Adaptive-Gradient Policy Optimization: Enhancing Policy Learning in Non-Smooth Differentiable Simulations

Feng Gao [1 *] **Liangzhi Shi** [1 *] **Shenao Zhang** [2] **Zhaoran Wang** [2] **Yi Wu** [1 3]

Abstract

Recent advancements in differentiable simulators highlight the potential of policy optimization using simulation gradients. Yet, these approaches are largely contingent on the continuity and smoothness of the simulation, which precludes the use of certain simulation engines, such as Mujoco. To tackle this challenge, we introduce the adaptive analytic gradient. This method views the Q function as a surrogate for future returns, consistent with the Bellman equation. By analyzing the variance of batched gradients, our method can autonomously opt for a more resilient Q function to compute the gradient when encountering rough simulation transitions. We also put forth the Adaptive-Gradient Policy Optimization (AGPO) algorithm, which leverages our proposed method for policy learning. On the theoretical side, we demonstrate AGPO's convergence, emphasizing its stable performance under non-smooth dynamics due to low variance. On the empirical side, our results show that AGPO effectively mitigates the challenges posed by non-smoothness in policy learning through differentiable simulation.

1. INTRODUCTION

Learning control policies is vital for robotics and computer animation. Deep reinforcement learning has recently shown great promise, excelling in games (Mnih et al., 2015; Silver et al., 2016), animation generation (Peng et al., 2021; Zhang et al., 2023a), and robot control (Levine et al., 2016; Lee et al., 2020; Kaufmann et al., 2023). Yet, these methods need extensive data to approximate policy gradients (Williams, 1992; Sutton et al., 1999) for model updates. In practice, the gradient estimation method based on zeroth-order sampling (Schulman et al., 2017), is characterized by a low sample efficiency. The recent rise of differentiable simulations (Freeman et al., 2021; Heiden et al., 2021; Howell et al., 2022) introduces a way to tackle the problem of efficiency. By leveraging auto-differentiation tools like PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018) or specially crafted differentiable kernels (Xu et al., 2022), these simulators are well-positioned to provide first-order simulation gradients of rewards in relation to control inputs. Such capability holds the potential to significantly speed up the policy optimization process and to considerably enhance policy performance.

While differentiable simulators are emerging tools, similar ideas on utilizing first-order gradients have been investigated in model-based RL. Parmas et al. (Parmas et al., 2018) proposed to use the inverse-variance weighting (Fleiss, 1993) for blending first-order reparameterization gradients (Kingma & Welling, 2013) and zeroth-order likelihood gradients, and showcased the problem of rough optimization landscape. They further extended this idea to use critic-based gradient estimates and made a scalable solution in (Parmas et al., 2023). Akin to that, leveraging simulation gradients may not invariably lead to improved policy learning outcomes (Metz et al., 2021; Xu et al., 2022; Suh et al., 2022; Zhong et al., 2022), especially in scenarios involving rich contact dynamics. Xu et al. (2022) proposed to truncate the full task trajectory into short horizons and incorporates a value function at the horizon's end to signify future implications. Though effective on their proposed simulation kernels, this method may falter when horizons encompass abrupt transitions, i.e., when the simulated dynamics are uneven or discontinuous. Such environments, affected by the design of the contact model, often present issues that make it challenging to employ certain methodologies in popular robot simulators.

This paper tackles policy learning in non-smooth differentiable simulations. While previous works(Parmas et al., 2018; Suh et al., 2022) introduced to blend zeroth-order policy gradients (PG) with first-order simulation gradients (SG), we find that PG cannot well detect the rough region in practice. Observing PG's challenges in control tasks, we utilize the Q function for future returns prediction, using its gradi-

[*]Equal contribution [1]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China [2]Northwestern University, Illinois, United States [3]Shanghai Qi Zhi Institute, Shanghai, China. Correspondence to: Feng Gao <feng.gao220@gmail.com>, Yi Wu <jxwuyi@gmail.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ents (QG) to stabilize policy optimization. We show that QG is a more stable gradient estimator than PG. Furthermore, we propose an adaptive analytic gradient estimator that can autonomously opt for more resilient QGs when encountering singular simulation steps. That's to say, our method adaptively utilizes the Q function as a shortcut to bypass the rough landscape. Specifically, we adopt the inverse-variance weighting scheme (Fleiss, 1993; Parmas et al., 2018; Suh et al., 2022) to mix QG and SG. Furthermore, we propose the adaptive-gradient policy optimization algorithm, dubbed AGPO. It uses the proposed adaptive analytic gradient for policy learning and presents better performance than both pure RL baselines and existing simulation-gradient-based methods. Theoretically, we establish the convergence of AGPO and its reliance on the variance and bias of the gradient. Moreover, AGPO has a low variance upper bound, which does not increase significantly as in the first-order SG when the horizon is long and the dynamics is non-smooth.

2. RELATED WORK

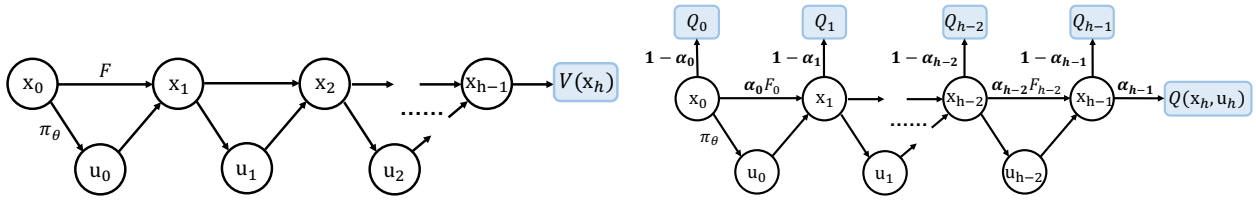
Differentiable Simulation. Differentiable simulation in contacts and collisions is a burgeoning field. Approaches range from modeling contacts as linear complementarity problems (LCPs) for impulse computation (de Avila Belbute-Peres et al., 2018; Heiden et al., 2021; Qiao et al., 2021; Zhang et al., 2023b), to convex optimization for velocity impulses based on maximum dissipation principles (Todorov, 2011; 2014). Other methods include compliant models with spring-damper systems for gradual interpenetration resolution (Carpentier & Mansard, 2018; Xu et al., 2022), and position-based dynamics (PBD) for direct position manipulation while conserving momentum (Müller et al., 2007; Macklin et al., 2020). Differentiability is achieved through implicit differentiation, automatic differentiation, or custom numerical solvers, facilitating direct simulation gradients for policy learning.

Policy Gradient Methods. Policy gradient methods optimize policies by estimating the gradient of the expected reward w.r.t. the policy parameters. Methods like REINFORCE (Williams, 1992) use Monte Carlo returns without an explicit value function, whereas actor-critic architectures (Sutton et al., 1999) incorporate a value function for refined gradient estimation. For continuous action spaces, DDPG (Lillicrap et al., 2015) implemented deterministic policy gradients. TD3 (Fujimoto et al., 2018) subsequently improved upon DDPG by addressing value overestimation concerns. The Soft Actor-Critic (SAC) method (Haarnoja et al., 2018) integrated entropy into the actor-critic framework, fostering enhanced exploration. Proximal Policy Optimization (PPO) (Schulman et al., 2017) sought to bolster stability through constrained policy updates. In addition to solely using zeroth-order gradients, PIPPS (Parmas et al., 2018) integrated zeroth-order likelihood gradients with first-order reparameterization gradients (Kingma & Welling, 2013), employing inverse variance weighting (Fleiss, 1993) for step-wise dynamic combination. Subsequent studies have expanded on this foundation, offering comprehensive theoretical analysis (Parmas & Sugiyama, 2021) and extending the methodology to more complicated model-based RL scenarios (Parmas & Seno, 2022; Parmas et al., 2023).

