A Cubic-regularized Policy Newton Algorithm for Reinforcement Learning

Mizhaan Prajit Maniyar[1] **Prashanth L.A.**[1] **Akash Mondal**[2] **Shalabh Bhatnagar**[2]

[1] Indian Institute of Technology Madras

[2] Indian Institute of Science

Abstract

We consider the problem of control in the setting of reinforcement learning (RL), where model information is not available. Policy gradient algorithms are a popular solution approach for this problem and are usually shown to converge to a stationary point of the value function. In this paper, we propose two policy Newton algorithms that incorporate cubic regularization. Both algorithms employ the likelihood ratio method to form estimates of the gradient and Hessian of the value function using sample trajectories. The first algorithm requires an exact solution of the cubic regularized problem in each iteration, while the second algorithm employs an efficient gradient descent-based approximation to the cubic regularized problem. We establish convergence of our proposed algorithms to a second-order stationary point (SOSP) of the value function, which results in the avoidance of traps in the form of saddle points. In particular, the sample complexity of our algorithms to find an ϵ-SOSP is $O(\epsilon^{-3.5})$, which is an improvement over the state-of-the-art sample complexity of $O(\epsilon^{-4.5})$.

1 Introduction

Markov decision processes (MDPs) provide a framework for analyzing sequential decision making problems under uncertainty. The aim here is to find a policy that optimizes a given performance objective, e.g., the discounted cumulative reward or cost. A direct solution approach for MDPs would require knowledge of the underlying transition dynamics. In practical settings, such information is seldom available, and one usually resorts to reinforcement learning (RL) algorithms [22] that find optimal policies using sample trajectories.

Classical RL algorithms based on lookup table representations suffer from the curse of dimensionality associated with large state spaces. A popular approach to overcoming this problem is through considering a prescribed parametric representation of policies and searching for the best policy within this class using a stochastic gradient (SG) algorithm. Policy gradient (PG) algorithms adopt this approach, and update the policy parameters using an estimate of the gradient of the expected sum of costs (or the value function) with respect to those parameters. The simplest such class of algorithms are trajectory-based methods that consider full Monte-Carlo returns for estimating the performance gradient such as REINFORCE [30]. These algorithms work by increasing the probabilities of actions that lead to higher returns and thereby reduce the probabilities of actions that lead to lower returns using data sampled through interactions with the environment. The gradient of the value function for the given policy is estimated using the aforementioned Monte-Carlo returns.

Incremental update algorithms such as actor-critic have been proposed as alternatives to trajectory-based methods [23]. A common approach here to estimate the value function for any given policy is to introduce a critic recursion that does this estimation for any given parameter update and which in turn is updated using a 'slower' actor recursion. Thus, actor-critic algorithms typically require two timescale recursions even though they are incremental-update procedures. Trajectory-based methods like REINFORCE involve a single-timescale in their update rule but typically suffer (like SGD) from high variance which leads to a high sample complexity for the algorithm. Many works in recent times that involve trajectory-based methods have tried to address this high variance problem. For example, Trust Region Policy Optimization (TRPO) [19] can improve training stability by constraining the step length to be within a certain "trust region" and thereby obtain better sampling efficiency than the vanilla policy gradient approach. TRPO optimizes the loss function via the Kullback-Leibler (KL) divergence. In addition, Proximal Policy Optimization (PPO) [20] is an improvement over TRPO as it employs objective function clipping and/or a

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

Table 1: Comparison of the sample complexities for finding approximate first-order and second-order stationary points (ϵ-FOSP and ϵ-SOSP, respectively). For precise definitions of these points, see Definitions 1 and 2 in Section 4. A ✓, e.g., under ϵ-FOSP, indicates that an algorithm is shown to converge to an ϵ-FOSP, while ✗, e.g., under ϵ-SOSP, indicates that an algorithm is not provably convergent to an ϵ-SOSP.

Algorithm	**Sample complexity**	**ϵ-FOSP**	**ϵ-SOSP**
REINFORCE	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	✓	✗
[21]	$\mathcal{O}\left(\frac{1}{\epsilon^3}\right)$	✓	✗
[31]	$\mathcal{O}\left(\frac{1}{\epsilon^{4.5}}\right)$	✓	✓
Our work	$\mathcal{O}\left(\frac{1}{\epsilon^{3.5}}\right)$	✓	✓

penalty on the KL divergence with the trust region update that is compatible with SGD and simplifies the algorithm by eliminating the KL divergence. Both TRPO and PPO aim at solving constrained optimization problems involving inequality constraints.

Policy gradient algorithms and their analyses have received a lot of research attention recently, cf. [1, 23, 17, 28, 32, 13]. Since the value function is usually non-convex, the analysis of policy gradient algorithms in general establishes convergence to first order stationary points (FOSP) in the long run or to approximate stationary points in the non-asymptotic regime. These points also include unstable equilibria or traps such as local maxima and saddle points. In the optimization literature, approaches to avoid traps either add extraneous noise in the gradient step [11], or show that the gradient estimates have omni-directional noise [14], or use second-order information in a stochastic Newton algorithm [16, 18].

In the context of RL, avoidance of traps has received very little attention and most of the previous works analyzing policy gradient algorithms show convergence to stationary points only. A few exceptions are [12, 32]. In [12], the authors explore addition of extraneous isotropic noise to avoid traps in the context of a policy gradient algorithm. However, their algorithm avoids traps under an additional condition that is hard to verify in typical RL settings. The latter condition requires the extraneous noise to dominate the martingale difference noise inherent to a policy gradient update. In [32], the authors explore a different approach to avoid traps by using larger stepsize periodically.

We propose a policy Newton algorithm that incorporates second-order information along with cubic regularization in the spirit of [16]. Our algorithm has an improved sample complexity from incorporating Hessian information and this also leads us to avoid saddle points and converge to an ϵ-SOSP.

We summarize our contributions below.

(a) Cubic-regularized policy Newton: For solving a finite horizon MDP, we propose a cubic regularized policy Newton (CR-PN) method that avoids saddle points and converges to an approximate second-order stationary point (SOSP), where the approximation is quantified by a parameter $\epsilon > 0$. Such a point is referred to as ϵ-SOSP. In this algorithm, we derive an estimate of the Hessian of the value function from sample episodes using Stein's identity.

(b) Approximate cubic-regularized policy Newton (ACR-PN): We propose the ACR-PN algorithm that finds an ϵ-SOSP with high probability, while employing a computationally efficient solver for the cubic model in CR-PN. The latter approximate solver, borrowed from [25], avoids calculating the Hessian and instead uses the Hessian-vector product for computational simplicity. Though CR-PN provides an exact solution, but in a practical scenario, the computation of Hessian is very expensive due to the presence of a complex cost objective. In such cases, ACR-PN is very efficient.

(c) Non-asymptotic convergence: We derive non-asymptotic bounds that quantify the convergence rate to ϵ-SOSP for both algorithms. First, we prove that under assumptions common to the analysis of policy gradient algorithms, CR-PN converges to an ϵ-SOSP within $O(\epsilon^{-1.5})$ iterations and with a sample complexity $O(\epsilon^{-3.5})$. These bounds hold both in expectation and with high probability. Second, we establish that ACR-PN converges to an ϵ-SOSP with high probability within a number of iterations and with a sample complexity that is comparable to that of CR-PN. These rates significantly improve upon the sample complexity of existing algorithms in the literature. For instance, a REINFORCE-type algorithm has $O(\epsilon^{-4})$ sample complexity, and the Hessian-aided policy gradient method [31] has $O(\epsilon^{-4.5})$ complexity. Table 1 provides a summary.

(d) Bounds for gradient and Hessian estimation: To establish the non-asymptotic bounds to ϵ-SOSP for our proposed algorithms, we require bounds on the gradient/Hessian estimation error of every-visit Monte Carlo sample averages. We derive error bounds that hold in expectation as well as with high-probability for the Monte Carlo gradient and Hessian estimates. These bounds may be of independent interest, for instance, in the analysis of other second-order policy Newton algorithms that do not incorporate a cubic regularizer.

Related work. Policy gradient algorithms and their analyses have received a lot of research attention, cf. [1, 23, 6, 17, 28, 32]. In [9] the authors propose policy Newton algorithms for solving an MDP in a setting where the model (or the transition dynamics) is known. In [21], the authors propose a policy gradient algorithm that incorporates second-order information and establish convergence to a first-order stationary point. In contrast, our proposed algorithms, which also use second-order information, are shown to converge to a second-order stationary point. In

[7], an actor-critic algorithm for constrained MDPs is developed in the average cost setting for the full-state case. In [4], a function approximation based actor-critic algorithm for average cost constrained MDPs is developed that does policy gradient on the Lagrangian along the actor update and involves a temporal-difference TD(λ) critic. In [3], the discounted cost version of the algorithm in [4] is developed except that a data-driven gradient estimation procedure [5] is adopted.

A preliminary version of this paper was part of the first author's master's thesis before a concurrent work, see [29]. In the aforementioned reference, the authors propose an approximate cubic-regularized Newton algorithm and provide sample complexity bounds. In contrast, we analyze both the exact and the approximate cubic-regularized Newton algorithm variants. For the former, we derive bounds using a proof technique that is radically different from that employed in [29]. Moreover, our analysis for the approximate Newton variant is a lot simpler, as we invoke the bounds in [25] after verifying the necessary assumptions, whereas the analysis in [29] mimics the proof in [25].

The rest of the paper is organized as follows: Section 2 describes the problem formulation and the policy gradient framework. Section 3 presents the cubic-regularized policy Newton algorithm, while Section 4 established non-asymptotic bounds for convergence to an approximate second-order stationary point of the objective. Section 5 describes the variant of the cubic-regularized policy Newton algorithm, which incorporates an approximate solution scheme for the cubic-regularized problem. The detailed proofs of convergence are provided in the supplementary material. Finally, Section 7 provides the concluding remarks.

2 Policy gradient framework

A Markov decision process (MDP) is a tuple of the form $(\mathcal{S}, \mathcal{A}, P, c, \gamma, \rho)$ where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $P(s'|s,a)$ represents the underlying transition probability function that governs the state evolution from s to s' under a given action $a \in \mathcal{A}$ of the MDP agent; $c(s,a)$ denotes the single-stage cost obtained by the agent in state s on taking action a; γ is the discount factor; and $\rho(s_0)$ is the distribution of the starting state s_0. The actions are chosen according to a probability distribution $\pi(a_h|s_h)$, $a_h \in \mathcal{A}$, which is conditioned over the current state. We shall call π as the policy used by the agent to select actions. For simplicity, we assume all actions are feasible in every state. We consider a finite horizon discounted setting. Here we denote the trajectory of states and actions until termination at instant H as $\tau := (s_0, a_0, \ldots, a_{H-1}, s_H)$, where $s_0 \sim \rho(s_0)$ and H is the trajectory horizon or episode length. The probability of trajectory τ following a policy π is given by

$$p(\tau;\pi) := \left(\prod_{h=0}^{H-1} P(s_{h+1}|s_h,a_h)\pi(a_h|s_h)\right)\rho(s_0). \quad (1)$$

We denote the discounted cumulative cost for a trajectory τ as $\mathcal{G}(\tau) := \sum_{h=0}^{H-1} \gamma^{h-1} c(s_h,a_h)$, where $\gamma \leq 1$ is the discount factor. Our objective is to minimize over π the *expected* discounted cumulative cost given by

$$J(\pi) := \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\mathcal{G}(\tau)\right] = \mathbb{E}_{\tau\sim p(\tau;\pi)}\left[\sum_{h=0}^{H-1}\gamma^{h-1}c(s_h,a_h)\right]. \quad (2)$$

We assume that the policy is parameterized by a vector $\theta \in \mathbb{R}^d$ and use the notation π_θ as a shorthand for the distribution $\pi(a_h|s_h;\theta)$. Also, by an abuse of notation, we denote $p(\tau;\theta) = p(\tau;\pi_\theta)$ and $J(\theta) = J(\pi)$ as they are both conditioned on the same information. For simplicity, we assume here that the terminal cost is 0.

Our aim is to find a parameter

$$\theta^* \in \operatorname*{argmin}_{\theta\in\mathbb{R}^d} J(\theta). \quad (3)$$

The gradient $\nabla J(\theta)$ of the expected cost $J(\theta)$ can be written as

$$\nabla J(\theta) = \sum_{h=0}^{H-1}\sum_{\tau_h}\gamma^{h-1}c(s_h,a_h)\nabla p(\tau_h;\theta)\,.$$

In this work, our focus is on solving (3) by employing a Newton-like second-order algorithm. For this purpose, we require an expression for the gradient as well as Hessian of the objective. Such an expression has been derived earlier in [9, 21].

For deriving the policy gradient and Hessian expressions, we require assumptions on the regularity of the MDP and the smoothness of our parameterized policy π_θ. These assumptions are specified below, with $\|\cdot\|$ denoting the l_2 norm for vectors and the operator norm for matrices.

(A1) (Bounded costs)**.** *The absolute value of the cost function of the MDP is bounded, i.e., $\exists K \in (0,\infty)$ such that*

$$|c(s,a)| \leq K, \qquad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

(A2) (Parameterization regularity)**.** *For any choice of the parameter θ, any state-action pair (s,a), there exist constants $0 < G, L_1 < \infty$ such that*

$$\|\nabla \log\pi(a|s;\theta)\| \leq G \quad \textit{and} \quad \left\|\nabla^2 \log\pi(a|s;\theta)\right\| \leq L_1.$$

(A3) (Lipschitz Hessian)**.** *For any pair of parameters (θ_1,θ_2), and any state-action pair (s,a), there exists a constant $L_2 > 0$ such that*

$$\left\|\nabla^2 \log\pi(a|s;\theta_1) - \nabla^2\log\pi(a|s;\theta_2)\right\| \leq L_2\|\theta_1-\theta_2\|\,.$$

Note that (A1) and (A2) are standard in the literature on policy gradient and actor-critic algorithms as shown in [21], while (A3) is common to the analysis of second-order policy search algorithms, cf. [32]. For a parameterized Boltzmann policy with a linear policy class, the assumptions (A2) and (A3) can be easily shown to hold. Further, if the policy parameter is restricted to a compact set, then by continuity of the gradient and the Hessian of $\log \pi$, one can infer (A2).

We now present the policy gradient and Hessian theorem.

Theorem 1 (***Policy gradient and Hessian theorem***). *Let*

$$\Psi_i(\tau) := \sum_{h=i}^{H-1} \gamma^{h-1} c(s_h, a_h) \text{ and}$$
$$\Phi(\theta;\tau) := \sum_{i=0}^{H-1} \Psi_i(\tau) \log \pi(a_i|s_i;\theta).$$

Then, under assumptions (A1)-(A3), the gradient $\nabla J(\theta)$ and the Hessian $\nabla^2 J(\theta)$ of the objective (2) *are given by*

$$\nabla J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)} \left[\nabla \Phi(\theta;\tau)\right],$$
$$\nabla^2 J(\theta) = \mathbb{E}_{\tau\sim p(\tau;\theta)} \left[\nabla \Phi(\theta;\tau)\nabla^\top \log p(\tau;\theta) + \nabla^2 \Phi(\theta;\tau)\right].$$

Proof. See [21]. For the sake of completeness, we provide the proof in Appendix B. □

The result below establishes that the expected cost objective is smooth, and its gradient, as well as Hessian, are well defined and Lipschitz continuous.

Proposition 1. *Under (A1)–(A3), for any $\theta_1, \theta_2 \in \mathbb{R}^d$, we have*

$$|J(\theta_1) - J(\theta_2)| \le M_{\mathcal{H}} \|\theta_1 - \theta_2\|,$$
$$\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \le G_{\mathcal{H}} \|\theta_1 - \theta_2\|, \text{ and}$$
$$\left\|\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2)\right\| \le L_{\mathcal{H}} \|\theta_1 - \theta_2\|, \tag{4}$$

where $M_{\mathcal{H}} := KGH^3, G_{\mathcal{H}} := H^3G^2K + L_1KH^2$ and $L_{\mathcal{H}} := H^4G^3K + 3H^3GL_1K + L_2KH^2$.

Proof. See Lemmas 4–5 and their proofs in Appendix A. □

3 Cubic-regularized policy Newton (CR-PN) algorithm

A stochastic gradient algorithm to find a local optimum of the objective function in the problem (3) would perform an incremental update of the policy parameter as follows:

$$\theta_{k+1} = \theta_k - \eta M(\theta_k) \nabla J(\theta_k),$$

where $\eta \in \mathbb{R}^+$ is the step-size and $M(\theta)$ is a preconditioning matrix that could depend on the policy parameter θ.

