s}[\mathbb{E}_{a\sim\pi(s)}[P(s, a)]]$, and (2) by introducing the t-step transition dynamics recursively as $\mathbb{P}^t(\mu_s, \pi) := \mathbb{P}(\mathbb{P}^{t-1}(\mu_s, \pi), \pi)$ with $\mathbb{P}^0(\mu_s, \pi) := \mu_s$, where μ_s is a distribution over $\mathcal{S}$. The visitation frequency can be defined as $\rho^\pi_\mu := (1-\gamma)\sum_{t=1}^\infty \gamma^{t-1}\mathbb{P}^{t-1}(\mu, \pi)$. Table 2 of the Supplementary Material summarizes all MDP notation.

The value function of π is defined as $V^\pi(s) := \mathbb{E}[\sum_{t=1}^\infty \gamma^{t-1} r_t \mid s_1 = s; \pi]$. Similarly, one can define $Q^\pi(s, a)$ by conditioning on the first action. The advantage function can then be defined as their difference (i.e. $A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s)$). Generally, one can define the advantage/value of one policy with respect to another using $A^\pi(s, \pi') := \mathbb{E}[Q^\pi(s, a) - V^\pi(s) \mid a \sim \pi'(\cdot|s)]$ and $Q^\pi(s, \pi') := \mathbb{E}[Q^\pi(s, a) \mid a \sim \pi'(\cdot|s)]$. Finally, the payoff of a policy $\eta_\pi := \mathbb{E}[V^\pi(s); s \sim \mu]$ is the average value over the initial states distribution of the MDP.

Probabilistic and mathematical notations. Sometimes we refer to $\int f(x)g(x)dx$ integrals as $\langle f, g\rangle_x$ Hilbert space inner products. Assuming that ζ and ν are two probabilistic densities, the Kulback-Liebler (KL) divergence is $D_{\mathrm{KL}}(\zeta|\nu) := \langle\zeta(x), \log(\frac{\zeta(x)}{\nu(x)})\rangle_x$, the Total-Variation (TV) distance is $\mathrm{TV}(\zeta, \nu) =: \frac{1}{2}\langle|\zeta(x) - \nu(x)|, 1\rangle_x$, and the Wasserstein distance is $W(\zeta, \nu) = \inf_{\gamma\in\Gamma(\zeta,\nu)}\langle\|x - y\|, \gamma(x, y)\rangle_{x,y}$ where $\Gamma(\zeta, \nu)$ is the set of couplings for ζ and ν. We define $\mathrm{Lip}(f(x, y); x) := \sup_x \|\nabla_x f(x, y)\|_2$ and assume the existence of $\mathrm{Lip}(Q^\pi(s, a); a)$ and $\|\mathrm{Lip}(\nabla_s Q^\pi(s, a); a)\|_2$ constants. Under this notation, the Rubinstein-Kantrovich (RK) duality states that the $|\langle\zeta(x) - \nu(x), f(x)\rangle_x| \le W(\zeta, \nu) \cdot \mathrm{Lip}(f; x)$ bound is tight for all f. For brevity, we may abuse the notation and denote $\sup_s W(\pi_1(\cdot|s), \pi_2(\cdot|s))$ with $W(\pi_1, \pi_2)$ (and similarly for other measures). For parameterized policies, we define $\nabla_\pi f(\pi) := \nabla_\theta f(\pi)$ where π is parameterized by the vector θ. Table 1 of the Supplementary Material summarizes all these mathematical definitions.

Policy gradient preliminaries. The advantage decomposition lemma provides insight into the relationship between payoff improvements and advantages [19]. That is,

$$\eta_{\pi_2} - \eta_{\pi_1} = \frac{1}{1-\gamma} \cdot \mathbb{E}_{s\sim\rho^{\pi_2}_\mu}[A^{\pi_1}(s, \pi_2)]. \tag{1}$$

We will denote the current and the candidate next policy as π_1 and π_2, respectively. Taking derivatives of both sides with respect to π_2 at π_1 yields

$$\nabla_{\pi_2}\eta_{\pi_2} = \frac{1}{1-\gamma}\left[\langle\nabla_{\pi_2}\rho^{\pi_2}_\mu(\cdot), A^{\pi_1}(\cdot, \pi_1)\rangle + \langle\rho^{\pi_1}_\mu(\cdot), \nabla_{\pi_2}A^{\pi_1}(\cdot, \pi_2)\rangle\right]. \tag{2}$$

Since π_1 does not have any advantage over itself (i.e., $A^{\pi_1}(\cdot, \pi_1) = 0$), the first term is zero. Thus, the Policy Gradient (PG) theorem is derived as

$$\nabla_{\pi_2}\eta_{\pi_2}\Big|_{\pi_2=\pi_1} = \frac{1}{1-\gamma} \cdot \mathbb{E}_{s\sim\rho^{\pi_1}_\mu}[\nabla_{\pi_2}A^{\pi_1}(s, \pi_2)]\Big|_{\pi_2=\pi_1}. \tag{3}$$

For policy iteration with function approximation, we assume π_2 and π_1 to be parameterized by θ_2 and θ_1, respectively. One can view the PG theorem as a Taylor expansion of the payoff at θ_1.

