IMPROVING TRUNCATED FIRST-ORDER GRADIENTS FOR FLOW POLICIES WITH SURROGATE OBJECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable simulation offers a way to improve the sample efficiency of reinforcement learning for robotics. However, correctly utlizing the first-order information from the simulator for training vision-based flow-matching policies remains challenging. Visual observations may be nondifferentiable with respect to the environment state, and iterative action generation in flow matching makes end-to-end backpropagation costly. Truncating the observation gradient path makes the analytical policy gradient (APG) tractable, but the resulting truncated APG is biased. Likelihood-ratio policy gradients (LRPGs) avoid this differentiable requirement, but can suffer from high variance. We introduce Differentiable Flow Policy Optimization (DFPO), a method that combines these complementary signals to improve policy updates under incomplete differentiation. DFPO uses LRPG, estimated with a flow-matching surrogate objective, to improve the update direction of the truncated APG. On classical control problems, we show that the combined update has low variance and is better aligned with the true policy gradient. We further evaluate DFPO on robotic tasks with visual and state observations. In the vision-based setting, DFPO consistently outperforms baselines using either APG or LRPG alone. In the state-based setting, it achieves performance comparable to methods using the full APG. These results demonstrate that DFPO, which combines truncated APG with LRPG, can support effective reinforcement learning of vision-based flow-matching policies.

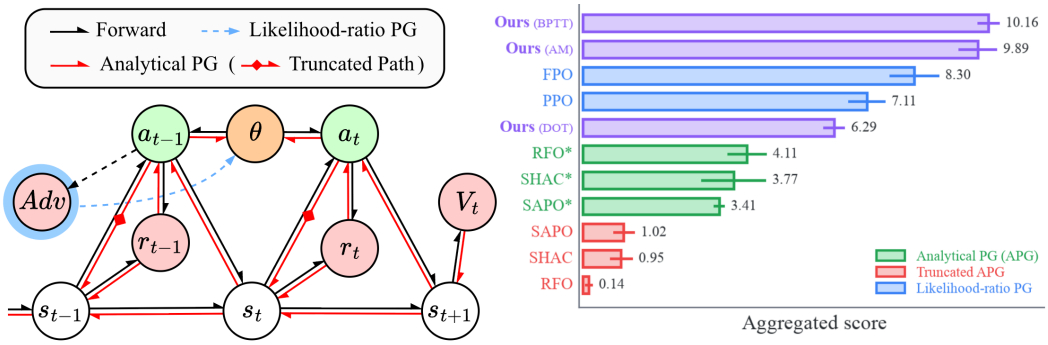


Figure 1: `Left`: Illustration of Analytical PG, Truncated Analytical PG, and Likelihood-ratio PG. `Right`: Aggregated normalized score across 12 tasks: 6 state-based and 6 RGB-based.
* Experiments were conducted on 6 state-based tasks only with access to the full gradient path.

1 INTRODUCTION

Differentiable simulation can substantially accelerate policy optimization by providing first-order gradients of the return: when the transition and reward functions are differentiable, the return gradient can be computed directly by backpropagating through the rollout (Xing et al., 2025; Zhong et al., 2026; Xu et al., 2022; You et al., 2025; Suh et al., 2022). In such a paradigm, the full analytical policy gradient (APG) with respect to policy parameters needs to differentiate through the policy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

observation, as illustrated by the green arrows in Fig. 1, which captures action sensitivity to observation changes. However, computing this gradient path is not trivial. Most simulation rendering process that generates the visual observations are non-differentiable, and even with differentiable rendering, backpropagating through high-dimensional images and visual policies (which are usually complex) makes it prohibitively expensive. As a result, most prior work using differentiable simulation focuses on state-based policies. A natural solution for generalizing this to visual policies is to truncate the observation gradient computation path (You et al., 2025; Zhang et al., 2025b). This truncation makes the computation of the analytic policy gradient practical, but introduces bias as it removes part of the dependence of future actions and returns on the policy parameters. In contrast, a likelihood-ratio policy gradient (LRPG) can optimize the policy without differentiating through the environment or observation path, but typically has higher variance because it relies on sampled trajectories (Schulman et al., 2015; Xu et al., 2019). The two signals provide complementary information: the truncated APG (TAPG) offers a low-variance but biased analytic gradient (which is a first-order signal), while the LRPG provides a higher-variance update direction. **The central question we explore in this work is: does combining these complementary signals lead to better policy updates?**

This question becomes particularly relevant for generative policies operating on high-dimensional observations, where computing the full APG is most challenging. Diffusion and flow-based policies can represent complex distributions over actions (Chi et al., 2023; Lipman et al., 2023), and have been shown to achieve strong performance in vision-based manipulation tasks and robot control (Intelligence et al., 2025b;a; Shukor et al., 2025). Unlike Gaussian policies, which parameterize the action distribution in a prescribed form, these generative models generate actions through iterative denoising from a simple noise distribution, making them well-suited to multi-modal behaviors where multiple actions may be valid under the same observation.

Applying common reinforcement learning (RL) algorithms based on LRPG to these policies is difficult, because the resulting action is the final output of a multi-step generative process, whose likelihood is generally unavailable in an analytical form. Recent approaches address this difficulty through stochastic denoising processes, likelihood surrogates, or policy distillation (Ren et al., 2025; Zhang et al., 2025a; McAllister et al., 2026; Park et al., 2025). However, these methods primarily adapt likelihood-ratio or value-based optimization to generative policies, while the use of differentiable simulation to directly optimize flow policies remains largely unexplored. This raises two practical questions: how to incorporate first-order gradients from differentiable simulation without constructing a prohibitively large end-to-end computation graph, and how to efficiently propagate these gradients through the iterative denoising process of a flow-matching policy.

In this work, we introduce Differentiable Flow Policy Optimization (DFPO), a first-order RL method for vision-based flow-matching policies. DFPO combines the biased truncated APG obtained from differentiable simulation with an LRPG. We evaluate the resulting update in terms of its variance and alignment with the true policy gradient, and study its performance in robot control tasks. We also compare different strategies in propagating action gradients into flow-matching policy parameters, and characterize their computational trade-offs. The same propagation problem also arises when action gradients are obtained from other sources, such as differentiable Q-functions.

Our contributions are summarized as follows:

- We introduce DFPO, which combines a low-variance but biased TAPG from differentiable simulation with an LRPG signal to improve policy-gradient updates. We show on linear-quadratic regulator problems that combining the signals gives better updates.
- We conduct a systematic evaluation across state-based and RGB-based control tasks, and demonstrate that DFPO consistently outperforms either gradient estimator alone and outperforms or matches the sample efficiency and converged performance of full-APG methods.
- To understand how action-level simulator gradients can be transported through a flow-matching policy to its parameters, we study three gradient-propagation mechanisms: direct BPTT, adjoint matching, and action-level regression. We characterize their trade-offs in forward and backward computational cost and optimization performance.

2 PRELIMINARIES

We consider a finite-horizon deterministic MDP with state s_t, observation $o_t = h(s_t)$, action a_t, transition dynamics $s_{t+1} = f(s_t, a_t)$, and reward $r_t = r(s_t, a_t)$. For a policy parameterized by θ,

$$a_t \sim \pi_\theta(a \mid o_t), \tag{1}$$

the objective of RL is

$$J(\theta) = \mathbb{E}\left[\hat{J}\right], \qquad \hat{J} \triangleq \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t), \tag{2}$$

where $\hat{J}$ denotes the return of a sampled trajectory. When the policy, dynamics, and reward are differentiable, $\nabla_\theta \hat{J}$ can be estimated by directly backpropagating through the sampled trajectory.