Policy Optimization with Differentiable Simulation. In recent advancements in policy optimization, differentiable engines have been leveraged to provide gradient information, enhancing policy optimization. While the traditional Backpropagation Through Time (BPTT) grapples with issues such as exploding/vanishing gradients (Bengio et al., 1994; Zhang et al., 2023c), many attempts have emerged to refine and craft more robust algorithms. For instance, PODS (Mora et al., 2021) taps into differentiable simulators to obtain analytic gradients of a policy's value function, leading to effective first- and second-order policy improvement strategies. SHAC (Xu et al., 2022) refines the BPTT approach by breaking trajectories into short optimization windows and embedding a value function at the end of each horizon, providing smoother learning. Zhang et al.(2023b) proposed ABS to reduce the gradient variance while controlling the gradient bias using a contact-aware adaptive central-path parameter. Nonetheless, the reliability of gradients from simulations remains a concern. To combat this, Suh et al. (2022) followed previous works (Parmas et al., 2018; Metz et al., 2021) to blend simulation gradients with zeroth-order policy gradients while additionally addressing the empirical bias phenomenon. Nonetheless, these existing algorithms cannot work well on complex continuous control tasks with non-smooth dynamics.

3. ADAPTIVE ANALYTIC GRADIENT

In this paper, we address the challenge posed by the non-smoothness of a differentiable simulation in continuous-state control problems. The problem is characterized by states, denoted as $\mathbf{x} \in \mathbb{R}^n$, and control inputs, represented as $\mathbf{u} \in \mathbb{R}^m$. The system dynamics are governed by transition functions $\phi(\mathbf{x}, \mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. In addition, there exist reward functions $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, where $\mathbf{x}'$ represents the subsequent state and is given by $\mathbf{x}' = \phi(\mathbf{x}, \mathbf{u})$. A salient feature of our study is the differentiability of both ϕ and r w.r.t. $\mathbf{x}$ and $\mathbf{u}$. This differentiability ensures that we can compute the subsequent partial derivatives (Jacobians) directly from the simulator: $\frac{\partial \phi}{\partial \mathbf{x}}, \frac{\partial \phi}{\partial \mathbf{u}}, \frac{\partial r}{\partial \mathbf{x}}, \frac{\partial r}{\partial \mathbf{u}}$. The non-smoothness suggests that, despite being differentiable, these derivatives may exhibit discontinuities or sudden variations across their domains. Outliers may arise due to hard contacts or from numerical issues.



(a) Short-Horizon Actor-Critic (SHAC)

(b) Adaptive-Gradient Policy Optimization (AGPO)

Figure 1. Computational graphs of SHAC and AGPO. For AGPO, we use F_t to denote the outcome of future trajectory starting from state $\mathbf{x}_t$.

3.1. Preliminaries

In the context of reinforcement learning (RL), a primary goal is to find an optimal policy π_θ, parameterized by θ, that maximizes the expected cumulative reward over time. This is succinctly captured by the objective function F with a discount factor γ. F is presented mathematically as:

$$F(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\right],$$

where $\mathbf{u}_t \sim \pi_\theta(\cdot|\mathbf{x}_t)$ and $\mathbf{x}_{t+1} = \phi(\mathbf{x}_t, \mathbf{u}_t)$.

The value function, denoted as $V(\mathbf{x}; \pi_\theta)$, represents the expected cumulative reward when starting from state $\mathbf{x}$ and acting according to policy π_θ:

$$V(\mathbf{x}; \pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathbf{u}_t, \phi(\mathbf{x}_t, \mathbf{u}_t)) \middle| \mathbf{x}_0 = \mathbf{x}\right].$$

The Q-function, denoted as $Q(\mathbf{x}, \mathbf{u}; \pi_\theta)$, encapsulates the expected return starting from state $\mathbf{x}$, taking action $\mathbf{u}$, and thereafter adhering to the policy π_θ:

$$Q(\mathbf{x}, \mathbf{u}; \pi_\theta) = r(\mathbf{x}, \mathbf{u}, \phi(\mathbf{x}, \mathbf{u})) + \gamma \mathbb{E}_{\pi_\theta}\left[V(\phi(\mathbf{x}, \mathbf{u}); \pi_\theta)\right].$$

In the actor-critic paradigm, neural networks are commonly used to approximate either the value function or the Q function. For clarification, the approximated value and Q functions are symbolized as $\hat{V}$ and $\hat{Q}$ respectively. For brevity, we will use V_t and Q_t to denote $V(\mathbf{x}_t)$ and $Q(\mathbf{x}_t, \mathbf{u}_t)$.

A central objective of our study is the development of an effective gradient estimator for ∇F. Since obtaining precise gradients is challenging, our estimator aims to provide a reliable approximation, crucial for steering the optimization towards the maximum of F.

Zeroth-order Gradient Estimators. The zeroth-order gradient estimators refer to the methods where ∇F is estimated without directly differentiating the simulation, opting instead for learning from samples. They can be classified primarily into two categories: **Policy Gradient (PG)** and **Q Gradient (QG)**. The PG method estimates the gradient of the expected reward w.r.t. the policy parameters, as illustrated by the equation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta(\mathbf{u}|\mathbf{x})}\left[\nabla_\theta \log \pi_\theta(\mathbf{u}|\mathbf{x}) \hat{Q}(\mathbf{x}, \mathbf{u})\right].$$

Conversely, the QG focuses on the Q-function's gradient, making it especially apt for deterministic policies or reparameterized stochastic policies. Notably, QG retains its zeroth-order character as the Q function is approximated by learning from samples. The QG formulation stands as $\nabla_\theta \hat{Q}(\mathbf{x}, \pi_\theta(\mathbf{x}))$.

In practice, we need a vast amount of data to approximate the policy gradient. Given a series of sampled data, the batched policy gradient is computed by

$$\bar{\nabla}^{[0]} J(\theta) := \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\mathbf{u}_i|\mathbf{x}_i) \hat{A}(\mathbf{x}_i, \mathbf{u}_i),$$

with $\hat{A}(\cdot)$ the estimated advantage function, and the batched Q gradient is computed by

$$\bar{\nabla}^{[0]} Q(\theta) := \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \hat{Q}(\mathbf{x}_i, \pi_\theta(\mathbf{x}_i)),$$

where we use $\bar{\nabla}^{[0]}$ to indicate the zeroth-order estimator in sample mean.

First-order Gradient Estimator. Given the differentiability of both the transition function ϕ and the reward function r, the first-order **Simulation Gradient (SG)** is the gradient of the objective function F w.r.t. the policy parameters θ that leverages the direct differentiation of these functions. To overcome the gradient explosion issue, we follow the method proposed in (Xu et al., 2022) whose computational graph is shown in Fig. 1. Suppose we truncate each trajectory into short horizons in the length of h, the optimization objective is defined as

$$\mathcal{L}(\theta; h) = \frac{1}{h}\left(\sum_{t=t_0}^{t_0+h-1} \gamma^{t-t_0} r(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) + \gamma^h V(\mathbf{x}_{t_0+h})\right).$$

Then the batched simulation gradient is

$$\bar{\nabla}^{[1]}F(\theta;h) := \frac{1}{N}\sum_{i=1}^{N}\hat{\nabla}^{[1]}F_i(\theta;h) = \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta\mathcal{L}(\theta;h)$$

where $\hat{\nabla}$ symbolizes the gradient estimation derived from a single sample and $\bar{\nabla}$ denotes the sample mean of batched gradients.

3.2. Validity of Gradient Estimates

Recent studies have underscored the importance of empirical variance as a metric for assessing the precision of gradient estimates (Parmas et al., 2018; Metz et al., 2021; Suh et al., 2022; Zhong et al., 2022). Here, we will further demonstrate that the Q-function Gradient (QG) serves as a more reliable and stable gradient estimator, particularly for outlier detection and gradient smoothing.

Definition 3.1 (Empirical Variance)**.** Empirical variance is calculated using the formula:

$$\hat{\sigma}^2_{[k]} = \frac{1}{N-1}\sum_{i=1}^{N}\left\|\hat{\nabla}^{[k]}F_i(\theta;h) - \bar{\nabla}^{[k]}F(\theta;h)\right\|^2,$$

where k can be either 0 or 1. Specifically, $\nabla^{[0]}F$ refers to the zeroth-order gradient estimators: PG ($\nabla^{[0]}J$) or QG ($\nabla^{[0]}Q$).