A brief introduction of the Conservative Policy Iteration (CPI) [19], the Trust Region Policy Optimization (TRPO) [38], the Proximal Policy Optimization (PPO) [39], the Deep Deterministic Policy Gradient (DDPG) [28], and the Twin-Delayed Deterministic Policy Gradient (TD3) [10] policy gradient methods is left to the Supplementary Material. Whether using deterministic policy gradients (e.g., DDPG and TD3) or stochastic policy gradients (e.g., TRPO and PPO), all these methods still perform stochastic search by adding stochastic noise to the deterministic policies to force exploration.

Reinforcement learning challenges. Non-local rewards and soft horizon scalability are two major challenges in reinforcement learning. Due to space constraints, we leave the discussion of these challenges with simple and intuitive numerical examples to the Supplementary Material. The rest of the paper assumes the reader's familiarity with these concepts.

2 Monotonic Policy Improvement Guarantee

We use the Wasserstein metric because it allows the effective measurement of distances between probability distributions or functions with non-overlapping support, such as deterministic policies, unlike the KL divergence or TV distance which are either unbounded or maximal in this case. The physical transition model's smoothness enables the use of the Wasserstein distance to regularize deterministic policies. Therefore, we make the following two assumptions about the transition model:

$$W(\mathbb{P}(\mu, \pi_1), \mathbb{P}(\mu, \pi_2)) \leq L_\pi \cdot W(\pi_1, \pi_2), \tag{4}$$
$$W(\mathbb{P}(\mu_1, \pi), \mathbb{P}(\mu_2, \pi)) \leq L_\mu \cdot W(\mu_1, \mu_2). \tag{5}$$

Also, we make the dynamics stability assumption $\sup \sum_{k=1}^{t} \hat{L}_{\mu,\pi_1,\pi_2}^{(k-1)} \prod_{i=k+1}^{t-1} \tilde{L}_{\mu,\pi_1,\pi_2}^{(i)} < \infty$, with the definitions of the new constants and further discussion of the implications deferred to the Supplementary Material where we also discuss Assumptions 4 and 5 and the existence of other Lipschitz constants which appeare as coefficients in the final lower bound.

The advantage decomposition lemma can be rewritten as

$$\eta_{\pi_2} = \eta_{\pi_1} + \frac{1}{1-\gamma} \cdot \mathbb{E}_{s \sim \rho_\mu^{\pi_1}}[A^{\pi_1}(s, \pi_2)] + \frac{1}{1-\gamma} \cdot \langle \rho_\mu^{\pi_2} - \rho_\mu^{\pi_1}, A^{\pi_1}(\cdot, \pi_2) \rangle_s. \tag{6}$$

The $\langle \rho_\mu^{\pi_2} - \rho_\mu^{\pi_1}, A^{\pi_1}(\cdot, \pi_2) \rangle$ term has zero gradient at $\pi_2 = \pi_1$, which qualifies it to be crudely called "the second-order term". We dedicate a full section of our Supplementary Material to the theoretical derivations and proofs necessary to lower-bound this second-order term into an objective well-suited form for practical optimization. Next, we present the theoretical bottom line and the final bound:

$$\mathcal{L}_{\pi_1}^{\sup}(\pi_2) = \eta_{\pi_1} + \frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_\mu^{\pi_1}}[A^{\pi_1}(s, \pi_2)] - C_2 \cdot \sup_s \left[W(\pi_2(a|s), \pi_1(a|s))^2 \right]$$
$$- C_1 \cdot \sup_s \left[\left\| \nabla_{s'} W\left(\frac{\pi_2(a|s') + \pi_1(a|s)}{2}, \frac{\pi_2(a|s) + \pi_1(a|s')}{2} \right) \Big|_{s'=s} \right\|_2^2 \right]. \tag{7}$$

For brevity, we denote the $\left\| \nabla_{s'} W(\cdots) \big|_{s'=s} \right\|_2^2$ expression as $\mathcal{L}_{G^2}(\pi_1, \pi_2; s)$ in the rest of the paper. We have $\eta_{\pi_2} \geq \mathcal{L}_{\pi_1}^{\sup}(\pi_2)$ and $\mathcal{L}_{\pi_1}^{\sup}(\pi_1) = \eta_{\pi_1}$. This facilitates the application of Theorem 2.1 as an instance of Minorization-Maximization algorithms [17].