If J is smooth and $M(\theta)$ is positive-definite, then the policy parameter update ensures a decrease in the objective, viz., the total expected cost for sufficiently small η. Note that if $M(\theta)$ is the identity matrix, then the update rule above corresponds to a gradient step, while $M(\theta) = \nabla^2 J(\theta)^{-1}$ would result in a Newton step.

In a typical RL setting, it is not feasible to find the exact gradient or Hessian of the objective function since the underlying transition dynamics of the environment is unknown. Instead, one has to form sample-based estimates of these quantities. Now, if we use an estimate of the Hessian in place of the preconditioning matrix, we cannot assure a stable descent as the Hessian estimate may not be positive-definite at each iterate as required. This makes the classical Newton update a bad candidate for our policy search algorithm. [15] motivates an algorithm called the cubic-regularized Newton method in a deterministic setting which tackles these issues and more. They show that the standard Newton step ($\eta = 1$) can alternatively be presented as follows:

$$\theta_{k+1} = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{ \langle \nabla J(\theta_k), \theta - \theta_k \rangle + \frac{1}{2} \left\langle \nabla^2 J(\theta_k)(\theta - \theta_k), \theta - \theta_k \right\rangle \right\}.$$

The cubic regularized Newton step adds a cubic term to the auxiliary function in the following manner:

$$\theta_{k+1} = \operatorname*{argmin}_{\theta\in\mathbb{R}^d} \left\{ \langle \nabla J(\theta_k), \theta - \theta_k \rangle + \frac{1}{2} \left\langle \nabla^2 J(\theta_k)(\theta - \theta_k), \theta - \theta_k \right\rangle + \frac{\alpha}{6} \|\theta - \theta_k\|^3 \right\},$$

where $\alpha \in \mathbb{R}^+$ is the regularization parameter. In a general stochastic optimization setting, using the cubic-regularized Newton step, the authors in [2] establish convergence to local minima. We adopt a similar approach in a RL setting to propose/analyze a cubic-regularized policy Newton method with gradient/Hessian estimates derived using the result in Theorem 1. Algorithm 1 presents the pseudo-code for the cubic-regularized policy Newton algorithm with a gradient and Hessian estimation scheme that is described below.

From Theorem 1, the policy gradient and Hessian can be seen as expectations of g and $\mathcal{H}$ defined below.

$$g(\theta;\tau) := \nabla \Phi(\theta;\tau),$$
$$\mathcal{H}(\theta;\tau) := \nabla \Phi(\theta;\tau)\nabla^\top \log p(\tau;\theta) + \nabla^2 \Phi(\theta;\tau). \tag{5}$$

The above estimates are calculated by the information obtained from a given trajectory τ and policy parameter θ. We

simulate multiple trajectories and calculate these estimates for each of them and then take their averages to obtain the final such estimates.

Algorithm 1: Cubic-regularized policy Newton (CR-PN)

Input : Initial parameter $\theta_0 \in \mathbb{R}^d$, a non-negative sequence $\{\alpha_k\}$, positive integer sequences $\{m_k\}$ and $\{b_k\}$, and an iteration limit $N \geq 1$.

for $k = 1, \ldots, N$ **do**

`/* Monte Carlo simulation */`

Simulate $\max\{m_k, b_k\}$ number of trajectories according to θ_{k-1}, randomly pick m_k trajectories for set $\mathcal{T}_m$ and b_k trajectories for set $\mathcal{T}_b$;

`/* Gradient estimation */`

$$\bar{g}_k = \frac{1}{m_k} \sum_{\tau \in \mathcal{T}_m} \sum_{h=0}^{H-1} \Psi_h(\tau) \nabla \log \pi(a_h|s_h; \theta_{k-1})$$

where the state-action pairs (s_h, a_h) belong to the respective trajectories τ;

`/* Hessian estimation */`

$$\bar{\mathcal{H}}_k = \frac{1}{b_k} \sum_{\tau \in \mathcal{T}_b} \Big(\sum_{h=0}^{H-1} \Psi_h(\tau) \nabla \log \pi(a_h|s_h; \theta_{k-1}) \times \sum_{h'=0}^{H-1} \nabla^\top \log \pi(a_{h'}|s_{h'}; \theta_{k-1}) \Big) + \frac{1}{b_k} \sum_{\tau \in \mathcal{T}_b} \sum_{h=0}^{H-1} \Psi_h(\tau) \nabla^2 \log \pi(a_h|s_h; \theta_{k-1});$$

`/* Policy update (cubic regularized Newton step) */`

Compute

$$\theta_k = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left\{ \tilde{J}^k(\theta) \equiv \tilde{J}(\theta, \theta_{k-1}, \bar{\mathcal{H}}_k, \bar{g}_k, \alpha_k) \right\},$$

where $\tilde{J}(x, y, \mathcal{H}, g, \alpha) =$

$$\langle g, x - y \rangle + \frac{1}{2} \langle \mathcal{H}(x - y), x - y \rangle + \frac{\alpha}{6} \|x - y\|^3. \tag{6}$$

end for

Output: Policy θ_N

The result that we state below provides error bounds on the single trajectory-based gradient and Hessian estimates defined in (5). These bounds will be used subsequently to establish convergence to a local minimum of the objective J.

Lemma 1. *Let $g(\theta; \tau), \mathcal{H}(\theta; \tau)$ be the gradient and Hessian estimates formed using* (5)*. Then, under (A1) (A2) and (A3) for any parameter θ and trajectory τ, we have almost surely*

$$\|g(\theta; \tau) - \nabla J(\theta)\| \leq M_1 \text{ and}$$
$$\left\|\mathcal{H}(\theta; \tau) - \nabla^2 J(\theta)\right\| \leq M_2,$$

where $M_1 := GKH^2(H+1)$, and $M_2 := 2G_{\mathcal{H}}$. The constants K, G are given in Assumptions (A1) and (A3), respectively, H is the horizon, and $G_{\mathcal{H}}$ is defined in Proposition 1.

Proof. See Appendix D. □

4 Main results

In this section, we first define a first and second-order stationary point of the expected cost objective. Subsequently, we prove that Algorithm 1 converges to a second-order stationary point.

At a first-order stationary point (FOSP), say $\bar{\theta}$, the gradient vanishes, i.e., $\nabla J(\bar{\theta}) = 0$. An ϵ-approximation to FOSP is a point $\bar{\theta}$ that satisfies $\left\|\nabla J(\bar{\theta})\right\| \leq \epsilon$. The definition below extends the notion of ϵ-FOSP to a stochastic optimization setting, and is standard in optimization literature, cf. [10].

Definition 1 (**ϵ-first-order stationary point**). *Fix $\epsilon > 0$. Let θ_R be the output of a stochastic iterative algorithm for solving* (3)*. Then, θ_R is said to be an ϵ-first-order stationary point (ϵ-FOSP) of J if*

$$\mathbb{E}\left[\|\nabla J(\theta_R)\|\right] \leq \epsilon,$$

where the expectation is over the randomness in the algorithm considered.

A FOSP could potentially be a saddle point. In order to avoid such points and find local optima of J, we need information with regard to the curvature of the underlying objective. The notion of second-order stationary point (SOSP) formalizes this notion and aids in escaping saddle points.

At a second-order stationary point (SOSP), say $\bar{\theta}$, we have $\nabla J(\bar{\theta}) = 0$ and $\lambda_{\min}\left(\nabla^2 J(\bar{\theta})\right) \geq 0$, where $\lambda_{\min}(A)$ denotes the minimum eigenvalue of a matrix A. If the objective J satisfies a strict saddle condition, i.e., there are no points where the aforementioned minimum eigenvalue is zero, then SOSPs coincide with local minima [16]. For non-asymptotic analysis, an ϵ-version of SOSP is defined below.

Definition 2 (**ϵ-second-order stationary point**). *Fix $\epsilon > 0$. Let θ_R be the output of a stochastic iterative algorithm for solving* (3)*. Then, for some $\rho > 0$, θ_R is said to be an ϵ-SOSP in expectation if*

$$\max\left\{\sqrt{\mathbb{E}\left[\|\nabla J(\theta_R)\|\right]}, \frac{-1}{\sqrt{\rho}} \mathbb{E}\left[\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right]\right\} \leq \sqrt{\epsilon}.$$

Furthermore, θ_R is said to be an ϵ-SOSP with high probability if the following bound holds with probability $1 - \delta$ for any $\delta \in (0, 1)$:

$$\max\left\{\sqrt{\|\nabla J(\theta_R)\|}, \frac{-1}{\sqrt{\rho}} \lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right\} \leq \sqrt{\epsilon},$$

Such definitions are standard in second-order optimization literature, and an algorithm that outputs an ϵ-SOSP approximates the local minimum better than one that outputs ϵ-FOSP, cf. [2, 25].

Before stating the main result that establishes convergence of Algorithm 1 to an ϵ-SOSP, we state a useful lemma that bounds the error in the gradient estimate $\bar{g}_k$ and the Hessian estimate $\bar{\mathcal{H}}_k$ as a function of number of trajectories m_k and b_k, respectively.

Lemma 2. *Let $\bar{g}_k$ and $\bar{\mathcal{H}}_k$ be computed as in Algorithm 1, and assume $b_k \geq 4(1+2\log 2d)$. Then, we have*

$$\mathbb{E}\left[\|\bar{g}_k - \nabla J(\theta_{k-1})\|^2\right] \leq \frac{G_g^2}{m_k}, \text{ and}$$
$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^3\right] \leq \frac{4\sqrt{15(1+2\log 2d)}dG_{\mathcal{H}}^3}{b_k^{\frac{3}{2}}}.$$

Proof. The first claim is easy to prove, using the definition of $\bar{g}_k$ in conjunction with algebraic manipulations. For the second claim concerning $\bar{\mathcal{H}}_k$, we follow the technique from [2]. In particular, we use a bound on the expected norm of an independent sum of random matrices from [27, Theorem 1] in conjunction with Rosenthal's inequality (see Lemma 16 in Appendix G). The reader is referred to Appendix C for the detailed proof. □

The main result that establishes convergence in expectation to an ϵ-SOSP is given below.

Theorem 2 (*Bound in expectation*). *Let $\{\theta_1, \ldots, \theta_N\}$ be computed by Algorithm 1 with the following parameters:*

$$\alpha_k = 3L_{\mathcal{H}}, \quad N = \frac{12\sqrt{L_{\mathcal{H}}}(J(\theta_0) - J^*)}{\epsilon^{\frac{3}{2}}}, \tag{7}$$
$$m_k = \frac{25G_g^2}{4\epsilon^2}, \quad b_k = \frac{36\sqrt[3]{30(1+2\log 2d)}d^{\frac{2}{3}}G_{\mathcal{H}}^2}{L_{\mathcal{H}}\epsilon}.$$

Let θ_R be picked uniformly at random from $\{\theta_1, \ldots, \theta_N\}$. Then, under (A1) (A2) and (A3), we have

$$5\sqrt{\epsilon} \geq \max\left\{\sqrt{\mathbb{E}\left[\|\nabla J(\theta_R)\|\right]}, \frac{-5}{6\sqrt{L_{\mathcal{H}}}}\mathbb{E}\left[\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right]\right\}. \tag{8}$$

where $G_{\mathcal{H}}$ and $L_{\mathcal{H}}$ are defined as in (4)*.*

Proof. See Appendix C. □

A few remarks are in order.

Remark 1. *Since $(J(\theta_0) - J^*)$ is unknown in a typical RL setting, to aid practical implementations, one could choose $N = \frac{24KH\sqrt{L_{\mathcal{H}}}}{\epsilon^{\frac{3}{2}}}$, and the bound in* (8) *would continue to hold, since $(J(\theta_0) - J^*) \leq 2KH$.*

Remark 2. *As a consequence of Theorem 2, to obtain an ϵ-SOSP of the problem, the total number of trajectories required to compute the gradient and the Hessian are bounded by $O\left(\frac{1}{\epsilon^{\frac{7}{2}}}\right)$ and $O\left(\frac{d^{\frac{2}{3}}}{\epsilon^{\frac{5}{2}}}\right)$, respectively. This is of a higher order in contrast to the HAPG algorithm proposed by [21], which requires $O\left(\frac{1}{\epsilon^3}\right)$ number of trajectories. However, the total number of time steps that it requires for our algorithm to converge is $O\left(\frac{1}{\epsilon^{1.5}}\right)$ versus the $O\left(\frac{1}{\epsilon^2}\right)$ required for HAPG. Furthermore, our algorithm ensures convergence to an ϵ-SOSP thereby avoiding saddle points, while HAPG is shown to converge to an ϵ-FOSP, which could potentially be a trap (e.g. saddle point).*

Before establishing convergence of Algorithm 1 to an ϵ-SOSP with high probability, we state a high-probability counterpart of Lemma 2.

Lemma 3. *Let $m_k = \max\left(\frac{M_1}{t}, \frac{M_1^2}{t^2}\right)\frac{8}{3}\log\frac{2d}{\delta'}, b_k = \max\left(\frac{M_2}{\sqrt{t_1}}, \frac{M_2^2}{t_1}\right)\frac{8}{3}\log\frac{2d}{\delta'}$ any positive constants and $\delta' \in (0,1)$. Then, with probability $1-\delta'$ we have*

$$\|\bar{g}_k - \nabla J(\theta_k)\|^2 \leq t^2, \text{ and } \left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta)\right\|^3 \leq t_1^{\frac{3}{2}}.$$

Proof. The main claim follows by an application of the matrix variant of Bernstein's concentration inequality, after verifying the necessary assumptions. See Appendix E for the detailed proof. □

The main result that shows Algorithm 1 converges to an ϵ-SOSP with high probability is given below.

Theorem 3 (*High-probability bound*). *Suppose (A1) (A2) and (A3) hold. Let $\{\theta_1, \ldots, \theta_N\}$ be computed by Algorithm 1 with α_k and N as in Theorem 2 and batch sizes m_k, b_k set as follows:*

$$m_k = \max\left(\frac{5M_1}{2\epsilon}, \frac{25M_1^2}{4\epsilon^2}\right)\frac{8}{3}\log\frac{2d}{\delta'},$$
$$b_k = \max\left(\frac{12M_2}{\sqrt{\epsilon}}, \frac{144M_2^2}{\epsilon}\right)\frac{8}{3L_{\mathcal{H}}}\log\frac{2d}{\delta'}, \tag{9}$$

where $\delta' \in (0,1)$. Let θ_R be picked uniformly at random from $\{\theta_1, \ldots, \theta_N\}$. Then, with probability $1-2\delta'$, we have

$$5\sqrt{\epsilon} \geq \max\left\{\sqrt{\|\nabla J(\theta_R)\|}, \frac{-5}{6\sqrt{L_{\mathcal{H}}}}\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)\right\}, \tag{10}$$

where $G_{\mathcal{H}}$ and $L_{\mathcal{H}}$ are defined as in (4)*.*

Proof. Follows in a similar manner as the proof of Theorem 2, while using the high-probability bounds for gradient and Hessian estimation in Lemma 3 in place of the corresponding bounds in expectation (see Lemma 2 above). The reader is referred to Appendix E for a detailed proof. □

From the bound above, it is apparent that Algorithm 1 will output an ϵ-SOSP with probability at least $1-2\delta'$ within $O\left(\frac{1}{\epsilon^{1.5}}\right)$ number of iterations. Further, as in the case of the expectation bound in Theorem 2, the total number of trajectories required to estimate the gradient and Hessian are bounded by

$$\sum_{k=1}^{N} m_k = O\left(\frac{1}{\epsilon^{\frac{7}{2}}}\right), \text{ and } \sum_{k=1}^{N} b_k = O\left(\frac{1}{\epsilon^{\frac{5}{2}}}\right),$$

respectively, when ϵ is small.

5 Approximate cubic-regularized policy Newton (ACR-PN) algorithm

The cubic regularized policy Newton algorithm, which is described in Algorithm 1, requires an exact solution to the following optimization problem in each iteration:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmin}} \left\{ \langle \bar{g}_k, \theta - \theta_k \rangle + \frac{1}{2} \left\langle \bar{\mathcal{H}}_k(\theta - \theta_k), \theta - \theta_k \right\rangle + \frac{\alpha}{6} \|\theta - \theta_k\|^3 \right\}. \tag{11}$$

In practice, it may not always be possible to obtain an exact solution to the problem above, while one can perform a few gradient descent steps to arrive at an approximate solution that may be 'good enough'. Such an approach has been explored in a general stochastic optimization context in [25].

We use the algorithm in [25] as a black-box to arrive at a bound for a variant of Algorithm 1 that solves the cubic-regularized problem approximately. In other words, this algorithm is an instantiation of the template from [25] with gradient and Hessian estimates along the lines of those employed in Algorithm 1. We present the pseudo-code of this approximate variant in Algorithm 2 in the supplementary material.

