ARE: Adaptive TD-λ Return Estimation for Learning Control in Differentiable Simulation

***Abstract*—Differentiable simulators have gained traction in robotics because they provide the analytic pathwise gradient of the agent's reward, promising better sample efficiency for model-based learning. However, in continuous control settings, the gradient grows exponentially with the decision horizon, making the pathwise gradient susceptible to the exploding/vanishing variance issue. Recent works addressed these fundamental challenges by introducing first-order actor-critic algorithms that optimize over very short horizons. In this work, we propose to further improve these algorithms by introducing the TD-λ return into the actor network's optimization as a *generalized return estimator*, thereby reducing the actor gradient estimator's variance and enabling consistent, robust learning. In addition, we propose to adaptively adjust λ using a value-fitting objective to avoid expensive manual tuning and further enhance the learning stability. Our algorithms demonstrate improvements over challenging locomotion tasks, with an average improvement of roughly 50% over the Ant environment and almost 100% with the simulated Unitree Go2 quadruped environment. Moreover, our design allows exploiting the gradient information over a much longer learning horizon, enabling more effective long-term credit assignment for first-order model-based reinforcement learning methods.**

***Index Terms*—differentiable simulation, pathwise gradient, first-order model-based reinforcement learning**

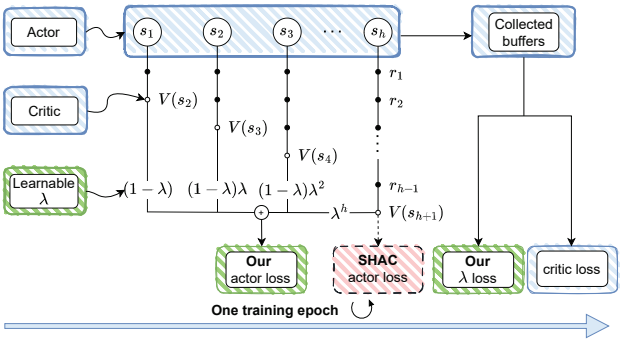


Fig. 1: Visual summary of the workflow of one training epoch following our proposed design with adaptive λ on the basis of SHAC. Instead of the h-step return, we consider the TD-λ return in the definition of actor loss. Moreover, we propose treating λ as a learnable parameter that is updated online immediately after an actor update. In the figure, blue, green, and red correspond to the existing, new, and replaced components in SHAC's structure.

I. Introduction

In recent years, learning-based methods for robot control have progressed at an astonishing pace. Thanks to the development of deep reinforcement learning techniques and high-throughput physics simulators that enable massive data generation for learning, a wide range of challenging tasks across diverse robotic platforms are being addressed, including vision-based drone navigation [1]–[4], legged locomotion [5]–[7] and contact-rich manipulation [8], [9]. Most of these approaches followed a zeroth-order policy optimization paradigm, using the score function (SF) gradient (or the REINFORCE gradient) to estimate policy updates directly from sampled trajectories. While this framework is broadly applicable and does not require differentiable system dynamics, it tends to suffer from unstable variance and limited sample efficiency. More recently, GPU-based end-to-end differentiable simulators (e.g., Warp [10], Brax [11], TinyDiffSim [12], and DFlex [13]) were proposed to enable large-scale first-order policy optimization by providing gradients that "flow" through the agent's simulated trajectories. First-order model-based reinforcement learning methods leverage these analytic gradients to achieve substantially lower-variance updates [14] and are therefore theoretically expected to achieve improved sample efficiency. Several works have explored the potential of such methods, particularly in real time deployment, e.g. for quadruped locomotion [15] and drone hovering [16].

The gradient obtained from differentiable simulators - the pathwise (PW) gradient [14], [17], also known as the reparameterization gradient [18], [19] or the first-order (FO) policy gradient [20], can be directly incorporated into policy optimization algorithms. In [21], policy optimization is decoupled into two stages: trajectory optimization in the action space using the PW gradient of the value function, followed by supervised learning in the policy space. Meanwhile, it is demonstrated in [11] that by replacing the high-variance SF gradient in the vanilla policy gradient (REINFORCE) algorithm with the PW gradient, one yield analytic policy gradient (APG) which leverages the differentiable transition dynamics for sample efficiency. Unfortunately, the potential for zero-bias and low-variance of PW gradient only hold under certain smoothness/Lipschitz conditions [14], which often fail in continuous control tasks involving contacts. In addition, the long-horizon decision-making nature of such tasks causes the variance of PW gradient to grow exponentially. To counter these effects, an on-policy first-order actor-critic (FO-AC) algorithm, namely short-horizon actor-critic (SHAC), that combines the analytic PW gradient over a truncated fixed-length learning horizon and a h-step bootstrapped return that smoothens the actor's loss landscape was proposed in [13] to learn legged locomotion policies over various environments. This work was then extended into an adaptive-horizon version

based on the norm of the simulation dynamics' Jacobian called adaptive-horizon actor-critic [22]. More recently, the challenge of stabilizing learning in high-dimensional action spaces (e.g. soft-bodies and deformable-bodies) was addressed with soft-analytic policy optimization (SAPO) [23] - a variant of SHAC which incorporates entropy regularization into the actor optimization.

In traditional reinforcement learning settings, extensive research has focused on reducing the variance of the SF gradient estimator. Among these techniques, the generalized advantage estimation (GAE) has become a cornerstone of modern policy-gradient algorithms for improving on-policy training efficiency and stability. Introduced in [24], this design provides a principled way to balance bias and variance in advantage estimation by combining the multi-step temporal-difference (TD) residuals through an exponentially decaying weighting factor λ. As reported in [24], the GAE significantly stabilized the training of trust-region policy optimization (TRPO) across classical continuous-control benchmarks. Nevertheless, its performance varied over combinations of the discount factor γ and the trace-decay parameter λ. Thus, the idea of adaptive λ was examined in [25], which proposed to automatically tune it via the indicators of the bias and variance of the return estimator. The main limitation of this work was that the adaptive λ depends on hand-designed heuristics (entropy, regression accuracy, and KL thresholds) whose reliability may vary across tasks.