Analytical and Likelihood-Ratio Policy Gradient The APG computes a pathwise gradient of the sampled return,

$$g_{\mathrm{APG}} = \nabla_\theta \hat{J} = \nabla_\theta \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t), \tag{3}$$

where gradients propagate through both the policy and differentiable environment. In contrast, the LRPG is computed as

$$g_{\mathrm{LRPG}} = \sum_{t=0}^{T-1} \hat{A}_t \nabla_\theta \log \pi_\theta(a_t \mid o_t), \tag{4}$$

where $\hat{A}_t$ is an advantage estimate. Under standard regularity conditions, both estimators target the same objective gradient,

$$\mathbb{E}[g_{\mathrm{APG}}] = \mathbb{E}[g_{\mathrm{LRPG}}] = \nabla_\theta J(\theta). \tag{5}$$

Thus, APG and LRPG are equivalent in expectation, although their sample-wise gradients and variance properties can differ substantially.

Truncated APG For a visual policy with $o_t = h(s_t)$, full APG differentiates through the observation path,

$$\frac{\mathrm{d}a_t}{\mathrm{d}\theta} = \frac{\partial a_t}{\partial \theta} + \frac{\partial a_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\frac{\mathrm{d}s_t}{\mathrm{d}\theta}, \quad \frac{\mathrm{d}s_t}{\mathrm{d}\theta} = \frac{\partial s_t}{\partial a_{t-1}}\frac{\mathrm{d}a_{t-1}}{\mathrm{d}\theta} + \frac{\partial s_t}{\partial s_{t-1}}\frac{\mathrm{d}s_{t-1}}{\mathrm{d}\theta}, \quad t \geq 1. \tag{6}$$

Backpropagating through high-dimensional rendering requires a differentiable renderer and is usually infeasible. Prior works (You et al., 2025; Zhang et al., 2025b) have shown that the visual observation can simply be detached,

$$\bar{o}_t = \mathrm{sg}(h(s_t)), \qquad a_t = \pi_\theta(\bar{o}_t, \epsilon_t), \tag{7}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator and ϵ_t the noise input of the reparameterized policy, yielding

$$\frac{\mathrm{d}a_t}{\mathrm{d}\theta} \approx \frac{\partial a_t}{\partial \theta}. \tag{8}$$

This truncates the $s_t \to o_t \to a_t$ gradient path while preserving gradients from actions through future dynamics and rewards. The resulting TAPG avoids differentiating through rendering, but is a biased approximation of the full APG because it omits the effect of policy-induced state changes on future observations and actions.

Flow-Matching Policy A flow policy generates each action through iterative denoising with a learned velocity field. We denote the flow time by $\tau \in [0, 1]$ to distinguish it from the environment timestep:

$$\frac{\mathrm{d}x_t^\tau}{\mathrm{d}\tau} = v_\theta(x_t^\tau, \tau; \bar{o}_t), \qquad x_t^0 = \epsilon_t \sim \mathcal{N}(0, I), \qquad a_t = x_t^1. \tag{9}$$

The policy is trained with the conditional flow matching (CFM) loss (Lipman et al., 2023), which regresses v_θ toward the constant velocity of a linear interpolant between a noise sample and the target action,

$$\ell_\theta(a_t, \bar{o}_t; \tau, \epsilon) = \left\| v_\theta(a_t^\tau, \tau; \bar{o}_t) - (a_t - \epsilon) \right\|^2, \qquad a_t^\tau = \tau a_t + (1 - \tau)\epsilon, \tag{10}$$

with $\epsilon \sim \mathcal{N}(0, I)$ and $\tau \sim \mathcal{U}[0, 1]$.

3 METHOD

Noticing the two estimators of policy gradients suffer from complementary errors, we propose DFPO which exploits both: the TAPG supplies a low-variance update direction, albeit potentially biased, from the differentiable simulator, and the LRPG suppliements the TAPG by correcting the bias introduced by truncation. The algorithm is outlined in Alg. 1.

In each iteration, DFPO rolls out the flow policy for H steps with detached observations (7) and runs a single backward pass through the simulator. This yields, for every executed action, the gradient of the sampled return with respect to executed actions,

$$\delta_t \triangleq \frac{\partial \hat{J}}{\partial a_t}, \tag{11}$$

which carries the derivatives of all subsequent dynamics and rewards. The same rollout provides advantage estimates $\hat{A}_t$ from the value function. Each minibatch step of the update phase then computes the TAPG g_{TAPG} from δ_t and the LRPG g_{LRPG} from $\hat{A}_t$, and applies their sum,

Algorithm 1 Differentiable Flow Policy Optimization (DFPO)

Require: flow policy v_θ; value function; differentiable simulator
transport operator $\mathcal{T}$: $\{\delta_t\} \to g_{\mathrm{TAPG}}$ (13)
FPO surrogate $\mathcal{J}_{\mathrm{LRPG}}$ (17)

for each training iteration **do**
  roll out H steps, $a_t = x_t^1$, observations detached
  simulator backward: $\delta_t \leftarrow \partial \hat{J} / \partial a_t$, clipped
  compute advantages $\hat{A}_t$
  for each minibatch update step **do**
    $g_{\mathrm{TAPG}} \leftarrow \mathcal{T}(\{\delta_t\})$ ▷ TAPG
    $g_{\mathrm{LRPG}} \leftarrow \nabla_\theta \mathcal{J}_{\mathrm{LRPG}}(\theta)$ ▷ LRPG
    $\theta \leftarrow \theta + \eta(g_{\mathrm{TAPG}} + g_{\mathrm{LRPG}})$
    update the value function
  end for
end for

$$g_{\mathrm{DFPO}} = g_{\mathrm{TAPG}} + g_{\mathrm{LRPG}}. \tag{12}$$

Two design choices remain: how to carry δ_t through the multi-step flow policy to its parameters, and how to estimate the LRPG without a tractable likelihood.

Propagating the action gradient. We write the mapping from action gradients to a parameter gradient as a transport operator $\mathcal{T}$,

$$g_{\mathrm{TAPG}} = \mathcal{T}(\{\delta_t\}), \tag{13}$$

and consider three choices of $\mathcal{T}$.

The most direct one, BPTT, keeps the complete flow integration graph and backpropagates δ_t through every flow step,

$$g_{\mathrm{BPTT}} = \sum_t \left(\frac{\partial a_t}{\partial \theta} \right)^\top \delta_t. \tag{14}$$

It gives the exact gradient of the truncated rollout, but storing and differentiating every integration step of every action is expensive with long rollouts and a visual encoder.

The second approach for action gradient backpropagation avoids differentiating through the flow trajectory by using adjoint matching (AM). Q-learning with Adjoint Matching (Li & Levine, 2026) transports the critic gradient $\nabla_a Q$ backward through flow time with the lean-adjoint construction and turns it into step-wise regression targets for the velocity field. We use the same construction with δ_t as the terminal condition in place of the critic gradient.

The third method requires neither a backward pass through the flow nor an adjoint solve. It forms a corrected action $a_t^* = a_t + \beta \delta_t$ and regresses the velocity field toward it while keeping the interpolant a_t^τ of (10) fixed:

$$\begin{aligned} \ell_\theta^*(a_t, \bar{o}_t; \tau, \epsilon) &\triangleq \left\| v_\theta(a_t^\tau, \tau; \bar{o}_t) - (a_t^* - \epsilon) \right\|^2 \\ &= \ell_\theta(a_t, \bar{o}_t; \tau, \epsilon) - 2\beta\, \delta_t^\top v_\theta(a_t^\tau, \tau; \bar{o}_t) + C, \end{aligned} \tag{15}$$

where C collects terms independent of θ. The simulator gradient thus enters only through the dot product between δ_t and the predicted velocity, which gives the channel its name.