Theorem 2.1. *Successive maximization of $\mathcal{L}^{\sup}$ yields non-decreasing policy payoffs.*

Proof. With $\pi_2 = \arg\max_\pi \mathcal{L}_{\pi_1}^{\sup}(\pi)$, we have $\mathcal{L}_{\pi_1}^{\sup}(\pi_2) \geq \mathcal{L}_{\pi_1}^{\sup}(\pi_1)$. Thus,

$$\eta_{\pi_2} \geq \mathcal{L}_{\pi_1}^{\sup}(\pi_2) \text{ and } \eta_{\pi_1} = \mathcal{L}_{\pi_1}^{\sup}(\pi_1) \implies \eta_{\pi_2} - \eta_{\pi_1} \geq \mathcal{L}_{\pi_1}^{\sup}(\pi_2) - \mathcal{L}_{\pi_1}^{\sup}(\pi_1) \geq 0. \quad \square \tag{8}$$

Successive optimization of $\mathcal{L}_{\pi_1}^{\sup}(\pi_2)$ generates non-decreasing payoffs. However, it is impractical due to the large number of constraints and statistical estimation of maximums. To mitigate this, we take a similar approach to TRPO and replace the maximums with expectations over the observations.

The coefficients C_1 and C_2 are dynamics-dependent. In the *basic variant* of our method, we used constant coefficients and a trust region. This yields the Truly Deterministic Policy Optimization (TDPO) as given in Algorithm 1. See the Supplementary Material for practical notes on the manual choice of C_1 and C_2. Alternatively, one could adopt processes similar to Schulman et al. [38] where the IS-based advantage estimator used a line search for proper step size selection, or the adaptive penalty coefficient setting in Schulman et al. [40]. In the *advanced variant* of our method, we implement such a line search procedure by collecting samples from the environment and picking the coefficient yielding the best improvement. This comes at an increase in the sample complexity, but the benefits can outweigh the added costs. Furthermore, the exploration scale in the sampling oracle is adaptively tuned using the collected payoffs in the advanced variant; by constructing an importance sampling derivative estimate for the exploration scale parameter, stochastic gradient descent can be used to tune the exploration scale individually.

2.1 On the interpretation of the surrogate function

For deterministic policies, the squared Wasserstein distance $W(\pi_2(a|s), \pi_1(a|s))^2$ degenerates to the Euclidean distance over the action space. Any policy defines a sensitivity matrix at a given

Algorithm 1 Truly Deterministic Policy Optimization (TDPO)

Require: Initial policy π_0.
Require: Advantage estimator and sample collector oracle $\mathbb{A}^{\pi}$.
1: **for** $k = 1, 2, \dots$ **do**
2: Collect trajectories and construct the advantage estimator oracle $\mathbb{A}^{\pi_k}$.
3: Compute the policy gradient g at θ_k : $g \leftarrow \nabla_{\theta'} \mathbb{A}^{\pi_k}(\pi')|_{\pi'=\pi_k}$
4: Construct a surrogate Hessian vector product oracle $v \to H \cdot v$ such that for $\theta' = \theta_k + \delta\theta$,

$$\mathbb{E}_{s\sim\rho_\mu^{\pi_k}}\left[W(\pi'(a|s), \pi_k(a|s))^2\right] + \frac{C_1}{C_2}\mathbb{E}_{s\sim\rho_\mu^{\pi_k}}\left[\mathcal{L}_{G^2}(\pi', \pi_k; s)\right] = \frac{1}{2}\delta\theta^T H \delta\theta + \text{h.o.t.}, \quad (9)$$

where h.o.t. denotes higher order terms in $\delta\theta$.
5: Find the optimal update direction $\delta\theta^* = H^{-1}g$ using the Conjugate Gradient algorithm.
6: **(Basic Variant)** Determine the best step size α^* within the trust region:

$$\begin{aligned} \alpha^* = &\arg\max_{\alpha} g^T(\alpha\delta\theta^*) - \frac{C_2}{2}(\alpha\delta\theta^*)^T H(\alpha\delta\theta^*) \\ &\text{s.t.} \qquad \frac{1}{2}(\alpha^*\delta\theta^*)^T H(\alpha^*\delta\theta^*) \le \delta_{\max}^2 \end{aligned} \quad (10)$$

7: **(Advanced Variant)** Determine the best step size α^* using a line-search procedure and pick the best one; each coefficient's performance can be evaluated by sampling from the environment.
8: **(Advanced Variant)** Update the exploration scale in $\mathbb{A}^{\pi}$ using the collected samples.
9: Update the policy parameters: $\theta_{k+1} \leftarrow \theta_k + \alpha^*\delta\theta^*$.
10: **end for**

