Momentum-Based Policy Gradient with Second-Order Information

Saber Salehkaleybar[1 2], **Sadegh Khorasani**[1], **Negar Kiyavash**[2], **Niao He**[3], **Patrick Thiran**[1]

[1]School of Computer and Communication Sciences, EPFL
[2]College of Management of Technology, EPFL
[3]Department of Computer Science, ETH Zurich

Abstract

Variance-reduced gradient estimators for policy gradient methods have been one of the main focus of research in the reinforcement learning in recent years as they allow acceleration of the estimation process. We propose a variance-reduced policy-gradient method, called SHARP, which incorporates second-order information into stochastic gradient descent (SGD) using momentum with a time-varying learning rate. SHARP algorithm is parameter-free, achieving ϵ-approximate first-order stationary point with $O(\epsilon^{-3})$ number of trajectories, while using a batch size of $O(1)$ at each iteration. Unlike most previous work, our proposed algorithm does not require importance sampling which can compromise the advantage of variance reduction process. Moreover, the variance of estimation error decays with the fast rate of $O(1/t^{2/3})$ where t is the number of iterations. Our extensive experimental evaluations show the effectiveness of the proposed algorithm on various control tasks and its advantage over the state of the art in practice.

1 Introduction

Reinforcement Learning (RL) has achieved remarkable success in solving various complex tasks in games (Silver et al. 2017), autonomous driving (Shalev-Shwartz, Shammah, and Shashua 2016), and robot manipulation (Deisenroth et al. 2013), among other fields. In RL setting, an agent tries to learn the best actions by interacting with the environment and evaluating its performance based on reward signals. More specifically, in Markov Decision Processes (MDPs), the mathematical formalism for RL, after taking an action, the state changes according to a transition probability model and a reward signal is received based on the action taken and the current state. The main goal of the learner is to find a policy that maps the state space to the action space, maximizing the expected cumulative rewards as the objective function.

Policy gradient methods (Sutton et al. 2000) are often used for obtaining good policies in MDPs, especially for high-dimensional continuous action space. In policy gradient methods, the policy is parameterized by an unknown parameter θ and it is directly optimized using the stochastic first-order gradient of cumulative rewards as it is infeasible to compute the gradient exactly. REINFORCE (Williams 1992), PGT (Sutton et al. 2000), and GPOMDP (Baxter and Bartlett 2001) are some classical methods that update the policy by applying a stochastic gradient ascent step. These methods generally require a large number of trajectories due to the large variance of gradient estimates, stemming from randomness of transitions over trajectories.

In the RL literature, several methods have been proposed to reduce the variance in policy gradient methods. For instance, (Sutton et al. 2000) proposed to consider a baseline in order to reduce variance of gradient estimation. (Konda and Tsitsiklis 2000) presented an actor-critic algorithm that estimates the value function and uses it to mitigate the effect of large variance. (Schulman et al. 2015b) proposed GAE to control both bias and variance by exploiting a temporal difference relation for the advantage function approximation. More recent work such as TRPO (Schulman et al. 2015a) considers a Kullback-Leibler (KL) divergence penalty term in order to ensure that the updated policy remains close to the current policy or PPO (Schulman et al. 2017) that uses clipped surrogate objective function to achieve the same goal. In practice, it has been shown that these algorithms have better performance compared with vanilla policy gradient method.

Most stochastic gradient based policy methods need $O(\epsilon^{-4})$ trajectories in order to achieve ϵ-approximate first-order stationary point (FOSP) of the objective function $J(\theta)$, i.e., $\mathbb{E}[\|\nabla J(\theta)\|] \leq \epsilon$ (Ghadimi and Lan 2013; Shani, Efroni, and Mannor 2020). In recent years, there have been several attempts to reduce the variance of policy gradient by adapting variance reduction techniques proposed previously in supervised learning context (a list of previous work is given in Section 4). These methods can achieve sample complexity of $O(\epsilon^{-3})$ in RL setting and this rate is optimal in stochastic optimization under some mild assumptions on the objective function and stochastic gradients (Arjevani et al. 2020). In supervised learning problems, the objective function is oblivious, in the sense that the randomness that selects the loss function does not depend on the parameters that are to be optimized. On the other hand, in RL setting, the distribution over trajectories is non-stationary and changes over time as the parameters of policy are updated. To resolve this issue, most previous work utilized importance sampling techniques, which may degrade the effectiveness of the variance reduction process (Yang et al. 2019). Moreover, to analyze the convergence rate of these methods, a strong assumption on the variance of importance sampling weights is assumed,

Preprint.

which may not hold in RL setting. Most importantly, these methods often need huge batch sizes, which is highly undesirable in practice.

In this paper, we propose Stochastic Hessian Aided Recursive Policy gradient (SHARP) algorithm, which incorporates second-order information into SGD with momentum. Our main contributions are summarized as follows:

- Under some common regularity assumptions on the parameterized policy, SHARP reaches ϵ-FOSP with a sample complexity of $O(\epsilon^{-3})$. Moreover, our algorithm does not use importance sampling techniques. As a result, we can relax the strong additional assumptions on importance sampling weights customary in the literature.
- The batch size of SHARP is $O(1)$ and it does not require checkpoints, thanks to the use of a second-order term in the updates and time-varying learning rate and momentum weight.
- SHARP is parameter-free in the sense that the initial learning rate and momentum weight do not depend on the parameters of the problem. Moreover, the variance of the estimation error decays with the rate of $O(1/t^{2/3})$, where t is the number of iterations.
- Our experimental results show that SHARP outperforms the state of the art on various control tasks, with remarkable performance in more complex environments.

The rest of this paper is organized as follows: In Section 2, we define the problem and provide some notations and background on variance reduction methods in supervised learning. In Section 3, we describe the proposed algorithm and analyze its convergence rate. In Section 4, we give a summary of previous work and discuss how our proposed algorithm differs from them. In Section 5, we evaluate the performance of the proposed algorithm against the related work experimentally. Finally, we conclude the paper in Section 6.

2 Preliminaries

Notations and problem definition

Consider a discrete-time MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, R, \gamma, \rho\}$ that models how an agent interacts with a given environment. $\mathcal{S}$ and $\mathcal{A}$ are state space and action space, respectively. $P(s'|s,a)$ denotes the probability of transiting to state s' from s after taking action a. The reward function R returns reward $r(s,a)$ when action a is taken in state s. Parameter $\gamma \in (0,1)$ denotes the discount factor and ρ is the distribution of starting state. The actions are chosen according to policy π where $\pi(a|s)$ is the probability of taking action a for a given state s. Here, we assume that the policy is parameterized with a vector $\theta \in \mathbb{R}^d$ and use shorthand notation π_θ for $\pi_\theta(a|s)$.

For a given time horizon H, according to policy π_θ, the agent observes a sequence of state-action pairs $\tau = (s_0, a_0, \cdots, s_{H-1}, a_{H-1})$ called a trajectory. The probability of observing a trajectory τ for a given policy π_θ is:

$$p(\tau|\pi_\theta) = \rho(s_0) \prod_{h=0}^{H-1} P(s_{h+1}|s_h, a_h)\pi_\theta(a_h|s_h). \quad (1)$$

The discounted cumulative reward for a trajectory τ is defined as $R(\tau) := \sum_{h=0}^{H-1} \gamma^h r(s_h, a_h)$ and the expected return for a policy π_θ is:

$$J(\theta) := \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \quad (2)$$

The main goal in policy-based RL is to find $\theta^* = \arg\max_\theta J(\theta)$. As in many applications, $J(\theta)$ is non-convex and we settle instead for obtaining ϵ-FOSP, $\hat{\theta}$, such that $\mathbb{E}[\|\nabla J(\hat{\theta})\|] \leq \epsilon$. It can be shown that:

$$\nabla J(\theta) = \mathbb{E}\left[\sum_{h=0}^{H-1} \Psi_h(\tau)\nabla \log \pi_\theta(a_h|s_h)\right], \quad (3)$$

where $\Psi_h(\tau) = \sum_{t=h}^{H-1} \gamma^t r(s_t, a_t)$. Therefore, for any trajectory τ, $g(\tau;\theta) := \sum_{h=0}^{H-1} \Psi_h(\tau)\nabla \log \pi_\theta(a_h|s_h)$ is an unbiased estimator of $\nabla J(\theta)$. The vanilla policy gradient updates θ as follows:

$$\theta \leftarrow \theta + \eta g(\tau;\theta), \quad (4)$$

where η is the learning rate.

The Hessian matrix of $J(\theta)$ can be written as follows (Shen et al. 2019):

$$\nabla^2 J(\theta) = \mathbb{E}[\nabla\Phi(\theta;\tau)\nabla \log p(\tau|\pi_\theta)^T + \nabla^2\Phi(\theta;\tau)], \quad (5)$$

where $\Phi(\theta;\tau) = \sum_{h=0}^{H-1}\sum_{t=h}^{H-1} \gamma^t r(s_t, a_t)\nabla \log \pi_\theta(a_h|s_h)$. For a given trajectory τ, $B(\tau;\theta) := \nabla\Phi(\theta;\tau)\nabla \log p(\tau|\pi_\theta)^T + \nabla^2\Phi(\theta;\tau)$ is an unbiased estimator of the Hessian matrix.