In DFPO, the truncated APG could be propagated to policy parameters with any of the above methods, although gradient exactness and cost of compute differ.

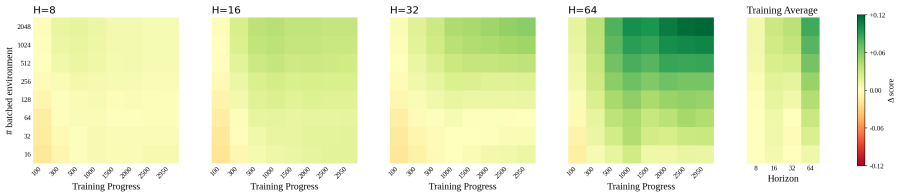


Figure 2: Policy update quality on 16-dimensional LQR. Each cell reports $\mathcal{U}(G_{\mathrm{DFPO}}) - \mathcal{U}(G_{\mathrm{TAPG}})$ (22) on normalized gradient direction for a given training checkpoint. The number of batched environments is used for batched sampling when computing gradients at a given checkpoint.

Estimating the likelihood-ratio gradient. Since the action log-likelihood in (4) is not tractable for a flow policy, we obtain the LRPG using the FPO++ surrogate (McAllister et al., 2026). FPO++ approximates the policy likelihood ratio using the change in CFM loss,

$$\hat{\rho}_{\mathrm{FPO}}(\theta) = \exp\left(\bar{\ell}_{\theta_{\mathrm{old}}} - \bar{\ell}_\theta\right), \qquad \bar{\ell}_\theta = \mathbb{E}_{\tau,\epsilon}\left[\ell_\theta(a_t, \bar{o}_t; \tau, \epsilon)\right], \tag{16}$$

where ℓ_θ is the CFM objective defined in (10). The resulting surrogate is optimized with the asymmetric trust-region objective

$$\mathcal{J}_{\mathrm{LRPG}}(\theta) = \mathbb{E}\left[\psi_{\mathrm{ASPO}}\left(\hat{\rho}_{\mathrm{FPO}}(\theta), \hat{A}_t\right)\right], \tag{17}$$

where

$$\psi_{\mathrm{ASPO}}(\rho, \hat{A}) = \begin{cases} \min\left(\rho\hat{A},\ \mathrm{clip}(\rho, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\hat{A}\right), & \hat{A} \geq 0, \\ \rho\hat{A} - \dfrac{|\hat{A}|}{2\epsilon_{\mathrm{clip}}}(\rho-1)^2, & \hat{A} < 0, \end{cases} \tag{18}$$

and ϵ_{clip} controls the trust-region size. We use the gradient of this surrogate as the LRPG signal,

$$g_{\mathrm{LRPG}} = \nabla_\theta \mathcal{J}_{\mathrm{LRPG}}(\theta), \tag{19}$$

which depends only on sampled advantages and policy quantities, and therefore does not require differentiating through the simulator.

Note that the sum in (12) is not an unbiased estimator of the true policy gradient, and the LRPG is not meant to estimate only the component removed by truncation. It rather acts as a better estimation that corrects the direction of biased TAPG. Whether this correction improves the update is an empirical question, which we first study in a setting where the quality of a gradient direction can be measured exactly.

Case study: linear–quadratic regulator. A linear–quadratic regulator (LQR) is the most controlled environment in which to compare gradient estimators, since its linear dynamics and quadratic cost, which could be stated as

$$s_{t+1} = As_t + Ba_t, \qquad r_t = -\left(s_{t+1}^\top Q s_{t+1} + a_t^\top R a_t\right), \qquad s_0 \sim \mathcal{N}(0, \Sigma_0), \tag{20}$$

make both the gradient of the return with respect to the actions and the optimal return $J^\star$ available in closed form through solution of the algebraic Riccati equation. We sample controllable linear systems and positive semi-definite cost matrices to formulate LQR problems in a 16-dimensional state space, and analytically solve for the optimal value function.

The three gradient estimators are evaluated on a series of Gaussian MLP policies with different levels of performance. The policies are obtained by taking different checkpoints from a Policy Gradient training to solve the sampled LQR problem. Since the policy parameter space is high-dimensional and sparse, cosine similarity between parameter gradients does not reliably reflect the quality of an update. We therefore measure a gradient estimate g by its effect on the return: we take a normalized step along g and record the fraction of the remaining optimality gap that it closes,

$$q(g) = \frac{J(\theta + \eta g/\|g\|) - J(\theta)}{J^\star - J(\theta)}. \tag{21}$$

For a gradient estimator G, the update quality

$$\mathcal{U}(G) = \mathbb{E}_{g\sim G}[q(g)] - \sigma_{g\sim G}[q(g)], \tag{22}$$

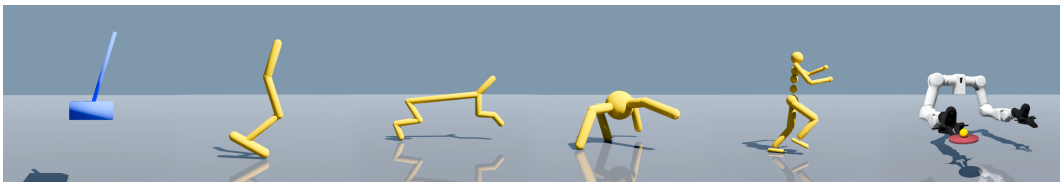


Figure 3: Evaluation tasks in differentiable simulator. From left to right: Cartpole, Hopper, HalfCheetah, Ant, Humanoid, and Pingpong Balance.

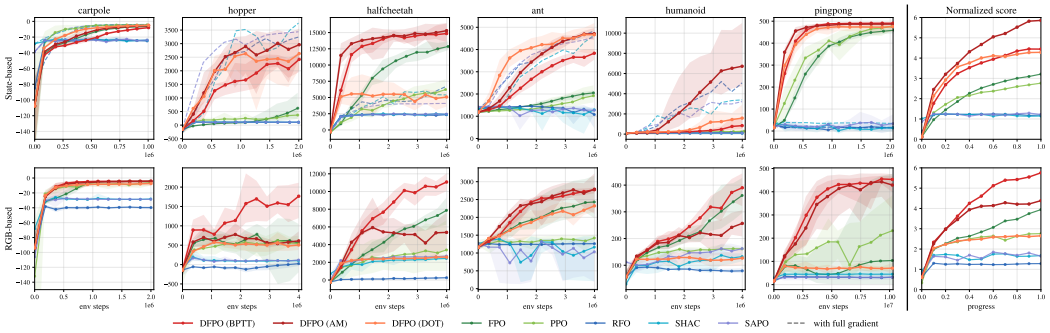


Figure 4: Average evaluated return versus environment steps on six tasks in the state-based (top) and RGB-based (bottom) settings. Solid lines use the TAPG; dashed lines ("with full gradient") use the full APG (state only). Shaded regions show min–max over 5 seeds.

with $\sigma[\cdot]$ the standard deviation, rewards directions that close the gap on average and do so consistently across sampled batches.