Algorithm 2 Deterministic Vine (DeVine) Policy Advantage Estimator

Require: The number of parallel workers K
Require: A policy π, an exploration policy q, discrete time-step distribution $\nu(t)$, initial state distribution $\mu(s)$, and the discount factor γ.
1: Sample an initial state s_0 from μ, and then roll out a trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$ using π.
2: **for** $k = 1, 2, \cdots, K$ **do**
3: Sample the integer number $t = t_k$ from ν.
4: Compute the value $V^{\pi_1}(s_t) = \sum_{i=t}^{\infty} \gamma^{t-i} R(s_i, a_i)$.
5: Reset the initial state to s_t, sample the first action a'_t according to $q(\cdot|s_t)$, and use π for the rest of the trajectory. This will create $\tau' = (s_t, a'_t, s'_{t+1}, a'_{t+1}, \cdots)$.
6: Compute the value $Q^{\pi_1}(s_t, a'_t) = \sum_{i=t}^{\infty} \gamma^{t-i} R(s'_i, a'_i)$.
7: Compute the advantage $A^{\pi_1}(s_t, a'_t) = Q^{\pi}(s_t, a'_t) - V^{\pi}(s_t)$.
8: **end for**
9: Define $\mathbb{A}^{\pi_1}(\pi_2) := \dfrac{1}{K}\displaystyle\sum_{k=1}^{K} \frac{\dim(\mathcal{A}) \cdot \gamma^{t_k}}{\nu(t_k)} \cdot \frac{(\pi_2(s) - a_{t_k})^T (a'_{t_k} - a_{t_k})}{(a'_{t_k} - a_{t_k})^T (a'_{t_k} - a_{t_k})} \cdot A^{\pi_1}(s_{t_k}, a'_{t_k})$.
10: Return $\mathbb{A}^{\pi_1}(\pi_2)$ and $\nabla_{\pi_2}\mathbb{A}^{\pi_1}(\pi_2)$ as unbiased estimators for $E_{s\sim\rho_\mu^{\pi_1}}[A^{\pi_1}(s, \pi_2)]$ and PG, respectively.

state s, which is the Jacobian matrix of the policy output with respect to s. The policy sensitivity term $\mathcal{L}_{G^2}(\pi_1, \pi_2; s)$ is essentially the squared Euclidean distance over the action-to-observation Jacobian matrix elements. In other words, our surrogate prefers to step in directions where the action-to-observation sensitivity is preserved within updates.

Although our surrogate uses a metric distance instead of the traditional non-metric measures for regularization, we do not consider this sole replacement a major contribution. The squared Wasserestein distance and the KL divergence of two identically-scaled Gaussian distributions are the same up to a constant (i.e., $D_{\text{KL}}(\mathcal{N}(m_1, \sigma)\|\mathcal{N}(m_2, \sigma)) = W(\mathcal{N}(m_1, \sigma), \mathcal{N}(m_2, \sigma))^2/2\sigma^2$). On the other hand, our surrogate's compatibility with deterministic policies makes it a valuable asset for our policy gradient algorithm; both $W(\pi_2(a|s), \pi_1(a|s))^2$ and $\mathcal{L}_{G^2}(\pi_1, \pi_2; s)$ can be evaluated for two deterministic policies π_1 and π_2 numerically without any approximations to overcome singularities.

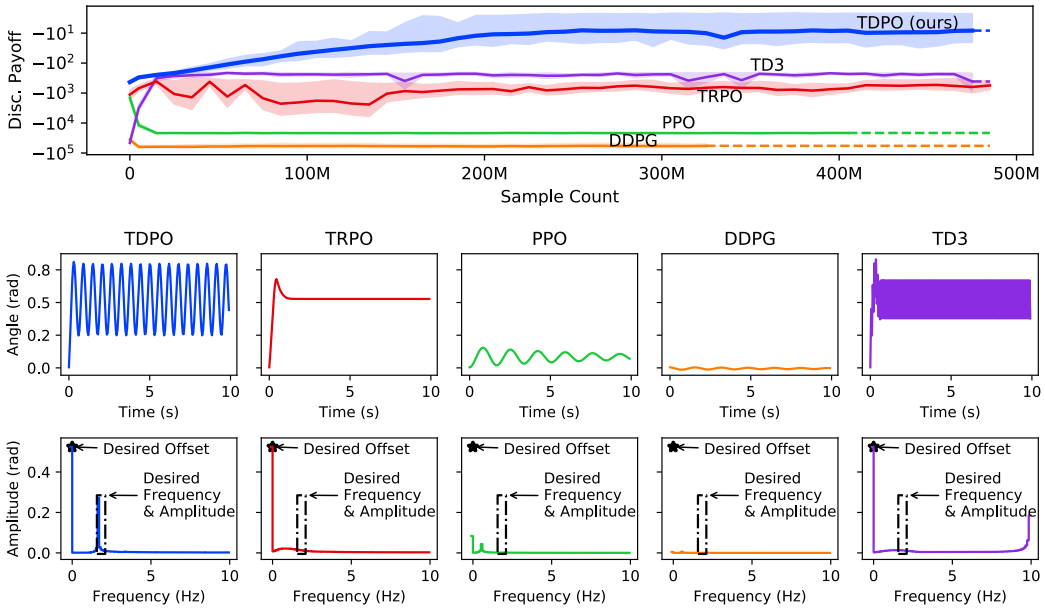

Figure 1: Results for the simple pendulum with non-local rewards. Upper panel: training curves with empirical discounted payoffs. Lower panels: trajectories in both the time domain and frequency domain, showing target values of oscillation frequency, amplitude, and offset. The basic variant of our method (non-adaptive exploration scales and update coefficients) was used in this experiment.