Inspired by these approaches, we address the high-variance problem of PW gradient in FO-AC algorithms for continuous control by integrating TD-λ return into the actor loss as a *generalized return estimator*. In addition, we show that the bias-variance control effect can be further improved by adopting an adaptive module that automatically adjusts λ by minimizing the value prediction error. The contributions of this work are:

- A variance-reduction method that can be adopted for any FO-AC algorithms using TD-λ estimation of return, which encourages more robust learning and enables learning over long horizons with pathwise gradient.
- An adaptive method for tuning the trace-decay parameter λ to adjust the bias-variance trade-off online and avoid expensive hyperparameter tuning.

II. Preliminaries

A. Reinforcement learning

Our problem is modeled as a discrete-time, continuous-space RL problem characterized by a deterministic Markov decision process $(S, A, f, r, \pi_\theta, \rho)$, where:

- $S \subset \mathbb{R}^n$ is the state space,
- $A \subset \mathbb{R}^m$ is the action space,
- $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n; (s_t, a_t) \mapsto s_{t+1}$ denotes the deterministic transition function,
- $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}; (s, a) \mapsto r(s, a)$ is the reward function, whose realization is $r_t := r(s_t, a_t)$,
- $\pi_\theta : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m; (s, a) \mapsto \pi_\theta(s, a)$ is a policy parameterized by a neural network with parameter vector $\theta \in \mathbb{R}^d$,
- ρ is the initial state distribution, such that $s \sim \rho(\cdot)$ denotes a sample from the initial state distribution.

B. Pathwise gradient

We consider an infinite-horizon setting where our control objective is to find a policy parametrization θ that maximizes the expected state-value, respectively action-value function starting from an initial state. Our optimization objective is to maximize:

$$J(\theta) = \mathop{\mathbb{E}}_{s_0 \sim \rho(s_0)} \Big[V^{\pi_\theta}(s) \Big] = \mathop{\mathbb{E}}_{\substack{s_0 \sim \rho(s_0) \\ a \sim \pi_\theta(s,\cdot)}} \Big[Q^{\pi_\theta}(s, a) \Big]. \quad (1)$$

Following previous works [20], [22], we impose the following regularity assumptions to ensure the existence and stability of PW gradient.

Assumption 1: The dynamics f and the reward function r are Lipschitz-smooth in both s and a, while the policy π_θ and the value function $V^{\pi_\theta}(s)$ are Lipschitz-smooth in θ. Moreover, their gradients are uniformly bounded:

- $\|\nabla f(s, a)\| \leq B_f$ and $\|\nabla r(s, a)\| \leq B_r \ \forall\, s \in \mathbb{R}^n, a \in \mathbb{R}^m$,
- $\|\nabla_\theta \pi(a, s)\| \leq B_\pi$ and $\|\nabla_\theta V(s_t)\| \leq B_V \ \forall\ \theta \in \mathbb{R}^d$,

where ∇ and $\|\cdot\|$ stand for the partial derivative and the Euclidean norm operators, respectively.

Then, our PW gradient of the objective admits the representation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0, \pi_\theta} \left[\nabla_\theta Q^{\pi_\theta}(s, a) \right] \quad (2)$$

$$= \mathbb{E}_{s_0, \pi_\theta} \left[\sum_{l=0}^{k-1} \gamma^l \nabla_\theta r_{t+l} + \gamma^k \nabla_\theta V^{\pi_\theta}(s_{t+k}) \right], \quad (3)$$

where the second equality follows from expressing the action-value function Q^{π_θ} via a truncated k-step return with a terminal bootstrap $G_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l r_{t+l} + \gamma^k V^{\pi_\theta}(s_{t+k})$. In practice, the true state-value function V^{π_θ} is unknown and is approximated by a parametric critic V_ϕ with parameters ϕ, yielding the following estimator of the PW gradient:

$$\hat{\nabla}_\theta^{[1]} J(\theta) = \sum_{l=0}^{k-1} \gamma^l \nabla_\theta r_{t+l} + \gamma^k \nabla_\theta V_\phi(s_{t+k}). \quad (4)$$

We argue that the bias–variance behavior of this gradient estimator as a function of the truncation horizon k mirrors that of its anti-derivative, namely the k-step return itself. Formally, the bias B_1 of $\hat{\nabla}_\theta^{[1]} J(\theta)$ is given by:

$$B_1 := \mathbb{E}_{s_0, \pi_\theta} \left[\hat{\nabla}_\theta^{[1]} J(\theta) \right] - \nabla_\theta J(\theta) \quad (5)$$

$$= \gamma^k \mathbb{E}_{s_0, \pi_\theta} \left[\nabla_\theta V_\phi(s_{t+k}) - \nabla_\theta V^{\pi_\theta}(s_{t+k}) \right], \quad (6)$$

which decays geometrically with k under Assumption 1. Meanwhile, the exponential growth of variance of $\hat{\nabla}_\theta^{[1]} J(\theta)$ with increasing truncation horizon, which is indirectly driven by those of the reward terms via the compositional state

transitions and the chain rule, is well studied; we refer the reader to [18], [20] for detailed analyses.

Such a close correspondence between the bias–variance properties of $\hat{\nabla}_\theta^{[1]} J(\theta)$ and those of $G_t^{(k)}$ naturally motivates the consideration of TD-λ-style/GAE-style constructions. We recall that the TD-λ return is defined as the exponentially weighted combination:

$$G_t^\lambda := (1-\lambda)\sum_{k=1}^{\infty} \lambda^{k-1} G_t^{(k)}. \tag{7}$$

By combining multi-step return estimates, the TD-λ return leverages both short- and long-horizon information while gradually diminishing the contribution of distant rewards. This smooth weighting mechanism enables fine control over the bias–variance trade-off: increasing λ incorporates more Monte Carlo information at the cost of bias, whereas decreasing λ reduces variance. The resulting estimator is biased but significantly less noisy, which greatly improves the stability of policy gradient updates.