Fig. 2 compares the combined estimator G_{DFPO} with the truncated estimator G_{TAPG} across training checkpoints, batch sizes, and horizons. Adding the LRPG yields better update directions in most settings, and the gain grows with both the horizon and the batch size. Longer horizons make truncation more harmful, because the omitted $s_t \to o_t \to a_t$ path compounds over more steps, and this is precisely the bias that the LRPG corrects. Larger batches, reduce the variance of the LRPG, so that its correction carries more signal than noise. Beyond this regime, the combined update consistently outperforms the TAPG alone, which motivates its use in all experiments of Sec. 4.

4 Experiments

4.1 Experimental Setup

We evaluate on six tasks in the differentiable simulator Genesis (Authors, 2024), as shown in Fig. 3: five control and locomotion tasks ported from the DiffRL benchmark (Xu et al., 2022), and Pingpong Balance, a contact-rich task where a 7-DoF arm keeps a ball bouncing on a paddle. Each task is evaluated in both *state-based* and *RGB-based* settings. In the RGB setting, the actor observes a rendered image and proprioception while the critic retains privileged state. Since rendering is non-differentiable, all first-order methods must truncate the observation path; we apply the same truncation in the state-based setting for comparison, and additionally evaluate baselines with the full APG.

We evaluate three DFPO variants corresponding to the propagation channels of Sec. 3: DFPO (BPTT), DFPO (AM), and DFPO (DOT). The zeroth-order baselines are FPO++ (Yi et al., 2026), which trains the same flow policy with an LRPG alone under an asymmetric trust region and is referred to as FPO below, and PPO (Schulman et al., 2017) for Gaussian policies. The first-order baselines are SHAC (Xu et al., 2022), SAPO (Xing et al., 2025), and RFO (Zhong et al., 2026), which follow the APG without an LRPG correction. All methods use 5 seeds under a fixed per-task environment-step budget and shared checkpoint-evaluation protocol. FPO and PPO are additionally trained with $4\times$ the budget for final-performance comparison. Task details, hyperparameters, and evaluation protocols are given in Appendices A and B.

	Method	cartpole	hopper	halfcheetah	ant	humanoid	pingpong	Normalized Sum
State-based	**DFPO (BPTT)**	-7.9 ± 0.6	2420 ± 260	$\mathbf{14851 \pm 910}$	3834 ± 168	839 ± 331	$\mathbf{491 \pm 3}$	4.17 ± 0.12
	DFPO (AM)	-5.3 ± 0.2	2960 ± 413	$\mathbf{15293 \pm 1231}$	4709 ± 233	$\mathbf{6721 \pm 2205}$	487 ± 7	$\mathbf{5.57 \pm 0.37}$
	DFPO (DOT)	-5.0 ± 0.3	2633 ± 393	5077 ± 773	4665 ± 524	1581 ± 993	476 ± 13	3.90 ± 0.23
	FPO	-5.9 ± 1.0	2894 ± 194	$\mathbf{15340 \pm 414}$	3735 ± 377	1395 ± 686	459 ± 11	4.42 ± 0.16
	PPO	-4.2 ± 0.4	2801 ± 383	11000 ± 1308	3280 ± 668	4853 ± 1557	$\mathbf{492 \pm 3}$	4.62 ± 0.32
	RFO	-24.9 ± 0.4	103 ± 3	2349 ± 21	1077 ± 310	106 ± 9	15.0 ± 11.4	0.01 ± 0.08
	SHAC	-24.1 ± 1.2	107 ± 2	2385 ± 66	1290 ± 103	89.1 ± 8.3	13.8 ± 14.3	0.10 ± 0.07
	SAPO	-23.8 ± 1.4	107 ± 3	2445 ± 66	1266 ± 130	106 ± 12	33.6 ± 26.2	0.16 ± 0.09
	RFO (with full gradient)	-4.0 ± 0.1	3339 ± 75	7482 ± 4079	$\mathbf{5027 \pm 294}$	5954 ± 2416	16.7 ± 10.1	4.11 ± 0.49
	SHAC (with full gradient)	$\mathbf{-3.9 \pm 0.0}$	$\mathbf{4027 \pm 2153}$	5342 ± 2466	4993 ± 338	3464 ± 3781	33.0 ± 14.0	3.77 ± 0.82
	SAPO (with full gradient)	$\mathbf{-3.9 \pm 0.0}$	3466 ± 33	4110 ± 19	4650 ± 49	3459 ± 919	12.5 ± 10.5	3.41 ± 0.14
RGB-based	**DFPO (BPTT)**	$\mathbf{-4.3 \pm 0.1}$	$\mathbf{1765 \pm 310}$	$\mathbf{11075 \pm 751}$	$\mathbf{2775 \pm 58}$	$\mathbf{390 \pm 44}$	$\mathbf{453 \pm 27}$	$\mathbf{5.99 \pm 0.25}$
	DFPO (AM)	-4.5 ± 0.5	611 ± 135	5384 ± 1377	$\mathbf{2790 \pm 399}$	257 ± 36	429 ± 55	4.32 ± 0.32
	DFPO (DOT)	-7.4 ± 2.5	507 ± 93	2583 ± 100	2322 ± 131	127 ± 26	70.7 ± 15.3	2.39 ± 0.15
	FPO	$\mathbf{-4.3 \pm 0.1}$	553 ± 434	7869 ± 970	2431 ± 414	361 ± 57	104 ± 187	3.88 ± 0.60
	PPO	-6.2 ± 2.3	541 ± 107	3398 ± 2078	1421 ± 57	162 ± 4	230 ± 111	2.49 ± 0.34
	RFO	-39.7 ± 1.6	24.3 ± 50.8	262 ± 40	1262 ± 17	79.1 ± 5.6	32.0 ± 1.4	0.13 ± 0.06
	SHAC	-28.2 ± 0.3	101 ± 18	2437 ± 94	1165 ± 479	133 ± 10	44.9 ± 12.5	0.85 ± 0.28
	SAPO	-28.6 ± 0.2	118 ± 24	2706 ± 47	1034 ± 445	162 ± 11	30.8 ± 7.4	0.86 ± 0.26

Table 1: Final evaluated return (mean $\pm$ std over 5 seeds) at each task's training budget, with the RGB-based results reported as separate rows. (FPO and PPO are extended to $4\times$ as both algorithms use only the LRPG.) The normalized sum rescales each task column to $[0, 1]$ by the min and max mean return across all methods on that task within the same observation setting, then sums the normalized scores over the six tasks, so the maximum attainable value is 6.

To ablate the effects of the LRPG correction, we evaluate all three DFPO variants while replacing the LRPG term with the conditional flow-matching regularizers used in RFO, which have been shown to be effective for first-order RL with flow-matching policies. Apart from this replacement, all variants use the same task setup, training configuration, and algorithmic settings.

4.2 Experimental Results

Our experiments aim to answer the following four questions:

Q1: Does combining the TAPG and LRPG help under truncation? Yes. In both the state-based and RGB-based settings, DFPO substantially outperforms FPO and the first-order-only baselines (Fig. 4). Compared with FPO, which uses the same LRPG surrogate as DFPO, incorporating the TAPG term consistently improves sample efficiency across all tasks, with particularly pronounced gains in the state-based setting. Moreover, within the standard training budget, DFPO achieves final performance comparable to or better than FPO trained with a $4\times$ larger environment-step budget. Conversely, truncated RFO, SHAC, and SAPO make little progress on most tasks in both settings. These results highlight the limitation of using the biased TAPG alone: the updates from truncated policy gradients are suboptimal for effective policy optimization.