3 Model-Free Estimation of Policy Gradient

The DeVine advantage estimator is formally defined in Algorithm 2. Unlike DDPG and TD3, the DeVine estimator allows our method to perform *deterministic search* by not consistently injecting noise in actions for exploration. Under deterministic dynamics and policies, if DeVine samples each dimension at each time-step exactly once then in the limit of small exploration scale σ it can produce exact advantages, as stated in Theorem 3.1, whose proof is deferred to the Supplementary Material.

Theorem 3.1. *Assume a finite horizon MDP with both deterministic transition dynamics P and initial distribution μ, with maximal horizon length of H. Define $K = H \cdot \dim(\mathcal{A})$, a uniform ν, and $q(s;\sigma) = \pi_1(s) + \sigma \mathbf{e}_j$ in Algorithm 2 with $\mathbf{e}_j$ being the j^{th} basis element for $\mathcal{A}$. If the (j, t_k) pairs are sampled to exactly cover $\{1, \cdots, \dim(\mathcal{A})\} \times \{1, \cdots, H\}$, then we have*

$$\lim_{\sigma \to 0} \nabla_{\pi_2} \mathbb{A}^{\pi_1}(\pi_2)\big|_{\pi_2=\pi_1} = \nabla_{\pi_2} \eta_{\pi_2}\big|_{\pi_2=\pi_1}. \tag{11}$$

Theorem 3.1 provides a guarantee for recovering the exact policy gradient if the initial state distribution was deterministic and all time-steps of the trajectory were used to branch vine trajectories. Although this theorem sets the stage for computing a fully deterministic gradient, stochastic approximation can be used in Algorithm 2 by randomly sampling a small set of states for advantage estimation. In other words, Theorem 3.1 would use ν to deterministically sample all trajectory states, whereas this is not a practical requirement for Algorithm 2 and the gradients are still unbiased if a random set of vine branches is used.

The DeVine estimator can be advantageous in at least two scenarios. First, in the case of rewards that cannot be decomposed into summations of immediate rewards. For example, overshoot penalizations or frequency-based rewards as used in robotic systems are non-local. DeVine can be robust to non-local rewards as it is insensitive to whether the rewards were applied immediately or after a long period. Second, DeVine can be an appropriate choice for systems that are sensitive to the injection of noise, such as high-bandwidth robots with natural resonant frequencies. In such cases, using white (or colored) noise for exploration can excite these resonant frequencies and cause instability, making learning difficult. DeVine avoids the need for constant noise injection.

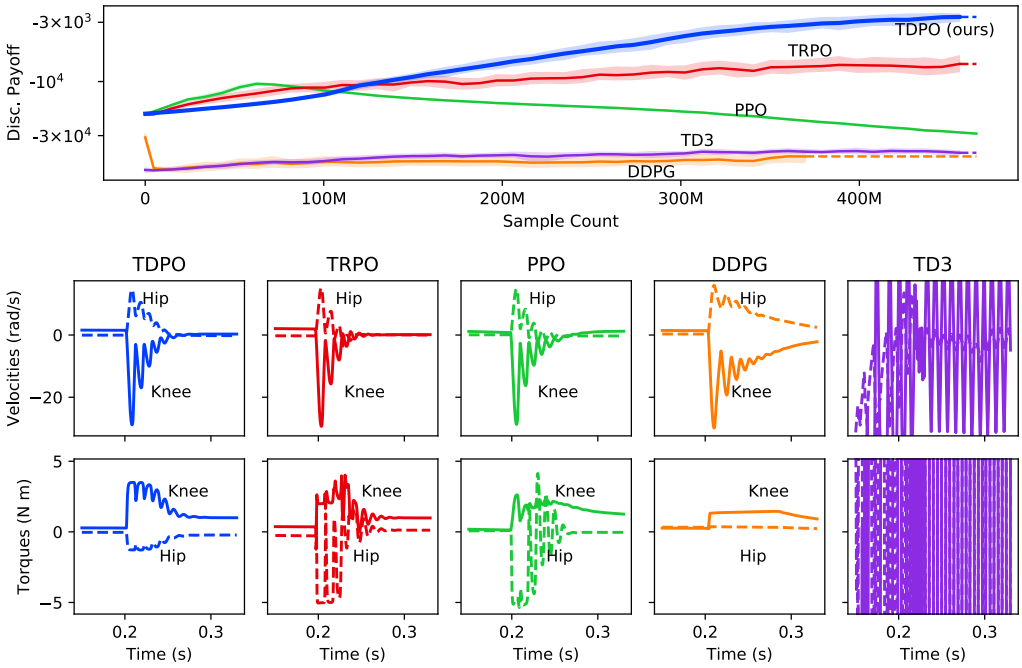

Figure 2: Results for the leg environment with a long horizon and resonant frequencies due to ground compliance. Upper panel: training curves with empirical discounted payoffs. Lower panel: partial trajectories, restricted to times shortly before and after impact with the ground. Note the oscillations at about 100 Hz that appear just after the impact at 0.2 s—these oscillations are evidence of a resonant frequency. The basic variant of our method (non-adaptive exploration scales and update coefficients) was used in this experiment.