C. Short-Horizon Actor-Critic

Our proposed algorithm is built upon the basis of SHAC [13] where one considers an FO-AC algorithm with a fixed-length truncated learning horizon. For this reason, we will consider a finite-horizon Markov decision process for the rest of this paper. In SHAC, instead of learning over the full task horizon H, one considers N mini-trajectories $\{\tau_j\}$ of length $h \ll H$ and the task horizon H serves as part of the termination conditions. Each mini-trajectory begins from the final state of the previous episode and is reinitialized to cut the gradient flow across mini-trajectories, thereby avoiding unstable gradients. The objective function to be maximized is exactly the expectation of the h-step bootstrapped return $G_t^{(h)}$ over the mini-trajectories:

$$J(\theta) = \mathop{\mathbb{E}}_{\substack{s_0\sim\rho(s_0)\\ a_i\sim\pi_\theta(s_i,\cdot)}} \left[G_t^{(h)}\right]. \tag{8}$$

Correspondingly, the actor loss is defined by the negative sample mean of $G_t^{(h)}$ over the mini-trajectories:

$$\mathcal{L}_\theta := -\frac{1}{Nh}\sum_{i=0}^{N-1} G_t^{(h),(i)}, \tag{9}$$

where i denotes the index of the i-th sampled mini-trajectory.

After updating the actor network, the critic network ϕ is optimized using a standard MSE loss to match the target TD-λ estimates. We recall the formulation of the truncated TD-λ return over a finite horizon h:

$$G_t^\lambda := \left(\sum_{k=1}^{h-t-1} \left[(1-\lambda)\lambda^{k-1}\right] G_t^{(k)}\right) + \lambda^{h-t-1} G_t^{(h-t)}. \tag{10}$$

Then, the corresponding critic loss can be defined as:

$$\mathcal{L}_\phi^\lambda := \frac{1}{Nh}\sum_{s\in\{\tau_j\}} \left\| V_\phi(s) - \tilde{V}^\lambda(s)\right\|^2, \tag{11}$$

where for $t \geq 0$, $\tilde{V}^\lambda(s_t) := G_t^\lambda$ represents the target value.

III. Proposed method

A. Optimizing the actor with TD-λ return

Our first proposal is to replace the h-step return $G_t^{(h)}$ in the actor loss $\mathcal{L}_\theta$ with the TD-λ return G_t^λ. Fig. 1 captures the main idea of our approach. Following this, the actor network is optimized using the following loss:

$$\mathcal{L}_\theta^\lambda := -\frac{1}{Nh}\sum_{i=0}^{N-1} G_t^{\lambda,(i)}, \tag{12}$$

whose gradient is thus:

$$\nabla_\theta \mathcal{L}_\theta^\lambda := -\frac{1}{Nh}\sum_{i=0}^{N-1} \nabla_\theta G_t^{\lambda,(i)}. \tag{13}$$

By construction, $\nabla_\theta G_t^\lambda$ naturally mitigates the exploding gradient variance issue of backpropagation through time by exponentially discounting contributions from the gradient of distant rewards while assigning greater weight to more recent outcomes. Consequently, it exhibits a bias–variance trade-off which can be employed to stabilize policy optimization in FO-AC frameworks. To ensure the consistency between the learning objectives of the actor and the critic, the same value of λ is used for both the actor and critic TD-λ returns. It is also worth noting that the computational overhead introduced by computing the TD-λ return is negligible compared to the total runtime thanks to the following recursive relationship:

$$\begin{aligned} G_t^\lambda = r_t &+ \gamma D V_\phi(s_{t+1}) \\ &+ \gamma(1-D)\Big[(1-\lambda)\, V_\phi(s_{t+1}) + \lambda\, G_{t+1}^\lambda\Big], \end{aligned} \tag{14}$$

where D stands for the binary termination mask.

B. Online adaptation of λ via a value fitting objective

In practice, there is no general rule to choose λ, and the performance of the TD-λ estimator can be highly sensitive to this value. Manually tuning λ is computationally expensive because it requires repeating the entire training process multiple times. To address this limitation, we propose treating λ as a learnable parameter and integrating its optimization into the training loop. Our next task, then, is to find an optimization criterion that yields the correct value of λ. We propose to use the MSE loss analogous to the critic loss:

$$\mathcal{L}_\lambda := \frac{1}{Nh}\sum_{s\in\{\tau_j\}} \left\| V_\phi(s) - \tilde{V}^\lambda(s)\right\|^2. \tag{15}$$

This formulation encourages λ to minimize the discrepancy between the predicted and target state values, thereby selecting a trace-decay coefficient that best matches the true value function under the current policy dynamics. During optimization, the actor and critic are treated as stop-gradient operators, allowing the gradient to flow solely through λ.

Our rationale for annealing λ is as follows. Early in training, the critic V_ϕ is inaccurate, leading to highly biased predictions. To mitigate this, λ is initialized at a relatively high value (e.g., 0.95), placing more trust in the Monte Carlo returns, which,

although high in variance, provide unbiased estimates based on actual rewards. As training progresses and the critic becomes more accurate, its predictions become a low-variance, low-bias signal. In this later stage, relying more on the critic's stability accelerates convergence, which can be achieved by gradually reducing λ. Empirically, we find that λ should not converge faster than the critic; rather, it must co-adapt with the critic to maintain training stability. As shown in our results, updating λ too frequently leads to overfitting of the value function V_ϕ, which in turn destabilizes policy learning and degrades overall performance.

C. Preliminary analysis of TD-λ gradient variance

In fact, as shall be discussed in Section V-D, the variance reduction effect of the adaptive λ scheme becomes particularly pronounced in long-horizon learning settings. We now present a preliminary analysis that establishes an upper bound on λ that ensures the gradient variance remains bounded and thus training stability as the horizon increases. Our main proposition is as follows.