The ablation in Fig. 5 further highlights the importance of the LRPG correction. Without this term, performance degrades substantially across all three propagation channels, especially in the RGB-based setting, where BPTT nearly collapses and AM/DOT plateau much earlier. This behavior is consistent with the LQR study in Fig. 2, which shows that the biased TAPG can be systematically improved by combining it with an LRPG signal.

Q2: How close is DFPO to methods with the full APG? With state observations, all three DFPO variants are competitive with methods that use the full APG, and DFPO (AM) achieves the highest normalized aggregate return (Tab. 1). The full APG remains advantageous on some tasks such as Hopper and Ant, but its advantage is not uniformly observed across environments. On Pingpong Balance, all first-order baselines fail, with or without truncation. Because successful control depends on constantly adjusting the paddle rotation to keep the ball balanced, small errors in the backpropagated trajectory gradient can be repeatedly amplified, which is shown in Tab.7, making the policy optimization particularly prone to gradient explosion. In terms of sample efficiency, Fig. 5 shows that RFO with the full APG does not provide a consistent advantage over DFPO on any of the evaluated tasks.

On HalfCheetah, as shown in Fig. 4, FPO discovers a policy with substantially higher performance than those reached by the first-order methods. Through the shared LRPG component, DFPO (BPTT)

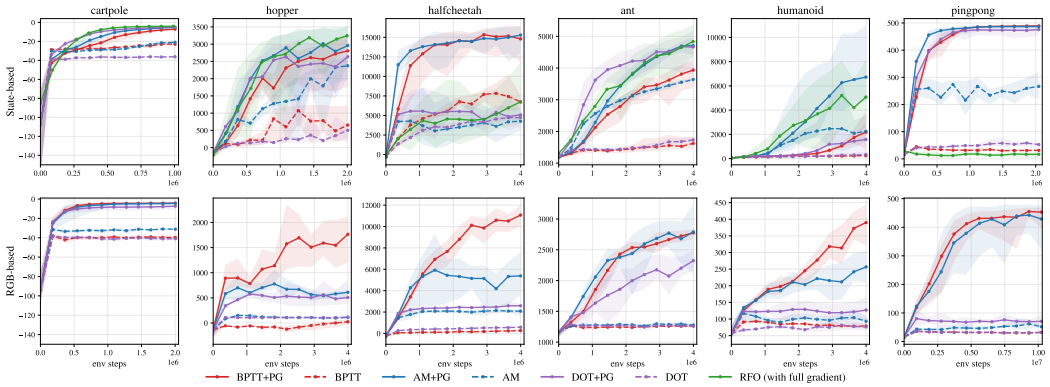


Figure 5: Ablation of the propagation channel and the LRPG correction. Solid lines ("+PG") combine each channel with the LRPG as in (12); dashed lines replace it with RFO's conditional flow-matching regularizers. The green reference is RFO with the full APG.

and DFPO (AM) also discover this behavior, whereas DFPO (DOT) converges to behaviors similar to the other first-order baselines. This suggests that, beyond correcting the truncated analytical gradient, the LRPG may also provide a complementary optimization signal that helps the policy escape poor local optima.

Q3: How do the propagation channels compare? BPTT preserves the full mapping from the action gradient to the policy parameters by differentiating through the complete flow trajectory. AM instead converts the action gradient into frozen regression targets at the predicted-velocity level of the flow policy through vector–Jacobian products through denoising steps. DOT performs a more aggressive approximation by converting the gradient into frozen regression targets directly at the action level, making it the computationally cheapest but also the least accurate propagation mechanism. As shown in Fig. 1, BPTT and AM achieve comparable overall performance among the three variants. BPTT performs better in the RGB-based setting, whereas AM performs better in the state-based setting. In contrast, DOT generally provides a lower-quality gradient signal and, overall, does not outperform the LRPG-only baseline. These results suggest that effective backward propagation through a flow policy should preserve gradient information at least at the predicted-velocity level, rather than collapsing it directly to an action-level regression target.

Because AM and DOT use intermediate regression targets, their actor-gradient norms are much smaller than those of BPTT (Table 2). They therefore use multiple optimization epochs per rollout, as in FPO and PPO. When action-level gradients explode, as in Pingpong Balance, repeatedly updating along a clipped but inaccurate direction can still destabilize learning. We thus apply δ_t clipping to both AM and DOT.

Q4: What are the gradient and computational trade-offs? We profile the three gradient channels at a shared mid-training checkpoint on all 12 tasks, measuring policy inference, policy-side backward, per-minibatch update time, and the pre-clip actor gradient norm (Tab. 2; full results in Table. 7-8). Since simulator backpropagation is shared across channels, the primary difference lies in policy-side computation.

		Peak actor grad norm	Policy inference (ms)	Update step (ms)	Peak gpu mem (G)
State	BPTT	8419	9.1 ± 2.5	62.9 ± 0.3	3.1
	AM	1.4	37.5 ± 0.6	1.3 ± 0.1	2.8
	DOT	1.9	7.2 ± 0.1	1.5 ± 0.1	2.9
RGB	BPTT	1457	26.7 ± 5.8	53.8 ± 2.6	8.0
	AM	6.1	80.0 ± 1.1	3.6 ± 0.2	8.6
	DOT	2.0	22.5 ± 0.5	3.8 ± 0.2	7.3

Table 2: Resource profile on Humanoid.

BPTT incurs the highest compute cost in its backward pass, differentiating through the full flow-integration graph. It also produces a single large gradient update per simulator backward. AM shifts this cost to inference by storing the flow trajectory for adjoint transport, but enables the full minibatch update schedule using frozen targets. DOT is inexpensive in both inference and backward. Consistently, RGB peak memory follows DOT<BPTT<AM. Overall, BPTT trades simulator gradients for a single high-magnitude update, AM amortizes each simulator backward over multiple policy updates at higher inference and memory cost, while DOT provides the lowest overhead but the shortest effective update horizon.

5 Related Work

Generative Policies for Robot Learning Continuous-control policies are often parameterized as Gaussians for tractable likelihoods and gradients (Rudin et al., 2022; Tang et al., 2024; Margolis & Agrawal, 2022; Chen et al., 2022), but their limited expressiveness can be restrictive for multimodal behaviors (Chi et al., 2023; Lipman et al., 2023). Diffusion and flow-matching policies instead generate actions through iterative transformations and have demonstrated strong performance in visuomotor control. In particular, flow-based policies have shown promising scalability to large-scale robot learning (Black et al., 2024; Intelligence et al., 2025b;a; Shukor et al., 2025). We study how such flow policies can be optimized when differentiable dynamics and rewards are available.

Reinforcement Learning with Diffusion and Flow Policies A growing body of work adapts RL to expressive generative policies. Value-based approaches include Diffusion-QL, IDQL, DIPO, QVPO, DACER, DIME, and Diffusion Actor-Critic (Wang et al., 2023; Hansen-Estruch et al., 2023; Yang et al., 2023; Ding et al., 2024; Wang et al., 2024; Celik et al., 2025; Fang et al., 2025), while Q-score matching directly relates diffusion scores to action-value gradients (Psenka et al., 2024). Policy-gradient approaches instead operate on the generative trajectory or construct surrogate likelihood ratios, including DPPO, ReinFlow, FPO, and its robot-control extension (Ren et al., 2025; Zhang et al., 2025a; McAllister et al., 2026; Yi et al., 2026). For flow policies, Flow Q-Learning, Q-Learning with Adjoint Matching, and FlowDPG propagate or distill critic-derived action gradients without full backpropagation through the generative trajectory (Park et al., 2025; Li & Levine, 2026; Shi et al., 2026), building on adjoint-based optimization of iterative generative models (Domingo-Enrich et al., 2025). In contrast, our TAPG is obtained directly from differentiable environment dynamics and rewards, and is combined with an LRPG to compensate for truncation bias.