Case Study. To assess the efficacy of the SG method on intricate tasks, we employed the canonical `Ant` task from Brax (Freeman et al., 2021). Initiating from 64 starting states, we concurrently simulated 128 steps within the `Ant` task. Within each step, the empirical variances of PG, QG, and SG were evaluated. For brevity, we set the horizon length to 1 when computing SG, simplifying it to $\nabla_\theta(r_t(\pi_\theta) + \gamma\hat{Q}_{t+1}(\pi_\theta))$. As depicted in Fig. 2, QG's variance remains stable, outshining both SG and PG by several orders of magnitude in consistency. While the majority of SG's variances are relatively small, outliers do exist and will cause unreliable gradients for policy learning. PG, on the other hand, exhibits a higher variance. To delve deeper into the impact of SG's large variance during training, we pinpointed two states from the rollout, emblematic of low and high variance. Starting with a random policy, we iteratively employed optimization for the same state using SG gradients, monitoring alterations in returns ($r_t + \gamma\hat{Q}_{t+1}$). Notably, the optimized return is erratic in high variance states, underscoring the challenges faced by the SG method in such scenarios. It highlights the importance of identifying and replacing the high-variance SG during training.

Different from previous works that combine first-order gradients with PG (Parmas et al., 2018; Suh et al., 2022), our experiments reveal that PG can exhibit significantly larger

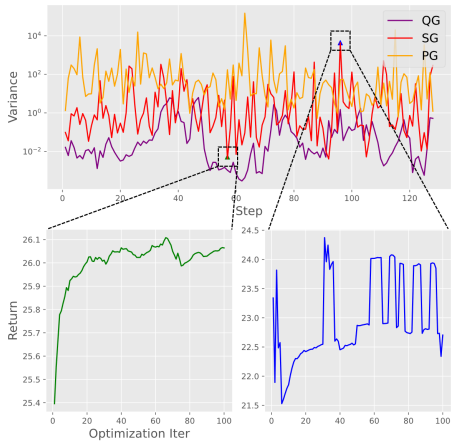


Figure 2. Empirical variance (in log scale) between SG, QG, and PG in a rollout (upper) and iterated optimization process using SG method in low(down left)/high(down right) variance states.

fluctuating empirical variance during policy rollouts. Instead, we posit that the Q function offers a more consistent proxy for predicting future returns, making QG a superior gradient estimator. Given the prevalence of Q functions in RL algorithms for policy learning (Lillicrap et al., 2015; Fujimoto et al., 2018), QG emerges as a compelling alternative gradient estimator. Our findings demonstrate the stability of QG, establishing it as a baseline for detecting deviations in SG.

3.3. Adaptive Gradient Estimator

We've demonstrated that outliers in SG can lead to policy collapse and empirical variance is an effective indicator for outliers. At the same time, QG consistently showcases low variance. This observation leads us to strategically blend SG and QG to address the challenge of non-smoothness. To this end, we introduce an adaptive analytic gradient, which merges SG and QG and adjusts their respective weights adaptively. When SG becomes unreliable, the weighting shifts in favor of QG. This moderates the impact of SG outliers and leads to stable policy optimization. Drawing from prior studies (Parmas et al., 2018; 2023; Suh et al., 2022), we employ a closed-form formula for blending gradients via inverse variance weighting. Specifically, we compute the mixture ratio with single-step SG and QG, as shown in Fig. 4. Importantly, the estimated Q value acts as a surrogate for expected returns, facilitating its use in place of multi-step returns in multi-step SG optimization.

Adaptive Mixture Ratio. Let the 1-step SG for ratio computation be represented as

$$\hat{\nabla}^{[1]}F(\theta;h=1) = \nabla_\theta\left[r(\mathbf{x},\mathbf{u},\mathbf{x}';\pi_\theta) + \gamma\hat{Q}(\mathbf{x}',\mathbf{u}';\pi_\theta)\right], \quad (1)$$

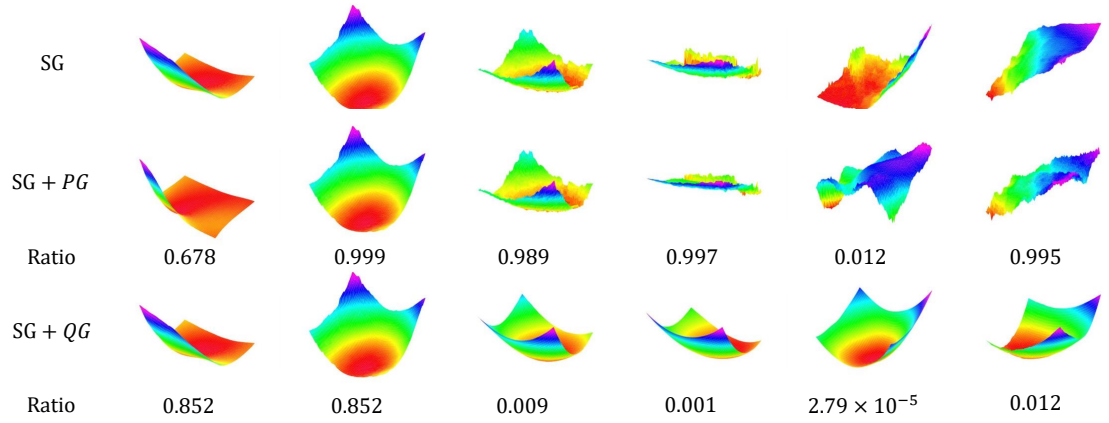


Figure 3. A comparison of the optimization landscapes among three types of gradient estimators: pure simulation gradient (SG), a mixture of simulation gradient and policy gradient (SG+PG), and a mixture of simulation gradient and Q-function gradient (SG+QG).

where $u \sim \pi_\theta(\mathbf{x})$, $u' \sim \pi_\theta(\mathbf{x}')$, and $\mathbf{x}' = \phi(\mathbf{x}, \mathbf{u})$. QG is computed at the current state by

$$\hat{\nabla}^{[0]} Q(\theta) = \nabla_\theta \hat{Q}(\mathbf{x}, \mathbf{u}; \pi_\theta), u \sim \pi_\theta(\mathbf{x}) \tag{2}$$

We follow previous works (Parmas et al., 2018; Parmas & Seno, 2022; Parmas et al., 2023) to compute the adaptive mixture ratio for inverse variance weighting, via

$$\alpha = \frac{\hat{\sigma}^2_{[0]}}{\hat{\sigma}^2_{[0]} + \hat{\sigma}^2_{[1]}}. \tag{3}$$

Thus, the adaptive analytic gradient is estimated by the weighted mixture of SG and QG as

$$\nabla^{[\alpha]} F(\theta; h) = \alpha \nabla^{[1]} F(\theta; h) + (1 - \alpha) \nabla^{[0]} Q(\theta).$$

Case Study. To assess the effects of different gradient mixtures, we follow a similar approach to previous works (Parmas et al., 2018; Xu et al., 2022), in which the objective landscapes are plotted to study issues with the gradients. We ran simulations across 64 `Ant` task environments using pre-trained policies and Q functions. Fig. 3 contrasts the optimization landscapes of six random simulation steps between the 1-step SG method and adaptive SG+PG and SG+QG mixtures. Each step involved random selection of two policy parameter space directions and plotting their loss landscapes. SG alone produced noisy landscapes with significant outliers, while SG+PG struggled to consistently address this issue. In contrast, our SG+QG mixture achieved smoother landscapes, showing our method's ability to navigate challenges and leverage SG effectively. During steps where SG was stable, the mixture inclined towards SG for optimization, resulting in similar landscapes between SG and the adaptive mixtures in stable scenarios.

4. ADAPTIVE-GRADIENT POLICY OPTIMIZATION

In this section, we further introduce a practical policy optimization algorithm that leverages the proposed adaptive analytic gradients, termed as Adaptive-Gradient Policy Optimization (AGPO). Fig. 1 compares the computational graphs of ours and the previous SHAC (Xu et al., 2022) algorithm. In line with SHAC, we segment the trajectory into shorter horizons to address gradient vanishing/explosion challenges. To account for future trajectory influences in policy learning, the Q value of the terminal state serves as an approximation for potential outcomes. For more stable transitions within these horizons, we estimate the Q value at every step rather than just at the end of the segment. Then we can compute the adaptive mixture ratio to evaluate the reliability of each transition.