Variance reduced methods for gradient estimation

Variance reduced methods for estimating the gradient vector were originally proposed for the stochastic optimization setting:

$$\min_{\theta\in\mathbb{R}^d} \mathbb{E}_{z\sim p(z)}[f(\theta, z)], \quad (6)$$

where a sample z is drawn from distribution $p(z)$ and $f(.,z)$ is commonly assumed to be smooth and non-convex function of θ. This setting is mainly considered in supervised learning context where θ corresponds to the parameters of the training model and $z = (x, y)$ is the training sample, with x the feature vector of the sample and y the corresponding label. In this setting, the distribution $p(z)$ is invariant with respect to parameter θ.

The common approach for reducing the variance of gradient estimation is to reuse past gradient vectors. The pseudocode for this general framework for variance reduction is given in Algorithm 1. After every pre-determined number of iterations Q, there is a checkpoint to obtain an unbiased estimate of the gradient, denoted by h_t, at the current parameter θ_t by taking a batch of samples $\mathcal{B}_{check}$. Between any two consecutive checkpoints, the gradient at the parameter θ_t is estimated according to (8) by taking a batch of samples $\mathcal{B}$ drawn from $p(z)$. The above framework appeared in several previous variance reduction methods in stochastic optimization such as SARAH (Nguyen et al. 2017) and SPIDER (Fang et al. 2018). (Zhang 2021) discusses how to choose the size of batches and the parameters Q and η. In fact, there is a

Algorithm 1: Common framework in variance reduction methods

```
1: for $t = 0, \cdots, T-1$ do
2:
```

$$h_t = \begin{cases} \frac{1}{|\mathcal{B}_{check}|} \sum_{z \in \mathcal{B}_{check}} \nabla f(\theta_t, z) & \text{if } t \equiv 0 \pmod{Q}, \quad (7) \\ h_{t-1} + \frac{1}{|\mathcal{B}|} \sum_{z \in \mathcal{B}} \nabla f(\theta_t, z) - \nabla f(\theta_{t-1}, z), & \text{otherwise.} \quad (8) \end{cases}$$

```
3:     $\theta_{t+1} \leftarrow \theta_t - \eta h_t$
4: end for
5: Return $\theta_t$ with $t$ chosen randomly from $\{0, \cdots, T-1\}$
```

trade-off between η and $|\mathcal{B}|$. If a small batch size is used, then η is also required to be small. The two extremes are SpiderBoost (Wang et al. 2019) ($|\mathcal{B}| = O(\epsilon^{-1}), \eta = O(1)$) and SARAH (Nguyen et al. 2017) ($|\mathcal{B}| = O(1)$, $\eta = O(\epsilon)$). Very recently, (Li et al. 2021) proposed PAGE, where in each iteration t, either a batch of samples is taken with probability p_t to update the gradient or the previous estimate of the gradient is used with a small adjustment, with probability $1 - p_t$.

In the context of RL, a sample z corresponds to a trajectory τ. Unlike supervised learning, the distribution of these trajectories depends on the parameters of policy generating them. Therefore, in the second term in the sum in (8), namely $\nabla f(\theta_{t-1}, z)$, z (or trajectory τ in RL context) is generated according to policy π_{θ_t} while θ_{t-1} is the parameter of the policy at the previous iteration. In RL setting, importance sampling technique is commonly used to account for the distribution shift as follows:

$$h_t = h_{t-1} + \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} g(\theta_t; \tau) - w(\tau|\theta_t, \theta_{t-1}) g(\theta_{t-1}; \tau), \tag{9}$$

with the weights $w(\tau|\theta_t, \theta_{t-1}) = \prod_{h=0}^{H-1} \frac{\pi_{\theta_{t-1}}(a_h|s_h)}{\pi_{\theta_t}(a_h|s_h)}$.

As we shall see in Section 4, nearly all variance reduction approaches in RL employing the general framework of Algorithm 1, use an importance sampling technique. This could significantly degrade the performance of the approach as the gradient estimates depend heavily on these weights (Yang et al. 2019). Besides, these variance reduction methods often need giant batch sizes at checkpoints, which is not practical in RL setting. Finally, the hyper-parameters of these approaches must be selected carefully as they often use non-adaptive learning rates. To resolve the issue of requiring huge batch-sizes, in the context of stochastic optimization, a recent variance reduction method called STORM (Cutkosky and Orabona 2019) was proposed with the following update rule:

$$\begin{aligned} h_t = (1 - \alpha_t) h_{t-1} + \alpha_t \nabla f(\theta_t, z_t) \\ + (1 - \alpha_t)(\nabla f(\theta_t, z_t) - \nabla f(\theta_{t-1}, z_t)) \quad (10) \\ \theta_{t+1} \leftarrow \theta_t - \eta_t h_t, \quad (11) \end{aligned}$$

where z_t is the sample drawn at iteration t and α_t and η_t are the adaptive momentum weight and learning rate, respectively. Compared with SGD with momentum, the main difference in STORM is the correction term $\nabla f(\theta_t, z_t) - \nabla f(\theta_{t-1}, z_t)$ in (10). (Cutkosky and Orabona 2019) showed that by adaptively updating α_t and η_t based on the norm of stochastic gradient in previous iterations, STORM can achieve the same convergence rate as previous methods without requiring checkpoints nor a huge batch size. Later, a parameter-free version, called STORM+ (Levy, Kavis, and Cevher 2021), has been introduced using new adaptive learning rate and momentum weight. However, to adapt these methods in RL setting, we still need to use importance sampling techniques because of the term $\nabla f(\theta_{t-1}, z_t)$. Recently, (Tran and Cutkosky 2021) showed that the correction term can be replaced with a second-order term $\nabla^2 f(\theta_t, z_t)(\theta_t - \theta_{t-1})$ by considering additional assumption that objective function is second-order smooth. Besides, the above Hessian vector product can be computed in $O(Hd)$ (similar to the computational complexity of obtaining the gradient vector) by executing Pearlmutter's algorithm (Pearlmutter 1994).

3 The SHARP Algorithm

In this section, we propose the SHARP algorithm, which incorporates second-order information into SGD with momentum and provide a convergence guarantee. SHARP algorithm is presented in Algorithm 2. At each iteration t, we draw sample b_t from a uniform distribution in the interval $[0, 1]$ (line 5) and then obtain θ_t^b as the linear combination of θ_{t-1} and θ_t with coefficients $1 - b_t$ and b_t (line 6). In line 7, we sample trajectories τ_t and τ_t^b according to policies π_{θ_t} and $\pi_{\theta_t^b}$, respectively. Afterwards, we update the momentum weight α_t and the learning rate η_t (line 8) and then compute the estimate of gradient at time t, i.e., v_t, using the Hessian vector product $B(\tau_t^b; \theta_t^b)(\theta_t - \theta_{t-1})$ and stochastic gradient $g(\tau_t; \theta_t)$ (line 9). Finally we update θ_t based on a normalized version of v_t in line 10.

Remark 1 *By choosing a point uniformly at random on the line between θ_{t-1} and θ_t, we can ensure that $B(\tau_t^b; \theta_t^b)(\theta_t - \theta_{t-1})$ is an unbiased estimate of $\nabla J(\theta_t) - \nabla J(\theta_{t-1})$ (see* (23) *in Appendix A). As we mentioned before, in the context of stochastic optimization, (Tran and Cutkosky 2021) used the second-order term $\nabla^2 f(\theta_t, z_t)(\theta_t - \theta_{t-1})$, which is biased as the Hessian vector product is evaluated at the point θ_t. As*

Algorithm 2: The SHARP algorithm

Input: Initial point θ_0, parameters α_0, η_0, and number of iterations T

```
1: Sample trajectory $\tau_0$ with policy $\pi_{\theta_0}$
2: $v_0 \leftarrow g(\tau_0; \theta_0)$
3: $\theta_1 \leftarrow \theta_0 + \eta_0 \frac{v_0}{\|v_0\|}$
4: for $t = 1, \cdots, T-1$ do
5:     Sample $b_t \sim U(0,1)$
6:     $\theta_t^b \leftarrow b_t\theta_t + (1-b_t)\theta_{t-1}$
7:     Sample trajectories $\tau_t$ and $\tau_t^b$ with policies $\pi_{\theta_t}$ and $\pi_{\theta_t^b}$, respectively
8:     $\eta_t \leftarrow \frac{\eta_0}{t^{2/3}}$, $\alpha_t \leftarrow \frac{\alpha_0}{t^{2/3}}$
9:     $v_t \leftarrow (1-\alpha_t)(v_{t-1} + B(\tau_t^b; \theta_t^b)(\theta_t - \theta_{t-1})) + \alpha_t g(\tau_t; \theta_t)$
10:    $\theta_{t+1} \leftarrow \theta_t + \eta_t \frac{v_t}{\|v_t\|}$
11: end for
12: Return $\theta_t$ with $t$ chosen randomly from $\{0, \cdots, T-1\}$
```

a result, in order to provide the convergence guarantee, it is further assumed that the objective function is second-order smooth in (Tran and Cutkosky 2021).