First-Order Reinforcement Learning. First-order RL exploits differentiable environment dynamics to obtain pathwise gradients of the return, often providing a lower-variance learning signal than likelihood-ratio estimators when accurate derivatives are available. Representative methods include Analytic Policy Gradient, SHAC, SAPO, decoupled first-order model-based RL, and differentiable-simulation approaches for locomotion and visual control (Wiedemann et al., 2023; Xu et al., 2022; Xing et al., 2025; Amigo et al., 2025; Zhang et al., 2025b; You et al., 2025). More recently, RFO (Zhong et al., 2026) extends reparameterization-based RL to flow-matching policies by jointly differentiating through the flow-generation process and the system dynamics, together with regularization terms for stable policy optimization. In contrast, we study settings in which this end-to-end gradient path is unavailable or prohibitively expensive, particularly for vision-based policies. DFPO therefore operates with a truncated pathwise gradient, investigates efficient mechanisms for transporting the resulting action gradient through the multi-step flow policy, and combines this biased first-order signal with an LRPG to mitigate the information lost through truncation.

6 Conclusion

We introduced DFPO, a reinforcement learning method for flow-matching policies that combines the truncated analytical policy gradient from differentiable simulation with a likelihood-ratio policy gradient. The two signals address complementary limitations: the TAPG provides low-variance first-order information but is biased under an incomplete gradient path, while the LRPG provides an unbiased but higher-variance update. Their combination improves the policy update and enables effective optimization when end-to-end differentiation is impractical.

Across state-based and RGB-based continuous-control tasks, DFPO consistently improves over using either gradient signal alone and, in the state-based setting, achieves performance comparable to methods with access to the full analytical gradient. We further study BPTT, adjoint matching, and action-level regression for transporting simulator gradients through the iterative flow policy, revealing different trade-offs between gradient fidelity and computational cost. These results suggest that combining partial first-order information with policy-gradient surrogates provides a practical approach to sample-efficient reinforcement learning of expressive flow policies, while future work could replace exact simulator derivatives with gradients from learned world models to extend this framework beyond differentiable simulators.

7 AI DISCLOSURE

In this work, we used generative AI tools for (design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, interpret results). We have not used generative AI tools for (help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, propose or refine hypotheses), and (generate synthetic data sets, clean and reformat dataset) are not applicable to this work. Additionally, we used generative AI tools for (searching for information, edit a research paper to improve readability, identify relevant literature). We have reviewed all AI-assisted text writing, and all codes generated by AI are evaluated by the first author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Joseph Amigo, Rooholla Khorrambakht, Elliot Chane-Sane, Nicolas Mansard, and Ludovic Righetti. First Order Model-Based RL through Decoupled Backpropagation. In *Proceedings of The 9th Conference on Robot Learning*, pp. 5234–5255. PMLR, 2025.

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/genesis-world.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv:2410.24164*, 2024.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 6958–6977. PMLR, 2025. URL https://proceedings.mlr.press/v267/celik25a.html.

Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuang Jiang, Stephen Marcus McAleer, Yiran Geng, Hao Dong, Zongqing Lu, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning, 2022. URL https://arxiv.org/abs/2206.08686.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *arXiv:2303.04137*, 2023.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.16173.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control. *arXiv:2409.08861*, 2025.

Linjiajie Fang, Ruoxue Liu, Jing Zhang, Wenjia Wang, and Bingyi Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2405.20555.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv:2304.10573*, 2023. URL https://arxiv.org/abs/2304.10573.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. URL https://arxiv.org/abs/2504.16054.

Qiyang Li and Sergey Levine. Q-Learning with Adjoint Matching. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior, 2022. URL https://arxiv.org/abs/2212.03238.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow Matching Policy Gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 48104–48127. PMLR, 2025.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a Diffusion Model Policy from Rewards via Q-Score Matching. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 41163–41182. PMLR, 2024.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022. URL https://arxiv.org/abs/2109.11978.

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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*, 2017.

Kexin Shi, Junyao Shi, Poorvi Hebbar, Zhuolun Zhao, Tarun Amarnath, Yifan Su, Shikhar Bahl, and Deepak Pathak. Flowdpg: Deterministic policy gradient on flow matching policies for real-world manipulation. *arXiv preprint arXiv:2606.22303*, 2026.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20668–20696. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/suh22b.html.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes, 2024. URL https://arxiv.org/abs/2408.03539.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, and Shengbo Eben Li. Diffusion actor-critic with entropy regulator. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6174c67b136621f3f2e4a6b1d3286f6b-Abstract-Conference.html.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Nina Wiedemann, Valentin Wüest, Antonio Loquercio, Matthias Müller, Dario Floreano, and Davide Scaramuzza. Training Efficient Controllers via Analytic Policy Gradient. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1349–1356, 2023.

Eliot Xing, Vernon Luk, and Jean Oh. Stabilizing reinforcement learning in differentiable multiphysics simulation. 2025.

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated Policy Learning with Parallel Differentiable Simulation. In *International Conference on Learning Representations*, 2022.

Ming Xu, Matias Quiroz, Robert Kohn, and Scott A. Sisson. Variance reduction properties of the reparameterization trick. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2711–2720. PMLR, 2019.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv:2305.13122*, 2023. URL https://arxiv.org/abs/2305.13122.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow Policy Gradients for Robot Control. *arXiv:2602.02481*, 2026.

Haoxiang You, Yilang Liu, and Ian Abraham. Accelerating visual-policy learning through parallel differentiable simulation. In *Advances in Neural Information Processing Systems*, 2025.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning Flow Matching Policy with Online Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 106282–106319. Curran Associates, Inc., 2025a.

Yuang Zhang, Yu Hu, Yunlong Song, Danping Zou, and Weiyao Lin. Learning vision-based agile flight via differentiable physics. *Nature Machine Intelligence*, 7(6):954–966, 2025b.

Hai Zhong, Zhuoran Li, Xun Wang, and Longbo Huang. Reparameterization flow policy optimization, 2026. URL https://arxiv.org/abs/2602.03501.

APPENDIX

A ENVIRONMENT DETAILS

All tasks are simulated in Genesis with differentiable dynamics and rewards. Table 3 summarizes the task dimensions and training budgets.

	Cartpole	Hopper	HalfCheetah	Ant	Humanoid	Pingpong
State dim.	5	11	17	37	76	15
Action dim.	1	3	6	8	21	4
Episode length	240	1000	1000	1000	1000	500
Budget (state)	1M	2M	4M	4M	4M	2M
Budget (RGB)	2M	4M	4M	4M	4M	10.2M

Table 3: Task specifications and per-task environment-step budgets.

Locomotion and control tasks. Cartpole swing-up, Hopper, HalfCheetah, Ant, and Humanoid are ports of the DiffRL benchmark tasks (Xu et al., 2022) to Genesis, keeping the original geometry, mass parameters, reward shaping, and early-termination rules. Actions are joint torques (Cartpole: cart force), and episodes reset with randomized initial states during training.