Proposition 1: Assume Assumption 1. For any parameterized policy π_θ, the variance of the pathwise gradient of the TD-λ return objective converges as the horizon $h \to \infty$ provided that $\lambda < \frac{1}{\gamma B_f}$.

Proof: By the chain rule and triangle inequality, the gradient norm of the h-step return $G_t^{(h)}$ is bounded by:

$$\left\| \frac{\partial G_t^{(h)}}{\partial \theta} \right\| \leq \sum_{l=0}^{h-1} \gamma^l \left\| \frac{\partial r_{t+l}}{\partial \theta} \right\| + \gamma^h \left\| \frac{\partial V^{\pi_\theta}(s_{t+h})}{\partial \theta} \right\|. \quad (16)$$

Given a timestep t, we first examine the sensitivity of the reward of l steps ahead r_{t+l} with respect to the policy parameters θ. The chain rule and triangle inequality yield:

$$\left\| \frac{\partial r_{t+l}}{\partial \theta} \right\| = \left\| \frac{\partial r_{t+l}}{\partial s_{t+l}} \frac{\partial s_{t+l}}{\partial \theta} + \frac{\partial r_{t+l}}{\partial a_{t+l}} \frac{\partial a_{t+l}}{\partial \theta} \right\| \quad (17)$$

$$\leq B_r \left\| \frac{\partial s_{t+l}}{\partial \theta} \right\| + B_r B_\pi. \quad (18)$$

The chain rule applies once again on $\partial s_{t+l}/\partial\theta$, resulting in:

$$\frac{\partial s_{t+l}}{\partial \theta} = \sum_{j=0}^{l-1} \left(\prod_{i=j+1}^{l-1} \frac{\partial s_{t+i+1}}{\partial s_{t+i}} \right) \frac{\partial s_{t+j+1}}{\partial a_{t+j}} \frac{\partial a_{t+j}}{\partial \theta}. \quad (19)$$

Using Assumption 1 and triangle inequality, we obtain:

$$\left\| \frac{\partial s_{t+l}}{\partial \theta} \right\| \leq \sum_{j=0}^{l-1} B_f^{l-j-1} B_\pi = B_\pi \frac{B_f^l - 1}{B_f - 1}. \quad (20)$$

Moreover, by Assumption 1, the value function has bounded gradient, so

$$\left\| \frac{\partial G_t^{(h)}}{\partial \theta} \right\| \in \mathcal{O}\left((\gamma B_f)^h \right). \quad (21)$$

The infinite TD-λ gradient is a weighted sum of these h-step return gradients. By Minkowski's inequality:

$$\sqrt{\mathrm{Var}\left(\frac{\partial G_t^\lambda}{\partial \theta} \right)} \leq \sum_{h=1}^{\infty} (1-\lambda)\lambda^{h-1} \sqrt{\mathbb{E}\left[\left\| \frac{\partial G_t^{(h)}}{\partial \theta} \right\|^2 \right]}. \quad (22)$$

Algorithm 1 SHAC with TD-λ actor loss

```
for epoch in 1, ... m do
  Sample mini-trajectories {τ_j} of length h ≪ H
  Update the actor network θ using L_θ^λ
  Collect dataset D for critic and λ training.
  if λ is learnable then
    for iter_λ in 1, ..., m_λ do
      for b_λ in D do
        Update λ using L_λ
      end for
    end for
  end if
  for iter_critic in 1, ..., m_critic do
    for b_critic in D do
      Update the critic network φ using L_φ^λ
    end for
  end for
end for
```

Substituting the growth rate from (21), we obtain:

$$\sqrt{\mathrm{Var}\left(\frac{\partial G_t^\lambda}{\partial \theta} \right)} \in \mathcal{O}\left(\sum_{h=1}^{\infty} \lambda^h (\gamma B_f)^h \right) = \mathcal{O}\left(\sum_{h=1}^{\infty} (\lambda \gamma B_f)^h \right). \quad (23)$$

This geometric series converges if and only if the ratio satisfies $\lambda \gamma B_f < 1$, or equivalently $\lambda < \frac{1}{\gamma B_f}$.

IV. Experimental settings

We conduct our experiments on a series of tasks in the Rewarped simulator [23], whose backend is written in NVIDIA Warp [10]. In particular, our experiments involve the Ant and SoftJumper environments from Rewarped, as well as a Unitree Go2 environment we created. Finally, we use the Mineral package, which was provided with Rewarped in [23], to implement our RL algorithms. All of our experiments were run on an NVIDIA A6000 GPU.

As baselines, we select two state-of-the-art FO-AC algorithms designed exclusively for differentiable simulators: SHAC [13] and SAPO [23]. As zeroth-order baselines like PPO and SAC have been shown to be inferior to SHAC and SAPO, we decided to skip such comparisons. For each baseline, we evaluate our proposed variants: TDFixed (fixed λ) and TDAdpt (adaptive λ). A pseudo-algorithm for our proposed algorithms is given in Alg. 1. By default, the collected dataset $\mathcal{D}$ is split into $b_{\text{critic}} = 4$ batches, and $m_{\text{critic}} = 16$ critic update iterations is performed. Meanwhile, we perform one ($m_\lambda = 1$) full-batch ($b_\lambda = 1$) update for λ using the same $\mathcal{D}$. For each experiment, we collect the results over 10 seeds. After training, we evaluate the algorithms on an additional 10 seeds, each with twice as many environments as during training.