Remark 2 *To give an intuition why the second-order term is helpful in the update in line 9, consider the following error term:*

$$\epsilon_t = v_t - \nabla J(\theta_t). \tag{12}$$

We can rewrite the above error term as follows:

$$\begin{aligned}\epsilon_t = (1-\alpha_t)(v_{t-1} - \nabla J(\theta_t) + B(\tau_t^b; \theta_t^b)(\theta_t - \theta_{t-1})) \\ + \alpha_t(g(\tau_t; \theta_t) - \nabla J(\theta_t)).\end{aligned} \tag{13}$$

Now, for a moment, suppose that $\mathbb{E}[v_{t-1}] = \mathbb{E}[\nabla J(\theta_{t-1})]$ (with total expectation on both sides). Then,

$$\mathbb{E}[v_{t-1} - \nabla J(\theta_t) + B(\tau_t^b; \theta_t^b)(\theta_t - \theta_{t-1})] = 0. \tag{14}$$

As v_0 is an unbiased estimate of gradient at θ_0, we can easily show by induction that according to above equation, $\mathbb{E}[v_t] = \mathbb{E}[\nabla J(\theta_t)]$ for any $t \geq 0$.

In the next part, we provide a theoretical guarantee on the convergence rate of SHARP algorithm.

Convergence Analysis

In this part, we analyze the convergence rate of Algorithm 2 under bounded reward function and some regularity assumptions on the policy π_θ.

Assumption 1 (Bounded reward) *For $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$, $|R(s,a)| < R_0$ where $R_0 > 0$ is some constant.*

Assumption 2 (Parameterization regularity) *There exist constants $G, L > 0$ such that for any $\theta \in \mathbb{R}^d$ and for any $s \in \mathcal{S}, a \in \mathcal{A}$:*
(a) $\|\nabla \log \pi_\theta(a|s)\| \leq G$,
(b) $\|\nabla^2 \log \pi_\theta(a|s)\| \leq L$.

Assumptions 1 and 2 are common in the RL literature (Papini et al. 2018; Shen et al. 2019) to analyze the convergence of policy gradient methods. Under these assumptions, the following upper bounds can be derived on $\mathbb{E}[\|g(\tau;\theta) - \nabla J(\theta)\|^2]$ and $\mathbb{E}[\|B(\tau;\theta) - \nabla^2 J(\theta)\|^2]$.

Lemma 1 ((Shen et al. 2019)) *Under Assumptions 1 and 2:*

$$\begin{aligned}\mathbb{E}[\|g(\tau;\theta) - \nabla J(\theta)\|^2] \leq \sigma_g^2 \\ \mathbb{E}[\|B(\tau;\theta) - \nabla^2 J(\theta)\|^2] \leq \sigma_B^2,\end{aligned} \tag{15}$$

where $\sigma_g^2 = \frac{G^2 R_0^2}{(1-\gamma)^4}$ and $\sigma_B^2 = \frac{H^2 G^4 R_0^2 + L^2 R_0^2}{(1-\gamma)^4}$.

Based on these bounds, we can provide the following guarantee on the convergence rate of SHARP algorithm. All proofs are provided in the appendix.

Theorem 1 *Suppose that the initial momentum weight $\alpha_0 \in (2/3, 1]$ and initial learning rate $\eta_0 > 0$. Under Assumptions 1 and 2, Algorithm 2 guarantees that:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \|\nabla J(\theta_t)\|\right] \leq \frac{8\sqrt{C} + 9C_J/\eta_0}{T^{1/3}} + \frac{6\sigma_B \eta_0}{T^{2/3}}, \tag{16}$$

where $C = 3\alpha_0(48\sigma_B^2\eta_0^2/\alpha_0 + (6\alpha_0 + 1/\alpha_0)\sigma_g^2)/(3\alpha_0 - 2)$ and $C_J = R_0/(1-\gamma)$.

Corollary 1 *The right hand side of* (16) *is dominated by the first term. If we set η_0 in the order of $\sqrt{C_J/\sigma_B}$, then the number of trajectories for achieving ϵ-FOSP would be $O(\frac{1}{(1-\gamma)^2\epsilon^3})$, where we assume that the time horizon H is set in the order of* $1/(1-\gamma)$.

Remark 3 *Along the iterations of the SHARP algorithm, it can be shown that the following inequality holds for any $t \geq 1$ (see* (21) *in Appendix A):*

$$\mathbb{E}[\|\epsilon_t\|^2] \leq (1-\alpha_t)\mathbb{E}[\|\epsilon_{t-1}\|^2] + O(\eta_t^2). \tag{17}$$

Therefore, the variance of the estimation error decays with the rate of $O(1/t^{2/3})$ (see Appendix B for the proof). To the best of our knowledge, existing variance reduction methods only guarantee the decay of accumulative variance. This appealing property of SHARP is largely due to the use of unbiased Hessian-aided gradient estimator and normalized gradient descent. Moreover, as a byproduct of these desirable properties, our convergence analysis turns out to be more simple, compared to existing work (Cutkosky and Orabona 2019; Tran and Cutkosky 2021). This could be of independent interest for better theory of variance-reduced methods.

Remark 4 *The SHARP algorithm is parameter-free in the sense that α_0 and η_0 are constants that do not depend on other parameters in the problem. Therefore, for any choice of $2/3 < \alpha_0 \leq 1$ and $\eta_0 > 0$, we can guarantee convergence to ϵ-FOSP with the sample complexity of $O(\epsilon^{-3})$. However, in practice, it is desirable to tune these constants to have smaller constants in the numerators of convergence rates in* (16). *For instance, σ_B might be large in some RL settings and we control the constant in the first term on the right hand side of* (16) *by tuning η_0. It is noteworthy that STORM+ is also parameter-free but it requires adaptive learning rate and momentum weight that depend on stochastic gradients in previous iterations.*

Method	SC	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{B}_{check}\rvert$	Checkpoint	IS	Further Assump.
SVRPG (Xu, Gao, and Gu 2020)	$O(\frac{1}{\epsilon^{10/3}})$	$O(\frac{1}{\epsilon^{4/3}})$	$O(\frac{1}{\epsilon^{4/3}})$	Needed	Needed	Assump. in (18)
HAPG (Shen et al. 2019)	$O(\frac{1}{\epsilon^3})$	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon^2})$	Needed	Not needed	-
VRMPO (Yang et al. 2019)	$O(\frac{1}{\epsilon^3})$	$O(\frac{1}{\epsilon})$	$O(\frac{1}{\epsilon^2})$	Needed	Not needed	-
SRVR-PG (Xu, Gao, and Gu 2019)	$O(\frac{1}{\epsilon^3})$	$O(\frac{1}{\sqrt{\epsilon}})$	$O(\frac{1}{\epsilon})$	Needed	Needed	Assump. in (18)
HSPGA (Pham et al. 2020)	$O(\frac{1}{\epsilon^3})$	$O(1)$	-	Not needed	Needed	Assump. in (18)
MBPG (Huang et al. 2020)	$\tilde{O}(\frac{1}{\epsilon^3})$	$O(1)$	-	Not needed	Needed	Assump. in (18)
PAGE-PG (Gargiani et al. 2022)	$O(\frac{1}{\epsilon^3})$	$O(1)$	$O(\frac{1}{\epsilon^2})$	Needed	Needed	Assump. in (18)
This paper	$O(\frac{1}{\epsilon^3})$	$O(1)$	-	Not needed	Not needed	-

Table 1: Comparison of main variance-reduced policy gradient methods to achieve ϵ-FOSP based on sample complexity (SC), batch size ($|\mathcal{B}|$), batch size at checkpoints ($|\mathcal{B}_{check}|$), and the need for checkpoints, importance sampling (IS), and additional assumptions.

Remark 5 *Regarding the dependency on ϵ, in the context of stochastic optimization, (Arjevani et al. 2020) have shown that under some mild assumptions on the objective function and stochastic gradient, the rate of $O(1/\epsilon^3)$ is optimal in order to obtain ϵ-FOSP, and cannot be improved with stochastic p-th order methods for $p \geq 2$.*

4 Related Work

In recent years, several variance-reduced methods have been proposed in order to accelerate the existing PG methods. (Papini et al. 2018) and (Xu, Gao, and Gu 2020) proposed SVRPG algorithm based on SVRG (Johnson and Zhang 2013), with sample complexity of $O(1/\epsilon^4)$ and $O(1/\epsilon^{10/3})$, respectively. This algorithm requires importance sampling techniques as well as the following further assumption for guaranteeing convergence to ϵ-FOSP:

- Bounded variance of importance sampling weights: For any trajectory τ, it is assumed that:

$$Var\left(\frac{p(\tau|\pi_{\theta_1})}{p(\tau|\pi_{\theta_2})}\right) \leq W, \qquad \forall \theta_1, \theta_2 \in \mathbb{R}^d, \qquad (18)$$

  where $W < \infty$ is a constant.