The ACR-PN algorithm uses the gradient and Hessian estimates to solve the cubic-regularized problem (11) using a gradient descent-type algorithm. In particular, the "Cubic-Subsolver" routine returns the parameter change Δ, which is used to update the policy parameter θ_k. If the corresponding change in $\bar{J}_k(\theta)$, i.e., $\delta_J := \bar{J}_k(\theta_k + \Delta) - \bar{J}_k(\theta_k)$ satisfies a certain stopping criterion, then ACR-PN calls the "Cubic-Finalsolver" subroutine to perform several gradient descent steps so that the cubic-regularized problem (11) is solved accurately. As an aside, we note that the ACR-PN uses the Hessian estimate only through Hessian-vector products. The reader is referred to [25] for the details of the two subroutines mentioned above.

We now turn to establishing convergence of the ACR-PN algorithm to an ϵ-SOSP with high probability. The main claim of Theorem 4 would follow from Theorem 1 of [25], provided we verify Assumptions 1 and 2 from the aforementioned reference. For the sake of completeness, we state these assumptions as B1 and B2 below. Note that these assumptions are equivalent to Proposition 1 and Lemma 1, respectively.

(B1): The function J satisfies the following properties:

- $G_{\mathcal{H}}$-Lipschitz gradients: for all θ_1 and θ_2, $\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \leq G_{\mathcal{H}} \|\theta_1 - \theta_2\|$, for some $G_{\mathcal{H}} > 0$.
- $L_{\mathcal{H}}$-Lipschitz Hessians: for all θ_1 and θ_2, $\left\|\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2)\right\| \leq \alpha \|\theta_1 - \theta_2\|$, for some $\alpha > L_{\mathcal{H}} > 0$.

(B2): The estimates $g(\theta;\tau)$ and $\mathcal{H}(\theta;\tau)$ satisfy

$$\|g(\theta;\tau) - \nabla J(\theta)\| \leq M_1 \text{ and } \left\|\mathcal{H}(\theta;\tau) - \nabla^2 J(\theta)\right\| \leq M_2,$$

The bounds in Proposition 1 imply (B1), while Lemma 1 implies (B2). Invoking Theorem 1 of [25] leads to the following result that establishes convergence of the ACR-PN algorithm to an ϵ-SOSP.

Theorem 4. *Assume (A1)–(A3). Fix $\delta \in (0,1]$ and let $\chi \geq J(\theta_0) - J^*$. Then, there exists a universal constant c such that setting $m_k = \max(\frac{M_1}{c\epsilon}, \frac{M_1^2}{c^2\epsilon^2}) \log\left(\frac{d\sqrt{\alpha}\chi}{\epsilon^{1.5}\delta' c}\right)$ and $b_k = \max(\frac{M_2}{c\sqrt{\alpha\epsilon}}, \frac{M_2^2}{c^2\alpha\epsilon}) \log\left(\frac{d\sqrt{\alpha}\chi}{\epsilon^{1.5}\delta' c}\right)$, the ACR-PN algorithm will output an ϵ-SOSP of J with probability at least $1-\delta'$ within $O\left(\frac{1}{\sqrt{\epsilon}}\right)$ number of iterations, while requiring*

$$O\left(\frac{\sqrt{\alpha}\chi}{\epsilon^{1.5}}\left(\max\left(\frac{M_1}{\epsilon}, \frac{M_1^2}{\epsilon^2}\right) + \max\left(\frac{M_2}{\sqrt{\alpha\epsilon}}, \frac{M_2^2}{\alpha\epsilon}\right)\frac{1}{\sqrt{\epsilon}}\right)\right)$$

number of episodes to compute gradient and Hessian-vector products.

We can conclude from the result above that the ACR-PN algorithm finds an ϵ-SOSP after $\bar{N} = O(\epsilon^{-1.5})$ iterations. The total number of trajectories required for gradient averaging and Hessian product averaging is $O(\frac{M_1^2}{\epsilon^{7/2}})$ and $O(\frac{M_2^2}{\alpha\epsilon^{5/2}})$, respectively when ϵ is small. These numbers are comparable to those of Algorithm 1, where the cubic-regularized problem was solved exactly.

6 Simulation experiments

We conduct experiments on three popular benchmarks, namely cart-pole, reacher and humanoid. We implement Algorithm 1 (CRPN), and its approximate variant ACR-PN that is described in Section 5. For the sake of comparison, we implement the vanilla policy gradient (REINFORCE) algorithm.

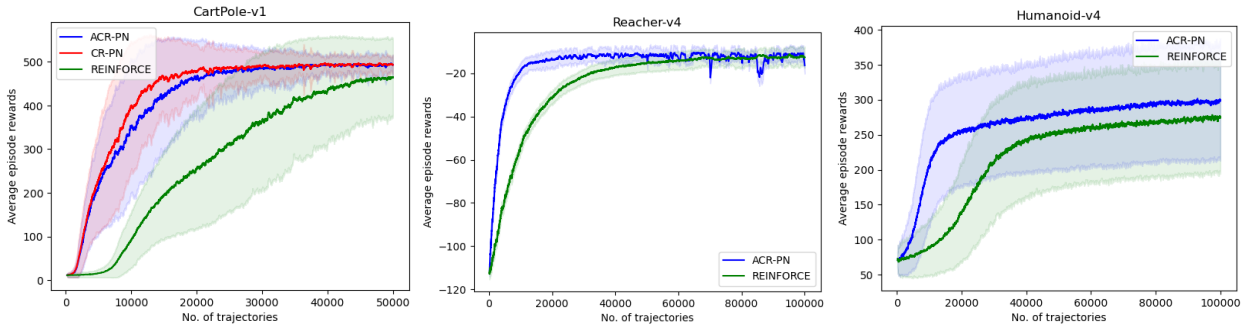

Figure 1: Performance evaluation of CR-PN, ACRP-PN and REINFORCE on gym environments. The results are averages over 10 independent replications and are smoothed by taking a moving average with an 80% confidence interval.

6.1 Cart-pole with linear function approximation

In this problem provided by OpenAI's Gym library, the observation space $\mathcal{S}$ is of the form $\{x, \dot{x}, \omega, \dot{\omega}\}$, where x is the cart position, $\dot{x}$ is the cart velocity, ω is the pole angle, and $\dot{\omega}$ is the pole angular velocity. These values are bounded according to OpenAI's Gym library documentation [1]. The reward for each time step is $+1$ until termination. The episode is terminated when $|\omega| > 12°$ or $|x| > 2.4$. One can observe that these conditions satisfy assumption (A1), which was made for the theoretical results in Section 4. All observations are initialized with a uniformly random value in the interval $(-0.05, 0.05)$ at the beginning of each episode. The action space $\mathcal{A} = \{0, 1\}$, with the first action pushing the cart to the left and the second one to the right, respectively.

For the simulation experiments, we use a linear function approximation architecture in conjunction with a Gibbs policy parametrization. The latter parametrization provides a distribution to sample the action as follows:

$$\pi(a|s;\theta) = \frac{\exp\left(\psi(s,a)^T\theta\right)}{\sum_{b\in\mathcal{A}}\exp\left(\psi(s,b)^T\theta\right)},$$

where $\psi(s,a)$ is a one-hot encoded feature w.r.t. the discrete action a and θ is the trainable parameter vector of length $\dim\mathcal{S}\times|\mathcal{A}|$ which is $4\times 2 = 8$ in this case.

In this setting, we can hard-code the gradient and Hessian to perform the updates of CRPN algorithm using the following expressions:

$$\log\pi(a|s;\theta) = \psi(s,a)^T\theta - \log\sum_{b\in\mathcal{A}}\exp\left(\psi(s,b)^T\theta\right),$$

$$\nabla\log\pi(a|s;\theta) = \psi(s,a) - \mathbb{E}_{\pi_\theta}\left[\psi(s,\cdot)\right], \text{ and}$$
$$\nabla^2\log\pi(a|s;\theta) = -\mathbb{E}_{\pi_\theta}\left[\psi(s,\cdot)\psi(s,\cdot)^T\right] + \mathbb{E}_{\pi_\theta}\left[\psi(s,\cdot)\right]\mathbb{E}_{\pi_\theta}\left[\psi(s,\cdot)\right]^T.$$

Using the quantities in the expressions above, we estimate the gradient and Hessian of the batch of trajectories as specified in Algorithm 1. For the theoretical results in Section 4, we assumed certain bounds for the parameterized policies in (A2) and (A3). Here, these follow from straightforward calculations using the expressions above in conjunction with the fact that the observation space $\mathcal{S}$ is bounded.

In order to keep the cubic sub-problem 6 well-defined, we set the regularizer α to a sufficiently large value, as suggested in [8]. We vary the corresponding learning rate in the benchmark REINFORCE algorithm as $\alpha^{-2/3}$ for a fair comparison to CR-PN. We get this relation when comparing the bounds between the two algorithms when their respective hyper-parameters are set in order to achieve the same ϵ-FOSP. The reader is referred to Theorems 5 and 6 in the supplementary material for a justification of the relation between regularizer α and REINFORCE learning rate.

In our implementation of CR-PN, the cubic sub-problem 6 is optimized numerically using the Newton-CG algorithm provided by the SciPy toolkit with a small enough tolerance to ensure accurate results[2]. The update iterations for both CRPN and REINFORCE algorithms are performed by estimating the gradient and Hessian over a batch size of 50 independent episodes.

Figure 1 shows the average episodic rewards for REINFORCE, CR-PN and ACR-PN algorithms. It is apparent that both CR-PN and ACR-PN outperform REINFORCE, implying convergence in a lower number of iterations as compared to REINFORCE. Notice that CR-PN and ACR-PN show improved performance initially as compared to REINFORCE and such a performance gain was observed for a variety of α-values. Further, we observe that CR-PN outperforms ACR-PN slightly owing to its higher precision while finding the optima of the sub-problem using

[1] OpenAI Gym library's Cart-Pole environment documentation is available at `https://gymnasium.farama.org/environments/classic_control/cart_pole/`.

[2] Newton-CG algorithm documentation by SciPy is available at `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-newtoncg.html`

Newton-based solvers. However, our implementation of ACR-PN, which requires Hessian-vector products, is extensible to cover neural networks as seen in the MuJoCo experiments, which we describe next.

The reader is referred to Appendix H for the details of the hyper-parameters in our implementation as well as additional simulation experiments and results.

6.2 MuJoCo environments with deep neural networks

For deep neural network policies, as computing the full Hessian is intractable, we implement the ACR-PN algorithm, which solves the cubic sub-problem in Algorithm 1 approximately using a gradient descent approach, as suggested in [8]. We perform experiments on the popularly used MuJoCo (Multi-Joint dynamics with Contact) environments. We consider two environments, namely Reacher (2 joints) and Humanoid (17 joints).

The Reacher environment consists of a two-jointed robotic arm that rotates in a horizontal 2-D arena. The goal in this problem is to take the end-effector (tip) of the arm to some target co-ordinate (random position) within the arena. The net reward is proportional to the negative of the distance of the tool tip and target position, while also having an additional penalty for large activation forces at the joints.

The Humanoid environment based on [24] is of a 3-D bipedal robot that has a torso and a pair of legs and arms. The legs have 3 body parts, i.e., the "knees" and "feet" while the arms have two joints by the "elbow". The net reward depends on how fast the humanoid is able to walk forward while staying upright and having some control costs to penalise large force inputs.

We use a standard Multi-Layer Perceptron (MLP) neural network architecture with softmax hidden activation, and a Gaussian policy where the parameters mean and standard-deviation are learnable, i.e., $\pi(a|s;\theta) = \mathcal{N}(\mu_\theta, \sigma_\theta^2)$, where μ_θ is in $(-1, 1)$ using a $\tanh$ function, and σ_θ is clipped.

As second-order methods have not been explored as much as their first-order counterparts in the implementation on popular RL benchmarks such as MuJoCo, we provide an efficient scheme for computing the Hessian-vector estimates from a Monte-Carlo simulation in libraries like PyTorch and TensorFlow that have auto-grad functionality. We now briefly describe this scheme for Hessian-vector estimation which can be found in Section I. Let $\mathbf{L_1}^{(i)} := \Phi(\theta;\tau_i)$ and $\mathbf{L_2}^{(i)} := \log p(\tau_i;\theta)$ denote the stacked "losses" corresponding to each trajectory in the mini-batch. Given $\theta \in \mathbb{R}^d$ and $\nabla \equiv \nabla_\theta$, let $\nabla\mathbf{L_1}^{(ij)} := \frac{\partial \mathbf{L_1}^{(j)}}{\partial \theta^{(i)}}$ and $\nabla\mathbf{L_2}^{(ij)} := \frac{\partial \mathbf{L_2}^{(j)}}{\partial \theta^{(i)}}$. Then, $\mathbf{L_1}, \mathbf{L_2} \in \mathbb{R}^n$ and $\nabla\mathbf{L_1}, \nabla\mathbf{L_2} \in \mathbb{R}^{d\times n}$ by convention. Now, from (1), the gradient and Hessian-vector products can be written in vector-matrix notation as follows:

$$\bar{g} = \frac{1}{n}\sum_i \nabla\mathbf{L_1}^{(i)} = \nabla\left(\frac{1}{n}\sum_i \mathbf{L_1}^{(i)}\right), \text{ and}$$

$$\bar{\mathcal{H}}\cdot v = \frac{1}{n}\sum_i (\nabla\mathbf{L_1}^{(i)}\nabla^\top\mathbf{L_2}^{(i)} + \nabla^2\mathbf{L_1}^{(i)})\cdot v$$

$$= \frac{1}{n}\nabla\mathbf{L_1}\nabla^\top\mathbf{L_2}v + \nabla\bar{g}^\top v.$$

The second term, i.e. $\nabla\bar{g}^\top v$ is straightforward to implement as we already have the computational graph of $\bar{g}$ that we retain and use it to re-differentiate its inner product with v, and this requires just **one** additional auto-grad call. On the other hand, estimating the first term, i.e. $\frac{1}{n}\nabla\mathbf{L_1}\nabla^\top\mathbf{L_2}v$ is non-trivial but can be performed using **three** calls to the back-propagation algorithm. To elaborate, we first define a function $ujp(f,u;x) := \frac{\partial f}{\partial x}u$, with f and u being vectors of the same dimension and x being the input to differentiate against. This function resembles the function signature and working of the auto-grad function in PyTorch [3]. Then for some $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^d$, we have

$$ujp(\mathbf{L_2}, u;\theta) = \nabla_\theta\mathbf{L_2}u = \nabla\langle\mathbf{L_2}, u\rangle, \quad \in \mathbb{R}^d$$

$$ujp(\nabla\mathbf{L_2}u, v; u) = \nabla_u(\nabla\mathbf{L_2}u)v = \nabla^\top\mathbf{L_2}v, \quad \in \mathbb{R}^n$$

$$ujp(\mathbf{L_1}, \nabla^\top\mathbf{L_2}v;\theta) = \nabla\mathbf{L_1}\nabla^\top\mathbf{L_2}v. \quad \in \mathbb{R}^d$$

The scheme outlined above provides the *exact* Hessian-vector product for a neural network and this is unlike previous works, cf. [21], which employed finite-differences to arrive at approximations to the Hessian-vector product. The reader is referred to the GitHub implementation, which is available at `https://github.com/mizhaan23/crpn_algo`.

7 Conclusions and future work

In this paper, we proposed policy Newton algorithms with cubic regularization. Our algorithms form unbiased estimates of the gradient as well as the Hessian of the value function using sample trajectories. Through a rigorous convergence analysis, we established that our policy Newton algorithms converge to a second-order stationary point (SOSP) of the value function, which implies the algorithms avoid saddle points. Further, the sample complexity of our algorithms to find an ϵ-SOSP is $O(\epsilon^{-3.5})$, and this result is an improvement over the $O(\epsilon^{-4.5})$ bound currently available in the literature.

As future work, it would be interesting to extend the policy Newton algorithms to incorporate feature-based representations and function approximation, and establish convergence of the resulting actor-critic algorithms to SOSPs. From an empirical standpoint, policy Newton algorithms may be explored in real-life applications.

[3] See `https://pytorch.org/docs/stable/generated/torch.autograd.grad.html`

Acknowledgments

S.Bhatnagar was supported in part by the J.C.Bose National Fellowship of SERB, Project No. DFTM/02/3125/M/04/AIR-04 from DRDO under DIA-RCOE, a grant from Walmart under the CSR initiative as well as the RBCCPS, IISc.

References

[1] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, volume 125, pages 64–66, Jul 2020.