In Fig. 4, we illustrate how both SG and QG are calculated at every step. Here, an important point is the node w.r.t. which we calculate the gradients. Early works (Parmas et al., 2018; Suh et al., 2022) calculated gradients w.r.t. policy parameters, while Parmas et al. (2023) showed that better scalability and computational efficiency can be obtained by computing the gradient variances at the last shared node between the gradient estimators (which in their case was the state node). Similarly, we achieve computational efficiency by computing gradient variances at the action node. To better determine whether the variance in gradients is caused by discontinuities in dynamics rather than differences in a specific action dimension due to varying actions taken in different environments, we calculated the L2 norm of the gradientss along the action dimension. Then, we compute a single adaptive mixture ratio across parallel environments with Eq. (3) , denoted as α_i, for the i-th step. This ratio acts as a gauge for the dependability of the current transition

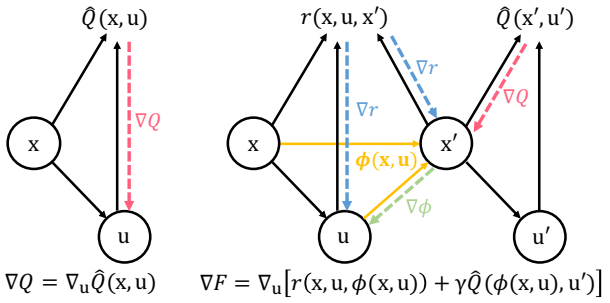


Figure 4. Detailed computational graphs of QG (left) and 1-step SG (right) for calculating α.

and its associated SG. When α_i is close to zero, it implies that the transition might be abrupt, potentially leading to inconsistencies in SG. Therefore, the SG associated with this transition becomes suspect for policy optimization. Such a scenario would consequently allocate lower credit to the left trajectory. During back-propagation, a near-zero ratio proficiently diminishes the influence of this rough step.

Throughout the truncated trajectory, we iteratively determine α_i as the credit assigned to future steps. For the t-th step, the cumulative credit, β_t, is given by $\beta_t = \prod_{k=0}^{t} \alpha_k$. If β_t descends below a specified threshold, the trajectory undergoes early truncation. Assuming $\beta_{-1} = 1$, the optimization objective for the policy π_θ is formulated as Eq. (4). The corresponding proof can be found in the Appendix.

$$\ell(\theta; h) = \frac{1}{h}\Big(\sum_{i=0}^{h-1} \gamma^i \Big((1-\alpha_i)\beta_{i-1}\hat{Q}_i + \beta_i r_i\Big) + \gamma^h \beta_{h-1}\hat{Q}_h\Big). \tag{4}$$

Our algorithm's pseudocode can be found in Alg. 1.

4.1. Theoretical Properties

In what follows, we establish the theoretical results of the proposed AGPO algorithm.

To begin, we impose a common regularity condition on the policy functions following previous studies (Agarwal et al., 2021; Zhang et al., 2023b).

Assumption 4.1 (Lipschitz and Bounded Score Function)**.** We assume that the score function of policy π_θ is Lipschitz continuous and has bounded norm, i.e.,

$$\big\|\log \pi_{\theta_1}(\mathbf{u}\,|\,\mathbf{x}) - \log \pi_{\theta_2}(\mathbf{u}\,|\,\mathbf{x})\big\|_2 \le L_1 \cdot \|\theta_1 - \theta_2\|_2,$$
$$\big\|\log \pi_\theta(\mathbf{u}\,|\,\mathbf{x})\big\|_2 \le B_\theta.$$

Now we characterize the convergence of AGPO with the following theorem.

Theorem 4.2 (Convergence to Stationary Points)**.** *Denote by b_m and v_m the gradient bias and variance at epoch $m \in [1, M]$, respectively, defined as*

$$b_m = \big\|\nabla_\theta \ell(\theta_m) - \mathbb{E}\big[\hat{\nabla}_\theta \ell(\theta_m)\big]\big\|_2,$$
$$v_m = \mathbb{E}\big[\big\|\hat{\nabla}_\theta \ell(\theta_m) - \mathbb{E}\big[\hat{\nabla}_\theta \ell(\theta_m)\big]\big\|_2^2\big].$$

Suppose the absolute value of the reward is bounded by $|r(\mathbf{x},\mathbf{u})| \le r_m$. Let $\delta = \sup \|\theta\|_2$, $L_0 = r_m \cdot L_1/(1-\gamma)^2 + (1+\gamma)\cdot r_m \cdot B_\theta^2/(1-\gamma)^3$, and $c = (\eta - L_0\eta^2)^{-1}$. Under Assumption 4.1, it holds for $M \ge 4L^2$ that

$$\min_{m\in[1,M]} \mathbb{E}\Big[\big\|\nabla_\theta \ell(\theta_m)\big\|_2^2\Big] \le \frac{4c}{M}\cdot \mathbb{E}\big[\ell(\theta_M) - \ell(\theta_1)\big] + \frac{4}{N}\Big(\sum_{m=1}^{M} c(2\delta\cdot b_m + \frac{\eta}{2}\cdot v_m) + b_m^2 + v_m\Big).$$

Besides, for a large enough N, $v_m \le 32L_{\hat{Q}}^2 L_\theta^2/h$ and $b_m \le \epsilon_m(\sum_{i=0}^{h-1}\gamma^i(1-\alpha_i)\beta_{i-1} + \gamma^h\beta_{h-1})/h$. Here, $L_{\hat{Q}} = \sup_{\psi,\mathbf{x}_1,\mathbf{x}_2,\mathbf{u}_1,\mathbf{u}_2} \|\hat{Q}_\psi(\mathbf{x}_1,\mathbf{u}_1) - \hat{Q}_\psi(\mathbf{x}_2,\mathbf{u}_2)\|_2/\|(\mathbf{x}_1 - \mathbf{x}_2, \mathbf{u}_1 - \mathbf{u}_2)\|_2$, $L_\theta = \sup_{\theta,\mathbf{x}} \|\nabla_\theta \pi_\theta(\mathbf{x})\|_2$, and $\epsilon_m = \max_i \|\nabla_\theta \hat{Q}(\mathbf{x}_i,\mathbf{u}_i) - \nabla_\theta Q^{\pi_{\theta_m}}(\mathbf{x}_i,\mathbf{u}_i)\|_2$.

Theorem 4.2 provides the variance and bias upper bound of the AGPO gradient and reveals the reliance between the convergence of AGPO and the variance, bias of the gradient estimators.

Notably, compared to the results in (Metz et al., 2021; Zhang et al., 2023b;c), their variance upper bounds exhibit an exponential dependency on h and the dynamics Lipschitz, resulting in chaotic optimization procedures and highly non-smooth optimization landscapes with exploding gradient

Algorithm 1 Adaptive-Gradient Policy Optimization

Input: Initialize policy π_{θ_0}, Q function Q_{ψ_0}, the target Q function $Q_{\psi_0'}$, and the learning rate η
repeat
  # Policy Rollout and Data Collection
  for step i = 1 to h **do**
    Sample actions $\mathbf{u} \sim \pi_{\theta_m}(\mathbf{x})$
    $\mathbf{x}', r, done \leftarrow \texttt{env.step}(\mathbf{x},\mathbf{u})$
    Compute $Q_{\psi_m'}(\mathbf{x},\mathbf{u})$ and $Q_{\psi_m'}(\mathbf{x}',\mathbf{u}')$
    Compute a single α_i using Eqs. (1) to (3)
    Add experience tuple for this step to buffer
    $\mathbf{x} \leftarrow \mathbf{x}'$
  end for
  # Update Policy
  $\theta_{m+1} \leftarrow \theta_m + \eta \cdot \nabla_\theta \ell(\theta_m; h)$
  # Update Q function and its moving average
  Compute Bellman target Q' as $r + \gamma Q_{\psi_m'}(\mathbf{x}',\mathbf{u}')$
  $\psi_{m+1} \leftarrow \psi_m + \eta \cdot \nabla_\psi \mathrm{MSE}(Q_{\psi_m}, Q')$
  $\psi'_{m+1} \leftarrow \tau\psi_{m+1} + (1-\tau)\psi'_m$
until termination condition is met