The above assumption is fairly strong as the importance sampling weight could grow exponentially with horizon length H (Zhang et al. 2021). In order to remove importance sampling weights, (Shen et al. 2019) proposed HAPG algorithm, which uses second-order information and achieves better sample complexity of $O(1/\epsilon^3)$. However, it still needs checkpoints and large batch sizes of $|\mathcal{B}| = O(1/\epsilon)$, and $|\mathcal{B}_{check}| = O(1/\epsilon^2)$.

In Table 1, we compare the main variance-reduced policy gradient methods achieving ϵ-FOSP in terms of sample complexity and batch size[1]. In this table, after HAPG (Shen et al. 2019), all the proposed variance reduction methods achieve a similar sample complexity. (Yang et al. 2019) proposed the VRMPO method based on stochastic mirror descent, and similarly to HAPG (Shen et al. 2019), they do not require importance sampling weights in the variance reduction part. The orders of batch sizes are also the same as in HAPG. (Xu, Gao, and Gu 2019) proposed SRVR-PG, and used stochastic path-integrated differential estimators for variance reduction. This algorithm uses important sampling weights and the required batch sizes are in the order of $|\mathcal{B}| = O(1/\sqrt{\epsilon})$ and $|\mathcal{B}_{check}| = O(1/\epsilon)$. Later, (Pham et al. 2020) proposed HSPGA by adapting the SARAH estimator for reducing the variance of REINFORCE. HSPGA still needs importance sampling weights, but the batch size is reduced to $O(1)$. (Huang et al. 2020) proposed three variants of momentum-based policy gradient (called MBPG), which are based on the STORM algorithm (Cutkosky and Orabona 2019). Thus, the required batch size is in the order of $O(1)$, similarly to STORM. However, it still needs to use importance sampling weights. (Zhang et al. 2021) proposed TSIVR-PG with a gradient truncation mechanism in order to resolve some of issues pertaining to the use of importance sampling weights. In their convergence analysis, they are restricted to soft-max policy with some specific assumptions on the parameterization functions. Recently, based on PAGE (Li et al. 2021), (Gargiani et al. 2022) proposed the PAGE-PG algorithm which takes a batch of samples of $O(\epsilon^{-2})$ for updating the parameters with probability p_t or reuse the previous estimate gradient with a small adjustment, with probability $1 - p_t$. The proposed algorithm requires importance sampling weights and thus the

[1]Please note that the sample complexity also depends on the other parameters such as horizon length H and discount factor γ. Here, we just mention the dependency of sample complexity on ϵ.

additional assumption in (18) to guarantee convergence to ϵ-FOSP with a sample complexity of $O(\epsilon^{-3})$.

There exist few recent work on the global convergence of policy gradient methods. For instance, (Liu et al. 2020) showed global convergence of policy gradient, natural policy gradient, and their variance reduced variants, in the case of positive definite Fisher information matrix of the policy. (Chung et al. 2021) studied the impact of baselines on the learning dynamics of policy gradient methods and showed that using a baseline minimizing the variance can converge to a sub-optimal policy. Recently, (Ding, Zhang, and Lavaei 2022) studied the soft-max and the Fisher non-degenerate policies, and showed that adding a momentum term improves the global optimality sample complexities of vanilla PG methods by $\tilde{O}(\epsilon^{-1.5})$ and $\tilde{O}(\epsilon^{-1})$, respectively.

The aforementioned discussion for the main methods are summarized in Table 1. For each method, we mention whether it needs checkpoints and importance sampling weights.[2] All the discussed aforementioned methods require Assumptions 1 and 2. In the last column, additional assumptions besides these two are listed for each method.

Comparing the sample complexity of our algorithm with previous work, note that all the algorithms (including ours) under SVRPG in Table 1 achieve the rate of $O(1/\epsilon^3)$ or $\tilde{O}(1/\epsilon^3)$. Without any further assumption, our proposed method is the only one that requires no checkpoints, no importance sampling weights, and has a batch size of the order of $O(1)$. As we will see in the next section, besides these algorithmic advantages, it has remarkable performance compared to the state of the art in various control tasks.

5 Experiments

In this section, we evaluate the performance of the proposed algorithm and compare it with two recent methods, HAPG (Shen et al. 2019) and MBPG (Huang et al. 2020), which are state-of-the-art representatives for variance reduced policy gradient methods. HAPG uses second order information (Hessian vector product) and does not need importance sampling weights. In addition, MBPG is the recent work for variance reduction based on STORM (Cutkosky and Orabona 2019). Besides these two methods, we further considered very recent work, PAGE-PG, which performs updates according to PAGE (Li et al. 2021). We also considered REINFORCE (Williams 1992) as the baseline method.

Regarding MBPG algorithm, we selected the IS-MBPG variant as its performance is superior to the other two variants of this algorithm. PAGE-PG was evaluated on just two simple environments of Acrobate and Cartpole with discrete actions in (Gargiani et al. 2022). Here, we compared SHARP with PAGE-PG in the these two environments (see Appendix D), showing that our method outperforms PAGE-PG in both. There is no complete implementation of PAGE-PG for continuous control tasks in (Gargiani et al. 2022). We extended their implementation for these tasks but the performance was not comparable with other methods. As such we did not report its performance on the control tasks.

We implemented SHARP in the Garage library (garage contributors 2019) as it allows for maintaining and integrating it in future versions of Garage library for easier dissemination. We utilized a Linux server with Intel Xeon CPU E5-2680 v3 (24 cores) operating at 2.50GHz with 377 GB DDR4 of memory, and Nvidia Titan X Pascal GPU. The computation was distributed over 48 threads to ensure a relatively efficient run time. The implementation of SHARP is available as the supplementary material.

In our experiments, we used the MuJoCo simulator (Todorov, Erez, and Tassa 2012) which is a physical engine, suitable for simulating robotic tasks with a good accuracy and speed in RL setting. We evaluated the following six control tasks with continuous action space: Reacher, Walker, HalfCheeta, Humanoid, Hopper, and Swimmer. For each task, we used a Gaussian multi-layer perceptron (MLP) policy whose mean and variance are parameterized by an MLP with two hidden layers of 64 neurons. We used the default implementation of the Gaussian MLP policy in the Garage library. For a fair comparison, we considered the same network architecture for the other methods.

A baseline term $b(s)$, which only depends on the current state, can be subtracted from the discounted cumulative reward $\Psi_h(\tau)$ in order to further reduce the variance as follows:

$$\Psi_h(\tau) = \sum_{t=h}^{H-1} \gamma^t r(s_t, a_t) - b(s_h), \tag{19}$$

where b is trained based on the previous observed trajectories. It is noteworthy that $g(\tau;\theta)$ and $B(\tau;\theta)$ remain unbiased with the above modification. For all the methods, we trained a linear feature baseline or Gaussian baseline in our experiments which have been implemented in the Garage library (the details of baselines are given in the appendix). We tuned the hyper-parameters of the proposed algorithm using grid search. We adjusted the hyper-parameters of the other methods by grid search if they were not explicitly specified.