For common network hyperparameters and task-related settings (e.g., the observation and action spaces, the reward function), we follow the standard definitions in [13], [22], [23]; readers are referred to these works. In particular, we set the number of parallel environments $N = 64$, the number of

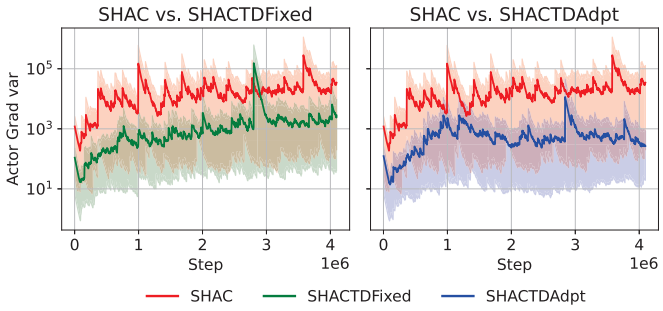


Fig. 2: Comparison of the variance of the actor network's gradient. We take the exponential moving average of the results and plot the mean (solid lines) and the maximum/minimum (shaded regions). Our methods demonstrate more well-behaved gradients throughout the training.

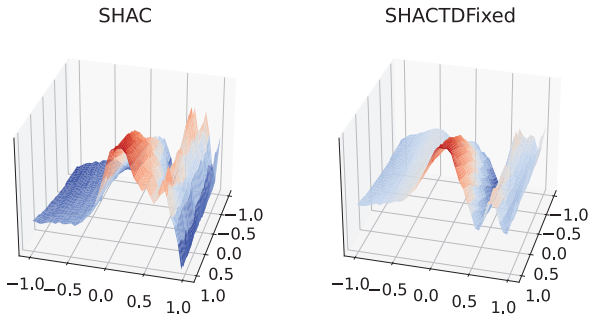


Fig. 3: Comparison between the loss landscapes computed using the h-step return versus the TD-λ return. It is visible that the TD-λ return yields a smoother loss landscape, without altering the optimal solution.

training epochs $m = 2000$, the maximum agent steps of 10^7, the task horizon $H = 1000$, the short learning horizon $h = 32$, and the discount factor $\gamma = 0.99$. Regarding our algorithm-related hyperparameters, we use a common value of the trace-decay parameter for both networks to maintain consistency between the actor and the critic objectives. For TDFixed, $\lambda = 0.95$. For TDAdpt, λ is initialized at 0.95 and is updated online immediately after the actor update using AdamW with a learning rate of 5.10^{-4} and $(\beta_1, \beta_2) = (0.7, 0.95)$. For the baselines, we set $\lambda = 0.95$. We also apply a sigmoid activation to λ to ensure its value is in $[0, 1]$.

V. Empirical results

A. Variance reduction and smoothing effect of TD-λ return

We first examine the variance reduction effect discussed previously (see Fig. 2). From the figure, it is evident that on average the TD-λ return reduces the variance of the actor network gradient. For SHAC, the mean of the gradient variance lies much closer to its maximum, indicating that SHAC receives noisy actor gradient signals. Compared to the high-variance h-step return, whose variance increases with h, the TD-λ return (with $\lambda = 0.95$) provides a cleaner, more stable, and lower-variance signal for evaluating the effect of actions on returns. Moreover, the adaptive adjustment of λ further lowers the variance of the gradient estimate. On average, while the gradient variance of SHAC remains close to 10^4, those of SHACTDFixed and SHACTDAdpt remain consistently between 10^1 and 10^3.

In Fig. 3, we empirically demonstrate the loss landscape computed by the TD-λ loss and the h-step return loss, obtained by taking a trained policy and perturbing two parameters over a 40×40 grid spanning $[-1, 1]$. Beyond reducing the empirical variance of the actor gradient, the TD-λ return also reshapes the loss landscape in a more favourable way for FO policy optimization. Specifically, Fig. 3 shows that the h-step return produces a rugged landscape with sharp curvature changes, whereas TD-λ yields a noticeably smoother surface without altering the optimal region. This smoothing effect is crucial for FO-AC algorithms because the analytic simulation gradient is highly sensitive to small perturbations in states and actions.

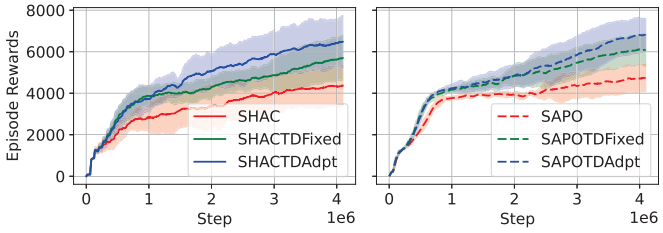
B. Learning results over the Ant environment

We now present the training results of our algorithms (see Fig. 4a). For SHAC, replacing the h-step return with TD-λ directly improves the reliability of the gradient estimator. This is reflected in the faster convergence and higher final performance for SHACTDFixed. The learnable variant of λ further improves performance by co-evolving with the critic, yielding the highest final returns among all SHAC variants. The behavior of learnable λ is demonstrated in the left subfigure of Fig. 4b, where it gradually decreases over time, consistent with our expectations. Overall, SHACTDAdpt demonstrates greater robustness than the vanilla SHAC, effectively enhancing both performance and consistency. We observe improvements of approximately 25% and 50% over the average performance of the baseline SHAC (Fig. 5), respectively.

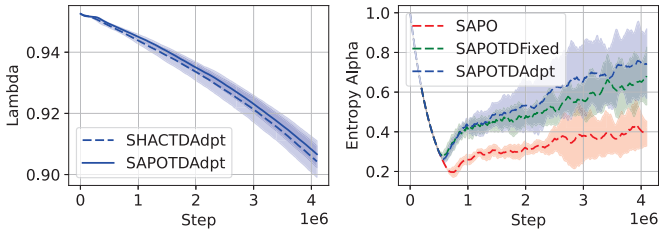
For the SAPO variants (Fig. 4a), we observe a trend similar to that of the SHAC variants. As shown in the right subfigure of Fig. 4b, the learnable temperature parameter α in SAPO collapses early during training, causing the policy to become prematurely deterministic. While α slowly recovers, the agent continues to follow the locally optimal policy, leading to a stall in performance. In contrast, the TD-λ return provides a more meaningful and stable gradient signal, allowing the actors trained with SAPOTDFixed and SAPOTDAdpt to be updated more smoothly toward a stable policy. This leads to a sharper drop in entropy, which in turn causes α to recover sooner once entropy falls below the target level. Since the agent is already near a stable policy, an earlier reintroduction of exploration triggers earlier policy improvements, which are then continued throughout training. Consequently, our variants SAPOTDFixed and SAPOTDAdpt outperform the baseline by roughly 40% and 50%, respectively (Fig. 5).