[2] K. Balasubramanian and S. Ghadimi. Zeroth-order nonconvex stochastic optimization: Handling constraints, high dimensionality, and saddle points. *Foundations of Computational Mathematics*, 22(1):35–76, Feb 2022.

[3] S. Bhatnagar. An actor–critic algorithm with function approximation for discounted cost constrained Markov decision processes. *Systems & Control Letters*, 59(12):760–766, 2010.

[4] S. Bhatnagar and K. Lakshmanan. An online actor–critic algorithm with function approximation for constrained Markov decision processes. *Journal of Optimization Theory and Applications*, 153:688–708, 2012.

[5] S. Bhatnagar, H. Prasad, and L. A. Prashanth. *Stochastic recursive algorithms for optimization. Simultaneous perturbation methods*, volume 434. Springer-Verlag London, 01 2013.

[6] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[7] V. S. Borkar. An actor-critic algorithm for constrained Markov decision processes. *Systems & control letters*, 54(3):207–213, 2005.

[8] Y. Carmon, J. Duchi, O. Hinder, and A. Sidford. Accelerated methods for non-convex optimization. *arXiv preprint*, arXiv:1611.00756, 2016.

[9] T. Furmston, G. Lever, and D. Barber. Approximate Newton Methods for Policy Search in Markov Decision Processes. *Journal of Machine Learning Research*, 17(226):1–51, 2016.

[10] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23:2341–2368, 2013.

[11] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. *ICML*, 2017.

[12] V. R. Konda and V. S. Borkar. Actor-critic–type learning algorithms for Markov decision processes. *SIAM Journal on control and Optimization*, 38(1):94–123, 1999.

[13] H. Kumar, A. Koppel, and A. Ribeiro. On the sample complexity of actor-critic method for reinforcement learning with function approximation. *Machine Learning*, pages 1–35, 2023.

[14] A. Mondal, L. A. Prashanth, and S. Bhatnagar. A Gradient Smoothed Functional Algorithm with Truncated Cauchy Random Perturbations for Stochastic Optimization. *arXiv preprint arXiv:2208.00290*, 2022.

[15] Y. Nesterov and B. Polyak. Cubic regularization of Newton method and its global performance. *Math. Program.*, 108:177–205, 08 2006.

[16] Y. Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 112:159–181, 2007.

[17] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli. Stochastic variance-reduced policy gradient. In *ICML*, volume 80, pages 4026–4035, Jul 2018.

[18] S. Paternain, A. Mokhtari, and A. Ribeiro. A Newton-Based Method for Nonconvex Optimization with Fast Evasion of Saddle Points. *Society for Industrial and Applied Mathematics*, 29:343–368, 2019.

[19] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *ICML*, pages 1889–1897, 2015.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimovl. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[21] Z. Shen, A. Ribeiro, H. Hassani, H. Qian, and C. Mi. Hessian aided policy gradient. In *International Conference on Machine Learning*, pages 5729–5738. PMLR, 2019.

[22] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2018.

[23] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, pages 1057–1063, 1999.

[24] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.

[25] N. Tripuraneni, M. Stern, C. Jin, J. Regier, and M. I. Jordan. Stochastic cubic regularization for fast nonconvex optimization. In *NeurIPS*, volume 31. Curran Associates, Inc., 2018.

[26] J. A Tropp. An Introduction to Matrix Concentration Inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[27] J. A. Tropp. The Expected Norm of a Sum of Independent Random Matrices: An Elementary Approach. In *High Dimensional Probability VII: The Cargèse Volume*, pages 173–202, 2016.

[28] N. Vijayan and L. A. Prashanth. Smoothed functional-based gradient algorithms for off-policy reinforcement learning. *Systems & Control Letters*, 155:104988, 2021.

[29] P. Wang, H. Wang, and N. Zheng. Stochastic cubic-regularized policy gradient method. *Knowledge-Based Systems*, 255:109687, 2022.

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

[31] L. Yang, Q. Zheng, and G. Pan. Sample complexity of policy gradient finding second-order stationary points. In *AAAI*, pages 10630–10638, 2021.

[32] K. Zhang, A. Koppel, H. Zhu, and T. Basar. Global convergence of policy gradient methods to (almost) locally optimal policies. *SIAM Journal on Control and Optimization*, 58(6):3586–3612, 2020.

A Proof of Proposition 1

Lemma 4. *Under Assumptions (A1) and (A2), for any parameter θ and trajectory τ, we have*

$$\|\nabla\Phi(\theta;\tau)\| \leq G_g,\ \left\|\nabla^2\Phi(\theta;\tau)\right\| \leq L_1KH^2,\ \|g(\theta;\tau)\| \leq G_g,\ \textit{and}\ \|\mathcal{H}(\theta;\tau)\| \leq G_{\mathcal{H}},$$

where $G_g = GKH^2$, and $G_{\mathcal{H}} := H^3G^2K + L_1KH^2$.

Proof. Using the definition of $\Phi(\theta;\tau)$, we have

$$\|\nabla\Phi(\theta;\tau)\| = \left\|\sum_{i=0}^{H-1}\Psi_i(\tau)\nabla\log\pi(a_i|s_i;\theta)\right\| \leq \sum_{i=0}^{H-1}|\Psi_i(\tau)|\,\|\nabla\log\pi(a_i|s_i;\theta)\| \leq G\sum_{i=0}^{H-1}|\Psi_i(\tau)|.$$

We establish a bound on $|\Psi_i(\tau)|$ as follows:

$$|\Psi_i(\tau)| = |\sum_{h=i}^{H-1}\gamma^{h-1}c(s_h,a_h)| \leq K\sum_{h=i}^{H-1}\gamma^{h-1} \leq KH.$$

Thus,

$$\|\nabla\Phi(\theta;\tau)\| \leq GKH^2.$$

Similarly,

$$\begin{aligned}\left\|\nabla^2\Phi(\theta;\tau)\right\| &= \left\|\sum_{i=0}^{H-1}\Psi_i(\tau)\nabla^2\log\pi(a_i|s_i;\theta)\right\| \leq \sum_{i=0}^{H-1}|\Psi_i(\tau)|\left\|\nabla^2\log\pi(a_i|s_i;\theta)\right\| \\ &\leq L_1\sum_{i=0}^{H-1}|\Psi_i(\tau)| \leq L_1KH^2.\end{aligned}$$

It is now easy to show that the gradient estimate $g(\theta;\tau)$ is bounded as follows:

$$\|g(\theta;\tau)\| = \|\nabla\Phi(\theta;\tau)\| \leq GKH^2 = G_g.$$

Next, we show that the Hessian estimate $\mathcal{H}(\theta;\tau)$ is bounded. Notice that

$$\begin{aligned}\|\mathcal{H}(\theta;\tau)\| &= \left\|\nabla\Phi(\theta;\tau)\nabla^\top\log p(\tau;\theta) + \nabla^2\Phi(\theta;\tau)\right\| \\ &\leq \|\nabla\Phi(\theta;\tau)\|\,\|\nabla\log p(\tau;\theta)\| + \left\|\nabla^2\Phi(\theta;\tau)\right\| \\ &\leq GKH^2\,\|\nabla\log p(\tau;\theta)\| + L_1KH^2.\end{aligned}$$

Using the relation $\nabla\log p(\tau;\theta) = \sum_{h=0}^{H-1}\nabla\log\pi(a_h|s_h;\theta)$, we have

$$\|\nabla\log p(\tau;\theta)\| \leq \sum_{h=0}^{H-1}\|\nabla\log\pi(a_h|s_h;\theta)\| \leq HG.$$

Therefore, we obtain

$$\|\mathcal{H}(\theta;\tau)\| \leq H^3G^2K + L_1KH^2 = G_{\mathcal{H}}.$$

Hence proved. □

From the above lemma, one can easily interpret that the objective function, i.e., $J(\theta)$ and its gradient, $\nabla J(\theta)$ are Lipschitz continuous. We now show that the Hessian of the objective, i.e., $\nabla^2 J(\theta)$ is Lipschitz.

Lemma 5. *Under Assumptions (A1), (A2) and (A3), we have for any (θ_1,θ_2),*

$$\begin{aligned}&\left\|\nabla^2J(\theta_1) - \nabla^2J(\theta_2)\right\| \leq L_{\mathcal{H}}\,\|\theta_1-\theta_2\|,\ \textit{where} \\ &L_{\mathcal{H}} := H^4G^3K + 3H^3GL_1K + L_2KH^2.\end{aligned} \tag{12}$$

Proof. We begin with the expression for the Hessian of our objective, i.e.,

$$\nabla^2 J(\theta) = \sum_\tau (\nabla\Phi(\theta;\tau)\nabla^\top p(\tau;\theta) + p(\tau;\theta)\nabla^2\Phi(\theta;\tau))\,.$$

Notice that

$$\begin{aligned}
\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2) &= \sum_\tau (\nabla\Phi(\theta_1;\tau)\nabla^\top p(\tau;\theta_1) + p(\tau;\theta_1)\nabla^2\Phi(\theta_1;\tau)) \\
&\quad - \sum_\tau (\nabla\Phi(\theta_2;\tau)\nabla^\top p(\tau;\theta_2) + p(\tau;\theta_2)\nabla^2\Phi(\theta_2;\tau)) \\
&= \sum_\tau \left(\nabla\Phi(\theta_1;\tau)\nabla^\top p(\tau;\theta_1) - \nabla\Phi(\theta_2;\tau)\nabla^\top p(\tau;\theta_2)\right) \\
&\quad + \sum_\tau \left(p(\tau;\theta_1)\nabla^2\Phi(\theta_1;\tau) - p(\tau;\theta_2)\nabla^2\Phi(\theta_2;\tau)\right) \\
\text{Hence, } \left\|\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2)\right\| &\le \sum_\tau \left\|\nabla\Phi(\theta_1;\tau)\nabla^\top p(\tau;\theta_1) - \nabla\Phi(\theta_2;\tau)\nabla^\top p(\tau;\theta_2)\right\| \\
&\quad + \sum_\tau \left\|p(\tau;\theta_1)\nabla^2\Phi(\theta_1;\tau) - p(\tau;\theta_2)\nabla^2\Phi(\theta_2;\tau)\right\|\,. \qquad (13)
\end{aligned}$$

For ease of notation, let $\Phi_j := \Phi(\theta_j;\tau)$ and $p_j := p(\tau;\theta_j)$. We bound the first summand in (13) as follows:

$$\begin{aligned}
\left\|\nabla\Phi_1\nabla^\top p_1 - \nabla\Phi_2\nabla^\top p_2\right\| &\le \left\|\nabla\Phi_1\nabla^\top p_1 - \nabla\Phi_1\nabla^\top p_2\right\| + \left\|\nabla\Phi_1\nabla^\top p_2 - \nabla\Phi_2\nabla^\top p_2\right\| \\
&\le \|\nabla\Phi_1\|\,\|\nabla p_1 - \nabla p_2\| + \|\nabla p_2\|\,\|\nabla\Phi_1 - \nabla\Phi_2\|\,. \qquad (14)
\end{aligned}$$

Using the mean-value theorem for vector-valued functions, we have

$$\nabla p_1 - \nabla p_2 = \nabla^2 p_{h_1}(\theta_1 - \theta_2),$$

where $p_{h_1} = p(\tau;\theta_{h_1})$, and $\theta_{h_1} = (1-h_1)\theta_1 + h_1\theta_2$ for some $h_1 \in [0,1]$. Hence,

$$\begin{aligned}
\|\nabla p_1 - \nabla p_2\| &\le \left\|\nabla^2 p_{h_1}\right\|\|\theta_1 - \theta_2\| \\
&= \|\nabla(p_{h_1}\nabla\log p_{h_1})\|\,\|\theta_1 - \theta_2\| \\
&= \left\|\nabla p_{h_1}\nabla^\top \log p_{h_1} + p_{h_1}\nabla^2\log p_{h_1}\right\|\|\theta_1 - \theta_2\| \\
&= \left\|p_{h_1}(\nabla\log p_{h_1}\nabla^\top\log p_{h_1} + \nabla^2\log p_{h_1})\right\|\|\theta_1 - \theta_2\| \\
&\overset{(a)}{\le} p_{h_1}\left(\|\nabla\log p_{h_1}\|^2 + \left\|\nabla^2\log p_{h_1}\right\|\right)\|\theta_1 - \theta_2\| \\
&\le p_{h_1}\left(H^2G^2 + HL_1\right)\|\theta_1 - \theta_2\|\,, \qquad (15)
\end{aligned}$$

where (a) uses the relation $\nabla\log p_j = \sum_{h=0}^{H-1}\nabla\log\pi(a_h|s_h;\theta_j)$ and $\nabla^2\log p_j = \sum_{h=0}^{H-1}\nabla^2\log\pi(a_h|s_h;\theta_j)$.

Similarly,

$$\|\nabla\Phi_1 - \nabla\Phi_2\| \le \left\|\nabla^2\Phi_{h_2}\right\|\|\theta_1 - \theta_2\|\,, \qquad (16)$$

where $\Phi_{h_2} = \Phi(\theta_{h_2};\tau)$, and $\theta_{h_2} = (1-h_2)\theta_1 + h_2\theta_2$ for some $h_2 \in [0,1]$. Using (15) and (16) in (14), we obtain

$$\begin{aligned}
\left\|\nabla\Phi_1\nabla^\top p_1 - \nabla\Phi_2\nabla^\top p_2\right\| &\le p_{h_1}\left(H^2G^2 + HL_1\right)\|\nabla\Phi_1\|\,\|\theta_1 - \theta_2\| + p_2\,\|\nabla\log p_2\|\left\|\nabla^2\Phi_{h_2}\right\|\|\theta_1 - \theta_2\| \\
&\le p_{h_1}(H^4G^3K + H^3L_1GK)\,\|\theta_1 - \theta_2\| + p_2H^3GL_1K\,\|\theta_1 - \theta_2\|\,, \qquad (17)
\end{aligned}$$

where the final inequality used the bounds obtained in Lemma 4.

Summing both the sides of (17) over τ, we obtain

$$\sum_\tau \left\|\nabla\Phi_1\nabla^\top p_1 - \nabla\Phi_2\nabla^\top p_2\right\| \le H^4G^3K + 2H^3GL_1K\,\|\theta_1 - \theta_2\|\,. \qquad (18)$$

In the above inequality, we used the fact that $\sum_\tau p(\tau;\theta) = 1$ for all θ.