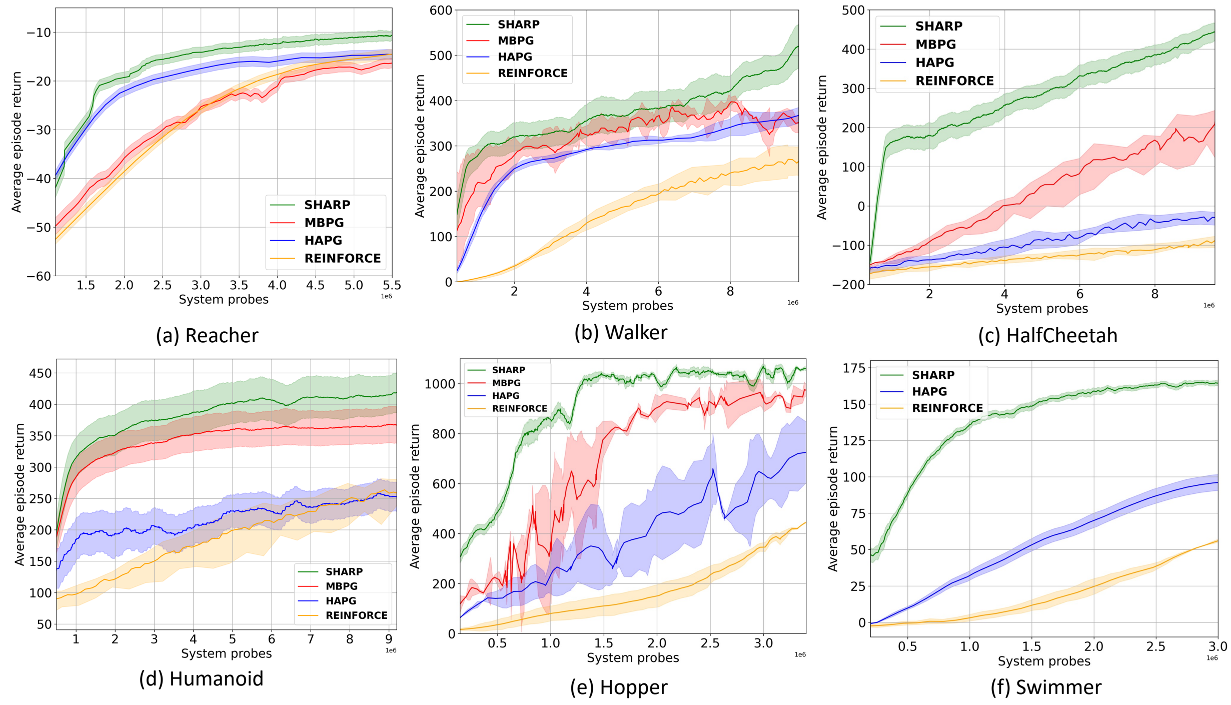
We considered average episode return as the performance measure versus system probes, i.e., the number of observed state-action pairs. System probes provide a better measure of sample complexity than the number of trajectories because of the varying lengths of trajectories. For each task, we executed the methods 10 times and reported the empirical estimation of mean and 90% confidence interval of the episode return.

In Figure 1, the average episode return against system probes is depicted for the six control tasks. In the following, we will discuss the results for each task.

Reacher environment: In Reacher environment, there is an arm with two degrees of freedom, aiming to reach a target point in the two dimensional plane. As can be seen in Figure 1 (a), SHARP exceeds the average episode return of -20 after about 2×10^6 system probes while for HAPG, REINFORCE, and MBPG this happens after about 2.5×10^6, 3.8×10^6, 4×10^6 system probes, respectively.

Walker2D environment: In this environment, a humanoid walker tries to move forward in a two dimensional space, i.e., it can only fall forward or backward. As shown

[2]To be more precise, although PAGE-PG, has no fixed checkpoints, it takes a batch of $O(\epsilon^{-2})$ to get an unbiased estimate of the gradient with probability p_t. Therefore, in this sense, it requires checkpoints.



(a) Reacher (b) Walker (c) HalfCheetah

(d) Humanoid (e) Hopper (f) Swimmer

Figure 1: Comparison of SHARP with other variance reduction methods on six control tasks.

in Figure 1 (b), HAPG and MBPG have a similar performance (average episode return of about 350) after 10^7 system probes and MBPG is slightly better. On the other hand, our proposed algorithm achieves much higher average episode return (about 500).

HalfCheetah environment: In this environment, we control a two-dimensional cheetah robot and the goal is to make it run as fast as possible in the forward direction. As depicted in Figure 1 (c), the average episode return of HAPG and REINFORCE is less than zero after 10^7 system probes while MBPG, achieves 200. SHARP has the best performance with average episode return of more than 400.

Humanoid environment: In this environment, we control a three-dimensional bipedal robot to make it walk forward as fast as possible, without falling over. The state space is 376-dimensional, containing the position and velocity of each joint, the friction of actuator, and contact forces. The action space is a 17-dimensional continuous space. As depicted in Figure 1 (d), HAPG and REINFORCE have similar performance (average episode return of about 250) after 10^7 system probes and for MBPG, this value is about 370. However, SHARP exceeds 400 for the same system probes.

Hopper environments: The hopper is a two-dimensional one-legged agent, trying to make hops in the right direction. As shown in Figure 1 (e), SHARP outperforms other algorithms by reaching the average episode return of about 1000 after 1.5×10^6 system probes.

Swimmer environment: In this environment, a swimmer is in a two-dimensional pool and the goal is to move as fast as possible towards the right direction. For this environment, we could not reproduce the good performance for MBPG, despite trying different hyper-parameters and random seeds. Figure 1 (f) shows that SHARP has a remarkable performance with respect to two other methods by achieving an average episode return of about 160 after 3×10^6 system probes.

In summary, the proposed algorithm outperforms the other methods in all considered environments. The improvement is even more drastic for some complex environments, such as HalfCheetah, than for simple environments, such as Reacher.

We also investigated the effect of batch size on the performance of SHARP. The experimental results show that the impact of batch size is negligible. For more details, please refer to Appendix D.

6 Conclusion

We proposed a variance-reduced policy-gradient method, which incorporates second-order information, i.e., Hessian vector product, into SGD with momentum. The Hessian vector product can be computed with an efficiency that is similar to that of obtaining the gradient vector. Therefore, the computational complexity of the proposed algorithm per iteration remains in the same order as first-order methods. More importantly, using the second-order correction term enables us to obtain an estimate of the gradient, completely bypassing importance sampling. Moreover, the batch size is $O(1)$ and there is no need for checkpoints, which makes the algorithm appealing in practice. Under some regularity assumptions on the parameterized policy, we showed that it achieves ϵ-FOSP with sample complexity of $O(\epsilon^{-3})$. SHARP is parameter-free in the sense that the initial learning rate and momentum weight do not depend on the parameters of the problem. Experimental results show its remarkable performance in various control tasks, especially in some complex environments, such as HalfCheetah, compared to the state of the art.

References

Arjevani, Y.; Carmon, Y.; Duchi, J. C.; Foster, D. J.; Sekhari, A.; and Sridharan, K. 2020. Second-order information in non-convex stochastic optimization: Power and limitations. In *Conference on Learning Theory*, 242–299. PMLR.

Baxter, J.; and Bartlett, P. L. 2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15: 319–350.

Chung, W.; Thomas, V.; Machado, M. C.; and Le Roux, N. 2021. Beyond variance reduction: Understanding the true impact of baselines on policy optimization. In *International Conference on Machine Learning*, 1999–2009. PMLR.

Cutkosky, A.; and Orabona, F. 2019. Momentum-Based Variance Reduction in Non-Convex SGD. *Advances in neural information processing systems*, 32.

Deisenroth, M. P.; Neumann, G.; Peters, J.; et al. 2013. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2): 388–403.

Ding, Y.; Zhang, J.; and Lavaei, J. 2022. On the Global Optimum Convergence of Momentum-based Policy Gradient. In *International Conference on Artificial Intelligence and Statistics*, 1910–1934. PMLR.

Fang, C.; Li, C. J.; Lin, Z.; and Zhang, T. 2018. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator. *arXiv preprint arXiv:1807.01695*.

garage contributors, T. 2019. Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage.

Gargiani, M.; Zanelli, A.; Martinelli, A.; Summers, T.; and Lygeros, J. 2022. PAGE-PG: A Simple and Loopless Variance-Reduced Policy Gradient Method with Probabilistic Gradient Estimation. In *International Conference on Machine Learning*.

Ghadimi, S.; and Lan, G. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4): 2341–2368.

Huang, F.; Gao, S.; Pei, J.; and Huang, H. 2020. Momentum-based policy gradient methods. In *International Conference on Machine Learning*, 4422–4433. PMLR.

Johnson, R.; and Zhang, T. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26: 315–323.

Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Levy, K.; Kavis, A.; and Cevher, V. 2021. STORM+: Fully Adaptive SGD with Momentum for Nonconvex Optimization. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, CONF.

Li, Z.; Bao, H.; Zhang, X.; and Richtárik, P. 2021. PAGE: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International Conference on Machine Learning*, 6286–6295. PMLR.

Liu, Y.; Zhang, K.; Basar, T.; and Yin, W. 2020. An Improved Analysis of (Variance-Reduced) Policy Gradient and Natural Policy Gradient Methods. In *NeurIPS*.

Nguyen, L. M.; Liu, J.; Scheinberg, K.; and Takáč, M. 2017. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, 2613–2621. PMLR.

Papini, M.; Binaghi, D.; Canonaco, G.; Pirotta, M.; and Restelli, M. 2018. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, 4026–4035. PMLR.

Pearlmutter, B. A. 1994. Fast exact multiplication by the Hessian. *Neural computation*, 6(1): 147–160.

Pham, N.; Nguyen, L.; Phan, D.; Nguyen, P. H.; Dijk, M.; and Tran-Dinh, Q. 2020. A hybrid stochastic policy gradient algorithm for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 374–385. PMLR.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015a. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Shani, L.; Efroni, Y.; and Mannor, S. 2020. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5668–5675.