C. Ablation study on the number of updates for λ

Since the adaptive λ module behaves similarly to a *secondary* value estimation component, one natural question is



(a) Learning curves.



(b) Behavior of the learnable hyperparameters.

Fig. 4: Training metrics of the Ant environment. The bold lines and the shaded part represent the means and standard deviation, respectively. Our TDFixed and TDAdpt algorithms outperform the baseline (SHAC and SAPO) by a good margin.

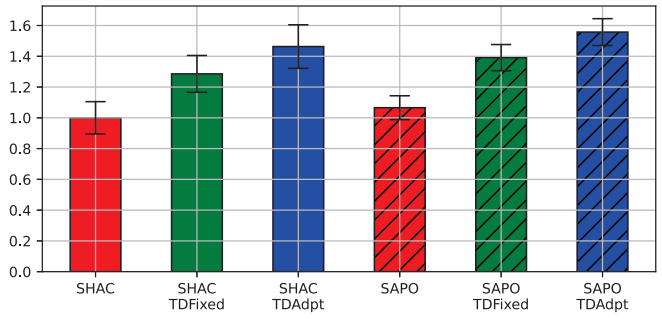


Fig. 5: Evaluation results of each algorithm. All metrics are normalized by the mean of SHAC episodic rewards. Our TDFixed and TDAdpt variants of SHAC, respectively, demonstrate significant improvements of approximately 25% and 50%, whereas these are 40% and 50% for SAPO.

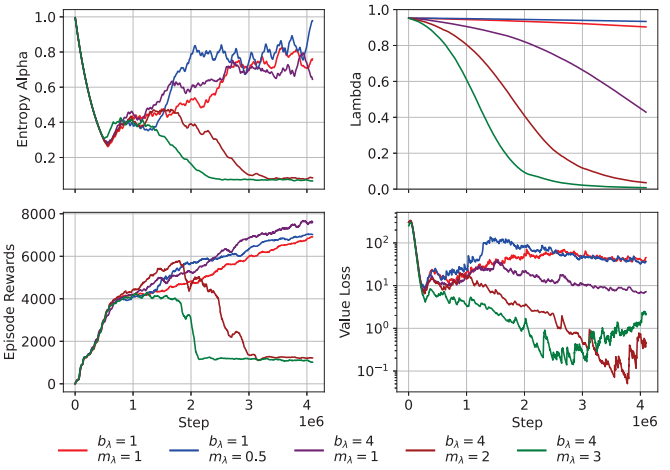


Fig. 6: Ablation study results on the update frequency of λ. The results were obtained using the SAPOTDAdpt algorithm and averaged over 10 seeds. The parameter m_λ indicates the number of iteration for the update of λ per 1 actor update, with $m_\lambda = 0.5$ indicating making 1 λ update per 2 actor update. The $(b_\lambda, m_\lambda) = (1, 1)$ setting is our baseline. Overall, updating λ too frequently overfits the critic, harming policy learning.

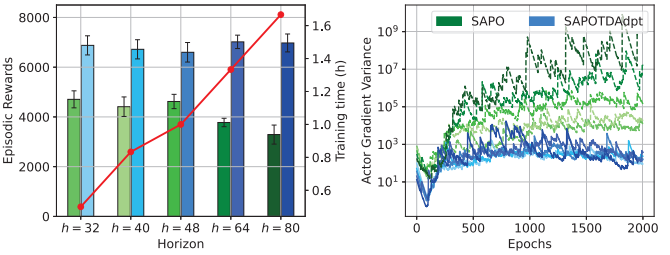
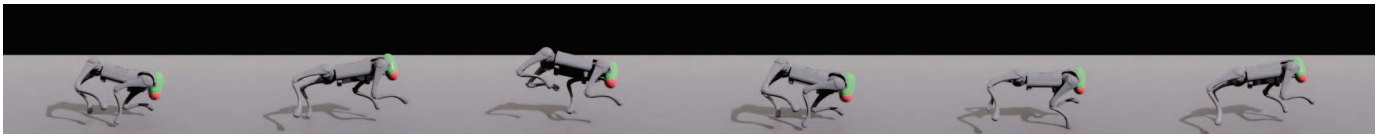


Fig. 7: Ablation study results on the short horizon length h. Right: Episodic rewards - Evaluation and Total training time. Left: Actor gradient variance - Training. The color green and blue represent SAPO and SAPOTDAdpt, respectively. The results are averaged over 10 seeds. Our SAPOTDAdpt shows the ability to mitigate the unstable variance issue of the PW gradient, fostering learning over a long learning horizon.