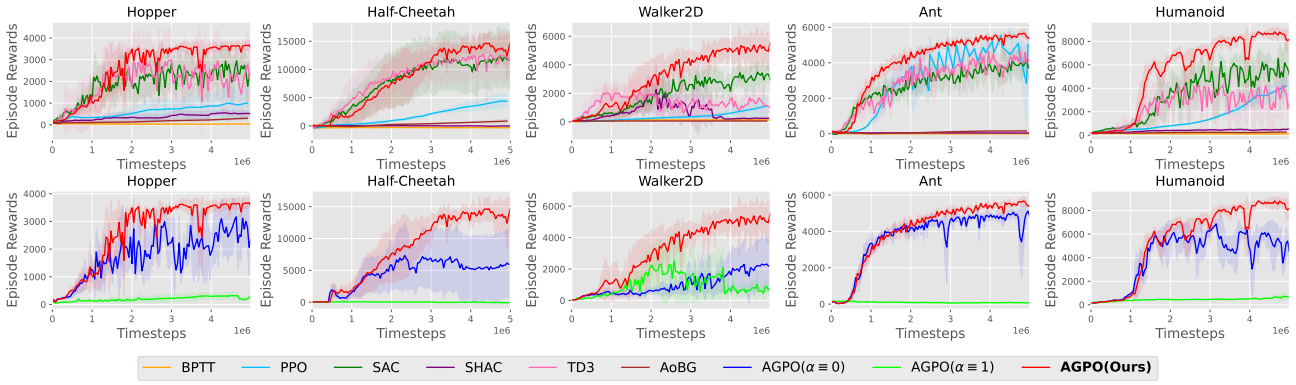


Figure 5. Experiment results on classic control tasks.

variance, causing slow convergence. This is especially the case when the dynamics are non-smooth and the horizon h is large. On the contrary, AGPO addresses this issue by adaptively mixing SG and QG, whose variance is low. Therefore, increasing the horizon h leads to a faster convergence with lower upper bound of the gradient variance and bias.

Finally, we establish the convergence rate of AGPO.

Corollary 4.3 (Convergence Rate)**.** *Let* $\varepsilon(M) = \sum_{m=1}^{M} b_m$. *Suppose that* $L_{\hat{Q}}$ *and* L_θ *are bounded. Then we have for* $M \geq 4L^2$ *that*

$$\min_{m\in[1,M]} \mathbb{E}\left[\left\|\nabla_\theta \ell(\theta_m)\right\|_2^2\right] \leq 16\delta \cdot \varepsilon(M)/\sqrt{M} + 4\varepsilon^2(M)/M + O\big(1/\sqrt{M}\big).$$

4.2. Implementation Details

We implemented our algorithm using JAX (Bradbury et al., 2018) for empirical analysis. Although compatible with both reparameterized stochastic and deterministic policies, we opted for a reparameterized Gaussian stochastic policy, incorporating entropy regularization to enhance exploration. To counteract Q learning's overestimation bias, we used double Q functions and stabilized training by employing the moving average of Q functions for QG computation and setting Bellman targets. A KL penalty, or "trust region" approach, was applied to moderate policy updates. We standardize gradients by the median of batched gradients before calculating α_i to manage scale differences between estimators.[1]

[1]We normalize the gradient magnitudes when computing the empirical variance during the calculation of α_i, then use the raw gradients for blending. As shown in (Parmas et al., 2023), rescaling gradient magnitudes for ratio estimation may lead to suboptimal performance. However, we find empirically that it works reasonably well in our tasks, leaving a deeper exploration for future work.

5. EXPERIMENTS

In this section, we conduct extensive experiments to answer these research questions: (1) How does AGPO's performance compare to pure RL methods and others utilizing differentiable simulation? (2) What effect does horizon length have on learning? (3) Does our adaptive method surpass other rule-based techniques in addressing SG outliers? (4) Is our method consistently effective across various simulation backends, encompassing both smooth and non-smooth differentiability?

5.1. Experiment Setup

Unless otherwise specified, all experiments are conducted in the Brax environment (Freeman et al., 2021) using its `generalized` backend, which computes motion via generalized coordinates similar to Mujoco (Todorov et al., 2012), resolving contact impulses through a constraint solver. Episodes are set to a length of 1000, with early termination possible based on task-specific conditions. We compare our AGPO algorithm with the following baselines:

- Backpropagation Through Time (BPTT): This is the most straightforward method to leverage simulation gradients, which directly optimizes cumulative rewards. As in (Freeman et al., 2021), we truncate the horizon into short segments to avoid gradient issues.
- Short-Horizon Actor-Critic (SHAC) (Xu et al., 2022) and Alpha-order Batched Gradient (AoBG) (Suh et al., 2022): We implement JAX-based SHAC and AoBG for Brax tasks, which improve the vanilla BPTT with either a value function or policy gradients.
- Pure RL methods: PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), and TD3 (Fujimoto et al., 2018). We utilize the implementations provided by Stable Baselines3 (Raffin et al., 2021) and add a custom wrapper for our simulation environments.

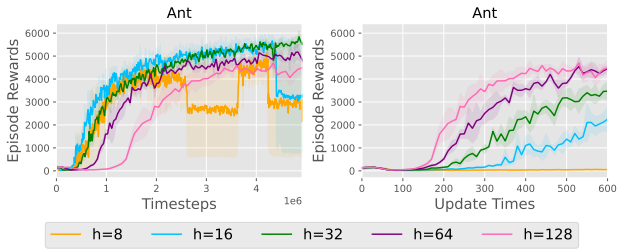


Figure 6. Comparison of performance for AGPO on the `Ant` task with increasing horizon lengths.

5.2. Continuous Control Benchmarks

We select five representative control tasks to evaluate the performance of our method, spanning from the control-less `Hopper` to the control-rich `Humanoid`. We run each method for five individual runs to report the average performance on each problem. The results are shown in the Fig. 5, in which the solid line indicates the average performance and the shaded area is the variance over random seeds.

Across all tasks, SHAC consistently outperforms the naive BPTT. However, both methods are surpassed by others, indicating that optimization solely relying on leveraging differentiability faces challenges in this testbed. The presence of hard contacts and solver-based contact modeling poses difficulties. In contrast, our proposed AGPO method not only significantly outperforms both SHAC and BPTT but also competes favorably with widely-recognized methods such as PPO, SAC, and TD3.

PPO, as an on-policy RL method, requires many samples for good performance and struggles with limited data, as our results show. In contrast, our method improves sample efficiency by exploiting the simulation's differentiability and outperforms PPO. SAC and TD3, as off-policy methods, train effectively with less data and converge faster than PPO, but do not use the simulation's differentiability. Our method surpasses both SAC and TD3, highlighting the advantages of differentiable simulation for policy learning.

Additionally, we compare our method against two fixed-ratio variants, where the mixture ratio α is set as a constant (either 0 or 1), to highlight the benefits of adaptivity. With $\alpha \equiv 1$, relying solely on SG leads to subpar performance across all tasks. Conversely, $\alpha \equiv 0$ shows improvement, emphasizing QG's adaptability. However, the adaptive mixture of SG and QG outperforms both, showcasing the strengths of SG's efficacy and QG's adaptability combined.

5.3. Ablation Study

To further probe the nuances of AGPO, we conduct several ablation studies in this section. We run each trial for three different seeds and present the average results.

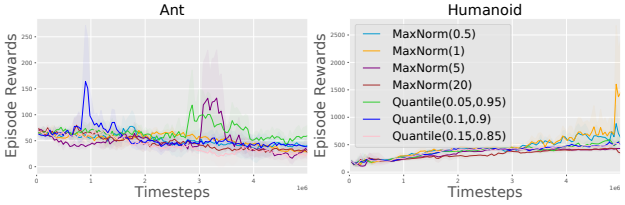


Figure 7. Rule-based methods to regulate SG outliers.

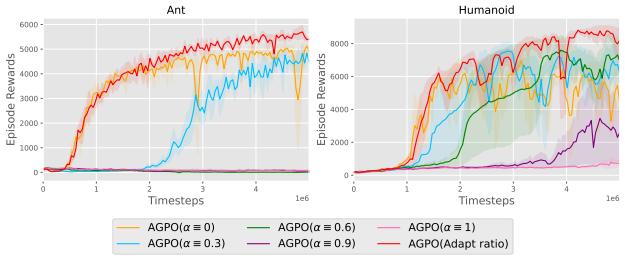


Figure 8. Experiments with fixed intermediate mixture ratio α.

Horizon length h. We conduct controlled experiments in the `Ant` environment to examine the impact of different horizon lengths. The results are illustrated in Fig. 6. We observe that as the horizon increases from $h = 8$ to $h = 128$, the performance monotonically improves under the same number of updates. This observation aligns with our theoretical findings that a large horizon leads to a faster convergence to the stationary points. Balancing training efficiency w.r.t. timesteps and w.r.t. update times, we choose $h = 32$ as the default value for other experiments.