Shen, Z.; Ribeiro, A.; Hassani, H.; Qian, H.; and Mi, C. 2019. Hessian aided policy gradient. In *International conference on machine learning*, 5729–5738. PMLR.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Tran, H.; and Cutkosky, A. 2021. Better SGD using Second-order Momentum. *arXiv preprint arXiv:2103.03265*.

Wang, Z.; Ji, K.; Zhou, Y.; Liang, Y.; and Tarokh, V. 2019. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32: 2406–2416.

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

Xu, P.; Gao, F.; and Gu, Q. 2019. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*.

Xu, P.; Gao, F.; and Gu, Q. 2020. An improved convergence analysis of stochastic variance-reduced policy gradient. In *Uncertainty in Artificial Intelligence*, 541–551. PMLR.

Yang, L.; Zheng, G.; Zhang, H.; Zhang, Y.; Zheng, Q.; Wen, J.; and Pan, G. 2019. Policy optimization with stochastic mirror descent. *arXiv preprint arXiv:1906.10462*.

Zhang, J.; Ni, C.; Yu, Z.; Szepesvari, C.; and Wang, M. 2021. On the convergence and sample efficiency of variance-reduced policy gradient method. *arXiv preprint arXiv:2102.08607*.

Zhang, L. 2021. *Variance Reduction for Non-Convex Stochastic Optimization: General Analysis and New Applications*. Master's thesis, ETH Zurich.

A Proof of Theorem 1

Let us define $\epsilon_t := v_t - \nabla J(\theta_t)$. Then, based on the update in line 9 of Algorithm 1, we have:

$$\epsilon_t = (1-\alpha_t)\epsilon_{t-1} + \alpha_t U_t + (1-\alpha_t)W_t, \tag{20}$$

where $U_t = (g(\tau_t;\theta_t) - \nabla J(\theta_t))$ and $W_t = B(\tau_t^b;\theta_t^b)(\theta_t - \theta_{t-1}) - (\nabla J(\theta_t) - \nabla J(\theta_{t-1}))$. Let $\mathcal{H}_t$ be the history up to time t, i.e., $\mathcal{H}_t := \{\theta_0, \tau_0, \tau_1, b_1, \tau_1^b, \cdots, \tau_t, b_t, \tau_t^b\}$.

By taking the square l_2 norm of both sides of the above equation,

$$\begin{aligned}\|\epsilon_t\|^2 &= (1-\alpha_t)^2\|\epsilon_{t-1}\|^2 + \alpha_t^2\|U_t\|^2 + (1-\alpha_t)^2\|W_t\|^2 + \\ &\quad 2\alpha_t(1-\alpha_t)\langle\epsilon_{t-1}, U_t\rangle + 2\alpha_t(1-\alpha_t)\langle U_t, W_t\rangle + 2(1-\alpha_t)^2\langle\epsilon_{t-1}, W_t\rangle \\ &\le (1-\alpha_t)^2\|\epsilon_{t-1}\|^2 + 2\alpha_t^2\|U_t\|^2 + 2(1-\alpha_t)^2\|W_t\|^2 + \\ &\quad 2\alpha_t(1-\alpha_t)\langle\epsilon_{t-1}, U_t\rangle + 2(1-\alpha_t)^2\langle\epsilon_{t-1}, W_t\rangle,\end{aligned} \tag{21}$$

where we used Young's inequality in the first inequality for the following term: $2\alpha_t(1-\alpha_t)\langle U_t, W_t\rangle \le (1-\alpha_t)^2\|W_t\|^2 + \alpha_t^2\|U_t\|^2$. Now, by taking expectations on both sides,

$$\begin{aligned}\mathbb{E}[\|\epsilon_t\|^2] &\le (1-\alpha_t)^2\mathbb{E}[\|\epsilon_{t-1}\|^2] + 2\alpha_t^2\mathbb{E}[\|U_t\|^2] + 2(1-\alpha_t)^2\mathbb{E}[\|W_t\|^2] + \\ &\quad 2\alpha_t(1-\alpha_t)\mathbb{E}[\langle\epsilon_{t-1}, U_t\rangle] + 2(1-\alpha_t)^2\mathbb{E}[\langle\epsilon_{t-1}, W_t\rangle] \\ &\overset{(a)}{\le} (1-\alpha_t)\mathbb{E}[\|\epsilon_{t-1}\|^2] + 2\alpha_t^2\mathbb{E}[\|U_t\|^2] + 2\mathbb{E}[\|W_t\|^2] + \\ &\quad 2\alpha_t(1-\alpha_t)\mathbb{E}[\langle\epsilon_{t-1}, U_t\rangle] + 2(1-\alpha_t)^2\mathbb{E}[\langle\epsilon_{t-1}, W_t\rangle] \\ &\overset{(b)}{\le} (1-\alpha_t)\mathbb{E}[\|\epsilon_{t-1}\|^2] + 2\alpha_t^2\mathbb{E}[\|U_t\|^2] + 2\mathbb{E}[\|W_t\|^2],\end{aligned} \tag{22}$$

(a) follows as $\alpha_t \le 1$ for all $t \ge 0$.
(b) follows because the following two terms are zero. First, $\mathbb{E}[\langle\epsilon_{t-1}, U_t\rangle] = \mathbb{E}[\mathbb{E}[\langle\epsilon_{t-1}, U_t\rangle|\mathcal{H}_{t-1}]] = 0$ as ϵ_{t-1} is determined given $\mathcal{H}_{t-1}$ and $\mathbb{E}[U_t|\mathcal{H}_{t-1}] = 0$ since $g(\tau_t;\theta_t)$ is an unbiased estimation of $\nabla J(\theta_t)$. Second, $\mathbb{E}[\langle\epsilon_{t-1}, W_t\rangle] = \mathbb{E}[\mathbb{E}[\langle\epsilon_{t-1}, W_t\rangle|\mathcal{H}_{t-1}]] = 0$ because

$$\begin{aligned}\mathbb{E}[W_t|\mathcal{H}_{t-1}] &= \mathbb{E}[B(\tau_t^b;\theta_t^b)(\theta_t - \theta_{t-1})|\mathcal{H}_{t-1}] - (\nabla J(\theta_t) - \nabla J(\theta_{t-1})) \\ &= \mathbb{E}_{b_t}[\mathbb{E}_{\tau_t^b}[B(\tau_t^b;\theta_t^b)(\theta_t - \theta_{t-1})|\mathcal{H}_{t-1}]] - (\nabla J(\theta_t) - \nabla J(\theta_{t-1})) \\ &\overset{(c)}{=} \mathbb{E}_{b_t}[\nabla^2 J(\theta_t^b)(\theta_t - \theta_{t-1})|\mathcal{H}_{t-1}] - (\nabla J(\theta_t) - \nabla J(\theta_{t-1})) \\ &\overset{(d)}{=} \int_0^1 \nabla^2 J(b\theta_t + (1-b)\theta_{t-1})(\theta_t - \theta_{t-1})db - (\nabla J(\theta_t) - \nabla J(\theta_{t-1})) = 0,\end{aligned} \tag{23}$$

(c) It is due to the fact that for a given θ_t^b, $B(\tau_t^b;\theta_t^b)$ is an unbiased estimation of $\nabla^2 J(\theta_t^b)$.
(d) The integral follows since $\theta_t^b = b_t\theta_t + (1-b_t)\theta_{t-1}$ with b_t uniformly distributed in $[0,1]$, and its value is $\nabla J(\theta_t) - \nabla J(\theta_{t-1})$.

Now, from (22), we have:

$$\begin{aligned}\mathbb{E}[\|\epsilon_{t-1}\|^2] &\overset{(a)}{\le} \frac{1}{\alpha_t}\left(\mathbb{E}[\|\epsilon_{t-1}\|^2] - \mathbb{E}[\|\epsilon_t\|^2]\right) + \frac{2}{\alpha_t}\mathbb{E}[\|W_t\|^2] + 2\alpha_t\sigma_g^2 \\ &\overset{(b)}{\le} \frac{1}{\alpha_t}\left(\mathbb{E}[\|\epsilon_{t-1}\|^2] - \mathbb{E}[\|\epsilon_t\|^2]\right) + 8\sigma_B^2\frac{\eta_{t-1}^2}{\alpha_t} + 2\alpha_t\sigma_g^2,\end{aligned} \tag{24}$$

(a) We use the bound in Lemma 1, i.e., $\mathbb{E}[\|U_t\|^2] \le \sigma_g^2$.
(b) It has been shown that $\|B(\tau;\theta)\| \le \sigma_B$ for any trajectory τ and $\theta \in \mathbb{R}^d$ (Shen et al. 2019). Therefore, $\|\nabla^2 J(\theta)\| = \|\mathbb{E}_\tau[B(\tau;\theta)]\| \le \sigma_B$. Hence, $\nabla J(\theta)$ is Lipschitz with constant σ_B and we have:

$$\begin{aligned}\|W_t\| &\le \|B(\tau_t^b;\theta_t^b)(\theta_t - \theta_{t-1})\| + \|\nabla J(\theta_t) - \nabla J(\theta_{t-1})\| \\ &\le \|B(\tau_t^b;\theta_t^b)\|\|\theta_t - \theta_{t-1}\| + \sigma_B\|\theta_t - \theta_{t-1}\| \\ &\le 2\sigma_B\|\theta_t - \theta_{t-1}\|,\end{aligned} \tag{25}$$

where the first inequality is due to the Lipschitzness of $\nabla J(\theta)$, and the second inequality results from the bound on $\|B(\tau_t^b;\theta_t^b)\|$.