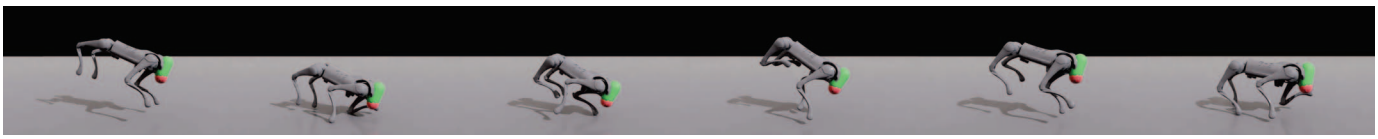
whether we should optimize them analogously. Specifically, we would like to determine an appropriate update frequency for λ per each actor update. Recall that, by default, we make only one full-batch gradient update for λ using all the collected data. In Fig. 6, we present our ablation study results of the SAPO-based algorithms with the following settings: $(b_\lambda, m_\lambda) = \{(1, 1); (1, 0.5); (4, 1); (4, 2); (4, 3)\}$. The results are as expected: more frequent updates to λ tend to overfit the value function to the TD targets and degrade the model's learning ability. Once overfitting occurs, the policy collapses into a near-deterministic regime, causing episodic rewards to drop sharply, aligning with the trends observed in the bottom right subfigure, where for the $(b_\lambda, m_\lambda) = \{(4, 2); (4, 3)\}$ settings, the performance plummets immediately after the value loss crosses a critical threshold at around half way into training. In contrast, a slower update schedule allows λ to evolve on a timescale closer to the critic's learning dynamics, preventing abrupt shifts in the return estimator. This stabilizes value prediction and leads to higher final episodic rewards. The comparable performance of the other three settings, with a slight advantage over the $(b_\lambda, m_\lambda) = (4, 1)$ setting, confirms that updating λ at a moderate frequency is more beneficial.

D. Ablation study on the short-horizon length h

As discussed, the variance of PW gradient increases with the length of the learning horizon, whereas the TD-λ return mitigates the negative effect of long learning horizons by exponentially discounting distant terms. Theoretically, this allows

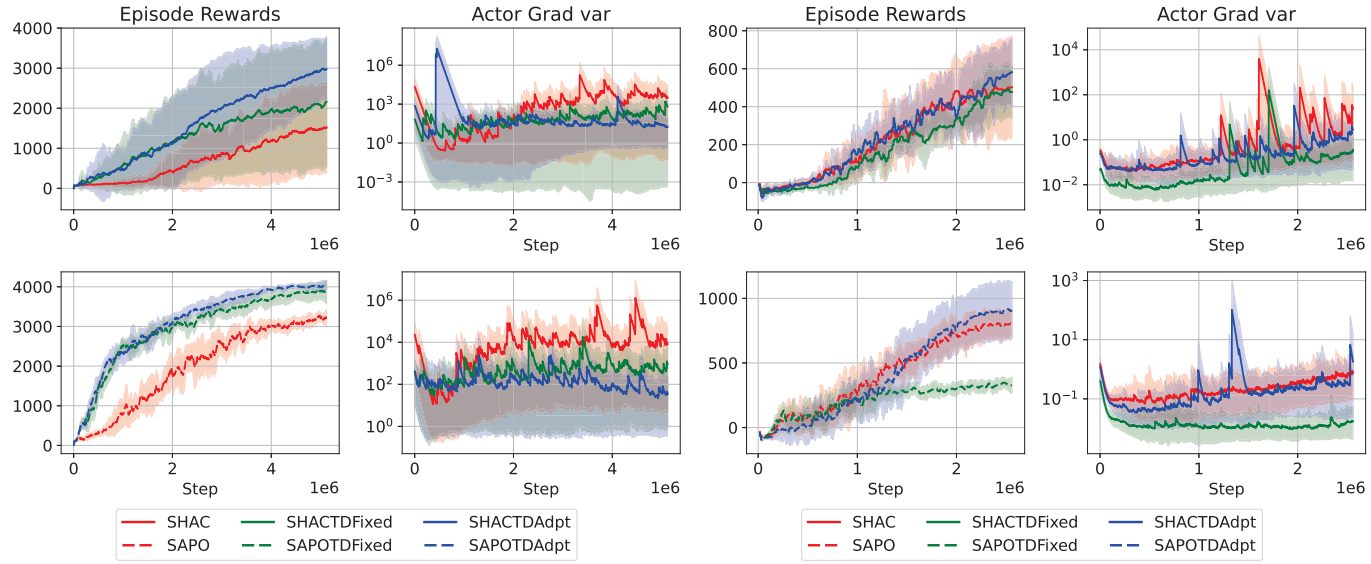


(a) Policy learned with SAPOTDAdpt



(b) Policy learned with SAPO

Fig. 8: Demonstration of the learned locomotion in the Go2 environment. Our algorithm produces a more natural forward gait: while the baseline SAPO tends to generate a hopping pattern, SAPOTDAdpt trains the robot to adopt a gallop-like motion, enabling it to achieve higher forward velocities.



(a) Go2

(b) SoftJumper

Fig. 9: Additional training results of Rewarped environments. Our methods demonstrate significant improvement over legged locomotion environments. However, learning soft-body locomotion task is particularly challenging due to their highly nonlinear dynamics with highly oscillatory, state-sensitive behavior. As demonstrated, without the adaptive λ module, the TD-λ estimations can be highly biased but still have low variance, leading to a value function that provides a bad gradient signal.

the TDFixed and TDAdpt algorithms to safely use larger h and extract richer gradient information. To empirically validate this observation, we train SHACTDAdpt and the baseline SHAC while varying the horizon $h \in \{32, 40, 48, 64, 80\}$. As demonstrated in Fig. 7, increasing h causes the actor gradient variance to increase by orders of magnitude - up to 10^9 - which makes optimization significantly more difficult for the baseline SAPO. In contrast, the TD-λ return effectively suppresses the high-variance issue and keeps the variance consistently around 10^3, allowing the SAPOTDAdpt to exploit longer gradient flow in its search for the optimal policy. Moreover, the policies trained with $h = 64$ and $h = 80$ even slightly outperform the baseline $h = 32$ by respectively achieving episodic rewards of 7018.64 ± 538.36 and 6974.35 ± 720.83 over 6877.21 ± 769.65 (Fig. 7, left). Although very long rollouts inevitably increase memory usage and computational cost, these results highlight the potential of our design to enable more sample-efficient FO-AC algorithms in the future.