Rule-based regulation and fixed-ratio mixture. To further assess the effect of the proposed adaptive analytic gradient, we compare our method with two rule-based outlier elimination methods as well as several fixed-ratio variants. For rule-based regulators, we devise two outlier elimination methods: *MaxNorm*, which clips the SG norm, and *Quantile*, using gradients within a specific quantile range. However, as Fig. 7 shows, neither method effectively overcomes non-smoothness in policy learning. While hard clipping removes outliers and prevents model collapse, it also discards crucial task information. Our approach, in contrast, leverages QG to compensate for outlier removal and employs zeroth-order methods to bypass non-smooth first-order optimization, resulting in more effective policy training. In addition, we carried out experiments to compare our method with some fixed-ratio variants using intermediate values of α (0.3, 0.6, 0.9) in `Ant` and `Humanoid`. These experiments were designed in a manner akin to those described in Sec. 5.3, where α was set to 0 and 1. Here, however, α was set to 0.3, 0.6, and 0.9 for the entire duration of the training process. As depicted in Fig. 8, the results suggest that consistently incorporating a fixed ratio of QG to SG can mitigate some challenges associated with the SG method to a certain de-

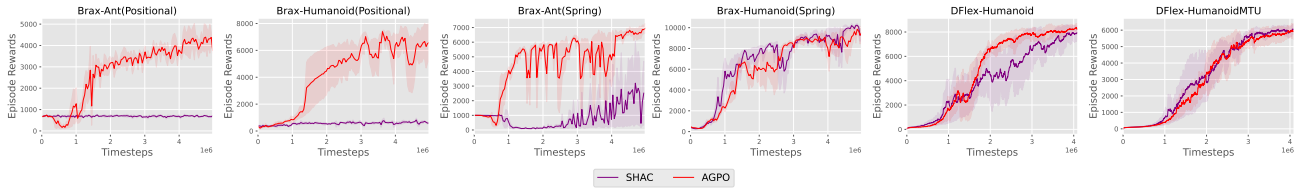
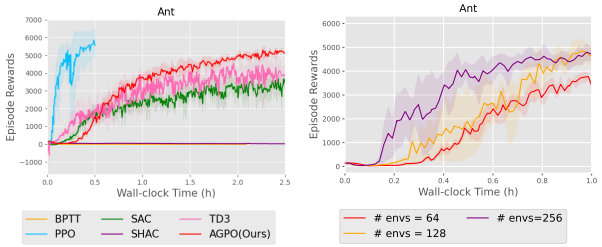


Figure 9. Comparison between SHAC and AGPO on additional various simulations in Brax and DFlex.



(a) Computational efficiency comparison. (b) Scalability with the number of environments.

Figure 10. Analysis on computational efficiency and scalability.

gree. Nonetheless, the performance of these fixed-ratio approaches still falls short when compared to the adaptive ratio method.

Computational efficiency and scalability. We conducted experiments to benchmark the computational efficiency of our proposed method against existing algorithms. These evaluations were carried out across 64 parallel environments, ensuring a fair comparison. The results on `Ant` are shown in Fig. 10 while more results can be found in the Appendix. Despite the integration of an extra backpropagation operation within our method, it demonstrated wall-clock efficiency on par with established methods such as SAC and TD3. Furthermore, an intriguing aspect of our method is its scalability: as we augmented the number of parallel environments, our approach exhibited enhanced wall-clock efficiency. This scalability suggests that our method is not only competitive under current testing conditions but also stands to gain significantly from increased parallelism, highlighting its potential for future applications requiring high computational efficiency.

Simulation backends. To demonstrate our method's versatility, we evaluate AGPO in three additional simulations, including Brax's `positional` and `spring` backends (Freeman et al., 2021) and the DFlex simulator used in SHAC (Xu et al., 2022). The `positional` backend, based on Position-based Dynamics (Müller et al., 2007), balances speed and stability, differing in complexity from `generalized` and `spring`. The `spring` backend uses simple impulse-based methods for cost-effective simulation. DFlex features a frictional contact model, enhancing contact dynamics smoothness, where SHAC has shown significant improvements compared to existing PG- or SG-based methods. We select two representative tasks, `Ant` and `Humanoid`, from Brax and high-dimensional control tasks, `Humanoid` and `HumanoidMTU`, from DFLex. As illustrated in Fig. 9, AGPO consistently surpasses or matches SHAC across these simulations, maintaining comparable performance even in the smoothly differentiable DFlex environments with high-dimensional action spaces.

Our experiments across different simulation backends reveal that SHAC struggles in simulations that lack smooth differentiability. This suggests that SG-based methods are significantly influenced by the specific design of the simulator. On the other hand, AGPO shows both adaptability and consistent performance across various simulation environments, underscoring its ability to handle diverse scenarios.

6. CONCLUSION

In this paper, we tackle policy learning in non-smooth differentiable simulations by introducing an adaptive analytic gradient that adeptly mixes unreliable simulation gradients (SG) with more reliable Q gradients (QG). Our adaptive-gradient policy optimization (AGPO) algorithm demonstrates resilience with proven convergence and low variance in non-smooth scenarios. Empirical results indicate that our method surpasses existing algorithms in non-smooth settings. Ablation studies further validate our approach's effectiveness, highlighting its reduced reliance on the smoothness of differentiable simulations and its capacity to navigate non-smooth landscapes for consistent policy improvement. This advancement marks significant progress in resilient policy learning across various differentiable simulations.

Limitations and future directions. Our method requires additional backpropagation steps to compute the mixture ratio, which could be accelerated with improved engineering. More advanced gradient composition techniques, such as incorporating covariance and using tensor-valued weights (Parmas et al., 2023), could potentially further enhance performance, significantly improving our approach's efficiency and effectiveness.

Impact Statement

This paper presents advancements in machine learning through the Adaptive-Gradient Policy Optimization (AGPO) algorithm, primarily impacting fields involving simulation-based learning and decision-making for robotic applications. Our work does not introduce specific ethical challenges beyond those already established in machine learning.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Carpentier, J. and Mansard, N. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and systems (RSS 2018)*, 2018.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

Fleiss, J. L. Review papers: The statistical basis of meta-analysis. *Statistical methods in medical research*, 2(2): 121–145, 1993.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Heiden, E., Millard, D., Coumans, E., Sheng, Y., and Sukhatme, G. S. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9474–9481. IEEE, 2021.

Howell, T. A., Le Cleac'h, S., Kolter, J. Z., Schwager, M., and Manchester, Z. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S., and Kim, T.-Y. Primal/dual descent methods for dynamics. In *Computer Graphics Forum*, volume 39, pp. 89–100. Wiley Online Library, 2020.

Metz, L., Freeman, C. D., Schoenholz, S. S., and Kachman, T. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Mora, M. A. Z., Peychev, M., Ha, S., Vechev, M., and Coros, S. Pods: Policy optimization via differentiable simulation. In *International Conference on Machine Learning*, pp. 7805–7817. PMLR, 2021.

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

Parmas, P. and Seno, T. Proppo: a message passing framework for customizable and composable learning algorithms. *Advances in Neural Information Processing Systems*, 35:29152–29165, 2022.

Parmas, P. and Sugiyama, M. A unified view of likelihood ratio and reparameterization gradients. In *International Conference on Artificial Intelligence and Statistics*, pp. 4078–4086. PMLR, 2021.

Parmas, P., Rasmussen, C. E., Peters, J., and Doya, K. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pp. 4065–4074. PMLR, 2018.

Parmas, P., Seno, T., and Aoki, Y. Model-based reinforcement learning with scalable composite policy gradient estimators. In *International Conference on Machine Learning*, pp. 27346–27377. PMLR, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Peng, X. B., Ma, Z., Abbeel, P., Levine, S., and Kanazawa, A. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4):1–20, 2021.

Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. C. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pp. 8661–8671. PMLR, 2021.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Suh, H. J., Simchowitz, M., Zhang, K., and Tedrake, R. Do differentiable simulators give better policy gradients? In *Proceedings of the 39th International Conference on Machine Learning*, pp. 20668–20696, 2022.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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

Todorov, E. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6054–6061. IEEE, 2014.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Xu, J., Makoviychuk, V., Narang, Y., Ramos, F., Matusik, W., Garg, A., and Macklin, M. Accelerated policy learning with parallel differentiable simulation. In *International Conference on Learning Representations*, 2022.