Summing the both sides of (24) from $t = 1$ to $t = T$, we have:

$$\mathbb{E}\left[\sum_{t=1}^{T} \|\epsilon_{t-1}\|^2\right] \leq -\frac{\mathbb{E}[\|\epsilon_T\|^2]}{\alpha_T} + \frac{\mathbb{E}[\|\epsilon_0\|^2]}{\alpha_1} + \underbrace{\sum_{t=1}^{T-1}\left(\frac{1}{\alpha_{t+1}} - \frac{1}{\alpha_t}\right)\mathbb{E}[\|\epsilon_t\|^2]}_{\text{(I)}} + 8\sigma_B^2 \underbrace{\sum_{t=1}^{T} \frac{\eta_{t-1}^2}{\alpha_t}}_{\text{(II)}} + 2\sigma_g^2 \underbrace{\sum_{t=1}^{T} \alpha_t}_{\text{(III)}} \tag{26}$$

First note that $\mathbb{E}[\|\epsilon_0\|^2]/\alpha_1 \leq \sigma_g^2/\alpha_0$. Now, we bound the other terms in the right hand side of the above inequality:
(I): For the coefficient in the sum, we have: $1/\alpha_{t+1} - 1/\alpha_t = ((t+1)^{2/3} - t^{2/3})/\alpha_0 \leq 2t^{-1/3}/(3\alpha_0) \leq 2/(3\alpha_0)$ where we used the gradient inequality for the concave function $f(z) = z^{2/3}$. Therefore, this term can be bounded by: $(2/(3\alpha_0)) \sum_{t=1}^{T-1} \mathbb{E}[\|\epsilon_t\|^2]$.
(II): $\sum_{t=1}^{T} \eta_{t-1}^2/\alpha_t = \eta_0^2/\alpha_0(1 + \sum_{t=1}^{T-1}(t+1)^{2/3}/t^{4/3}) \leq \eta_0^2/\alpha_0(1 + 2^{2/3}\sum_{t=1}^{T-1} t^{-2/3}) \leq 6\eta_0^2 T^{1/3}/\alpha_0$.
(III): $\sum_{t=1}^{T} \alpha_t = \alpha_0 \sum_{t=1}^{T} t^{-2/3} \leq 3\alpha_0 T^{1/3}$.
Plugging these bounds in (26), we get:

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \|\epsilon_t\|^2\right] \leq CT^{1/3}, \tag{27}$$

where $C := 3\alpha_0((48\sigma_B^2\eta_0^2 + \sigma_g^2)/\alpha_0 + 6\alpha_0\sigma_g^2)/(3\alpha_0 - 2)$.
The previous inequality yields that:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\epsilon_t\|] \leq \frac{1}{T}\sum_{t=0}^{T-1} \sqrt{\mathbb{E}[\|\epsilon_t\|^2]} \leq \sqrt{\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\epsilon_t\|^2]} \leq \frac{\sqrt{C}}{T^{1/3}}, \tag{28}$$

where we used Jensen's inequality in the second inequality above.
Now, if we average from $t = 0$ to $t = T - 1$ in (33) in Lemma 2 and then take expectations, we have:

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla J(\theta_t)\|\right] &\leq \frac{8}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\epsilon_t\|] + \frac{3\sigma_B}{2T}\sum_{t=0}^{T-1} \eta_t + \frac{3}{T}\mathbb{E}\left[\sum_{t=0}^{T-1} \frac{J(\theta_{t+1}) - J(\theta_t)}{\eta_t}\right] \\
&\overset{(a)}{\leq} \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \mathbb{E}\left[\frac{3}{T}\sum_{t=0}^{T-1} \frac{J(\theta_{t+1}) - J(\theta_t)}{\eta_t}\right] \\
&\leq \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \mathbb{E}\left[\frac{3}{T}\left(\frac{J(\theta_T)}{\eta_{T-1}} - \frac{J(\theta_0)}{\eta_0} + \sum_{t=1}^{T-1} J(\theta_t)\left(\frac{1}{\eta_{t-1}} - \frac{1}{\eta_t}\right)\right)\right] \\
&\overset{(b)}{\leq} \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \mathbb{E}\left[\frac{3}{T}\left(\frac{C_J(T-1)^{2/3}}{\eta_0} + \frac{C_J}{\eta_0} + \sum_{t=1}^{T-1} |J(\theta_t)|\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right)\right] \\
&\leq \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \mathbb{E}\left[\frac{3}{T}\left(\frac{C_J(T-1)^{2/3}}{\eta_0} + \frac{C_J}{\eta_0} + \sum_{t=1}^{T-1} C_J\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right)\right] \\
&\overset{(c)}{\leq} \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \mathbb{E}\left[\frac{3}{T}\left(\frac{C_J(T-1)^{2/3}}{\eta_0} + \frac{C_J}{\eta_0} + \frac{C_J(T-1)^{2/3}}{\eta_0}\right)\right] \\
&\leq \frac{8\sqrt{C}}{T^{1/3}} + \frac{6\sigma_B\eta_0}{T^{2/3}} + \frac{9C_J}{\eta_0 T^{1/3}},
\end{aligned} \tag{29}$$

(a) We used the bound in (28) for the first term and $\sum_{t=0}^{T-1} \eta_t \leq 4\eta_0 T^{1/3}$.
(b) $|J(\theta)| = |\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]| = |\mathbb{E}_{\tau\sim\pi_\theta}[\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h)]| \leq \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{h=0}^{H-1} \gamma^h |r(s_h, a_h)|] \leq \mathbb{E}_{\tau\sim\pi_\theta}[R_0/(1-\gamma)] = R_0/(1-\gamma)$. Hence, we have: $|J(\theta)| \leq C_J$ for all $\theta \in \mathbb{R}^d$ where $C_J := R_0/(1-\gamma)$.
(c) Due to $\sum_{t=1}^{T-1} 1/\eta_t - 1/\eta_{t-1} = 1/\eta_{T-1} - 1/\eta_0 \leq (T-1)^{2/3}/\eta_0$.

B Proof of Remark 3

According to (22), we have:

$$\mathbb{E}[\|\epsilon_t\|^2] \leq (1-\alpha_t)\mathbb{E}[\|\epsilon_{t-1}\|^2] + 8\sigma_B^2\eta_{t-1}^2 + 2\alpha_t^2\sigma_g^2. \tag{30}$$

Let us define $Z_t := \mathbb{E}[\|\epsilon_t\|^2]$. Then, we can rewrite the above equation as follows:

$$Z_t \leq (1-\alpha_t)Z_{t-1} + \frac{C_Z}{t^{4/3}}, \tag{31}$$

where $C_z = 8 \times 2^{4/3}\eta_0^2\sigma_B^2 + 2\alpha_0^2\sigma_g^2$. Now, by induction, we will show that: $Z_t \leq C'_Z/t^{2/3}$ where $C'_Z = C_Z/(\alpha_0 - 2/3)$. It can be easily seen that for the base case Z_1, this statement holds. Now, for the induction step, suppose that $Z_{t-1} \leq C'_Z/(t-1)^{2/3}$ for some $t \geq 2$. Then, from above equation, we have:

$$\begin{aligned}
&\frac{C_Z}{t^{4/3}} \overset{(a)}{\leq} \frac{\alpha_0 C'_Z - 2C'_Z/3}{t^{2/3}(t-1)^{2/3}} \\
&\Longrightarrow \frac{2C'_z/3}{(t-1)t^{2/3}} \leq \frac{\alpha_0 C'_Z}{t^{2/3}(t-1)^{2/3}} - \frac{C_Z}{t^{4/3}} \\
&\overset{(b)}{\Longrightarrow} C'_Z\left(\frac{1}{(t-1)^{2/3}} - \frac{1}{t^{2/3}}\right) \leq \frac{\alpha_0 C'_Z}{t^{2/3}(t-1)^{2/3}} - \frac{C_Z}{t^{4/3}} \\
&\Longrightarrow \left(1 - \frac{\alpha_0}{t^{2/3}}\right)\frac{C'_Z}{(t-1)^{2/3}} + \frac{C_Z}{t^{4/3}} \leq \frac{C'_Z}{t^{2/3}} \\
&\overset{(c)}{\Longrightarrow} (1-\alpha_t)Z_{t-1} + \frac{C_Z}{t^{4/3}} \leq \frac{C'_Z}{t^{2/3}} \\
&\overset{(d)}{\Longrightarrow} Z_t \leq \frac{C'_Z}{t^{2/3}},
\end{aligned} \tag{32}$$

where (a) is due to definition of C'_Z, (b) is based on using gradient inequality for the concave function $f(z) = z^{2/3}$, i.e., $t^{2/3} - (t-1)^{2/3} \leq 2/3(t-1)^{-1/3}$, (c) is according to induction hypothesis, and (d) is due to (31).