E. Additional results with locomotion tasks

We further evaluate our methods on two additional locomotion tasks from the Rewarped suite, as shown in 9. In the Unitree Go2 quadruped environment, SHACTDAdpt delivers a significant improvement over the baseline SHAC, which fails to acquire meaningful forward locomotion over the same number of training steps. For SAPO, both SAPOTDFixed and SAPOTDAdpt consistently outperform the baseline by roughly 25% in asymptotic performance.

In the SoftJumper environment, the performance gains of our algorithms are more modest. To this end, we hypothesize that the gradients obtained from the h-step return are already accurate and have low variance. Interestingly, SAPOTDFixed fails to learn a locomotion policy altogether, whereas SAPOTDAdpt succeeds. This failure highlights the importance of adaptive λ scheduling: a fixed trace-decay parameter may be ill-suited for environments with highly nonlinear dynamics. Due to its challenging dynamics, the TD-λ return is prone to yielding a highly biased estimate (though low variance), which will eventually provide a worse gradient signal to the actor. In contrast, the adaptive approach automatically adjusts the bias–variance balance, preventing collapse and enabling stable policy learning.

VI. Conclusion

In this study, we demonstrated that incorporating the lower-variance, higher-bias TD-λ return into the optimization objective significantly enhances on-policy FO-AC algorithms. Our approach stabilizes the PW gradient and mitigates the exploding/vanishing issues that commonly arise in continuous-control tasks with long decision horizons and contact-rich dynamics. To eliminate the need for manual λ tuning, we further introduced an adaptive mechanism that optimizes λ online through a value-fitting objective. This mechanism allows the algorithm to automatically balance the bias–variance trade-off throughout training, leading to more consistent learning behavior. Our empirical evaluations on challenging continuous-control tasks demonstrate clear performance gains over SHAC and SAPO, both in average return and training stability. Ablation studies further reveal that TD-λ facilitates the use of larger short-horizon lengths, allowing richer dynamics to be captured without sacrificing robustness. More importantly, our design is not restricted to these baselines, but can be integrated into any on-policy FO-based frameworks. Therefore, for future work, we are interested in exploring extensions to first-order variants of well-established on-policy zeroth-order actor-critic methods.

References

[1] V. H. Dang, A. Redder, H. X. Pham, A. Sarabakha, and E. Kayacan, "Vds-nav: Volumetric depth-based safe navigation for aerial robots–bridging the sim-to-real gap," *IEEE Robotics and Automation Letters*, vol. 10, no. 10, pp. 11 038–11 045, 2025.

[2] H. I. Ugurlu, A. Redder, and E. Kayacan, "Lyapunov-inspired deep reinforcement learning for robot navigation in obstacle environments," in *2025 IEEE Symposium on Computational Intelligence on Engineering/Cyber Physical Systems (CIES)*, 2025, pp. 1–8.

[3] Z. Qiao, X. H. Pham, S. Ramasamy, X. Jiang, E. Kayacan, and A. Sarabakha, "Continual learning for robust gate detection under dynamic lighting in autonomous drone racing," 2024. [Online]. Available: https://arxiv.org/abs/2405.01054

[4] K. F. Andersen, H. X. Pham, H. I. Ugurlu, and E. Kayacan, "Event-based navigation for autonomous drone racing with sparse gated recurrent network," 2022. [Online]. Available: https://arxiv.org/abs/2204.02120

[5] S. Bohara, M. A. Hanif, and M. Shafique, "Curiousrl: Curiosity-driven reinforcement learning for adaptive locomotion in quadruped robots," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–8.

[6] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adi7566

[7] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, Jan. 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abk2822

[8] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang, "Industreal: Transferring contact-rich assembly tasks from simulation to reality," 2023. [Online]. Available: https://arxiv.org/abs/2305.17110

[9] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," 2018. [Online]. Available: https://arxiv.org/abs/1806.07851

[10] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).

[11] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax – a differentiable physics engine for large scale rigid body simulation," 2021. [Online]. Available: https://arxiv.org/abs/2106.13281

[12] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting differentiable simulators with neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://github.com/google-research/tiny-differentiable-simulator

[13] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin, "Accelerated policy learning with parallel differentiable simulation," in *International Conference on Learning Representations*, 2021.

[14] S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih, "Monte carlo gradient estimation in machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, Jan. 2020.

[15] Y. Song, S. Kim, and D. Scaramuzza, "Learning quadruped locomotion using differentiable simulation," 2024. [Online]. Available: https://arxiv.org/abs/2403.14864

[16] J. Pan, J. Xing, R. Reiter, Y. Zhai, E. Aljalbout, and D. Scaramuzza, "Learning on the fly: Rapid policy adaptation via differentiable simulation," 2025. [Online]. Available: https://arxiv.org/abs/2508.21065

[17] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," 2016. [Online]. Available: https://arxiv.org/abs/1506.05254

[18] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," 2022. [Online]. Available: https://arxiv.org/abs/2111.05803

[19] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," 2015. [Online]. Available: https://arxiv.org/abs/1506.02557

[20] H. J. T. Suh, M. Simchowitz, K. Zhang, and R. Tedrake, "Do differentiable simulators give better policy gradients?" in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246472918

[21] M. A. Z. Mora, M. Peychev, S. Ha, M. Vechev, and S. Coros, "Pods: Policy optimization via differentiable simulation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 7805–7817. [Online]. Available: https://proceedings.mlr.press/v139/mora21a.html

[22] I. Georgiev, K. Srinivasan, J. Xu, E. Heiden, and A. Garg, "Adaptive horizon actor-critic for policy learning in contact-rich differentiable simulation," 2024. [Online]. Available: https://arxiv.org/abs/2405.17784

[23] E. Xing, V. Luk, and J. Oh, "Stabilizing reinforcement learning in differentiable multiphysics simulation," *International Conference on Learning Representations (ICLR)*, 2025.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018. [Online]. Available: https://arxiv.org/abs/1506.02438

[25] Y. Chen, F. Zhang, and Z. Liu, "Adaptive advantage estimation for actor-critic algorithms," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.