Zhang, H., Yuan, Y., Makoviychuk, V., Guo, Y., Fidler, S., Peng, X. B., and Fatahalian, K. Learning physically simulated tennis skills from broadcast videos. *ACM Trans. Graph.*, 42(4), jul 2023a. ISSN 0730-0301. doi: 10.1145/3592408. URL `https://doi.org/10.1145/3592408`.

Zhang, S., Jin, W., and Wang, Z. Adaptive barrier smoothing for first-order policy gradient with contact dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 41219–41243. PMLR, 2023b.

Zhang, S., Liu, B., Wang, Z., and Zhao, T. Model-based reparameterization policy gradient methods: Theory and practical algorithms. *arXiv preprint arXiv:2310.19927*, 2023c.

Zhong, Y. D., Han, J., and Brikis, G. O. Differentiable physics simulations with contacts: Do they have correct gradients wrt position, velocity and control? In *ICML 2022 2nd AI for Science Workshop*, 2022.

A. Proofs

A.1. Proof of Theorem 4.2

The theorem contains three results: the convergence rates to stationary points, the bound of the gradient variance, and the bound of the gradient bias.

We refer to Theorem 4.2 of (Zhang et al., 2023b) for the proof of the first result, i.e., the convergence rates to stationary points. Below we give the proof of the bound of the gradient variance and bias, respectively.

Proof. For the bound of the gradient variance, we first upper-bound the variance of the QG $\sigma^2_{[0]} = \mathbb{E}[\|\hat{\nabla}^{[0]}Q(\theta) - \mathbb{E}[\hat{\nabla}^{[0]}Q(\theta)]\|_2^2]$, we characterize the norm inside the outer expectation.

Consider an *arbitrary* action sampled from the policy π_θ at a fixed state $\overline{\mathbf{x}}$. We denote its QG $\hat{\nabla}^{[0]}Q(\theta)$ as g'. Then we have

$$\sigma^2_{[0]} \leq \max_{g'} \left\| g' - \mathbb{E}\big[\hat{\nabla}^{[0]}Q(\theta)\big] \right\|_2^2 = \left\| g - \mathbb{E}\big[\hat{\nabla}^{[0]}Q(\theta)\big] \right\|_2^2 = \left\| \mathbb{E}\big[g - \hat{\nabla}^{[0]}Q(\theta)\big] \right\|_2^2,$$

where g is the pathwise gradient $\hat{\nabla}^{[0]}Q(\theta)$ of action $\overline{\mathbf{u}}$ such that the maximum is achieved.

Using the fact that $\|\mathbb{E}[\cdot]\|_2 \leq \mathbb{E}[\|\cdot\|_2]$, we further obtain

$$\begin{aligned}
\sigma^2_{[0]} &\leq \mathbb{E}\Big[\big\|g - \hat{\nabla}_\theta \ell(\pi_\theta)\big\|_2\Big]^2 \\
&\leq \mathbb{E}_{\mathbf{u}}\Big[\big\|\nabla_u \hat{Q}(\overline{\mathbf{x}}, \overline{\mathbf{u}})\nabla_\theta \overline{\mathbf{u}} - \nabla \hat{Q}(\mathbf{x}, \mathbf{u})\nabla_\theta \mathbf{u}\big\|_2\Big]^2 \\
&= \mathbb{E}_{\mathbf{u}}\Big[\big\|\nabla_u \hat{Q}(\overline{\mathbf{x}}, \overline{\mathbf{u}})\nabla_\theta \overline{\mathbf{u}} - \nabla_u \hat{Q}(\overline{\mathbf{x}}, \overline{\mathbf{u}})\nabla_\theta \mathbf{u} + \nabla_u \hat{Q}(\overline{\mathbf{x}}, \overline{\mathbf{u}})\nabla_\theta \mathbf{u} - \nabla_u \hat{Q}(\mathbf{x}, \mathbf{u})\nabla_\theta \mathbf{u}\big\|_2\Big]^2 \\
&\leq 16 L^2_{\hat{Q}} L^2_\theta.
\end{aligned}$$

When N is large enough, we have

$$\begin{aligned}
\nabla^{[\alpha]}F(\theta; h) &= \alpha \nabla^{[1]}F(\theta; h) + (1-\alpha)\nabla^{[0]}Q(\theta) \\
&= \frac{\sigma^2_{[0]}}{\sigma^2_{[0]} + \sigma^2_{[1]}} \nabla^{[1]}F(\theta; h) + \frac{\sigma^2_{[1]}}{\sigma^2_{[0]} + \sigma^2_{[1]}} \nabla^{[0]}Q(\theta).
\end{aligned}$$

Therefore, we obtain for the variance that

$$\begin{aligned}
v_m &\leq \frac{1}{h}\left(\frac{\sigma^2_{[0]}}{\sigma^2_{[0]} + \sigma^2_{[1]}} \sigma^2_{[1]} + \frac{\sigma^2_{[1]}}{\sigma^2_{[0]} + \sigma^2_{[1]}} \sigma^2_{[0]} \right) \\
&= \frac{2\sigma^2_{[1]}}{h(\sigma^2_{[0]} + \sigma^2_{[1]})} \sigma^2_{[0]} \\
&\leq \frac{32\sigma^2_{[1]}}{h(\sigma^2_{[0]} + \sigma^2_{[1]})} L^2_{\hat{Q}} L^2_\theta \\
&\leq 32 L^2_{\hat{Q}} L^2_\theta / h,
\end{aligned}$$

where the terms in the brackets of the first inequality are the variance of each time step in the h-horizon.

The bias can be upper bounded by establishing the error of the gradient of the state-action value function at each timestep in (4) with ϵ_m. □

A.2. Proof of Corollary 4.3

Proof. We let the learning rate $\eta = 1/\sqrt{M}$. Then for $M \geq 4L^2$, we have $c = (\eta - L_0\eta^2)^{-1} \leq 2\sqrt{T}$ and $L_0\eta \leq 1/2$. By setting $N = O(\sqrt{T})$, we obtain

$$\begin{aligned}
\min_{m\in[1,M]} \mathbb{E}\Big[\big\|\nabla_\theta \ell(\theta_m)\big\|_2^2\Big] &\leq \frac{4}{M}\cdot\left(\sum_{m=0}^{M-1} c\cdot(2\delta\cdot b_m + \frac{\eta}{2}\cdot v_m) + b_m^2 + v_m\right) + \frac{4c}{T}\cdot\mathbb{E}\big[\ell(\theta_M) - \ell(\theta_1)\big] \\
&\leq \frac{4}{M}\left(\sum_{m=0}^{M-1} 4\sqrt{M}\delta\cdot b_m + b_m^2 + 2v_m\right) + \frac{8}{\sqrt{M}}\cdot\mathbb{E}\big[\ell(\theta_M) - \ell(\theta_1)\big] \\
&\leq \frac{4}{M}\left(\sum_{m=0}^{M-1} 4\sqrt{T}\delta\cdot b_m + b_m^2\right) + O\big(1/\sqrt{M}\big) \\
&\leq \frac{16\delta}{\sqrt{M}}\varepsilon(M) + \frac{4}{M}\varepsilon^2(M) + O\big(1/\sqrt{M}\big).
\end{aligned}$$

This concludes the proof. □

A.3. Proof of Eq. (4)

Proof. Let $\beta_t = \prod_{k=0}^{t}\alpha_k$, $\beta_{-1} = 1$, and $F_{m:n}$ denote the first-order optimization objective for the trajectory from state $\mathbf{x}_m$ to $\mathbf{x}_n$.