Pingpong Balance. A 7-DoF arm holds a paddle rigidly attached to its gripper; a ball is launched toward the paddle with randomized position and velocity at every reset, and the reward encourages keeping the ball bouncing on the paddle near a target height. The policy outputs a 4-dimensional task-space command (paddle translation and pitch) that is mapped to joint targets with damped-least-squares inverse kinematics, and the episode terminates early when the ball is missed. Unlike the locomotion tasks, the return gradient must flow through intermittent paddle–ball contact, which makes the full APG ill-conditioned (Sec. 4.2): its pre-clip norm exceeds that of the smooth tasks by roughly four orders of magnitude at matched checkpoints.

RGB setting. In the RGB setting the actor observes a $64\times64\times3$ image from a body-mounted camera (locomotion) or a wrist-mounted camera above the paddle (Pingpong), rendered with a batched GPU rasterizer, together with proprioceptive inputs; the critic always receives the compact simulator state. Images are treated as constants in the computation graph: rendering is not differentiated, so all first-order methods operate in the truncated regime of (7). In the state-based truncated regime we reproduce the same structure by detaching the observation tensor at the environment boundary while rewards keep their gradients; for the closed-loop baselines this isolates each action as a gradient leaf, matching what these methods can access in the RGB setting.

Simulation settings. All tasks run in Genesis's differentiable rigid-body pipeline; Table 4 lists the integration settings. Locomotion tasks apply scaled joint torques directly; cartpole actuates the cart's prismatic joint; Pingpong maps 4-dimensional Cartesian commands to arm joint targets through damped-least-squares IK, tracked by a torch-side PD controller running at substep rate.

Task	Δt (s)	Substeps	Episode length	Actions	Actuation
Cartpole	$1/60$	4	240	1	prismatic force
Hopper	$1/60$	16	1000	3	joint torque
HalfCheetah	$1/60$	16	1000	6	joint torque
Ant	$1/60$	16	1000	8	joint torque
Humanoid	$1/60$	48	1000	21	joint torque
Pingpong	$1/100$	4	500	4	DLS-IK + per-substep PD

Table 4: Simulation parameters per task: control timestep, physics substeps per control step, episode length in control steps, action dimension, and actuation mode.

B Algorithm and Training Settings

DFPO. The flow policy is an MLP velocity field integrated with $K{=}32$ Euler steps; in the RGB setting a shared convolutional encoder processes the image and its features are concatenated with the proprioceptive inputs. Rollouts use short horizons in the SHAC style with a critic bootstrapping the return beyond the horizon, and the adjoint-matching channel rebuilds its regression targets once per rollout. Table 5 lists all hyperparameters.

Zeroth-order baselines. FPO shares DFPO's policy class, architecture, surrogate objective, and data collection, differing only in the absence of the TAPG term (KL-adaptive learning rate on RGB Cartpole, where the fixed schedule loses one seed); PPO uses a Gaussian policy with the same network sizes and matched batch shapes. Table 6 lists the settings.

First-order baselines. All first-order baselines share DFPO's horizon settings. SHAC follows the released recipe (Xu et al., 2022); SAPO adds entropy regularization with a learned temperature; RFO trains a flow policy of the released SHAC-flow architecture by backpropagating the short-horizon return through both the rollout and the sampling chain, regularized by three conditional flow-matching regressions. The variants with the full APG differ from the truncated ones only in keeping the observation path in the graph; Table 6 lists the settings.

DFPO parameter	State	RGB
Actor (velocity field) MLP	$(128, 64, 32)$, ELU, flow-time embed 16	
Critic MLP	$(256, 128, 64)$, ELU	
Image encoder	–	shared CNN, $64{\times}64{\times}3$
Flow integration steps K	32	
Rollout horizon H	8 (cartpole: 32)	16
Parallel environments	64–256	32
Epochs $\times$ minibatches	8×4	
Optimizer	AdamW, $\beta{=}(0.9, 0.999)$, weight decay 10^{-4}	
Learning rate	10^{-3} (KL-adaptive)	$3{\times}10^{-4}$
Discount γ / GAE λ	0.99 / 0.95	
CFM samples per action	16	
Surrogate clip (ASPO)	0.05	
EMA behavior policy decay	0.95	
Global grad-norm clip	1.0	
Action-gradient clip (δ_t, per action)	5.0	
Adjoint-matching β	0.1	
Loss coefficients (TAPG, LRPG)	1.0, 1.0	

Table 5: DFPO hyperparameters. Empirical observation normalization is used in the state setting; RGB inputs are normalized inside the encoder.

Parameter	FPO	PPO	SHAC	SAPO	RFO
Policy	flow, $(128, 64, 32)$	Gaussian, $(128, 64, 32)$	Gaussian, $(128, 64, 32)$	Gaussian, $(128, 64, 32)$	flow, $(400, 200, 100)$
Critic	$(256, 128, 64)$	$(256, 128, 64)$	$(64, 64)$	$(64, 64)$	$(400, 200, 100)$, $n{=}2$
Flow steps	32	–	–	–	4
Horizon H	32	32	8	8	8
Actor lr	10^{-3}, decay to 10%	KL-adaptive	$2{\times}10^{-3}$, linear	$2{\times}10^{-3}$, linear	$2{\times}10^{-3}$, linear
Critic lr	(shared)	(shared)	$2{\times}10^{-3}$	$5{\times}10^{-4}$	$5{\times}10^{-4}$
Adam (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.7, 0.95)$	$(0.7, 0.95)$	$(0.7, 0.95)$
Grad clip	1.0	1.0	1.0	1.0	0.5
Other	CFM $k{=}16$, clip 0.05	clip 0.2	TD(λ) target critic	learned α, $\mathcal{H}^* = -0.5\lvert\mathcal{A}\rvert$	3 CFM reg. (0.5 each)

Table 6: Baseline hyperparameters. RGB variants prepend the same convolutional encoder as DFPO and use 32 environments; horizons match Table 5.

Evaluation protocol. All reported performance uses saved checkpoints rather than training rewards: each run stores checkpoints throughout training, and at 11 evenly spaced environment-step marks the nearest checkpoint is evaluated by running one deterministic episode in each of 16 independently seeded environments. Curves report the mean and min–max band over 5 seeds; tables

report the final mark. Gradient-based methods are evaluated in the same simulator build they were trained in, since the differentiable build is not bit-identical to the standard one. For the $4\times$-budget FPO and PPO entries, training continues from the budget checkpoint with the same hyperparameters, and the table reports the best of the standard-budget and extended-budget endpoints.

C Baseline Formulations

We briefly state the objective of each baseline; shared quantities follow the notation of Sec. 2, with $\hat{A}_t$ a GAE advantage estimate and V_ψ a learned critic.

PPO (Schulman et al., 2017). A Gaussian policy trained with the clipped likelihood-ratio surrogate,

$$\mathcal{J}_{\mathrm{PPO}}(\theta) = \mathbb{E}\Big[\min\Big(\rho_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\,\hat{A}_t\Big)\Big], \qquad \rho_t(\theta) = \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid o_t)}. \tag{23}$$

It uses no simulator gradients and serves as the LRPG reference for Gaussian policies.

FPO++ (Yi et al., 2026). An extension of FPO that improves training stability through per-sample CFM surrogate ratios and an asymmetric trust region. For positive advantages it uses standard PPO clipping, while for negative advantages it replaces hard clipping with the SPO penalty,

$$\psi_{\mathrm{ASPO}}(\rho, \hat{A}) = \begin{cases} \min\Big(\rho\hat{A},\ \mathrm{clip}(\rho, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\hat{A}\Big), & \hat{A} \geq 0, \\ \rho\hat{A} - \dfrac{|\hat{A}|}{2\epsilon_{\mathrm{clip}}}(\rho-1)^2, & \hat{A} < 0, \end{cases} \tag{24}$$

which continues to pull overly large ratios back toward the trust region rather than zeroing their gradients. FPO++ uses only the LRPG and does not differentiate through the environment dynamics.