We now bound the second summand in (13). Notice that

$$\begin{aligned}\left\|p_1\nabla^2\Phi_1 - p_2\nabla^2\Phi_2\right\| &\leq \left\|p_1\nabla^2\Phi_1 - p_1\nabla^2\Phi_2\right\| + \left\|p_1\nabla^2\Phi_2 - p_2\nabla^2\Phi_2\right\|, \\ &\leq p_1\left\|\nabla^2\Phi_1 - \nabla^2\Phi_2\right\| + \left\|\nabla^2\Phi_2\right\||p_1 - p_2|. \end{aligned} \tag{19}$$

Using the mean value theorem, we obtain

$$\begin{aligned}|p_1 - p_2| &\leq \|\nabla p_{h_3}\|\,\|\theta_1 - \theta_2\| \\ &\leq p_{h_3}\|\nabla \log p_{h_3}\|\,\|\theta_1 - \theta_2\| \\ &\leq p_{h_3}(HG)\,\|\theta_1 - \theta_2\|.\end{aligned}$$

We now bound the first term in (19) as follows:

$$\begin{aligned}\left\|\nabla^2\Phi_1 - \nabla^2\Phi_2\right\| &= \left\|\sum_{i=0}^{H-1}\Psi_i(\tau)\left(\nabla^2\log\pi(a_i|s_i;\theta_1) - \nabla^2\log\pi(a_i|s_i;\theta_2)\right)\right\| \\ &\leq \sum_{i=0}^{H-1}|\Psi_i(\tau)|\left\|\nabla^2\log\pi(a_i|s_i;\theta_1) - \nabla^2\log\pi(a_i|s_i;\theta_2)\right\| \\ &\leq L_2\|\theta_1 - \theta_2\|\sum_{i=0}^{H-1}KH \leq L_2KH^2\|\theta_1 - \theta_2\|.\end{aligned}$$

We used here (A3) in the second last inequality above. Plugging the above results in (19), we have

$$\left\|p_1\nabla^2\Phi_1 - p_2\nabla^2\Phi_2\right\| \leq p_1L_2KH^2\|\theta_1 - \theta_2\| + p_{h_3}H^3GL_1K\|\theta_1 - \theta_2\|.$$

Summing over trajectories on both sides above, we obtain

$$\sum_\tau\left\|p_1\nabla^2\Phi_1 - p_2\nabla^2\Phi_2\right\| \leq (L_2KH^2 + H^3GL_1K)\|\theta_1 - \theta_2\|. \tag{20}$$

Using (18) and (20) in (13), we obtain

$$\begin{aligned}\left\|\nabla^2 J(\theta_1) - \nabla^2 J(\theta_2)\right\| &\leq (H^4G^3K + 2H^3GL_1K)\|\theta_1 - \theta_2\| + (L_2KH^2 + H^3GL_1K)\|\theta_1 - \theta_2\| \\ &= (H^4G^3K + 3H^3GL_1K + L_2KH^2)\|\theta_1 - \theta_2\|.\end{aligned}$$

The claim follows. □

Remark 3. *From* (12)*, it can be easily seen that*

$$\begin{aligned}\left\|\nabla J(\theta_1) - \nabla J(\theta_2) - \nabla^2 J(\theta_2)(\theta_1 - \theta_2)\right\| &\leq \frac{L_{\mathcal{H}}}{2}\|\theta_1 - \theta_2\|^2, \\ |J(\theta_1) - J(\theta_2) - \langle\nabla J(\theta_2), \theta_1 - \theta_2\rangle - \frac{1}{2}\left\langle\theta_1 - \theta_2, \nabla^2 J(\theta_2)(\theta_1 - \theta_2)\right\rangle| &\leq \frac{L_{\mathcal{H}}}{6}\|\theta_1 - \theta_2\|^3.\end{aligned}$$

B Proof of Theorem 1

Proof. The result is available in [21], and we provide the proof here for the sake of completeness. First, we re-write the objective function (2) as follows:

$$J(\theta) := \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\mathcal{G}(\tau)\right] = \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[\sum_{h=0}^{H-1}\gamma^{h-1}c(s_h, a_h)\right] = \sum_{h=0}^{H-1}\mathbb{E}_{\tau_h\sim p(\tau_h;\theta)}\left[\gamma^{h-1}c(s_h, a_h)\right].$$

The last equality above holds as the term inside the expectation is independent of future events, i.e., the trajectory $(s_{0:h}, a_{0:h})$ does not depend on the trajectory $(s_{h+1:H-1}, a_{h+1:H-1})$. Replacing the expectation by a summation over all trajectories

$$J(\theta) = \sum_{h=0}^{H-1}\sum_{\tau_h}\gamma^{h-1}c(s_h, a_h)p(\tau_h;\theta).$$

Differentiating on both sides, we obtain

$$\nabla J(\theta) = \sum_{h=0}^{H-1} \sum_{\tau_h} \gamma^{h-1} c(s_h, a_h) \nabla p(\tau_h; \theta) \,.$$

Using now the fact that $\nabla p(\tau_h; \theta) = p(\tau_h; \theta) \nabla \log p(\tau_h; \theta)$, we obtain

$$\begin{aligned}
\nabla J(\theta) &= \sum_{h=0}^{H-1} \sum_{\tau_h} \gamma^{h-1} c(s_h, a_h) \nabla \log p(\tau_h; \theta)\, p(\tau_h; \theta) \\
&= \sum_{h=0}^{H-1} \mathbb{E}_{\tau_h \sim p(\tau_h;\theta)} \left[\gamma^{h-1} c(s_h, a_h) \nabla \log p(\tau_h; \theta) \right] .
\end{aligned}$$

From (1), we can show that $\nabla \log p(\tau; \theta) = \sum_{h=0}^{H-1} \nabla \log \pi(a_h | s_h; \theta)$, and thus

$$\begin{aligned}
\nabla J(\theta) &= \sum_{h=0}^{H-1} \mathbb{E}_{\tau_h \sim p(\tau_h;\theta)} \left[\gamma^{h-1} c(s_h, a_h) \sum_{i=1}^{h} \nabla \log \pi(a_i | s_i; \theta) \right] \\
&= \sum_{h=0}^{H-1} \sum_{i=0}^{h} \mathbb{E}_{\tau_h \sim p(\tau_h;\theta)} \left[\gamma^{h-1} c(s_h, a_h) \nabla \log \pi(a_i | s_i; \theta) \right] \\
&= \sum_{h=0}^{H-1} \sum_{i=0}^{h} \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\gamma^{h-1} c(s_h, a_h) \nabla \log \pi(a_i | s_i; \theta) \right] .
\end{aligned}$$

where in the last equality we use the fact that $\gamma^{h-1} c(s_h, a_h) \nabla \log \pi(a_i | s_i; \theta)$ with $i \leq h$ is independent of the randomness after a_h. Interchanging the order of summation, we obtain

$$\begin{aligned}
\nabla J(\theta) &= \sum_{i=0}^{H-1} \sum_{h=i}^{H-1} \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\gamma^{h-1} c(s_h, a_h) \nabla \log \pi(a_i | s_i; \theta) \right] \\
&= \sum_{i=0}^{H-1} \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\left(\sum_{h=i}^{H-1} \gamma^{h-1} c(s_h, a_h) \right) \nabla \log \pi(a_i | s_i; \theta) \right] \\
&= \sum_{i=0}^{H-1} \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\Psi_i(\tau) \nabla \log \pi(a_i | s_i; \theta) \right] .
\end{aligned}$$

This concludes the proof of the first claim. For the second claim, notice that

$$\begin{aligned}
\nabla^2 J(\theta) &= \nabla \left(\sum_{\tau} \nabla \Phi(\theta; \tau) p(\tau; \theta) \right) \\
&= \sum_{\tau} (\nabla \Phi(\theta; \tau) \nabla^\top p(\tau; \theta) + \nabla^2 \Phi(\theta; \tau) p(\tau; \theta) \\
&= \sum_{\tau} \left(\nabla \Phi(\theta; \tau) \nabla^\top \log p(\tau; \theta) + \nabla^2 \Phi(\theta; \tau) \right) p(\tau; \theta)) \\
&= \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[\nabla \Phi(\theta; \tau) \nabla^\top \log p(\tau; \theta) + \nabla^2 \Phi(\theta; \tau) \right] .
\end{aligned}$$

Hence proved. □

C Proof of Theorem 2

The proof proceeds through a sequence of lemmas while following the technique from [2]. However, unlike the aforementioned reference, we operate in an RL framework and more importantly, with unbiased gradient and Hessian estimates, leading to a major deviation in the proof as compared to [2].

Lemma 6. *Let $\bar{\theta} = \operatorname{argmin}_{x\in\mathbb{R}^d} \tilde{J}(x,\theta,\mathcal{H},g,\alpha)$. Then, we have*

$$g + \mathcal{H}(\bar{\theta}-\theta) + \frac{\alpha}{2}\left\|\bar{\theta}-\theta\right\|(\bar{\theta}-\theta) = 0,$$
$$\mathcal{H} + \frac{\alpha}{2}\left\|\bar{\theta}-\theta\right\| I_d \succeq 0.$$

where I_d is the identity matrix.

Proof. See Lemma 4.3 from [2] and [15]. □

We now derive the second and third-order error bounds on our Hessian estimate.

Lemma 7. *Let $\bar{g}_k$ and $\bar{\mathcal{H}}_k$ be computed as in Algorithm 1, and assume $b_k \geq 4(1+2\log 2d)$.*

Then we have

$$\mathbb{E}\left[\|\bar{g}_k - \nabla J(\theta_{k-1})\|^2\right] \leq \frac{G_g^2}{m_k}, \qquad \mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^3\right] \leq \frac{4\sqrt{15(1+2\log 2d)}dG_{\mathcal{H}}^3}{b_k^{\frac{3}{2}}}. \tag{21}$$

Proof. Using the fact that the estimate $\bar{g}_k$ is unbiased, we have

$$\begin{aligned}
\mathbb{E}\left[\|\bar{g}_k - \nabla J(\theta_{k-1})\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{m_k}\sum_{\tau\in\mathcal{T}_m}(g(\theta_{k-1};\tau) - \nabla J(\theta_{k-1}))\right\|^2\right] \\
&= \frac{1}{m_k^2}\sum_{\tau}\mathbb{E}\left[\|g(\theta_{k-1};\tau) - \nabla J(\theta_{k-1})\|^2\right] \\
&+ \frac{1}{m_k^2}\sum_{\tau\neq\tau'}\mathbb{E}\left[\langle g(\theta_{k-1};\tau) - \nabla J(\theta_{k-1}), g(\theta_{k-1};\tau') - \nabla J(\theta_{k-1})\rangle\right], \qquad (22) \\
&\leq \frac{1}{m_k^2}\mathbb{E}\left[\sum_{\tau\in\mathcal{T}_m}\|g(\theta_{k-1};\tau)\|^2\right] \leq \frac{G_g^2}{m_k},
\end{aligned}$$

where the final inequality follows by equating the second summand in (22) to zero. This zeroing is justified from the fact that the trajectories are independent. This establishes the first bound in (21).

Now we turn to proving the second bound in (21). By Theorem 1 in [27], we have

$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^2\right] \leq \frac{2C(d)}{b_k^2}\left(\left\|\sum_{\tau\in\mathcal{T}_b}\mathbb{E}\left[\Delta_{k,\tau}^2\right]\right\| + C(d)\mathbb{E}\left[\max_{\tau}\|\Delta_{k,\tau}\|^2\right]\right), \tag{23}$$

where $\Delta_{k,\tau} = \mathcal{H}(\theta_{k-1};\tau) - \nabla^2 J(\theta_{k-1})$ and $C(d) = 4(1+2\log 2d)$. It is easy to see that

$$\mathbb{E}\left[\|\Delta_{k,\tau}\|^2\right] \leq \mathbb{E}\left[\|\mathcal{H}(\theta_{k-1};\tau)\|^2\right] \leq G_{\mathcal{H}}^2, \quad \text{and} \tag{24}$$

$$\left\|\sum_{\tau\in\mathcal{T}_b}\mathbb{E}\left[\Delta_{k,\tau}^2\right]\right\| \leq \sum_{\tau\in\mathcal{T}_b}\left\|\mathbb{E}\left[\Delta_{k,\tau}^2\right]\right\| \leq \sum_{\tau\in\mathcal{T}_b}\mathbb{E}\left[\|\Delta_{k,\tau}\|^2\right]. \tag{25}$$

Using (24) and (25) in (23), we obtain

$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^2\right] \leq \frac{2C(d)}{b_k^2}\left(b_k G_{\mathcal{H}}^2 + C(d)G_{\mathcal{H}}^2\right) \leq \frac{4C(d)}{b_k}G_{\mathcal{H}}^2,$$

where in the last inequality we use the assumption that $b_k \geq C(d)$. Letting $\|\|_F$ denote the Frobenius norm, and using Holder's inequality, we obtain

$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^3\right] \leq \mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\| \cdot \left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|_F^2\right] \tag{26}$$

$$\leq \left(\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|^2\right] \cdot \mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|_F^4\right]\right)^{\frac{1}{2}}.$$

Note that $\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1}) = \frac{1}{b_k}\sum_{\tau\in\mathcal{T}_b}\Delta_{k,\tau}$, therefore we have

$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|_F^4\right] = \mathbb{E}\left[\left\|\frac{1}{b_k}\sum_{\tau\in\mathcal{T}_b}\Delta_{k,\tau}\right\|_F^4\right] = \frac{1}{b_k^4}\mathbb{E}\left[\left\|\sum_{\tau\in\mathcal{T}_b}\Delta_{k,\tau}\right\|_F^4\right] \leq \frac{3\mathbb{E}\left[\|\Delta_{k,\tau}\|_F^4\right]}{b_k^2},$$

where the final inequality comes from Rosenthal's inequality (see Lemma 16 in Appendix G). Using the fact that $\|\cdot\|_F \leq \sqrt{d}\,\|\cdot\|$ and the inequality from Lemma 15 in Appendix G, we have

$$\mathbb{E}\left[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta_{k-1})\right\|_F^4\right] \leq \frac{3d^2\mathbb{E}\left[\|\Delta_{k,\tau}\|^4\right]}{b_k^2} \leq \frac{15d^2\mathbb{E}\left[\|\mathcal{H}(\theta_{k-1};\tau_i)\|^4\right]}{b_k^2} \leq \frac{15d^2G_{\mathcal{H}}^4}{b_k^2},$$

which when combined in (26) leads to the second bound in (21). □

We next state a result that will be used in a subsequent lemma.

Lemma 8. *If for any two matrices A and B, and a scalar c, we have*

$$A \preceq B + cI,$$

where I is the identity matrix of the appropriate dimension, then the following holds:

$$c \geq \lambda_{max}(A) - \|B\|$$

Proof. See Appendix F. □

Lemma 9. *Let $\{\theta_k\}$ be computed by Algorithm 1. Then, we have*

$$\begin{aligned}&\sqrt{\mathbb{E}\left[\|\theta_k - \theta_{k-1}\|^2\right]}\\ &\geq \max\left\{\sqrt{\frac{\mathbb{E}\left[\|\nabla J(\theta_k)\|\right] - \delta_k^g - \delta_k^{\mathcal{H}}}{L_{\mathcal{H}} + \alpha_K}}, \frac{-2}{\alpha_k + 2L_{\mathcal{H}}}\left[\mathbb{E}\left[\lambda_{\min}\left(\nabla^2 J(\theta_k)\right)\right] + \sqrt{2(\alpha_k + L_{\mathcal{H}})\delta_k^{\mathcal{H}}}\right]\right\},\end{aligned} \tag{27}$$

where $\delta_k^g, \delta_k^{\mathcal{H}} > 0$ are chosen such that

$$\mathbb{E}\left[\|\nabla J(\theta_{k-1}) - \bar{g}_k\|^2\right] \leq \left(\delta_k^g\right)^2, \quad \textit{and} \quad \mathbb{E}\left[\left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|^3\right] \leq \left(2(L_{\mathcal{H}} + \alpha_k)\delta_k^{\mathcal{H}}\right)^{\frac{3}{2}}. \tag{28}$$

Proof. From Lemma 7, note that δ_k^g and $\delta_k^{\mathcal{H}}$ are inversely proportional to $\sqrt{m_k}$ and b_k, respectively and are therefore well-defined. Now, by the equality condition in Lemma 6, we have

$$\begin{aligned}\|\nabla J(\theta_k)\| &\leq \left\|\nabla J(\theta_k) - \nabla J(\theta_{k-1}) - \nabla^2 J(\theta_{k-1})(\theta_k - \theta_{k-1})\right\| + \|\nabla J(\theta_{k-1}) - \bar{g}_k\| \\ &\quad + \left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|\|\theta_k - \theta_{k-1}\| + \frac{\alpha_k}{2}\|\theta_k - \theta_{k-1}\|^2 \\ &\leq \frac{(L_{\mathcal{H}} + \alpha_k)}{2}\|\theta_k - \theta_{k-1}\|^2 + \|\nabla J(\theta_{k-1}) - \bar{g}_k\| + \left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|\|\theta_k - \theta_{k-1}\| \\ &\leq (L_{\mathcal{H}} + \alpha_k)\|\theta_k - \theta_{k-1}\|^2 + \|\nabla J(\theta_{k-1}) - \bar{g}_k\| + \frac{\left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|^2}{2(L_{\mathcal{H}} + \alpha_k)},\end{aligned} \tag{29}$$

where we used Young's inequality. In the last step, we take expectation on both sides and use the relations in (28) to obtain

$$\frac{\left(\mathbb{E}\left[\|\nabla J(\theta_k)\| - \delta_k^g - \delta_k^{\mathcal{H}}\right]\right)}{L_{\mathcal{H}} + \alpha_k} \leq \mathbb{E}\left[\|\theta_k - \theta_{k-1}\|^2\right]. \tag{30}$$

By the inequality in Lemma 6, and the smoothness result in Lemma 5, we have

$$\begin{aligned}\nabla^2 J(\theta_k) &\succeq \nabla^2 J(\theta_{k-1}) - L_{\mathcal{H}} \|\theta_k - \theta_{k-1}\| I_d = \nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k + \bar{\mathcal{H}}_k - L_{\mathcal{H}} \|\theta_k - \theta_{k-1}\| I_d \\ &\succeq \nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k - \frac{(\alpha_k + 2L_{\mathcal{H}}) \|\theta_k - \theta_{k-1}\|}{2} I_d,\end{aligned}$$

which, from Lemma 8, implies that

$$\begin{aligned}\frac{(\alpha_k + 2L_{\mathcal{H}}) \|\theta_k - \theta_{k-1}\|}{2} &\geq \lambda_{\max}\left(-\nabla^2 J(\theta_k)\right) - \left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|, \\ &= -\lambda_{\min}\left(\nabla^2 J(\theta_k)\right) - \left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|.\end{aligned} \tag{31}$$