4 Experiments

The next three subsections show challenging robotic control tasks including frequency-based non-local rewards, long horizons, and sensitivity to resonant frequencies. In Sections 4.1 and 4.2, we use the basic variant of our method (i.e., fixed exploration scale and update coefficient hyper-parameters throughout the training). This will facilitate a better understanding of our core method's capabilities without any additional tweaks. See the Supplementary Material for a comparison on traditional gym environments, where the basic variant of TDPO works similarly to existing methods. Section 4.3 includes the most difficult setting in our paper, where we use the advanced variant of our method (i.e., with line-search for the update coefficient and adaptive exploration scales).

4.1 An Environment with Non-Local Rewards [1]

The first environment that we consider is a simple pendulum. The transition function is standard—the states are joint angle and joint velocity, and the action is joint torque. The reward function is non-standard—rather than define a local reward in the time domain with the goal of making the pendulum stand upright (for example), we define a non-local reward in the frequency domain with the goal of making the pendulum oscillate with a desired frequency and amplitude about a desired offset. In particular, we compute this non-local reward by taking the Fourier transform of the joint angle signal over the entire trajectory and by penalizing differences between the resulting power spectrum and a desired power spectrum. We apply this non-local reward at the last time step of the trajectory. Implementation details and similar results for more pendulum variants are left to the Supplementary Material.

[1] Non-local rewards are reward functions of the entire trajectory whose payoffs cannot be decomposed into the sum of terms such as $\eta = \sum_t f_t(s_t, a_t)$, where functions f_t only depend on nearby states and actions. An example non-local reward is one that depends on the Fourier transform of the complete trajectory signal.

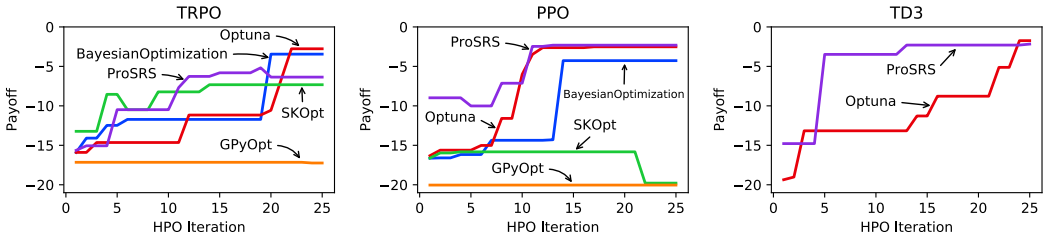

Figure 3: The best payoff vs. the Hyper-Parameter Optimization (HPO) iteration on a short-horizon variant of the legged robotic environment. The HPOs are performed for each of the TRPO, PPO, and TD3 methods in a separate panel. DDPG is a special case of TD3 with HPO. Since TD3 was considerably more expensive, we only show Optuna and ProSRS for it.

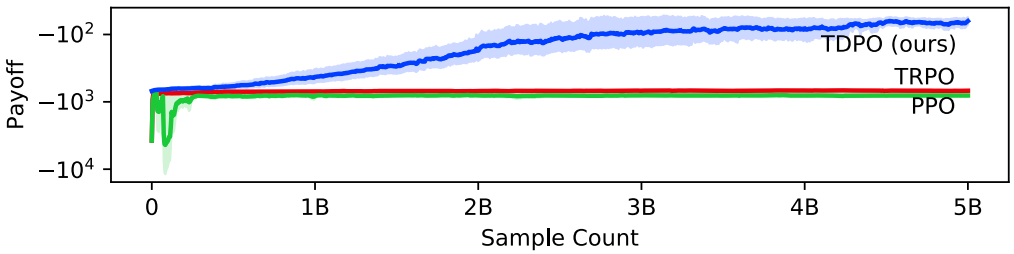

Figure 4: Post Hyper-Parameter Optimization (HPO) training curves with the best settings found for TRPO and PPO compared to the advanced variant of our method (TDPO with adaptive exploration scales and line search). TD3 had a significantly poor performance in the initial parameter sweeps. Due to resource limitations and poor initial performance, we excluded TD3 from this experiment.

Figure 1 shows training curves for TDPO (our method) as compared to TRPO, PPO, DDPG, and TD3. These results were averaged over 25 experiments in which the desired oscillation frequency was 1.7 Hz (different from the pendulum's natural frequency of 0.5 Hz), the desired oscillation amplitude was 0.28 rad, and the desired offset was 0.52 rad. Figure 1 also shows trajectories obtained by the best agents from each method. TDPO (our method) was able to learn high-reward behavior and to achieve the desired frequency, amplitude, and offset. TRPO was able to learn the correct offset but did not produce any oscillatory behavior. TD3 also learned the correct offset, but could not produce desirable oscillations. PPO and DDPG failed to learn any desired behavior.