SHAC (Xu et al., 2022). A first-order actor–critic that maximizes the short-horizon differentiable return with a terminal critic,

$$\mathcal{J}_{\mathrm{SHAC}}(\theta) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) + \gamma^H V_\psi(s_H)\right], \tag{25}$$

where the expectation is differentiated pathwise through the dynamics, rewards, and (when not truncated) observations of the H-step rollout; the critic is regressed on TD(λ) targets.

SAPO. SHAC's objective augmented with maximum-entropy regularization,

$$\mathcal{J}_{\mathrm{SAPO}}(\theta) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t (r(s_t, a_t) + \alpha\,\mathcal{H}(\pi_\theta(\cdot \mid o_t))) + \gamma^H V_\psi(s_H)\right], \tag{26}$$

with the temperature α adapted toward a target entropy as in soft actor–critic.

RFO. A flow policy trained with the SHAC objective: each action is generated by K Euler steps of the velocity field, $a_t = \Phi_\theta(x_t^0; \bar{o}_t)$, and (25) is differentiated through both the rollout and the full sampling chain. Three conditional flow-matching regressions regularize the policy,

$$\mathcal{L}_{\mathrm{reg}} = \lambda\left(\ell_\theta(a^{\mathrm{roll}}) + \ell_\theta(a^{\mathrm{prev}}) + \ell_\theta(a^{\mathrm{unif}})\right), \tag{27}$$

regressing the velocity field toward the current rollout actions, the previous iteration's actions, and uniform random actions, respectively. RFO is the closest first-order baseline to DFPO: it consumes the same TAPG but propagates it exclusively by backpropagation and anchors the policy with regressions instead of a reward-carrying LRPG term.

	Component	Actor grad norm (max)	Policy inference (ms)	Update step (ms)	Peak mem (GB)
cartpole	BPTT	332	51.4 ± 11.7	109.5 ± 10.9	2.2
	AM	0.2	151.4 ± 6.8	3.1 ± 0.7	1.8
	DOT	0.3	20.3 ± 0.1	1.9 ± 0.7	1.9
hopper	BPTT	7894	7.8 ± 3.0	39.0 ± 3.0	2.2
	AM	2.9	34.9 ± 0.1	1.3 ± 0.1	2.0
	DOT	5.0	6.0 ± 0.0	1.5 ± 0.2	2.1
halfcheetah	BPTT	3006	7.4 ± 2.7	35.4 ± 19.8	2.1
	AM	4.9	34.5 ± 0.1	1.3 ± 0.1	1.9
	DOT	2.7	5.7 ± 0.0	1.4 ± 0.1	2.0
ant	BPTT	1618	8.2 ± 2.9	69.0 ± 0.4	2.7
	AM	0.9	36.3 ± 0.1	1.3 ± 0.1	2.4
	DOT	0.3	6.5 ± 0.0	1.5 ± 0.1	2.5
humanoid	BPTT	8419	9.1 ± 2.5	62.9 ± 0.3	3.1
	AM	1.4	37.5 ± 0.6	1.3 ± 0.1	2.8
	DOT	1.9	7.2 ± 0.1	1.5 ± 0.1	2.9
pingpong	BPTT	2.6×10^8	7.6 ± 2.7	91.2 ± 2.8	2.4
	AM	5.1	37.5 ± 0.7	1.3 ± 0.1	2.2
	DOT	1.3	5.8 ± 0.0	1.5 ± 0.1	2.3

Table 7: Per-component resource profile on all six tasks, state-based setting (protocol in Appendix D; columns as in Table 2).

D Resource Study Protocol

The measurements of Table 2 isolate the per-component cost of the three propagation channels from the shared cost of data collection. All numbers are collected on a single RTX 5090 with the campaign batch shapes on Hopper: 256 environments with rollout horizon 8 in the state-based setting, and 32 environments with horizon 16 and 64×64 rendering in the RGB setting.

Each measurement loads the same mid-training checkpoint (at 20% of the task budget, so gradients are informative but not vanishing), collects one differentiable rollout, and then applies 32 sequential actor-only update steps of a *single* loss component, sampling a fresh minibatch (one quarter of the rollout data) at each step. We record three quantities. *Gradient magnitude*: the pre-clip ℓ_2 norm of the actor-parameter gradient at every step, which shows how the channel's signal is distributed over the update phase — BPTT concentrates its entire signal in the single backward pass through the simulator and flow graph, AM releases a nearly constant per-step signal that decays only as its regression converges, and the LRPG replenishes its signal at every minibatch through the ratio. *Time*: the wall-clock duration of the shared rollout-plus-backward phase, the one-off target construction (nonzero only for AM, which performs K velocity-field vector–Jacobian products to transport the adjoint), and the median per-update-step time, all CUDA-synchronized. *Memory*: the peak GPU memory after the rollout/backward phase and at the end of the update phase, reported both as the framework allocator peak and as the process footprint, since the simulator tape is held outside the training framework's allocator. Each mode runs in a fresh process so that memory attribution is not contaminated across components. Table 2 in the main text reports Hopper; Tables 7 and 8 report the same measurement on all six tasks, using each task's campaign batch shape and a checkpoint at 20% of its budget. The profiles are consistent across tasks: BPTT's impulse-and-heavy-backward pattern, AM's sustained per-step signal with cheap targets, and DOT's early trust-gate collapse hold on every task, with the contact-rich Pingpong showing the same structure at larger magnitudes.

	Component	Actor grad norm (max)	Policy inference (ms)	Update step (ms)	Peak mem (GB)
cartpole	BPTT	107	24.8 ± 5.0	38.3 ± 1.3	7.4
	AM	5.0	80.7 ± 0.2	3.6 ± 0.2	8.1
	DOT	0.1	20.5 ± 0.1	3.9 ± 0.2	6.8
hopper	BPTT	575	25.6 ± 4.6	49.1 ± 19.7	7.5
	AM	3.8	79.9 ± 2.7	3.6 ± 0.2	8.2
	DOT	2.1	21.8 ± 0.2	4.0 ± 0.2	6.9
halfcheetah	BPTT	782	24.9 ± 4.7	46.0 ± 19.9	7.5
	AM	8.4	82.7 ± 0.1	3.6 ± 0.2	8.2
	DOT	1.6	21.1 ± 0.0	3.9 ± 0.2	6.8
ant	BPTT	317	25.1 ± 4.4	48.0 ± 8.7	7.6
	AM	4.0	79.3 ± 1.8	3.7 ± 0.3	8.3
	DOT	0.6	21.7 ± 0.1	4.0 ± 0.4	6.9
humanoid	BPTT	1457	26.7 ± 5.8	53.8 ± 2.6	8.0
	AM	6.1	80.0 ± 1.1	3.6 ± 0.2	8.6
	DOT	2.0	22.5 ± 0.5	3.8 ± 0.2	7.3
pingpong	BPTT	1.0×10^{12}	28.2 ± 6.5	246.5 ± 5.3	12.6
	AM	3.4	82.6 ± 1.6	3.6 ± 0.2	8.8
	DOT	2.0	23.0 ± 0.5	4.2 ± 0.2	7.4

Table 8: Per-component resource profile on all six tasks, rgb-based setting (protocol in Appendix D; columns as in Table 2).