Taking expectations on both sides, and using the definition of $\delta_k^{\mathcal{H}}$ in (28), we have

$$\begin{aligned}\sqrt{\mathbb{E}\left[\|\theta_k - \theta_{k-1}\|^2\right]} &\geq \mathbb{E}\left[\|\theta_k - \theta_{k-1}\|\right] \\ &\geq \frac{-2}{\alpha_k + 2L_{\mathcal{H}}}\left[\mathbb{E}\left[\lambda_{\min}\left(\nabla^2 J(\theta_k)\right)\right] + \sqrt{2(\alpha_k + L_{\mathcal{H}})\delta_k^{\mathcal{H}}}\right].\end{aligned} \tag{32}$$

Combining the above inequality with (30), we obtain (27). □

Lemma 10. *Let $\{\theta_k\}$ be computed by Algorithm [1] for a given iteration limit $N \geq 1$, we have*

$$\begin{aligned}&\mathbb{E}\left[\|\theta_R - \theta_{R-1}\|^3\right] \\ &\leq \frac{36}{\sum_{k=1}^N \alpha_k}\left[J(\theta_0) - J^* + \sum_{k=1}^N \frac{4\left(\delta_k^g\right)^{\frac{3}{2}}}{\sqrt{3\alpha_k}} + \sum_{k=1}^N \left(\frac{18\sqrt[4]{2}}{\alpha_k}\right)^2 \left((L_{\mathcal{H}} + \alpha_k)\delta_k^{\mathcal{H}}\right)^{\frac{3}{2}}\right],\end{aligned} \tag{33}$$

where R is a random variable whose probability distribution $P_R(\cdot)$ is supported on $\{1, \ldots, N\}$ and given by

$$P_R(R = k) = \frac{\alpha_k}{\sum_{k=1}^N \alpha_k}, \qquad k = 1, \ldots, N, \tag{34}$$

and $\delta_k^g, \delta_k^{\mathcal{H}} > 0$ are defined as before in (28)*.*

Proof. We can see that by Lemma 5, (6) and the fact that $\alpha_k \geq L_{\mathcal{H}}$, we have

$$J(\theta_k) \leq J(\theta_{k-1}) + \tilde{J}^k(\theta_k) + \|\nabla J(\theta_{k-1} - \bar{g}_k)\| \|\theta_k - \theta_{k-1}\| + \frac{1}{2}\left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\| \|\theta_k - \theta_{k-1}\|^2. \tag{35}$$

Moreover, by Lemma 6, we have

$$\tilde{J}^k(\theta_k) = -\frac{1}{2}\left\langle \bar{\mathcal{H}}_k(\theta_k - \theta_{k-1}), (\theta_k - \theta_{k-1})\right\rangle - \frac{\alpha_k}{3}\|\theta_k - \theta_{k-1}\|^3 \leq -\frac{\alpha_k}{12}\|\theta_k - \theta_{k-1}\|^3. \tag{36}$$

Combining (35) and (36), we obtain

$$\begin{aligned}\frac{\alpha_k}{12}\|\theta_{k-1} - \theta_k\|^3 &\leq J(\theta_{k-1}) - J(\theta_k) + \|\nabla J(\theta_{k-1} - \bar{g}_k)\| \|\theta_k - \theta_{k-1}\| \\ &\quad + \frac{1}{2}\left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\| \|\theta_k - \theta_{k-1}\|^2 \\ &\leq J(\theta_{k-1}) - J(\theta_k) + \frac{4}{\sqrt{3\alpha_k}}\|\nabla J(\theta_{k-1} - \bar{g}_k)\|^{\frac{3}{2}} \\ &\quad + \left(\frac{9\sqrt{2}}{\alpha_k}\right)^2 \left\|\nabla^2 J(\theta_{k-1} - \bar{\mathcal{H}}_k)\right\|^3 + \frac{\alpha_k}{18}\|\theta_k - \theta_{k-1}\|^3,\end{aligned} \tag{37}$$

where the last inequality follows from the fact $ab \leq \frac{a^p}{\lambda^p p} + \frac{\lambda^q b^q}{q}$ for p, q satisfying $\frac{1}{p} + \frac{1}{q} = 1$ and $\lambda > 0$.

We now take expectation on both sides of (37) and use (28) to obtain

$$\frac{\alpha_k}{36}\mathbb{E}\left[\|\theta_k - \theta_{k-1}\|^3\right] \leq J(\theta_{k-1}) - J(\theta_k) + \frac{4\left(\delta_k^g\right)^{\frac{3}{2}}}{\sqrt{3\alpha_k}} + \left(\frac{18\sqrt[4]{2}}{\alpha_k}\right)^2 \left((L_{\mathcal{H}} + \alpha_k)\delta_k^{\mathcal{H}}\right)^{\frac{3}{2}}.$$

Summing over $k = 1, \ldots, N$, dividing both sides by $\sum_{k=1}^N \alpha_k$ and noting (34), we obtain the bound in (33). □

Proof of Theorem 2

Proof. First, note that by (7), Lemma 21, we can ensure that (28) is satisfied by $\delta_k^g = 2\epsilon/5$ and $\delta_k^{\mathcal{H}} = \epsilon/144$. Moreover, by Lemma 10, we have

$$\begin{aligned}\mathbb{E}\left[\|\theta_R - \theta_{R-1}\|^3\right] &\leq \frac{12}{L_{\mathcal{H}}}\left[\frac{J(\theta_0) - J^*}{N} + \frac{4\,(2/5)^{\frac{3}{2}}}{3\sqrt{L_{\mathcal{H}}}}\epsilon^{\frac{3}{2}} + \frac{18^2\sqrt{2}}{9\cdot 6^3\sqrt{L_{\mathcal{H}}}}\epsilon^{\frac{3}{2}}\right] \\ &\leq \frac{1}{L_{\mathcal{H}}^{\frac{3}{2}}}\left[\frac{12\sqrt{L_{\mathcal{H}}}(J(\theta_0) - J^*)}{N} + 6.88\epsilon^{\frac{3}{2}}\right] \\ &\leq \frac{8\epsilon^{\frac{3}{2}}}{L_{\mathcal{H}}^{\frac{3}{2}}}. \end{aligned} \tag{38}$$

The inequality in (38) follows by substituting the value of N specified in the theorem statement. Furthermore, from Lemma 9 and using Lyapunov inequality, we obtain

$$\left[\mathbb{E}\left[\|\theta_R - \theta_{R-1}\|^2\right]\right]^{1/2} \leq \left[\mathbb{E}\left[\|\theta_R - \theta_{R-1}\|^3\right]\right]^{1/3} \leq \frac{2\epsilon^{\frac{1}{2}}}{L_{\mathcal{H}}^{\frac{1}{2}}}.$$

Using the bound above in conjunction with (30) and (32), we obtain

$$\sqrt{\mathbb{E}\left[\|\nabla J(\theta_k)\|\right]} \leq \sqrt{\left(16 + \frac{2}{5} + \frac{1}{144}\right)\epsilon} \leq 5\sqrt{\epsilon},$$

and

$$\frac{\mathbb{E}\left[-\lambda_{\min}\left(\nabla^2 J(\theta_k)\right)\right]}{\sqrt{L_{\mathcal{H}}}} \leq \left(5 + \frac{1}{3\sqrt{2}}\right)\sqrt{\epsilon} \leq 6\sqrt{\epsilon}.$$

The main result in (8) follows from the two inequalities above.

Finally, note that the total number of required samples to obtain such a solution is bounded by

$$\sum_{k=1}^{N} m_k = O\left(\frac{1}{\epsilon^{\frac{7}{2}}}\right), \qquad \sum_{k=1}^{N} b_k = O\left(\frac{d^{\frac{2}{3}}}{\epsilon^{\frac{5}{2}}}\right).$$

□

D Proof of Lemma 1

Proof. Recall that $g(\theta;\tau) = \nabla\Phi(\theta,\tau)$. By Lemma 4 we have $\|\nabla\Phi(\theta;\tau)\| \leq GKH^2$. Further, from (4) $\|\nabla J(\theta)\| \leq KGH^3$. Hence,

$$\begin{aligned}\|g(\theta;\tau) - \nabla J(\theta)\| &\leq \|g(\theta;\tau)\| + \|\nabla J(\theta)\| \\ &\leq GKH^2 + KGH^3 \\ &= GKH^2(H+1) = M_1.\end{aligned}$$

Squaring and taking expectations, we obtain

$$\mathbb{E}\,\|g(\theta;\tau) - \nabla J(\theta)\|^2 \leq M_1^2.$$

Next, we establish bounds on the Hessian estimate. Note that $\|\mathcal{H}(\theta;\tau)\| \leq H^3G^2K + L_1KH^2 = G_{\mathcal{H}}$. Further from (4) we have $\left\|\nabla^2 J(\theta)\right\| \leq G_{\mathcal{H}}$. Hence,

$$\begin{aligned}\left\|\mathcal{H}(\theta,\tau) - \nabla^2 J(\theta))\right\| &\leq 2G_{\mathcal{H}} = M_2, \\ \mathbb{E}\left\|\mathcal{H}(\theta,\tau) - \nabla^2 J(\theta))\right\|^2 &\leq M_2^2.\end{aligned}$$

□

E Proof of Theorem 3

For ease of exposition, we restate Lemma 3 below. This result will be used subsequently in the proof of Theorem 3.

Lemma 11. *Let* $m_k = \max\left(\frac{M_1}{t}, \frac{M_1^2}{t^2}\right)\frac{8}{3}\log\frac{2d}{\delta'}$, $b_k = \max\left(\frac{M_2}{\sqrt{t_1}}, \frac{M_2^2}{t_1}\right)\frac{8}{3}\log\frac{2d}{\delta'}$ *any positive constants and* $\delta' \in (0,1)$. *Then, with probability* $1-\delta'$ *we have*

$$\|\bar{g}_k - \nabla J(\theta_k)\|^2 \leq t^2, \text{ and } \left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta)\right\|^3 \leq t_1^{\frac{3}{2}}.$$

Proof. Following [26, equation 2.2.8], we define the matrix variance statistic of a random matrix Z as

$$\begin{aligned} v(Z) &= \max\{\|\mathbf{Var}_1(Z)\|, \|\mathbf{Var}_2(Z)\|\}, \text{ where} \\ \mathbf{Var}_1(Z) &= \mathbb{E}\left[(Z-\mathbb{E}(Z))(Z-\mathbb{E}(Z))^T\right], \text{ and} \\ \mathbf{Var}_2(Z) &= \mathbb{E}\left[(Z-\mathbb{E}(Z))^T(Z-\mathbb{E}(Z))\right]. \end{aligned}$$

Letting $\nabla\tilde{\Phi}(\theta;\tau) = \nabla\Phi(\theta;\tau) - \nabla J(\theta)$, we have the following expression for the centered gradient estimate:

$$\tilde{g}_k = \frac{1}{m_k}\sum_{\tau\in\mathcal{T}_m}\left(\nabla\tilde{\Phi}(\theta;\tau)\right).$$

Using the triangle inequality and Jensen's inequality, the matrix variance $v(\tilde{g}_k)$ is simplified bounded as follows:

$$\begin{aligned} v(\tilde{g}_k) &= \frac{1}{m_k^2}\max\left\{\left\|\mathbb{E}\sum_{\tau\in\mathcal{T}_m}\nabla\tilde{\Phi}(\theta;\tau)\nabla\tilde{\Phi}(\theta;\tau)^T\right\|, \left\|\mathbb{E}\sum_{\tau\in\mathcal{T}_{m_k}}\nabla\tilde{\Phi}(\theta;\tau)^T\nabla\tilde{\Phi}(\theta;\tau)\right\|\right\}, \\ &\leq \frac{1}{m_k^2}\max\left\{\mathbb{E}\sum_{\tau\in\mathcal{T}_m}\left\|\nabla\tilde{\Phi}(\theta;\tau)\nabla\tilde{\Phi}(\theta;\tau)^T\right\|, \mathbb{E}\sum_{\tau\in\mathcal{T}_{m_k}}\left\|\nabla\tilde{\Phi}(\theta;\tau)^T\nabla\tilde{\Phi}(\theta;\tau)\right\|\right\} \leq \frac{M_1^2}{m_k}. \end{aligned}$$

From an application of matrix Bernstein inequality, see [26, Theorem 7.3.1], we obtain

$$\mathbb{P}[\|\bar{g}_k - \nabla J(\theta)\| \geq t] \leq 2d\exp\left(-\frac{t^2/2}{v(\tilde{g}_k) + M_1 t/(3m_k)}\right) \leq 2d\exp\left(-\frac{3m_k}{8}\min\left\{\frac{t}{M_1}, \frac{t^2}{M_1^2}\right\}\right).$$

Thus, for $m_k \geq \max\left(\frac{M_1}{t}, \frac{M_1^2}{t^2}\right)\frac{8}{3}\log\frac{2d}{\delta'}$, we have

$$\|\bar{g}_k - \nabla J(\theta)\| \leq t \text{ with probability } 1-\delta'.$$

The first claim follows.

Next, we turn to proving the second claim concerning the Hessian estimate $\bar{\mathcal{H}}_k$. As in the case of the high-probability bound for the gradient estimate above, we define $\tilde{\mathcal{H}}(\theta;\tau) = \mathcal{H}(\theta;\tau) - \nabla^2 J(\theta)$, and the centered Hessian $\tilde{\mathcal{H}}_k = \frac{1}{b_k}\sum_{\tau\in\mathcal{T}_b}\tilde{\mathcal{H}}(\theta;\tau)$. The variance of $\tilde{\mathcal{H}}_k$ can be bounded as follows:

$$v[\tilde{\mathcal{H}}_k] = \frac{1}{b_k^2}\left\|\sum_{\tau\in\mathcal{T}_{b_k}}\mathbb{E}\left[\left(\tilde{\mathcal{H}}(\theta;\tau)\right)^2\right]\right\| \leq \frac{M_2^2}{b_k}.$$

Applying the matrix Bernstein inequality for the centered Hessian leads to the following bound:

$$\mathbb{P}[\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta)\right\| \geq t'] \leq 2d\exp\left(-\frac{3b_k}{8}\min\left\{\frac{t'}{M_2}, \frac{t'^2}{M_2^2}\right\}\right)$$

Thus, for $b_k \geq \max\left(\frac{M_2}{t'}, \frac{M_2^2}{t'^2}\right)\frac{8}{3}\log\frac{2d}{\delta'}$, we have

$$\left\|\bar{\mathcal{H}}_k - \nabla^2 J(\theta)\right\| \leq t' \text{ with probability } 1-\delta'.$$

The claim concerning $\bar{\mathcal{H}}_k$ follows by setting $t' = \sqrt{t_1}$. □

Lemma 12. *Let $\{\theta_k\}$ be computed by Algorithm 1. Then with,m_k, b_k as in Lemma 11, we have*

$$\|\theta_k - \theta_{k-1}\| \geq \max\left\{\sqrt{\frac{\|\nabla J(\theta_k)\| - t - t_1}{L_{\mathcal{H}} + \alpha_K}}, \frac{-2}{\alpha_k + 2L_{\mathcal{H}}}\left[\lambda_{\min}\left(\nabla^2 J(\theta_k)\right) + \sqrt{2(\alpha_k + L_{\mathcal{H}})t_1}\right]\right\}, \tag{39}$$

with probability $1 - 2\delta'$.