Starting with the mixture strategy for a horizon of length h, the objective is:

$$\ell(\theta; h) = (1-\alpha_0)\hat{Q}_0 + \alpha_0 F_{0:h}.$$

Utilizing the Bellman equation to expand $F_{0:h}$:

$$\ell(\theta; h) = (1-\alpha_0)\hat{Q}_0 + \alpha_0(r_0 + \gamma Q_1).$$

By recursively applying the mixture strategy and Bellman equation, the expression evolves to:

$$\ell(\theta; h) = (1-\alpha_0)\hat{Q}_0 + \gamma\alpha_0(1-\alpha_1)\hat{Q}_1 + \alpha_0 r_0 + \gamma\alpha_0\alpha_1 F_{1:h}.$$

Iterating for all states up to h, the objective consolidates to:

$$\ell(\theta; h) = \sum_{i=0}^{h-1}\gamma^i\left((1-\alpha_i)\beta_{i-1}\hat{Q}_i + \beta_i r_i\right) + \gamma^h\beta_{h-1}\hat{Q}_h.$$

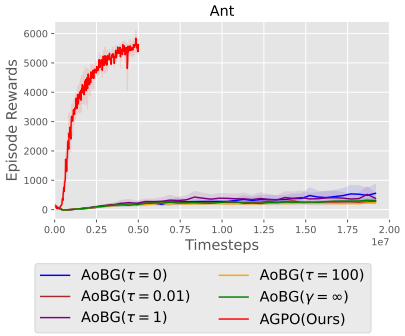
□

B. Details of PG implementation

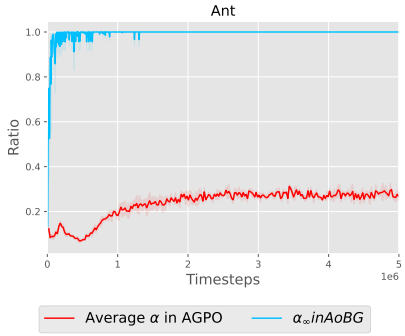
For the case studies in Sec 3.2 and Sec 3.3, we utilized policy function π and Q-function selected from a successful run of our AGPO algorithm. The policy function can be represented as a Gaussian policy, $\mu(\mathbf{x}) + \sigma(\mathbf{x}) * \mathcal{N}(0, I)$. For computing the advantage function in PG, we use $\hat{Q}(\mathbf{x}, \mu(\mathbf{x}))$ to serve as the baseline instead of using the value function in PPO (Schulman et al., 2017). This modification is based on two assumptions. Firstly, if $\hat{V} = V^*$ and $\hat{Q} = Q^*$, where * denotes the optimal value and Q-function, then we have $V^*(\mathbf{x}) = \max_a Q^*(\mathbf{x}, \mathbf{u})$. Secondly, based on most experimental results, we observe that the optimal value of Q often occurs when the action is chosen as $\mu(\mathbf{x})$.

C. Comparison with AoBG (Suh et al., 2022)

In this section, we expanded our experimental scope to include a direct comparison with the method proposed by Suh et al.(2022), on the `Ant` task. Due to the absence of open-source code from Suh et al.(2022), we implemented their method by



(a) Comparison of overall performance.



(b) Comparison of mixture ratio

Figure 11. Comparison with (Suh et al., 2022) on `Ant` task.

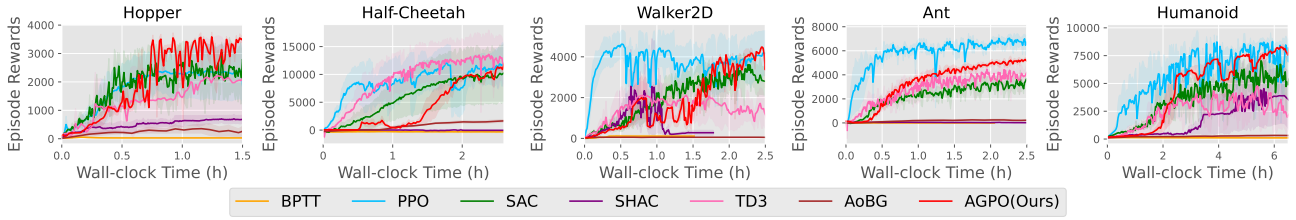


Figure 12. Computational efficiency comparison on classic control tasks in Brax.

ourselves. Specifically, we define SG as described in the main body

$$\bar{\nabla}^{[1]} F(\theta; h) := \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(\theta; h).$$

For PG, we implemented as

$$\bar{\nabla}^{[0]} F(\theta; h) := \frac{1}{M} \sum_{i=1}^{M} A(x_i, u_i) \nabla_\theta \log \pi(u_i | x_i; \theta),$$

where M denotes the number of transitions sampled, and $A(x_i, u_i)$, representing the advantage, follows the same definition as in PPO. To combine SG and PG, we use the adaptive ratio define as Eq (5) in (Suh et al., 2022). By denote $\tau = \gamma - \epsilon$, we test the performance of algorithms with setting $\tau = 0, 0.01, 1, 100$ and $\gamma = \infty$. Importantly, when $\gamma = \infty$, it will fall back to the original inverse-variance weighting scheme used by Parmas et al. (2018). Our findings reveal that (Suh et al., 2022) exhibits poor performance on the Brax environment (Fig. 11a). Furthermore, we plotted the adaptive ratio α_∞ in the experiment with $\gamma = \infty$ in the PG+SG method. For comparison, we also provided the change of the average adaptive ratio α when AGPO was run in the `Ant` task. Notably, we found α_∞ to be almost equal to 1 (Fig. 11b), indicating that the PG method itself has a large variance. This finding is consistent with the result in Fig. 2 and 3. Conversely, the average α in AGPO is reasonable, suggesting that QG has a greater ability to detect and complement SG outliers.

D. Computational Efficiency

Here, we evaluate the computational efficiency of our proposed method against existing algorithms across all five tasks. The results in Fig. 12 show that the wall-clock efficiency of our AGPO is on par with established methods such as SAC and TD3.

E. More Intermediate Fixed-alpha Experiments

In addition to the experiments in Fig. 8, we conducted further fixed-alpha experiments with $\alpha \in 0.1, 0.2, 0.3$, encompassing the range of the learned adaptive ratio. As shown in Fig. 13, while a carefully selected fixed-ratio gradient can achieve comparable final performance, our AGPO consistently demonstrates superior efficiency due to its adaptive mixture schedule.

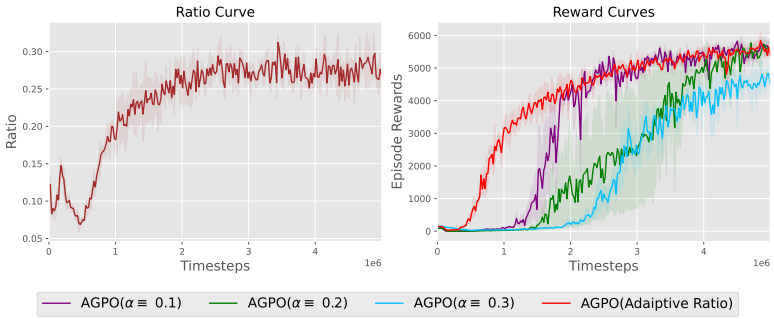


Figure 13. Experiments with fixed intermediate mixture ratio α.

	Ant	Humanoid	Hopper	Walker2d	Half-Cheetah
num_envs	64	64	64	64	64
num_eval_envs	128	128	128	128	128
truncation_length	32	32	32	32	32
actor_hidden_sizes	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$
critic_hidden_sizes	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$	$[256, 128, 64]$
stochastic_policy	True	True	True	True	True
reward_scaling	0.1	0.1	0.1	0.1	0.1
discounting_factor γ	0.99	0.99	0.99	0.99	0.99
soft_update_rate τ	0.8	0.2	0.8	0.8	0.2
learning_rate	7e-4	7e-4	2e-4	2e-4	2e-4
actor_max_grad_norm	1	1	1	1	1
critic_max_grad_norm	10	10	10	10	10
KL_penalty_coef	0.3	0.3	0.3	0.3	0.3
entropy_coef	0.01	0.01	0.01	0.01	0.01
critic_num_minibatch	8	8	8	8	8
critic_num_iteration	16	16	16	16	4
critic_buffer_size	64×32	64×32	64×32	64×32	64×32
β_truncation_threshold	0.1	0.1	0.1	0.01	0.1

Table 1. Training hyper-parameters for AGPO.

F. Experiment Details

F.1. Computation Resources

We conducted our experiments on one NVIDIA GeForce RTX 3090 GPU with 24 GB GDDR6X memory. We implemented our codes on the JAX framework, supporting XLA and automatic differentiation.

F.2. Hyper-parameters

We carried out our experiments using the Brax environment. For more specific implementation details related to each task, please refer to (Freeman et al., 2021). Below are the hyper-parameters for our proposed AGPO algorithm applied to each task.