4.2 An Environment with Long Horizon and Resonant Frequencies[2]

The second environment that we consider is a single leg from a quadruped robot [34]. This leg has two joints, a "hip" and a "knee," about which it is possible to exert torques. The hip is attached to a slider that confines motion to a vertical line above flat ground. We assume the leg is dropped from some height above the ground and the task is to recover from this drop and to stand upright at rest after impact. States given to the agent are the angle and velocity of each joint (slider position and velocity are hidden), and actions are the joint torques. The reward function penalizes difference from an upright posture, slipping or chattering at the contact between the foot and the ground, non-zero joint velocities, and steady-state joint torque deviations. We use the open-source MuJoCo software for simulation [43], with high-fidelity models of ground compliance, motor nonlinearity, and joint friction. The control loop rate is 4000 Hz and the rollout length is 2 s, resulting in a horizon of 8000 steps. Implementation details are left to the Supplementary Material.

[2]Resonant frequencies are a concept from control theory. In the frequency domain, signals of certain frequencies are excited more than others when applied to a system. This is captured by the frequency-domain transfer function of the system, which may have a peak of magnitude greater than one. The resonant frequency is the frequency at which the frequency-domain transfer function has the highest amplitude. Common examples of systems with a resonant frequency include the undamped pendulum, which oscillates at its natural frequency, and RLC circuits which have characteristic frequencies at which they are most excitable. See Chapter 8 of Kuo and Golnaraghi [27] for more information.

Figure 2 shows training curves for TDPO (our method) as compared to TRPO, PPO, DDPG and TD3. These results were averaged over 75 experiments. A discount factor of $\gamma = 0.99975$ was chosen for all methods, where $(1-\gamma)^{-1}$ is half the trajectory length. Similarly, the GAE factors for PPO and TRPO were scaled up to 0.99875 and 0.9995, respectively, in proportion to the trajectory length. Figure 2 also shows trajectories obtained by the best agents from each method. TDPO (our method) was able to learn high-reward behavior. TRPO, PPO, DDPG, and TD3 were not.

We hypothesize that the reason for this difference in performance is that TDPO better handles the combination of two challenges presented by the leg environment—an unusually long time horizon (8000 steps) and the existence of a resonant frequency that results from compliance between the foot and the ground (note the oscillations at a frequency of about 100 Hz that appear in the trajectories after impact). Both high-speed control loops and resonance due to ground compliance are common features of real-world legged robots to which TDPO seems to be more resilient.

4.3 Practical Training and Hardware Implementation

For the most realistic setting, we take the environment from the previous section and make it highly stochastic by (a) injecting noise into the transition dynamics P, and (b) making the initial state distribution μ as random as physically possible. We also systematically perform Hyper-Parameter Optimization (HPO) on all methods to allow the most fair comparison.

The choice of the HPO method can have a significant impact on the RL agent's performance. We consider a list of five off-the-shelf HPO implementations and run them in their default settings: Optuna [2], BayesianOptimization [32], Scikit-Optimize [14], GPyOpt [12], and ProSRS [41]. These implementations include a range of HPO methods, including Gaussian processes and tree Parzen estimators. For better performance, HPO methods need a reasonable set of initial hyper-parameter guesses. For this, we perform a one-variable-at-a-time parameter sweep along every hyper-parameter near the RL method's default hyper-parameters. These parameter sweep results are then input to each HPO method for full optimization. Using all HPO algorithms for all RL methods in the long-horizon environment (where each full training run takes 5 billion samples) is computationally infeasible. To pick the best HPO method, we benchmark a short-horizon environment with only 200 time-steps in a trajectory. The result is shown in Figure 3 (see the Supplementary Material for full details on the HPO methods). Overall, we found that Optuna and ProSRS are the best HPO methods on the test problem. Since Optuna is widely-test and arguably the most popular HPO library, we pick it as the main HPO method for our long-horizon environment.

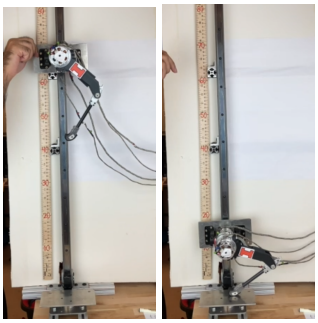

Figure 5: The simulation-to-real transfer of the best TDPO agent to perform a successful drop test at 4 kHz control rate.