Proof. We first recall (29) from the proof of Lemma 9 below.

$$\|\nabla J(\theta_k)\| \leq (L_{\mathcal{H}} + \alpha_k)\|\theta_k - \theta_{k-1}\|^2 + \|\nabla J(\theta_{k-1}) - \bar{g}_k\| + \frac{\left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|^2}{2(L_{\mathcal{H}} + \alpha_k)},$$

From Lemma 11, with probability $1 - \delta'$, we have

$$\|\nabla J(\theta_{k-1}) - \bar{g}_k\|^2 \leq t^2, \qquad \left\|\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k\right\|^3 \leq \left(2(L_{\mathcal{H}} + \alpha_k)t_1\right)^{\frac{3}{2}}. \tag{40}$$

Thus, with probability $1 - 2\delta'$,

$$\sqrt{\frac{(\|\nabla J(\theta_k)\| - t - t_1)}{L_{\mathcal{H}} + \alpha_k}} \leq \|\theta_k - \theta_{k-1}\|. \tag{41}$$

Recall that (31) from the proof of Lemma 9 established the following inequality:

$$\frac{(\alpha_k + 2L_{\mathcal{H}})\|\theta_k - \theta_{k-1}\|}{2} \geq \lambda_{min}(\nabla^2 J(\theta_{k-1}) - \bar{\mathcal{H}}_k) - \lambda_{\min}\left(\nabla^2 J(\theta_k)\right).$$

Using the bounds from (40) in the inequality above, we obtain

$$\|\theta_k - \theta_{k-1}\| \geq \frac{-2}{\alpha_k + 2L_{\mathcal{H}}}\left[\lambda_{\min}\left(\nabla^2 J(\theta_k)\right) + \sqrt{2(\alpha_k + L_{\mathcal{H}}))t_1}\right].$$

Combining the above inequality with (41), we obtain (39). Hence proved. □

Lemma 13. *Let $\{\theta_k\}$ be computed by Algorithm [1] for a given iteration limit $N \geq 1$. Then under the setting of Lemma 11 we have*

$$\begin{aligned}&\|\theta_R - \theta_{R-1}\|^3 \\ &\leq \frac{36}{\sum_{k=1}^N \alpha_k}\left[J(\theta_0) - J^* + \sum_{k=1}^N \frac{4t^{\frac{3}{2}}}{\sqrt{3\alpha_k}} + \sum_{k=1}^N \left(\frac{18\sqrt[4]{2}}{\alpha_k}\right)^2 \left((L_{\mathcal{H}} + \alpha_k)t_1\right)^{\frac{3}{2}}\right],\end{aligned} \tag{42}$$

with probability $1 - 2\delta' N$ and R is a random variable with distribution specified in Lemma 10

Proof. From (35),(36) and (37) in the proof of Lemma 10, we have

$$\begin{aligned}\frac{\alpha_k}{12}\|\theta_{k-1} - \theta_k\|^3 \leq J(\theta_{k-1}) - J(\theta_k) + \frac{4}{\sqrt{3\alpha_k}}\|\nabla J(\theta_{k-1} - \bar{g}_k)\|^{\frac{3}{2}} + \left(\frac{9\sqrt{2}}{\alpha_k}\right)^2 \left\|\nabla^2 J(\theta_{k-1} - \bar{\mathcal{H}}_k)\right\|^3 \\ + \frac{\alpha_k}{18}\|\theta_k - \theta_{k-1}\|^3.\end{aligned}$$

Rearranging the terms above and using the bounds from (40), we obtain

$$\frac{\alpha_k}{36}\|\theta_k - \theta_{k-1}\|^3 \leq J(\theta_{k-1}) - J(\theta_k) + \frac{4t^{\frac{3}{2}}}{\sqrt{3\alpha_k}} + \left(\frac{18\sqrt[4]{2}}{\alpha_k}\right)^2 \left((L_{\mathcal{H}} + \alpha_k)t_1\right)^{\frac{3}{2}}.$$

Summing over $k = 1, \ldots, N$, dividing both sides by $\sum_{k=1}^N \alpha_k$, and noting the fact that after N-th iteration of Algorithm 1, the concentration bounds in Lemma 11 hold with probability $1 - 2\delta' N$, we obtain the bound in (42) with probability $1 - 2\delta' N$. □

Proof of Theorem 3

Proof. By Lemma 13 and (38) with probability $1 - 2\delta' N$, we have

$$\|\theta_R - \theta_{R-1}\|^3 \leq \frac{8\epsilon^{\frac{3}{2}}}{L_{\mathcal{H}}^{\frac{3}{2}}},$$

where we choose N according to (9).

From Lemma 12, with probability $1 - 2\delta' N$, we have

$$\sqrt{\|\nabla J(\theta_R)\|} \leq \sqrt{\left(16 + \frac{2}{5} + \frac{1}{144}\right)\epsilon} \leq 5\sqrt{\epsilon} \quad \textit{and} \quad \frac{-\lambda_{\min}\left(\nabla^2 J(\theta_R)\right)}{\sqrt{L_{\mathcal{H}}}} \leq \left(5 + \frac{1}{3\sqrt{2}}\right)\sqrt{\epsilon} \leq 6\sqrt{\epsilon}.$$

Thus, (10) follows implying θ_R is a ϵ-SOSP with high-probability. □

F Proof of Lemma 8

We state and prove a useful result that will imply bound in Lemma 8.

Lemma 14. *For any square matrix $A \in \mathbb{R}^{d\times d}$ and for all vectors $v \in \mathbb{R}^d$, the following holds*

$$v^\top A v \leq \lambda \|v\|^2, \tag{43}$$

for some $\lambda \in \mathbb{R}$, if and only if

$$\lambda_{\max}(A) \leq \lambda. \tag{44}$$

Proof. We shall first prove the forward argument which is quite trivial. We define the map

$$\lambda_v(A) := \frac{v^\top A v}{\|v\|^2}$$

Now, given (43), we shall find a v_* such that $Av_* = \lambda_{max}(A)v_*$, i.e. v_* is the eigenvector associated with the largest eigenvalue of A. As $v_* \in \mathcal{C}(A) \subseteq \mathbb{R}^d$, the following should hold

$$\lambda_{v_*}(A) \leq \lambda.$$

But,

$$\lambda_{v_*}(A) = \frac{v_*^\top A v_*}{\|v_*\|^2} = \lambda_{\max}(A)\frac{v_*^\top v_*}{\|v_*\|^2} = \lambda_{\max}(A) \leq \lambda.$$

Now to check if the converse holds, we start by arguing that $\lambda_v(A) \leq \lambda_{\max}(A)$ for all v and A. We argue that $\lambda_v(A) \in [\lambda_{\min}(A), \lambda_{\max}(A)]$, as it has the form

$$\lambda_v(A) = \frac{\sum_{i=1}^r \lambda_i a_i^2}{\sum_{i=1}^r a_i^2},$$

where r is the rank of A and a_i are the coefficients of v. Hence, $\lambda_v(A)$ can be thought of as a weighted average of all eigenvalues of A. Therefore, given (44), we have for all $v \in \mathbb{R}^d$,

$$\lambda_v \leq \lambda,$$

which satisfies (43). □

Proof. (Lemma 8) For all $v \in \mathbb{R}^d$, we have

$$\begin{aligned} v^\top A v &\leq v^\top B v + c\|v\|^2 \\ &\leq \|B\| \cdot \|v\|^2 + c\|v\|^2 \\ &= (\|B\| + c)\|v\|^2, \end{aligned}$$

where in the second line, we used the Cauchy-Schwartz inequality. Now by using, Lemma 14 in the last line, we obtain

$$\lambda_{max}(A) \leq \|B\| + c, \quad \text{implying} \quad c \geq \lambda_{max}(A) - \|B\|.$$

□

G A few probabilistic inequalities

We state and prove two probabilistic inequalites, which are used in the proof of Theorem 2. In particular, the result below as well as Rosenthal's inequality (stated in Lemma 16) are used in the proof of Lemma 7

Lemma 15. *Let $Z \in \mathbb{R}^{d\times d}$ be a random matrix. Then, we have*

$$\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^4\right] \leq 5\mathbb{E}\left[\|Z\|^4\right].$$

Proof. We can re-write the expectation as

$$\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^4\right] = \mathrm{Var}\left(\|Z - \mathbb{E}[Z]\|^2\right) + \left(\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^2\right]\right)^2.$$

Consider the first term

$$\begin{aligned}
\mathrm{Var}\left(\|Z - \mathbb{E}[Z]\|^2\right) &= \mathrm{Var}\left(\|Z\|^2 + \|\mathbb{E}[Z]\|^2 - 2\langle Z, \mathbb{E}[Z]\rangle\right) \\
&= \mathrm{Var}\left(\|Z\|^2 - 2\langle Z, \mathbb{E}[Z]\rangle\right) \qquad \left(\because \mathrm{Var}\left(\|\mathbb{E}[Z]\|^2\right) = 0\right) \\
&\leq \mathrm{Var}\left(\|Z\|^2\right) + 4\mathrm{Var}\left(\langle Z, \mathbb{E}[Z]\rangle\right) + 4\sqrt{\mathrm{Var}\left(\|Z\|^2\right)}\sqrt{\mathrm{Var}\left(\langle Z, \mathbb{E}[Z]\rangle\right)}.
\end{aligned}$$

Now for the second term

$$\begin{aligned}
\left(\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^2\right]\right)^2 &= \left(\mathbb{E}\left[\|Z\|^2\right] - \|\mathbb{E}[Z]\|^2\right)^2 \\
&= \left(\mathbb{E}\left[\|Z\|^2\right]\right)^2 + \|\mathbb{E}[Z]\|^4 - 2\mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2.
\end{aligned}$$

Simplifying the terms under the root

$$\begin{aligned}
\sqrt{\mathrm{Var}\left(\|Z\|^2\right)} &= \sqrt{\mathbb{E}\left[\|Z\|^4\right] - \left(\mathbb{E}\left[\|Z\|^2\right]\right)^2} \\
&= \sqrt{\mathbb{E}\left[\|Z\|^4\right]}\sqrt{1 - \frac{\left(\mathbb{E}\left[\|Z\|^2\right]\right)^2}{\mathbb{E}\left[\|Z\|^4\right]}} \\
&\leq \sqrt{\mathbb{E}\left[\|Z\|^4\right]}\left(1 - \frac{\left(\mathbb{E}\left[\|Z\|^2\right]\right)^2}{2\mathbb{E}\left[\|Z\|^4\right]}\right)
\end{aligned}$$

where in the last inequality we used the fact that $\sqrt{1-x} \leq 1 - \frac{x}{2}$.

$$\begin{aligned}
\mathrm{Var}\left(\langle Z, \mathbb{E}[Z]\rangle\right) &= \mathbb{E}\left[\langle Z, \mathbb{E}[Z]\rangle^2\right] - \left(\mathbb{E}\left[\langle Z, \mathbb{E}[Z]\rangle\right]\right)^2 \\
&\leq \mathbb{E}\left[\|Z\|^2\|\mathbb{E}[Z]\|^2\right] - \|\mathbb{E}[Z]\|^4 \\
&\leq \mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2.
\end{aligned}$$

Putting these results together

$$\begin{aligned}
\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^4\right] \leq\ & \mathrm{Var}\left(\|Z\|^2\right) + \left(\mathbb{E}\left[\|Z\|^2\right]\right)^2 \\
& + 4\mathrm{Var}\left(\langle Z, \mathbb{E}[Z]\rangle\right) + \|\mathbb{E}[Z]\|^4 - 2\mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2 \\
& + 4\sqrt{\mathrm{Var}\left(\|Z\|^2\right)}\sqrt{\mathrm{Var}\left(\langle Z, \mathbb{E}[Z]\rangle\right)}
\end{aligned}$$

$$\leq \mathbb{E}\left[\|Z\|^4\right] + 2\mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2 - 3\|\mathbb{E}[Z]\|^4 + 4\sqrt{\mathbb{E}\left[\|Z\|^4\right]}\left(1 - \frac{\left(\mathbb{E}\left[\|Z\|^2\right]\right)^2}{2\mathbb{E}\left[\|Z\|^4\right]}\right)\sqrt{\mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2}.$$

Note that by Jensen's inequality, we have $\mathbb{E}\left[\|Z\|^2\right]\|\mathbb{E}[Z]\|^2 \leq \left(\mathbb{E}\left[\|Z\|^2\right]\right)^2 \leq \mathbb{E}\left[\|Z\|^4\right]$. Substituting these results above and further simplification, we have

$$\mathbb{E}\left[\|Z - \mathbb{E}[Z]\|^4\right] \leq 5\mathbb{E}\left[\|Z\|^4\right] - 3\|\mathbb{E}[Z]\|^4 \leq 5\mathbb{E}\left[\|Z\|^4\right].$$

□

Lemma 16 (Rosenthal's inequality)**.** *Let $\{X_1, \ldots, X_n\}$ be a sequence of random $d \times d$ square matrices with $\mathbb{E}[X_i] = 0$ for all i. Then the following inequality holds*

$$\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|_F^4\right] \leq 3n^2\mathbb{E}\left[\|X_i\|_F^4\right],$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix.

Proof. We start by using the definition of variance as follows

$$\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|_F^4\right] = \mathrm{Var}\left(\left\|\sum_{i=1}^n X_i\right\|_F^2\right) + \left(\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|_F^2\right]\right)^2. \tag{45}$$

Consider the first term in (45)

$$\begin{aligned}
\mathrm{Var}\left(\left\|\sum_{i=1}^n X_i\right\|_F^2\right) &= \mathrm{Var}\left(\mathrm{Tr}\left(\sum_i X_i^\top \sum_j X_j\right)\right) \\
&= \mathrm{Var}\left(\mathrm{Tr}\left(\sum_i X_i^\top X_i + 2\sum_{i<j} X_i^\top X_j\right)\right) \\
&= \mathrm{Var}\left(\sum_i \|X_i\|_F^2 + 2\sum_{i<j}\mathrm{Tr}\left(X_i^\top X_j\right)\right) \\
&= \sum_i \mathrm{Var}\left(\|X_i\|_F^2\right) + 4\sum_{i<j}\mathrm{Var}\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right).
\end{aligned}$$

Expanding the terms under summation

$$\begin{aligned}
\sum_i \mathrm{Var}\left(\|X_i\|_F^2\right) &= \sum_i \mathbb{E}\left[\|X_i\|_F^4\right] - \sum_i \left(\mathbb{E}\left[\|X_i\|_F^2\right]\right)^2, \\
\sum_{i<j}\mathrm{Var}\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right) &= \sum_{i<j}\mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right)^2\right] - \sum_{i<j}\left(\mathbb{E}\left[\mathrm{Tr}\left(X_i^\top X_j\right)\right]\right)^2 \\
&= \sum_{i<j}\mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right)^2\right]. \qquad (\because \mathbb{E}\left[\mathrm{Tr}\left(X_i^\top X_j\right)\right] = 0 \text{ for } i \neq j)
\end{aligned}$$

Therefore,

$$\mathrm{Var}\left(\left\|\sum_{i=1}^n X_i\right\|_F^2\right) = \sum_i \mathbb{E}\left[\|X_i\|_F^4\right] + 4\sum_{i<j}\mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right)^2\right] - \sum_i\left(\mathbb{E}\left[\|X_i\|_F^2\right]\right)^2$$

Table 2: Simulation hyper-parameter settings

	Cart-Pole	**Reacher**	**Humanoid**
Horizon	500	50	1000
Batch-size	50	100	100
NN hidden sizes	-	32×32	64×64
NN hidden activation	-	softmax	softmax
REINFORCE λ	0.01	$10^{-8/3}$	$10^{-8/3}$
ACR-PN α	10^3	10^4	10^4
CR-PN α	10^3	-	-

$$
\begin{aligned}
&= \sum_i \mathbb{E}\left[\|X_i\|_F^4\right] + 2\sum_{i\neq j}\mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right)^2\right] - \sum_i\left(\mathbb{E}\left[\|X_i\|_F^2\right]\right)^2 \\
&\leq 2\sum_i\sum_j \mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_j\right)\right)^2\right] \leq 2\sum_i\sum_j \mathbb{E}\left[\left(\mathrm{Tr}\left(X_i^\top X_i\right)\right)^2\right] \\
&= 2n^2\mathbb{E}\left[\|X_i\|_F^4\right].
\end{aligned}
$$

In the second last line we used the property of inner products, i.e. $\langle X_i, X_j\rangle \leq \langle X_i, X_i\rangle = \|X_i\|^2$ for all (i,j) pairs. Now taking the second term in (45)

$$
\begin{aligned}
\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|_F^2\right] &= \mathbb{E}\left[\mathrm{Tr}\left(\sum_i X_i^\top \sum_j X_j\right)\right] \\
&= \mathrm{Tr}\left(\sum_i\sum_j \mathbb{E}\left[X_i^\top X_j\right]\right) \\
&= \mathrm{Tr}\left(\sum_i \mathbb{E}\left[X_i^\top X_i\right]\right) = \sum_i \mathbb{E}\left[\|X_i\|_F^2\right] = n\mathbb{E}\left[\|X_i\|_F^2\right].
\end{aligned}
$$

Plugging these results in (45)

$$
\begin{aligned}
\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|_F^4\right] &\leq 2n^2\mathbb{E}\left[\|X_i\|_F^4\right] + n^2\left(\mathbb{E}\left[\|X_i\|_F^2\right]\right)^2 \\
&\leq 2n^2\mathbb{E}\left[\|X_i\|_F^4\right] + n^2\mathbb{E}\left[\|X_i\|_F^4\right] \qquad \text{(Jensen's inequality.)} \\
&= 3n^2\mathbb{E}\left[\|X_i\|_F^4\right].
\end{aligned}
$$

□

H Additional simulation details

Table 2 lists the various hyper-parameter values used in our experiments.