C Supplemental Lemma

Lemma 2 *Suppose that θ_t's are generated by executing Algorithm 2. Let $\epsilon_t := v_t - \nabla J(\theta_t)$. Then, at any time t, we have:*

$$\|\nabla J(\theta_t)\| \leq 8\|\epsilon_t\| + \frac{3\sigma_B\eta_t}{2} + \frac{3}{\eta_t}(J(\theta_{t+1}) - J(\theta_t)). \tag{33}$$

From σ_B-smoothness of $J(\theta_t)$ (Shen et al. 2019), we have:

$$\begin{aligned}
J(\theta_{t+1}) - J(\theta_t) &\geq \langle \nabla J(\theta_t), \theta_{t+1} - \theta_t\rangle - \frac{\sigma_B}{2}\|\theta_{t+1} - \theta_t\|^2 \\
&= \eta_t\left\langle \nabla J(\theta_t), \frac{v_t}{\|v_t\|}\right\rangle - \frac{\sigma_B}{2}\eta_t^2.
\end{aligned} \tag{34}$$

Regarding the first term above, we consider two cases, whether $\|\nabla J(\theta_t)\| \geq 2\|\epsilon_t\|$ or not. For the former case, we have:

$$\begin{aligned}
\left\langle \nabla J(\theta_t), \frac{v_t}{\|v_t\|}\right\rangle &= \frac{\|\nabla J(\theta_t)\|^2 + \langle \nabla J(\theta_t), \epsilon_t\rangle}{\|\nabla J(\theta_t) + \epsilon_t\|} \\
&\geq \frac{\|\nabla J(\theta_t)\|^2}{2\|\nabla J(\theta_t) + \epsilon_t\|} \\
&\geq \frac{\|\nabla J(\theta_t)\|^2}{2(\|\nabla J(\theta_t)\| + \|\epsilon_t\|)} \\
&\geq \frac{\|\nabla J(\theta_t)\|}{3} \\
&\geq \frac{\|\nabla J(\theta_t)\|}{3} - \frac{8}{3}\|\epsilon_t\|,
\end{aligned} \tag{35}$$

where in the first inequality, we used the bound $\langle \nabla J(\theta_t), \epsilon_t\rangle \geq -\|\nabla J(\theta_t)\|\|\epsilon_t\| \geq -\|\nabla J(\theta_t)\|^2/2$. For the latter case,

$$\begin{aligned}
\left\langle \nabla J(\theta_t), \frac{v_t}{\|v_t\|}\right\rangle &\geq -\|\nabla J(\theta_t)\| \\
&= \frac{\|\nabla J(\theta_t)\|}{3} - \frac{4\|\nabla J(\theta_t)\|}{3} \\
&\geq \frac{\|\nabla J(\theta_t)\|}{3} - \frac{8\|\epsilon_t\|}{3}.
\end{aligned} \tag{36}$$

Plugging the bound on $\langle \nabla J(\theta_t), v_t/\|v_t\|\rangle$ in (34), we get:

$$\begin{aligned} J(\theta_{t+1}) - J(\theta_t) &\geq \eta_t \left(\frac{\|\nabla J(\theta_t)\|}{3} - \frac{8\|\epsilon_t\|}{3} \right) - \frac{\sigma_B}{2}\eta_t^2 \\ \implies \|\nabla J(\theta_t)\| &\leq 8\|\epsilon_t\| + \frac{3\sigma_B \eta_t}{2} + \frac{3}{\eta_t}(\nabla J(\theta_{t+1}) - \nabla J(\theta_t)). \end{aligned} \tag{37}$$

D Additional Experimental Results

Experimental Results on Acrobat and Cartpole Environments

We compared SHARP with PAGE-PG (Gargiani et al. 2022) on two simple environments of Acrobat and Cartpole.

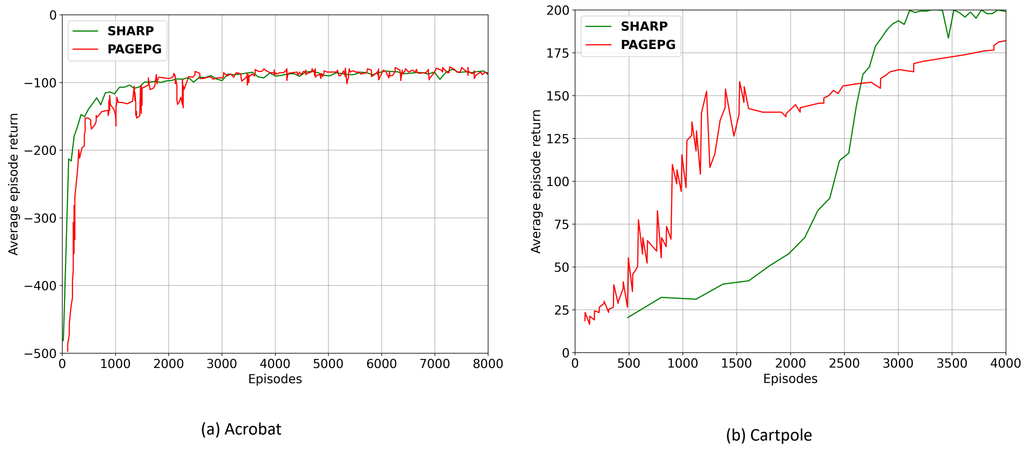


Figure 2: Comparison of SHARP with PAGE-PG (Gargiani et al. 2022) on two simple environments of Acrobat and Cartpole.

Acrobat environment: The Acrobot has two links. The two links are connected linearly to form a chain. The goal is to swing the free end of the linear chain to a given height. A reward of -1 is returned each time the goal is not achieved. An episode terminates whenever the target height is reached or 500 steps are elapsed. The state space is continuous with dimension 6 and the action space is discrete with 3 possible actions: applying a positive torque, or applying a negative torque, or do nothing.

As can be seen in Figure 2 (a), SHARP performs a little bit better than PAGE-PG on this environment.

Cartpole environment: The Cartpole comprises a pole attached by an un-actuated joint to a cart which moves along a friction-less track. The pendulum is placed upright at the beginning, and the goal is to balance the pole. A reward of $+1$ is returned for every step that the pole remains within 15 degrees from the upright position. An episode terminates whenever the pole is more than 15 degrees from upright position, or the cart is moved more than 2.4 units from its initial position. The state space is continuous with dimension 4 and the action space is discrete with 2 actions of pushing the cart left or right.

As can be seen in Figure 2 (b), SHARP outperforms PAGE-PG by achieving the average episode return of 200 after about 3000 episodes while PAGE-PG obtains the score of about 175 after 4000 episodes.

The effect of batch size on the performance

In Section 3, we provided a theoretical convergence guarantee for the proposed algorithm with a batch size equal to one for computing stochastic gradient and Hessian vector product. However, in practice, we can use a batch size of $O(1)$. Herein, we study the impact of batch size on the performance of the proposed algorithm empirically. More specifically, in Reacher environment, we executed SHARP with batch sizes of $|\mathcal{B}| = 1, 10, 20$. The average episode return versus system probes for different batch sizes is depicted in Figure 3; The impact of small batch size is negligible and the proposed algorithm still achieves good performance with small batch size.

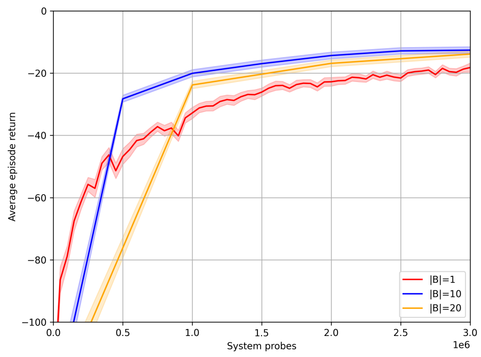


Figure 3: The performance of SHARP for different batch sizes in Reacher environment.

E Details of Baselines in Experiments

We used the default implementation of linear feature baseline and Gaussian MLP baseline from Garage library. The employed linear feature baseline is a linear regression model which takes observations for each trajectory and extracts new features such as different powers of their lengths from the observations. These extracted features are concatenated to the observations and used to fit the parameters of the regression with least square loss function. The Gaussian MLP baseline is a two layer neural network using 32 neurons per hidden layer with $\mathtt{tanh}(.)$ activation function.

In the experiments, we used linear baseline for all the environments and methods, except for HalfCheetah environment in our algorithm, which we used Gaussian MLP baseline.