We repeat the same HPO procedure on the long-horizon environment using Optuna, and pick the best hyper-parameters found in the course of HPO for a final training. Figure 4 shows this final training. TDPO shows superior performance in this highly stochastic environment, and such benefits cannot be obtained by merely performing HPO on other methods. To showcase the practicality of our method, we picked the best TDPO trained agent, and implemented it on the physical hardware. The transferred agent was able to successfully perform drop-and-catch tests on the robot system at 4 kHz, with both global control and suppression of high-frequency transients. Figure 5 shows a glimpse of this test, and a short video is also included in the code repository.

5 Discussion

We proposed a deterministic policy gradient method (TDPO: Truly Deterministic Policy Optimization) based on the use of a deterministic Vine (DeVine) gradient estimator and the Wasserstein metric. We proved monotonic payoff guarantees for our method, and defined a novel surrogate for policy optimization. We showed numerical evidence for superior performance with non-local rewards defined in the frequency domain and a realistic long-horizon resonant environment. This method enables applications of policy gradient to customize frequency response characteristics of agents. Our work assumed continuous environments, and future work should include the adaptation of our method to environments with discrete state and action spaces.

References

[1] Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Kavosh Asadi, Dipendra Misra, and Michael L Littman. Lipschitz continuity in model-based reinforcement learning. *arXiv preprint arXiv:1804.07193*, 2018.

[5] Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in neural information processing systems*, pages 1204–1212, 2009.

[6] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[7] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[8] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[9] Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for markov decision processes with infinite state spaces. *arXiv preprint arXiv:1207.1386*, 2012.

[10] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[12] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657. PMLR, 2016.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[14] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, Loïc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. scikit-optimize/scikit-optimize: v0.5.2, March 2018. URL `https://doi.org/10.5281/zenodo.1207017`.

[15] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[16] Karl Hinderer. Lipschitz continuity of value functions in markovian decision processes. *Mathematical Methods of Operations Research*, 62(1):3–22, 2005.

[17] David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

[18] Nan Jiang and Alekh Agarwal. Open problem: The dependence of sample complexity lower bounds on planning horizon. In *Conference On Learning Theory*, pages 3395–3398, 2018.

[19] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[20] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[21] Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

[22] Nathan Kallus and Masatoshi Uehara. Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning. *arXiv*, pages arXiv–1909, 2019.

[23] Nathan Kallus and Masatoshi Uehara. Statistically efficient off-policy policy gradients. *arXiv preprint arXiv:2002.04014*, 2020.

[24] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2-3):193–208, 2002.

[25] Michael J Kearns, Yishay Mansour, and Andrew Y Ng. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems*, pages 1001–1007, 2000.

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

[27] Benjamin C. Kuo and Farid Golnaraghi. *Automatic Control Systems*. John Wiley & Sons, Inc., USA, 8th edition, 2002. ISBN 0471134767.

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

[29] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018.

[30] Leonard Meirovitch. *Elements of vibration analysis*. McGraw-Hill Science, Engineering & Mathematics, 1975.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[32] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL `https://github.com/fmfn/BayesianOptimization`.

[33] Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Anna Choromanska, Krzysztof Choromanski, and Michael Jordan. Learning to score behaviors for guided policy optimization. *arXiv preprint arXiv:1906.04349*, 2019.

[34] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36 (2):167–192, 2017.

[35] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2-3):255–283, 2015.

[36] André Preumont and Kazuto Seto. *Active control of structures*. John Wiley & Sons, 2008.

[37] Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. In *11th International Symposium on Artificial Intelligence and Mathematics (ISIAM 2010)*, pages 1–8, Fort Lauderdale, US, 2010. URL https://oatao.univ-toulouse.fr/17977/.

[38] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[39] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[41] Chenchao Shou and Matthew West. A tree-based radial basis function method for noisy parallel surrogate optimization. *arXiv preprint arXiv:1908.07980*, 2019.

[42] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[43] 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.

[44] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[45] Ruosong Wang, Simon S Du, Lin F Yang, and Sham M Kakade. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*, 2020.

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

[47] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The claims are well-supported with both theoretical derivations and numerical examples.
   (b) Did you describe the limitations of your work? [Yes] The last sentence of the discussion section talks about the current limitation of our paper to non-discrete environments.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We dedicated a section of the appendix to discussing potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All theoretical results and mathematical notations are clearly stated in the form of lemmas and theorems, both in the main paper and the appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] All the theoretical theorems and lemmas are proven either in the main paper or in the appendix.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included detailed instructions to reproduce all experiments in the appendix, and our anonymized code repository includes all the code we used to produce the paper and the appendix results.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes, training details and experiment specifications are discussed at length in the appendix. For best reproducibility, we also released the code along with the specific hyper-parameters (in JSON format) in our anonymized code repository. Our code repository includes all software/library version details that we used to produce the results.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All our experiments include 95% confidence intervals, and are ran for 100 independent random seeds.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This was left to the appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We cited the scholar article for the Mujoco physical simulator used in our paper.

(b) Did you mention the license of the assets? [Yes] After de-anonymization, we will open-source our code with an MIT license. We used the open-source Mujoco simulator which is available under the Apache 2.0 lisence.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]