On comparing CRPN's α and REINFORCE's λ

The performance of CRPN and REINFORCE algorithms is highly sensitive to the selection of their respective hyper-parameters. For a fair comparison, we chose the hyper-parameters for the two algorithms so that both converge to an ϵ-FOSP. Towards this, we first provide a bound for REINFORCE algorithm in the result below.

Theorem 5 (***Bound in expectation for REINFORCE***). *Let θ_k be computed by REINFORCE where λ_k is the step-size at k, such that*

$$\theta_k = \theta_{k-1} - \lambda_k \bar{g}_k,$$

For any k, the step-size λ_k and batch size m_k are set as follows:

$$\lambda_k = \lambda = \frac{25^3}{4G_{\mathcal{H}}}, m_k = \frac{25G_g^2}{4\epsilon^2}.$$

Set $N = \frac{2(J(\theta_0)-J^*)}{25^2\lambda\epsilon^2}$ and choose θ_R uniformly at random from $\{\theta_1, \ldots, \theta_N\}$. Then, we have

$$\mathbb{E}\left[\|\nabla J(\theta_R)\|\right] \leq 25\epsilon.$$

where $G_{\mathcal{H}}$, $L_{\mathcal{H}}$ and G_g are defined as in (1)*.*

Proof. Assuming the objective is $G_{\mathcal{H}}$-smooth, we have the following according to Taylor's theorem

$$J(\theta_k) = J(\theta_{k-1} - \lambda_k \bar{g}_k) \leq J(\theta_{k-1}) - \lambda_k \bar{g}_k^T \nabla J(\theta_{k-1}) + \frac{\lambda_k^2}{2} G_{\mathcal{H}} \|\bar{g}_k\|^2,$$

taking expectation on both sides

$$\begin{aligned}
\mathbb{E}\left[J(\theta_k)\right] &\leq \mathbb{E}\left[J(\theta_{k-1})\right] - \lambda_k \mathbb{E}\left[\bar{g}_k\right]^T \nabla J(\theta_{k-1}) + \frac{\lambda_k^2}{2} G_{\mathcal{H}} \mathbb{E}\left[\|\bar{g}_k\|^2\right], \\
&\leq \mathbb{E}\left[J(\theta_{k-1})\right] - \lambda_k \mathbb{E}\left[\left\|\nabla J(\theta_{k-1})^2\right\|\right] + \frac{\lambda_k^2}{2} \frac{G_{\mathcal{H}} G_g^2}{m_k}, \\
\lambda_k \mathbb{E}\left[\left\|\nabla J(\theta_{k-1})^2\right\|\right] &\leq \mathbb{E}\left[J(\theta_{k-1})\right] - \mathbb{E}\left[J(\theta_k)\right] + \frac{\lambda_k^2}{2} \frac{G_{\mathcal{H}} G_g^2}{m_k}.
\end{aligned}$$

Taking summation on both sides, and noticing the telescoping term

$$\sum_{k=0}^{N-1} \lambda_k \mathbb{E}\left[\left\|\nabla J(\theta_{k-1})^2\right\|\right] \leq J(\theta_0) - J^* + \sum_{k=0}^{N-1} \frac{\lambda_k^2}{2} \frac{G_{\mathcal{H}} G_g^2}{m_k},$$

If we sample θ_R according to the probability distribution $P_R(\theta_R) = \frac{\lambda_R}{\sum_{k=0}^N \lambda_k}$, then we can treat the L.H.S as a an expectation over this distribution and obtain

$$\begin{aligned}
\frac{\sum_{k=0}^{N-1} \lambda_k \mathbb{E}\left[\left\|\nabla J(\theta_{k-1})^2\right\|\right]}{\sum_{k=0}^{N-1} \lambda_k} &\leq \frac{J(\theta_0) - J^*}{\sum_{k=0}^{N-1} \lambda_k} + \frac{G_{\mathcal{H}} G_g^2}{2m_k} \frac{\sum_{k=0}^{N-1} \lambda_k^2}{\sum_{k=0}^{N-1} \lambda_k}, \\
\mathbb{E}\left[\left\|\nabla J(\theta_R)^2\right\|\right] &\leq \frac{J(\theta_0) - J^*}{\lambda N} + \frac{\lambda G_{\mathcal{H}} G_g^2}{2m_k} := 25^2\epsilon^2.
\end{aligned}$$

Setting $\frac{J(\theta_0)-J^*}{\lambda N} = \frac{\lambda G_{\mathcal{H}} G_g^2}{2m_k} = \frac{25^2\epsilon^2}{2}$, we obtain the above relations. □

Remark 4. *Note that the sample complexity of REINFORCE is $O(1/\epsilon^4)$, greater than our CRPN algorithm.*

Theorem 6 (*Relation between step-size and cubic-regularizer*). *Borrowing the results from* (2) *and* (5) *and by treating the control hyper-parameters, i.e. N, $\Delta_J := J(\theta_0) - J^*$, m_k, $G_{\mathcal{H}}$, $L_{\mathcal{H}}$ and G_g same for both. Then from the following relation*

$$\lambda = \frac{2}{25^2 \cdot 3^{2/3} \cdot 4^{4/3}} \left(\frac{N}{\Delta_J}\right)^{1/3} \alpha^{-2/3},$$

we can assure that both these algorithms converge to the same ϵ-FOSP.

Proof. Invoking the bounds of iterations N from (2) and (5), we can rearrange the terms to remove dependency on ϵ as follows:

$$\epsilon^2 = \frac{2(J(\theta_0) - J^*)}{25^2\lambda N} = \left(\frac{12\sqrt{L_{\mathcal{H}}}(J(\theta_0) - J^*)}{N}\right)^{4/3}.$$

Note that number of iterations N for both algorithms in our experiments are assumed to be the same. Setting $L_{\mathcal{H}} = \alpha/3$, $\Delta_J := J(\theta_0) - J^*$ and re-arranging we obtain the above relation. □

Remark 5. *The above results answer the question as to what is a "fair" relation between λ and α if both were to converge to the same ϵ-FOSP given an identical setup. For our experiments, we set $\lambda = \alpha^{-2/3}$ while taking α large enough to avoid the ill-conditioned sub-problem as described in [8].*

I Computing the *exact* Hessian-vector product efficiently using Auto-grad in an RL setting

Usually calculating the Hessian-vector product (or a linear map $\bar{\mathcal{H}}[\cdot]$) is straight-forward by re-differentiating the quantity given by the inner product of the gradient and the arbitrary vector. For e.g. for a function f and arbitrary vector $v \in \mathbb{R}^d$, $\nabla^2 f \cdot v = \nabla \langle \nabla f, v \rangle$ which just requires 2 calls to the auto-grad function as given by packages like PyTorch etc. However, our policy Hessian estimate consists of two quantities which we shall discuss below. As per convention, let $\mathbf{L_1}^{(i)} := \Phi(\theta; \tau_i)$ and $\mathbf{L_2}^{(i)} := \log p(\tau_i; \theta)$ denote the stacked "losses" corresponding to each of the n trajectories in the mini-batch. Given $\theta \in \mathbb{R}^d$ and $\nabla \equiv \nabla_\theta$, let $\nabla \mathbf{L_1}^{(ij)} := \frac{\partial \mathbf{L_1}^{(j)}}{\partial \theta^{(i)}}$ and $\nabla \mathbf{L_2}^{(ij)} := \frac{\partial \mathbf{L_2}^{(j)}}{\partial \theta^{(i)}}$. Then, $\mathbf{L_1}, \mathbf{L_2} \in \mathbb{R}^n$ and $\nabla \mathbf{L_1}, \nabla \mathbf{L_2} \in \mathbb{R}^{d \times n}$ by construction. Recall from (1), we can now re-write our unbiased estimates in the following matrix-vector form

$$\bar{g} = \frac{1}{n} \sum_i \nabla \mathbf{L_1}^{(i)} = \nabla \left(\frac{1}{n} \sum_i \mathbf{L_1}^{(i)} \right),$$
$$\bar{\mathcal{H}} = \frac{1}{n} \sum_i (\nabla \mathbf{L_1}^{(i)} \nabla^\top \mathbf{L_2}^{(i)} + \nabla^2 \mathbf{L_1}^{(i)}) = \frac{1}{n} \sum_i (\nabla \mathbf{L_1}^{(i)} \nabla^\top \mathbf{L_2}^{(i)}) + \frac{1}{n} \sum_i \nabla^2 \mathbf{L_1}^{(i)}.$$

The resulting Hessian-vector product can be shown as the following in matrix form,

$$\begin{aligned}
\bar{\mathcal{H}} \cdot v &= \frac{1}{n} \sum_i (\nabla \mathbf{L_1}^{(i)} \nabla^\top \mathbf{L_2}^{(i)}) \cdot v + \frac{1}{n} \sum_i \nabla^2 \mathbf{L_1}^{(i)} \cdot v, \\
&= \frac{1}{n} \sum_i \nabla \mathbf{L_1}^{(i)} (\nabla^\top \mathbf{L_2}^{(i)} v) + \nabla \frac{1}{n} \sum_i (\nabla^\top \mathbf{L_1}^{(i)} v), \\
&= \frac{1}{n} \nabla \mathbf{L_1} \nabla^\top \mathbf{L_2} v + \nabla \bar{g}^\top v.
\end{aligned}$$

The second term, which we call $hvp_2 := \nabla \bar{g}^\top v$ is straight-forward to implement, as we already have the computational graph of $\bar{g}$ that we have to retain and re-differentiate its inner product with v. Thus, we require just **one** additional auto-grad call. However, estimating the first term, i.e. hvp_1 is a little tricky. We could use brute-force and calculate the gradient for each element in that mini-batch but that would scale according to $O(n)$. But we can do it in *constant* time $O(1)$ by using the following trick.

Forward auto-diff trick [4]

We first define a function $ujp(f, u; x) := \frac{\partial f}{\partial x} u$, with f and u being vectors of the same dimension and x being the input to differentiate against. This function resembles the function signature and working of the auto-grad function in PyTorch. Recall that $\nabla \mathbf{L_1}, \nabla \mathbf{L_2} \in \mathbb{R}^{d \times n}$ and $u \in \mathbb{R}^n$, $v \in \mathbb{R}^d$, we have

$$ujp(\mathbf{L_2}, u; \theta) = \nabla_\theta \mathbf{L_2} u = \nabla \langle \mathbf{L_2}, u \rangle . \quad \in \mathbb{R}^d$$

The above quantity can be computed in constant time. Now we take the inner product of this quantity with our arbitrary vector input v, and differentiate it w.r.t u to get,

$$ujp(\nabla \mathbf{L_2} u, v; u) = \nabla_u (\nabla \mathbf{L_2} u) v = \nabla_u \langle \nabla \mathbf{L_2} u, v \rangle = \nabla^\top \mathbf{L_2} v. \quad \in \mathbb{R}^n$$

Note that this quantity is now independent of u and thus, theoretically u can be initialized with any non-zero value. We initialize u as a vector of ones. Making our third and final call to our ujp function in the following manner,

$$ujp(\mathbf{L_1}, \nabla^\top \mathbf{L_2} v; \theta) = \nabla \mathbf{L_1} \nabla^\top \mathbf{L_2} v. \quad \in \mathbb{R}^d$$

Notice that this resembles the quantity hvp_1 exactly by a factor of $\frac{1}{n}$. And therefore,

$$hvp_1 = \frac{1}{n} \nabla \mathbf{L_1} \nabla^\top \mathbf{L_2} v = \frac{1}{n} ujp(\mathbf{L_1}, \nabla^\top \mathbf{L_2} v; \theta).$$

Therefore, we make a total of **three** additional auto-grad calls for hvp_1, where the first call, i.e. $ujp(\mathbf{L_2}, u; \theta)$ is independent of the input vector v and thus can be stored in memory instead of re-calculating every-time our linear map $\bar{\mathcal{H}}[\cdot]$ is called. This brings down the time complexity to $O(1)$ making it an algorithm comparable to REINFORCE or other first order methods in practice.

[4] inspired from : `https://j-towns.github.io/2017/06/12/A-new-trick.html`

J Pseudo-code of ACR-PN algorithm

Algorithm 2: Approximate cubic-regularized policy Newton (ACR-PN)

Input : mini-batch sizes m_k, b_k, initialization θ_0, number of iterations $\bar{N}$, and final tolerance ϵ.

for $k = 0$ ***to*** $\bar{N}$ **do**

`/* Monte Carlo simulation */`

Simulate $\min\{m_k, b_k\}$ number of trajectories according to θ_{k-1}, randomly pick m_k trajectories for set $\mathcal{T}_m$ and b_k trajectories for set $\mathcal{T}_b$.;

`/* Gradient estimation */`

$$\bar{g}_k = \frac{1}{m_k} \sum_{\tau \in \mathcal{T}_m} \sum_{h=0}^{H-1} \Psi_h(\tau) \nabla \log \pi(a_h|s_h;\theta_{k-1});$$

`/* Hessian estimation */`

$$\bar{\mathcal{H}}_k = \frac{1}{b_k} \sum_{\tau \in \mathcal{T}_b} \Big(\sum_{h=0}^{H-1} \Psi_h(\tau) \nabla \log \pi(a_h|s_h;\theta_{k-1}) \sum_{h'=0}^{H-1} \nabla^\top \log \pi(a_{h'}|s_{h'};\theta_{k-1}) \Big) + \frac{1}{b_k} \sum_{\tau \in \mathcal{T}_b} \sum_{h=0}^{H-1} \Psi_h(\tau) \nabla^2 \log \pi(a_h|s_h;\theta_{k-1});$$

$\Delta, \delta_J \leftarrow$ Cubic-Subsolver($\bar{g}_k, \bar{\mathcal{H}}_k[\cdot], \epsilon$)

$\theta_{k+1} \leftarrow \theta_k + \Delta$

if $\delta_J \geq -\frac{1}{100}\sqrt{\frac{\epsilon^3}{\alpha}}$ **then**

$\Delta \leftarrow$ Cubic-Finalsolver($\bar{g}_k, \bar{\mathcal{H}}_k[\cdot], \epsilon$)

$\theta^* \leftarrow \theta_k + \Delta$;

break;

end if

end for

Output: θ^* if the early termination condition was reached, otherwise the final iterate $\theta_{\bar{N}}$

Algorithm 3: Cubic-Subsolver via Gradient Descent

Input : gradient estimate $\bar{g}$, Hessian-vector function $\bar{\mathcal{H}}[\cdot]$, and final tolerance ϵ.

if $\|\bar{g}\| \geq \frac{G_{\mathcal{H}}^2}{\alpha}$ **then**

$$R_c \leftarrow -\frac{\bar{g}^\top \bar{\mathcal{H}}[\bar{g}]}{\alpha \|\bar{g}\|^2} + \sqrt{\left(\frac{\bar{g}^\top \bar{\mathcal{H}}[\bar{g}]}{\alpha \|\bar{g}\|^2}\right)^2 + \frac{2\|\bar{g}\|}{\alpha}}$$

$$\Delta \leftarrow -R_c \frac{\bar{g}}{\|\bar{g}\|}$$

end if

else

$\Delta \leftarrow 0, \sigma \leftarrow c' \frac{\sqrt{\epsilon\alpha}}{G_{\mathcal{H}}}, \eta \leftarrow \frac{1}{20 G_{\mathcal{H}}}$

$\tilde{g} \leftarrow \bar{g} + \sigma\zeta$ for $\zeta \sim$ Unif $(\mathbb{S}^{d-1})$

for $k = 0$ ***to*** $\hat{N}$ **do**

$\Delta \leftarrow \Delta - \eta(\tilde{g} + \bar{\mathcal{H}}[\Delta] + \frac{\alpha}{2}\|\Delta\|\Delta)$

end for

end if

$\delta_J \leftarrow \bar{g}^\top \Delta + \frac{1}{2}\Delta\bar{\mathcal{H}}[\Delta] + \frac{\alpha}{6}\|\Delta\|^3$

Output: Δ, δ_J.

Algorithm 4: Cubic-Finalsolver via Gradient Descent

Input : gradient estimate $\bar{g}$, Hessian-vector function $\bar{\mathcal{H}}[\cdot]$, and final tolerance ϵ.

$\Delta \leftarrow 0, \bar{g}_J \leftarrow \bar{g}, \eta \leftarrow \frac{1}{20G_{\mathcal{H}}}$

while $\|\bar{g}_J\| \geq \frac{\epsilon}{2}$ **do**

$\Delta \leftarrow \Delta - \eta \bar{g}_J$

$\bar{g}_J \leftarrow \bar{g}_J + \bar{\mathcal{H}}[\Delta] + \frac{\alpha}{2} \|\Delta\| \Delta$

end while

Output: Δ.